

Moduł szkoleniowy Java

poziom podstawowy 15 godzinny

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Spis treści

Spis treści	2
1 Wprowadzenie	4
2 Cel	5
3 Opis sposobu realizacji celów	5
4 Treści kształcenia	5
5 Opis założonych osiągnięć ucznia	5
5.1 Korelacja z treściami podstawy programowej	6
6 Sposoby osiągania celów	8
6.1 Lista czynności w zależności od roli	8
7 Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia	9
7.1 Wymagania na poszczególne oceny szkolne	11
7.2 Końcowy test teoretyczny	18
8 Lekcje	20
8.1 Lekcja 1 - Wstęp do Javy i środowiska pracy	20
8.1.1 Cel lekcji	20
8.1.2 Treść - slajdy z opisem	21
8.1.3 Opis założonych osiągnięć ucznia	62
8.2 Lekcja 2 - Typy podstawowe i instrukcje sterujące	63
8.2.1 Cel lekcji	63
8.2.2 Treść – slajdy z opisem	63
8.2.3 Opis założonych osiągnięć ucznia	83
8.3 Lekcja 3 - Wstęp do programowania obiektowego	83
8.3.1 Cel lekcji	83
8.3.2 Treść - slajdy z opisem	84
8.3.3 Opis założonych osiągnięć ucznia	105
8.4 Lekcja 4 – Interfejsy, klasy abstrakcyjne i dziedziczenie	105
8.4.1 Cel lekcji	105
8.4.2 Treść - slajdy z opisem	106
8.4.3 Opis założonych osiągnięć ucznia	122
8.5 Lekcja 5 - Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe	123
8.5.1 Cel lekcji	123

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



8.5.2	Treść - slajdy z opisem	123
8.5.3	Opis założonych osiągnięć ucznia	138
8.6	Lekcja 6 - Typ wyliczeniowy, JavaDocs.....	139
8.6.1	Cel lekcji.....	139
8.6.2	Treść - slajdy z opisem	139
8.6.3	Opis założonych osiągnięć ucznia	154
8.7	Lekcja 7 - Tablice jednowymiarowe i wielowymiarowe, kolekcje	154
8.7.1	Cel lekcji.....	154
8.7.2	Treść - slajdy z opisem	154
8.7.3	Opis założonych osiągnięć ucznia	168
8.8	Lekcja 8 - Operacje wejścia – wyjścia i wyjątki	168
8.8.1	Cel lekcji.....	168
8.8.2	Treść - slajdy z opisem	169
8.8.3	Opis założonych osiągnięć ucznia	191
8.9	Lekcja 9 - Graficzny interfejs użytkownika - Swing	192
8.9.1	Cel lekcji.....	192
8.9.2	Treść - slajdy z opisem	192
8.9.3	Opis założonych osiągnięć ucznia	213
8.10	Lekcja 10 - Przechwytywanie zdarzeń, tworzenie plików wykonywalnych	213
8.10.1	Cel lekcji.....	213
8.10.2	Treść - slajdy z opisem	213
8.10.3	Opis założonych osiągnięć ucznia	226
8.11	Lekcja 11 – Projekt Interdyscyplinarny	226
8.11.1	Cel lekcji.....	226
8.11.2	Treść – slajdy z opisem	227
8.11.3	Opis założonych osiągnięć ucznia	271

Człowiek - najlepsza inwestycja

1 Wprowadzenie

Powstanie języka Java miało być remedium na niedoskonałości języka C++, a zarazem krzywa nauczania dla osób przechodzących z tego języka miała być łagodna. W osiągnięciu tego śmiałego celu pomóc twórcom miały następujące założenia:

- **Obiektowość** – w Javie podstawową jednostką programów jest obiekt. Posiada on swój stan (wartości pól) i zachowanie (metody, które mogą zmieniać jego stan). Java jest silnie ukierunkowana obiektowo, w przeciwieństwie to C++ gdzie możliwe jest programowanie proceduralno-obiektowe.
- **Dziedziczenie** – w C++ możliwe było dziedziczenie wielobazowe, w Javie jednak projektanci zdecydowali, że należy uprościć ten model i pozwolić jedynie na dziedziczenie z jednego obiektu. Decyzja ta usunęła niedoskonałość języka C++ polegającą na powstawaniu konfliktów w przypadku, gdy dwie klasy nadrzędne posiadały składowe wzajemnie się wykluczające. Java wprowadziła nową konstrukcję zwaną interfejsem, która to miała na celu przywrócić elastyczność utraconą poprzez pojedyncze dziedziczenie. Kolejną istotną zmianą jest to, iż w Javie każda klasa posiada klasę rodzicielską. U korzenia dziedziczenia klas zawsze znajduje się klasa podstawowa Object.
- **Niezależność od architektury** – jest to jedna z największych przewag Javy nad innymi językami programowania. Kod źródłowy kompiluje się tylko raz, niezależnie od architektury docelowej, na której będzie on uruchamiany. Kompilacja powoduje powstanie bytecode'u, który z kolei uruchamiany jest w maszynie wirtualnej. Podejście to tworząc dodatkową abstrakcyjną warstwę uwolniło programy od konieczności dostosowania się do danej architektury, jednak wprowadziło spadek wydajności (było to zauważalne, zwłaszcza w pierwszych wersjach Javy)
- **Bezpieczeństwo i niezawodność** – w Javie wprowadzono tak zwany system wyjątków. Pozwala on na obsługę nieprzewidywalnych sytuacji w programie, takich jak stracone połączenie do pliku czy próba odczytu indeksu tablicy z poza zakresu tej tablicy. Dzięki umiejętnemu posługiwaniu się tym mechanizmem można obsłużyć takie nietypowe sytuacje, a program może podjąć kolejną próbę wykonania danego zadania lub też kontynuować swoje wykonanie. W języku C++ często takie zdarzenie prowadziło do zamknięcia działania całego programu.
- **Sieciowość i obsługa programowania rozproszonego** – w Javie od samego początku wbudowane są obiekty pozwalające na komunikację zdalną (socket, http, ftp), wywoływanie metod czy przysyłanie całych obiektów między dwoma programami Javy działającymi nawet na odrębnych maszynach (RMI). Java może być uruchamiana w przeglądarkach internetowych pod postacią appletów, jak i działać po stronie serwera.

Po latach, kiedy język dobrze się przyjął i ma rzeszę programistów można śmiało stwierdzić, iż podstawowy cel zastąpienia C++ nie został osiągnięty i raczej nigdy nie zostanie. Prostota języka jest kwestią dyskusyjną, jest on mniej elastyczny niż C++, zarazem chroniąc programistę przed popełnianiem prostych błędów. W chwili obecnej należy traktować Javę, jako w pełni wykształcony

Człowiek - najlepsza inwestycja

odrębny język programowania, który swoją popularności ustępuje jedynie całej rodzinie języków z rodziny C razem wziętych (C, C++, Objective C).

2 Cel

Poznanie zastosowania i zalet języka programowania Java. Umiejętność wykorzystania podstawowych możliwości języka. Praktyczne wykorzystanie elementów środowiska pracy. Podstawowa umiejętność projektowania aplikacji zorientowanej obiektowo. Tworzenie wieloplatformowych aplikacji z interfejsem graficznym. Zwiększenie efektywności pracy poprzez umiejętność korzystania i tworzenia z dokumentacji.

3 Opis sposobu realizacji celów

10 półtoragodzinnych lekcji składających się z treści teoretycznych wraz z ćwiczeniami. Praktyczne uwagi dla osób początkujących dotyczące środowiska pracy Eclipse. Po zakończeniu całego cyklu - przeprowadzenie egzaminu sprawdzającego wiedzę.

Dodatkowo wprowadzono lekcję 11 zawierającą część projektu interdyscyplinarnego łączącego 4 dziedziny wiedzy z zakresu informatyki (SQL, PHP, JavaScript oraz JAVA) w zakresie stworzenia aplikacji mobilnej na platformę Android, wykorzystującej API REST z aplikacji napisanej w PHP.

4 Treści kształcenia

Treść kursu została podzielona na 10 bloków tematycznych – po jednym do każdej lekcji:

1. Wstęp do Javy i środowiska pracy,
2. Typy podstawowe i instrukcje sterujące,
3. Wstęp do programowania obiektowego,
4. Interfejsy, klasy abstrakcyjne i dziedziczenie,
5. Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe,
6. Typ wyliczeniowy, JavaDocs,
7. Tablice jednowymiarowe i wielowymiarowe, kolekcje,
8. Operacje wejścia – wyjścia i wyjątki,
9. Graficzny interfejs użytkownika - Swing,
10. Przechwytywanie zdarzeń, tworzenie plików wykonywalnych.
11. Projekt interdyscyplinarny – aplikacja mobilna dla systemu Android umożliwiająca przeglądanie bazy kolegów i koleżanek ze szkoły

5 Opis założonych osiągnięć ucznia

Uczeń po odbyciu kursu pozna teoretyczne podstawy języka Java, będzie umiał zaprojektować prostą aplikację zorientowaną obiektowo z interfejsem graficznym.

Człowiek - najlepsza inwestycja

5.1 Korelacja z treściami podstawy programowej

Domyślnym podmiotem celów kształcenia jest „Uczeń”.

Cele kształcenia	Treść kształcenia	Podstawa programowa
<ul style="list-style-type: none"> - zna podstawowe typy danych - umie przechowywać obiekty w tablicach, lub kolekcjach - potrafi wyjaśnić pojęcie listy - umie używać klas zdefiniowanych w frameworku Swing 	Lekcje: 2, 7, 8	Uczeń korzysta z wbudowanych typów danych
<ul style="list-style-type: none"> - poznaje ideę programowania obiektowego - umie budować klasy, klasy abstrakcyjne, interfejsy w Javie - umie tworzyć konstruktory i metody - poznaje klasy wewnętrzne - rozumie specyfikę typu wyliczeniowego - tworzy własne wyjątki 	Lekcje: 3, 4, 5, 6, 8	Uczeń tworzy własne typy danych
<ul style="list-style-type: none"> - potrafi bezpiecznie przechowywać dane w zmiennych - poznaje ideę programowania obiektowego - umie efektywnie tworzyć aplikacje - umie zapewnić bezpieczeństwo integralności aplikacji poprzez stosowanie dziedziczenia - rozumie pojęcie polimorfizmu 	Lekcje: 2, 3, 4, 5	Uczeń przestrzega zasad programowania
<ul style="list-style-type: none"> - poznaje pętle i instrukcje sterujące - umie zarządzać przebiegiem wykonania programu 	Lekcje: 2, 7, 8	Uczeń stosuje instrukcje, funkcje, procedury, obiekty, metody wybranych języków programowania

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Cele kształcenia	Treść kształcenia	Podstawa programowa
<ul style="list-style-type: none"> - umie przechowywać obiekty w tablicach, lub kolekcjach - potrafi wyjaśnić pojęcie listy - umie wykorzystywać widżety z biblioteki Swing 		
<ul style="list-style-type: none"> - umie budować klasy, klasy abstrakcyjne, interfejsy w Javie - umie tworzyć konstruktory i metody - poznaje klasy wewnętrzne 	Lekcje: 3, 4, 5	Uczeń tworzy własne funkcje, procedury, obiekty, metody wybranych języków programowania
<ul style="list-style-type: none"> - poznaje specyfikę języka Java, działanie, wady i zalety - umie poprawnie skonfigurować środowisko pracy 	Lekcje: 1	Uczeń wykorzystuje środowisko programistyczne: edytor, kompilator, debugger
<ul style="list-style-type: none"> - uruchamia i buduje programy - umie budować pliki wykonywalne jar 	Lekcje: 1, 10	Uczeń kompiluje i uruchamia kody źródłowe
- poznaje środowiska, w których Java jest wykorzystywana	Lekcje: 1 oraz moduł dotyczący języka PHP	Uczeń wykorzystuje języki programowania do tworzenia aplikacji internetowych realizujących zadania po stronie serwera
- poznaje możliwości zastosowania Javy w wykonywaniu programu po stronie klienta	Lekcje: 1 oraz moduł dotyczący języka JavaScript	Uczeń stosuje skrypty wykonywane po stronie klienta przy tworzeniu aplikacji internetowych
<ul style="list-style-type: none"> - zna możliwości graficznego frameworku Swing - umie obsługiwać zdarzenia w frameworku Swing 	Lekcje: 9, 10	Uczeń wykorzystuje frameworki do tworzenia własnych aplikacji

Człowiek - najlepsza inwestycja

Cele kształcenia	Treść kształcenia	Podstawa programowa
<ul style="list-style-type: none"> - umie wymieniać dane z zasobami z i poza programu (sieć, pliki lokalne). - rozumie, kiedy używać plikowej bazy danych 	Lekcje: 8, 9 oraz moduł dotyczący SQL	Uczeń pobiera dane aplikacji i przechowuje je w bazie danych
<ul style="list-style-type: none"> - umie edytować kod źródłowy aplikacji 	Lekcje: 2, 3, 4, 5, 6, 7, 8, 9, 10	Uczeń testuje tworzoną aplikację i modyfikuje jej kod źródłowy
<ul style="list-style-type: none"> - umie dokumentować kod komentarzami - poznaje metodę dokumentowania kodu za pomocą JavaDocs 	Lekcje: 1, 6	Uczeń dokumentuje tworzoną aplikację
<ul style="list-style-type: none"> - poznaje sposób integracji Javy na stronach internetowych 	Lekcje: 1	Uczeń zamieszcza opracowane aplikacje w Internecie
<ul style="list-style-type: none"> - wie jak zabezpieczyć integralności aplikacji poprzez właściwe użycie typów widoczności - umie zabezpieczać aplikację przed błędami 	Lekcje: 4, 8	Uczeń zabezpiecza dostęp do tworzonych aplikacji

6 Sposoby osiągania celów

Po odbyciu każdej lekcji uczeń powinien samodzielnie wykonać proste ćwiczenia w celu dokładnego zrozumienia omawianych mechanizmów. Programowanie w języku Java bez odpowiedniego środowiska jest bardzo trudne, dlatego jednym z celów jest jego poznanie. Umożliwi to wykonywanie ćwiczeń w czasie efektywnym.

6.1 Lista czynności w zależności od roli

Czynności nauczyciela	Czynności ucznia
Prezentowanie przygotowanego materiału.	Sporządzanie notatek z prezentowanego materiału

Człowiek - najlepsza inwestycja

Zadawanie pytań kontrolnych odnośnie już pokazanego materiału	Udzielanie odpowiedzi na pytania nauczyciela
Zadawanie pytań odnośnie materiału, który dopiero będzie pokazany, jeśli możliwe jest uzyskanie odpowiedzi w drodze dedukcji	
Objaśnianie ćwiczeń lekcyjnym, pilnowanie czasu ich wykonania przez uczniów, udzielanie wskazówek	Rozwiązywanie zadań lekcyjnych
Objaśnienie zadań na koniec lekcji, udzielanie wskazówek podczas rozwiązywania, zlecenie dokończenia zadań, jako pracę domową	Odrabianie zadań domowych
Sprawdzanie ćwiczeń lekcyjnych i zadań domowych: należy zwrócić uwagę na poprawne formatowanie kodu (wcięcia) poprawne użycie konstrukcji języka dokumentowanie kodu komentarzami otrzymywanie poprawnych rezultatów	
Zapisywanie uwag odnośnie materiałów szkoleniowych i sugestie ewentualnych korekt.	

7 Propozycje kryteriów oceny i metod sprawdzania osiągnięć ucznia

Ocena końcowa powinna uwzględniać wiele aspektów aktywności ucznia. Podlegać ocenie powinny:

- ćwiczenia wykonywane podczas lekcji
- odpowiedzi na pytania nauczyciela
- zadania domowe
- aktywność podczas lekcji
- ćwiczenia sprawdzające

W celu umożliwienia wiarygodnej oceny ucznia materiał szkoleniowy zawiera wiele przykładowych

Człowiek - najlepsza inwestycja

ćwiczeń lekcyjnych, pytań nauczyciela do uczniów, ćwiczeń na zakończenie każdej lekcji, których dokończenie może być zadaniem domowym oraz propozycja końcowego testu sprawdzającego wiedzę teoretyczną.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



7.1 Wymagania na poszczególne oceny szkolne

Określone wymagania uwzględniają jedynie znajomość ucznia z zakresu programowania w języku Java. Należy je traktować, jako składową oceny końcowej, uwzględniającej inne kryteria.

Domyślny podmiot to uczniów:

Zagadnienie	2	3	4	5	6
Prymitywne typy danych	umie rozróżnić rodzaje: łańcuch znaków (tekst), liczba całkowita, liczby zmiennoprzecinkowe, logiczna	umie wymienić i zadeklarować większość (może zapomnieć do 3 typów numerycznych)	umie wymienić i zadeklarować zmienne wszystkich typów prymitywnych	rozumie kiedy i jak używać typów zmiennoprzecinkowych	zna zakresy dla każdego typu prymitywnego
Przechowywanie obiektów w tablicach, lub kolekcjach	umie wyjaśnić różnicę między tablicą a kolekcją	umie poprawnie zadeklarować tablicę	umie przepisać elementy z tablicy do listy i odwrotnie	rozumie tablice wielowymiarowe, umie poprawnie je zadeklarować	uczni umie posortować tablicę lub kolekcję

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Zagadnienie	2	3	4	5	6
Framework Swing	wie, co to jest i do czego służy	umie czytając przykładowy kod źródłowy wytłumaczyć każdą linię oraz opisać, co zostanie wyświetlone wie, że można obsłużyć zdarzenia myszki i klawiatury	umie sam napisać prostą aplikację okienkową. W szczególności wie jak poprawnie zaimplementować wyłączenie i inicjalizację. umie obsługiwać zdarzenia myszki i klawiatury wie, co to jest manager layoutu	umie użyć managera layoutu i uzasadnić wybór takiego a nie innego w zależności od zaistniałej sytuacji	umie napisać własny manager layoutu

Człowiek - najlepsza inwestycja

Zagadnienie	2	3	4	5	6
Paradygmat programowania obiektowego	umie wytłumaczyć, co to jest obiekt, podać prosty przykład modelu przedmiotu z świata rzeczywistego np. samochód z kołami, z felgami i radiem	umie rozróżnić pojęcie klasy od obiektu zna składowe klasy (metody i pola) umie zaimplementować prostą klasę w osobnym pliku wie co to jest dziedziczenie i z ilu klas można naraz dziedziczyć	wie, co to są typy widoczności umie zaimplementować klasę zawierającą, jako pole inną klasę umie stworzyć klasę, która dziedziczy z innej klasy umie wyjaśnić, co to jest i do czego służy słowo „this”	umie wyjaśnić widoczność pól i metod w aspekcie dziedziczenia umie wywoływać metody z klasy rodzica (super)	umie tworzyć konstruktory umie wywoływać konstruktor rodzica umie wywoływać z jednego konstruktora, drugi konstruktor tej samej klasy
Pojęcie interfejs, klasa abstrakcyjna, klasa anonimowa	umie wytłumaczyć każdy z terminów	wie jak stworzyć interfejs i klasę abstrakcyjną	umie zaimplementować 2 i więcej interfejsów w jednej nowej klasie umie dziedziczyć po klasie abstrakcyjnej	umie stworzyć obiekt klasy anonimowej z interfejsu oraz klasy abstrakcyjnej	wie, że klasa abstrakcyjna może dziedziczyć z innej klasy (również abstrakcyjnej) wie że interfejsy mogą dziedziczyć po innych interfejsach

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Zagadnienie	2	3	4	5	6
Klasy wewnętrzne	umie powiedzieć czy jest możliwe zdefiniowanie klasy wewnątrz innej klasy	umie zdefiniować klasę wewnątrz innej klasy	wie, jaki typ widoczności może posiadać klasa wewnętrzna	umie stworzyć obiekt klasy wewnętrznej w dowolnej innej części programu	umie odwołać się do pola klasy zewnętrznej, które ma taką samą nazwę jak pole w klasie wewnętrznej
Typ wyliczeniowy	wie że istnieje coś takiego i kiedy powinno się go użyć	umie zaimplementować przykładowy typ wyliczeniowy			umie stworzyć pole wewnątrz typu wyliczeniowego rozumie, iż jest to specyficzny typ klasy, dla którego nie można stworzyć obiektu za pomocą słowa new

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Zagadnienie	2	3	4	5	6
Wyjątki	wie, po co są wyjątki wie, że można je łapać	umie rzucić wyjątek i go złapać	umie stworzyć własny wyjątek	umie stworzyć własny wyjątek zawierający dodatkowe pole i łapiąc ten wyjątek użyć tej dodatkowej informacji	potrafi rozróżnić RuntimeException od innych podklas Exception
Polimorfizm		wie, co to jest i potrafi przytoczyć przykłady z życia codziennego	umie użyć polimorfizmu w programie potrafi sprawdzić czy dany obiekt jest typu bardziej szczegółowego	wie jak działa nadpisywanie metod i która wersja metody zostanie wywołana w zależności o typu referencji do obiektu	
Pętle i instrukcje warunkowe	potrafi wyjaśnić, po co są pętle. potrafi podać przykład instrukcji warunkowej rozpoznaje słowa kluczowe	umie napisać poprawną pętlę for oraz while umie napisać poprawny warunek if/else	umie napisać poprawne zagnieżdżone instrukcje if/else umie napisać poprawną pętlę do-while oraz for w wersji foreach (bez trzech członów)	zna i umie użyć trójargumentowego operatora warunkowego	Potrafi zamienić prostą pętlę na rekurencyjne wywołanie metody

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Zagadnienie	2	3	4	5	6
Specyfika języka Java	<p>wie, że Java to język programowania</p> <p>wie, że uruchamia się niezależnie od platformy, gdyż wykonuje się w maszynie wirtualnej</p> <p>potrafi podać kilka przykładów środowisk, w jakich można spotkać aplikacje napisane w Javie</p>	<p>wie, że Java kompiluje się do bytecodu</p> <p>wie jak uruchomić program w Javie z linii komend</p> <p>rozdzieli pojęcia JRE oraz SDK</p>	<p>wie jak skompilować program w Javie</p> <p>zna rozszerzenia plików kodu źródłowego, jak i plików po kompilacji</p> <p>wie jak uruchomić program z pliku JAR w javie</p> <p>wie, co to są pakiety i jak importować klasy</p>	<p>wie, że każda klasa czy interfejs po kompilacji łączy się w osobnym pliku</p> <p>umie stworzyć plik JAR w elipsie</p>	<p>umie stworzyć plik JAR w linii komend</p> <p>umie uruchomić aplikację składającą się z kilku osobnych plików JAR</p>
Obsługa wejścia/wyjścia	<p>wie, że Java potrafi czytać i zapisywać pliki na dysku lokalnym</p> <p>wie, że Java umie komunikować się przez sieć</p>	<p>umie napisać program czytający lub zapisujący plik</p>	<p>Poprawnie obsługuje możliwe wyjątki I/O</p>	<p>umie napisać program łączy się z serwerem</p>	<p>umie napisać program czytający plik binarny, wyszukujący zadany ciąg bitów, zamieniający ten ciąg na inny i zapisujący wynik w nowym pliku</p>

Człowiek - najlepsza inwestycja

Zagadnienie	2	3	4	5	6
Środowisko programistyczne	wie, że pliki źródłowe są zwykłymi plikami tekstowymi	umie stworzyć nowy projekt w Eclipse umie uruchomić aplikację w Eclipse	korzysta z znanych skrótów klawiaturowych: podpowiadanie składni, kopiuje, wkleja	korzysta z bardziej zaawansowanych funkcji Eclipse: refactoring/zmiana nazwy zmiennej/metody/klasy, automatyczne czyszczenie importów	umie używać debugera
Dokumentacja kodu	potrafi wyjaśnić, po co dokumentuje się kod potrafi podać przykład komentarza	rozróżnia trzy typy komentarzy: JavaDoc, blokowy i do końca linii używa komentarzy w swoim kodzie	potrafi napisać poprawny komentarz JavaDoc używa JavaDoc w swoim kodzie	potrafi użyć anotacji w JavaDoc	potrafi wygenerować dokumentację HTML z kodu aplikacji

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

7.2 Końcowy test teoretyczny

Ocena końcowa powinna zależeć od wyniku testu sprawdzającego wiadomości teoretyczne (20 pytań zamkniętych pojedynczego wyboru - 20 punktów).

Propozycje kryteriów oceny:

- 0-9 punktów - ocena 1,
- 10-12 punktów - ocena 2,
- 13-15 punktów - ocena 3,
- 16-18 punktów - ocena 4,
- 29-20 punktów - ocena 5, Test końcowy sprawdzający wiedzę

20 zadań na 30 minutowy test sprawdzający wiedzę. Pogrubioną czcionką zaznaczono prawidłowe odpowiedzi.

1. Java kompiluje się do:
 - a. Kodu maszynowego
 - b. Kodu bajtowego
 - c. Pliku skryptowego
2. Kompilator Javy znajduje się w pakiecie:
 - a. JDK
 - b. JRE
 - c. Eclipse
3. Aby przechować wartość 100 jakiego najmniejszego typu możemy użyć:
 - a. long
 - b. int
 - c. short
4. Która pętla wykona jeden krok przebiegu nawet w przypadku fałszywego warunku:
 - a. for
 - b. do-while
 - c. while
5. Słowo „new” tworzy:
 - a. nową klasę;
 - b. nowy obiekt
 - c. nowy typ
6. Czy kiedy w klasie występuje konstruktor „public MojaKlasa(int x) „ mogą stworzyć obiekt za pomocą instrukcji „new MojaKlasa()”?
 - a. tak
 - b. nie
 - c. tylko wtedy, gdy istnieje drugi konstruktor „public MojaKlasa()”

Człowiek - najlepsza inwestycja

7. Która ze struktur nie posiada implementacji metod:
- klasa
 - interfejs
 - klasa niewłaściwa
8. Jeżeli chcemy, aby dana metoda była widoczna tylko w bieżącym pakiecie, to jakiego typu widoczności użyjemy?
- private
 - public
 - nie podajemy typu, co oznacza użycie widoczności package (default)
9. Które zdanie jest nieprawdziwe:
- Każdy obiekt typu Integer jest również obiektem typu Object
 - Mogę używać obiektu typu Integer tak, jak każdego innego obiektu typu Object
 - Mogę stworzyć obiekt, który jest jednocześnie typu Integer i String.
10. Zaznacz prawidłowe stwierdzenie
- Mogę stworzyć instancję interfejsu, pod warunkiem, że zaimplementuję jego metody
 - Jedyną możliwością stworzenia instancji interfejsu jest użycie słowa „implements”
 - Mogę stworzyć instancję interfejsu tak, jak zwykłej klasy.
11. Pisanie JavaDoc rozpoczynamy od znaku:
- /**
 - //
 - @
12. Podpowiadanie składni w środowisku Eclipse wywołujemy skrótem:
- „Ctrl” + „Shift” + „O”
 - „Ctrl” + „Spacja”
 - „F5”
13. Jeżeli chcemy udostępnić innym programistom wybór jednego z 5 słów, powinniśmy:
- Skorzystać z typu wyliczeniowego - enum
 - Umieścić słowa w tablicy
 - Poprosić o liczbę z przedziału od 1 do 5
14. Jeżeli chcemy wstawiać liczby w różne miejsca zbioru, powinniśmy skorzystać z:
- tablicy
 - typu wyliczeniowego
 - listy
15. Odczytując plik używamy obiektu:
- System.in
 - OutputStream
 - InputStream
16. Aby stworzyć własny wyjątek należy dziedziczyć po klasie:
- Exception
 - Error

Człowiek - najlepsza inwestycja

- c. StackTrace
- 17. Aby umożliwić użytkownikowi wybór kilku opcji w interfejsie graficznym, skorzystamy z:
 - a. JComboBox
 - b. JCheckBox
 - c. JRadioButton
- 18. Aby stworzyć obiekt reagujący na zdarzenia myszy, należy zaimplementować interfejs:
 - a. LayoutManager
 - b. MouseEvent
 - c. MouseListener
- 19. Zarządca rozkładu w interfejsie graficznym to w przypadku Swinga:
 - a. Layout
 - b. Listener
 - c. JPanel
- 20. Plik wykonywalny JAR to:
 - a. Archiwum ZIP
 - b. Plik z pojedynczym kodem bajtowym
 - c. Kod maszynowy

8 Lekcje


8.1 Lekcja 1 - Wstęp do Javy i środowiska pracy

8.1.1 Cel lekcji

Celem lekcji jest poznanie, czym jest język programowania Java, w jaki sposób działa i jakie są jego zalety. Wy tłumaczone jest jak poprawnie skonfigurować środowisko pracy i co jest potrzebne, aby móc uruchamiać zbudowane programy.

Człowiek - najlepsza inwestycja

8.1.2 Treść - slajdy z opisem

<p>Slajd 1</p>		
--------------------	--	--

Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd
2

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Główne koncepcje:

- Obiektość
- Niezależność od architektury
- Sieciowość i obsługa programowania rozproszonego
- Niezawodność i bezpieczeństwo

 **DAILY**
G R O U P
 Wstęp do języka i środowiska pracy

Java jest językiem kompilowanym do kodu bajtowego (bytecode). Oznacza to, że aplikacje napisane w Javie i skompilowane nie mogą zostać uruchomione bezpośrednio przez system operacyjny. Niezbędna jest maszyna wirtualna javy (JVM), która interpretuje i wykonuje kod bajtowy.

Java została stworzona przez grupę roboczą pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems, obecnym właścicielem tej technologii jest Oracle Corporation.

Jego podstawowe koncepcje zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz z języka C++ (duża część składni i słów kluczowych).

Główne koncepcje

Autorzy języka Java określili kilkanaście kluczowych koncepcji swojego języka. Najważniejsze z nich to:

Obiektość

W przeciwieństwie do proceduralno-obiektowego języka C++, Java jest silnie ukierunkowana na

Człowiek - najlepsza inwestycja

obiektość. O obiekcie można myśleć jako o samoistnej części programu, która może przyjmować określone stany i ma określone zachowania, które mogą zmieniać te stany bądź przysyłać dane do innych obiektów. Wyjątkiem od całkowitej obiektości są typy proste (inaczej: typy wbudowane, prymitywy).

Dziedziczenie

W Javie wszystkie obiekty są pochodną obiektu nadrzędnego (jego klasa nazywa się po prostu Object), z którego dziedziczą podstawowe zachowania i właściwości. Dzięki temu wszystkie mają wspólny podzbiór podstawowych możliwości, takich jak ich: identyfikacja, porównywanie, kopiowanie, niszczenie czy wsparcie dla programowania współbieżnego.

Niezależność od architektury

Tę właściwość Java ma dzięki temu, że kod źródłowy programów pisanych w Javie kompiluje się do kodu pośredniego (kodu bajtowego). Powstały kod jest niezależny od systemu operacyjnego i procesora, a wykonuje go tzw. wirtualna maszyna Javy, która (między innymi) tłumaczy kod uniwersalny na kod dostosowany do specyfiki konkretnego systemu operacyjnego i procesora. W tej chwili wirtualna maszyna Javy jest już dostępna dla większości systemów operacyjnych i procesorów. Jednak z uwagi na to, że kod pośredni jest interpretowany, taki program jest wolniejszy niż kompilowany do kodu maszynowego.

Sieciowość i obsługa programowania rozproszonego

Dzięki wykorzystaniu reguł obiektości, Java nie widzi różnicy między danymi płynącymi z pliku lokalnego a danymi z pliku dostępnego przez HTTP czy FTP.

Niezawodność i bezpieczeństwo

W zamierzeniu Java miała zastąpić C++ – obiektowego następcę języka C. Jej projektanci zaczęli od rozpoznania cech języka C++, które są przyczyną największej liczby błędów programistycznych, by stworzyć język prosty w użyciu, bezpieczny i niezawodny. O ile po siedmiu odsłonach Javy jej prostota jest dyskusyjna, o tyle język faktycznie robi dużo, by utrudnić programiście popełnienie błędu.

Człowiek - najlepsza inwestycja

Slajd
 3



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Możliwości zastosowania:

Typy aplikacji:

- desktop (wiersza poleceń i okienkowe)
- serwerowe (strony WWW)
- mobilne (telefony)
- wbudowane (centrale telefoniczne, telewizory, dekodery telewizyjne, pralki, piekarniki, karty SIM...)

Wirtualna maszyna Javy – jako uniwersalne środowisko uruchomieniowe







Wstęp do języka i środowiska pracy

Z uwagi na swoją popularność Java dorobiła się ogromnej ilości bibliotek i projektów. Dzięki takiej popularności można ją spotkać praktycznie w każdym środowisku.

Aplikacje – jako pełnoprawny język programowania Java umożliwia tworzenie standardowych aplikacji uruchamianych przez użytkownika. Mogą to być aplikacje linii komend lub też okienkowe. Środowisko okienkowe tworzymy korzystając z frameworka Swing lub zyskującej coraz większą popularność JavvyFX.

WWW – w dzisiejszych czasach Java głównie dostarcza zaplecza serwerowego dla stron WWW. Pozwala na tworzenie bezpiecznych i skalowalnych aplikacji działających po stronie serwera. Frontend często tworzony przy pomocy innych technologii komunikuje się z stroną serwerową aby uzyskać lub przekazać potrzebne dane. Obecnie mniej modnym i zarazem mniej dynamicznym rozwiązaniem jest generowanie po stronie serwera całych stron HTML. W tym przypadku możemy mówić o tym iż Java dostarcza również frontend, jednak interakcja z aplikacją powoduje zazwyczaj przeładowanie strony. Istnieją jeszcze technologie oparte na Javie, które pozwalają na częściowe renderowanie zawartości strony WWW w przeglądarce np.. Java Server Faces (JSF), jednak po stronie przeglądarki nie znajdziemy ani odrobiny Javy jako takiej. W JSF komponenty generują kod HTML (widok) oraz JavaScript odpowiedzialny za interakcję z użytkownikiem, komunikację z serwerem i uaktualnienie widoku.

Człowiek - najlepsza inwestycja

Aplet – jest to program napisany w Javie, przeglądarka korzystając z wtyczki maszyny wirtualnej uruchamia go w oknie przeglądarki. Obecnie technologia uznana za przestarzałą, jednak czasem jeszcze spotykana w specyficznych rozwiązaniach. Zazwyczaj używana gdy strona WWW potrzebuje dostępu do zasobów lokalnych np.. czytnika kart podpisu cyfrowego lub też operacji na plikach np.. policzenie hash'a z pliku wideo i automatyczne znalezienie pasujących napisów w serwisie.

Aplikacje mobilne – tutaj znów możemy wyróżnić dwa podstawowe warianty

Jako aplikacje Java ME (Java mobile edition) – jest to zubożona wersja Javy, dostosowana do słabych obliczeniowo i z małą ilością pamięci urządzeń, takich jak poprzednie generacje telefonów komórkowych, dekodery telewizyjne, karty SIM czy chipy w kartach bankowych. Nie zawiera wszystkich bibliotek, jednak sama składnia języka jest taka sama.

Jako aplikacje na ostatnio bardzo popularną platformę Android. Java jest językiem wybranym przez twórców platformy, jednak kompiluje się do niestandardowej formy bytecodu, interpretowanej przez maszynę wirtualną Dalvik.

JVM – sama maszyna wirtualna okazała się być dobrą, szybką i niezawodną platformą uruchomieniową. W związku z tym powstały inne języki programowania kompilujące się do kodu bajtowego. Oto kilka bardziej popularnych:

- a. Clojure, dialekt języka funkcyjnego Lisp
- b. Groovy, język skryptowo-programistyczny
- c. Scala, programowanie funkcyjno-obiektowe
- d. JRuby, implementacja Ruby
- e. Jython, implementacja Python'a

Człowiek - najlepsza inwestycja

Slajd
 4



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Przygotowanie środowiska pracy:

www.oracle.com

Java JRE (Java Runtime Environment)

Java JDK (Java Development Kit)

<http://www.eclipse.org/downloads/>

Eclipse Classic



Wstęp do języka i środowiska pracy

Aby móc uruchomić aplikację napisaną w Javie musimy mieć maszynę wirtualną. Znajduje się ona w pakiecie Java JRE (Java Runtime Environment). Można go łatwo wyszukać korzystając z wyszukiwarki Google lub pobrać bezpośrednio ze strony www.oracle.com.


Aby móc stworzyć aplikację napisaną w Javie, musimy mieć większy zestaw narzędzi o nazwie Java JDK (Java Development Kit). Znajduje się tam przede wszystkim kompilator. Cały pakiet dostępny jest również na stronie www.oracle.com.

Należy przede wszystkim pamiętać, że Java stworzona jest na licencji GNU (General Public License). Na slajdzie przedstawione są jedynie oficjalne narzędzia. Każdy może napisać własną maszynę wirtualną i istnieje wiele alternatywnych źródeł. Np. otwarta implementacja Javy - OpenJDK.


Bardzo ważnym elementem jest również środowisko pracy. Stanowczo odradzam korzystanie z prostych edytorów tekstowych. Jednym z najpopularniejszych środowisk jest Eclipse. Można pobrać również środowisko NetBeans lub IntelliJ IDEA. Wszystkie trzy oferują podobne możliwości i są powszechnie wykorzystywane na całym świecie. Zalecam pobranie pakietu Eclipse Classic, ponieważ właśnie w nim będę pokazywał przykłady.

Człowiek - najlepsza inwestycja

Slajd
5





B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






Wstęp do języka i środowiska pracy


W pierwszej kolejności instalujemy pakiet Java JRE

Człowiek - najlepsza inwestycja

Slajd
6

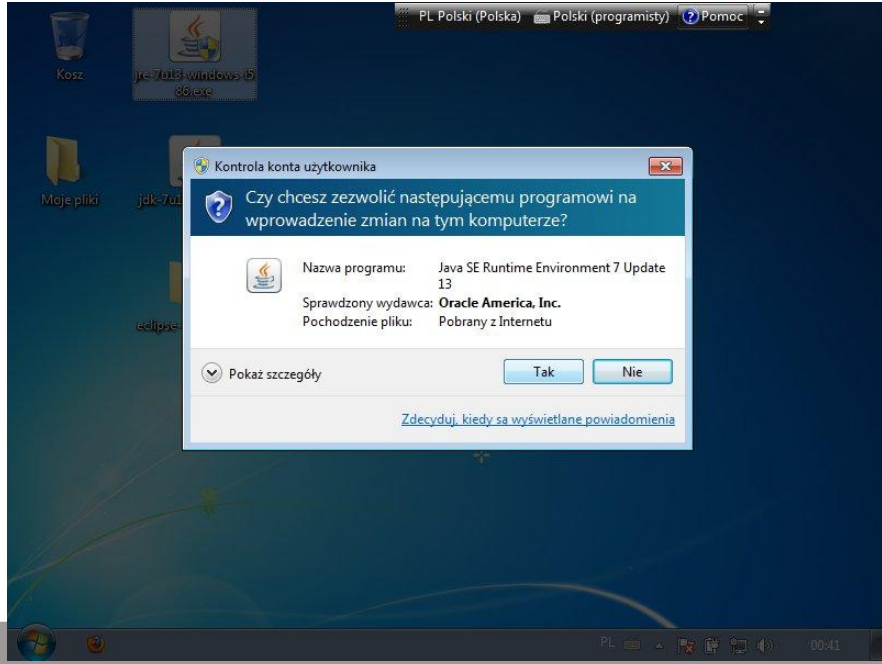



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących







DAILY
GROUP

Wstęp do języka i środowiska pracy

Slajd
7





B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





DAILY
GROUP

Wstęp do języka i środowiska pracy

Przechodzimy kolejne kroki instalacji.

Człowiek - najlepsza inwestycja

Slajd
8

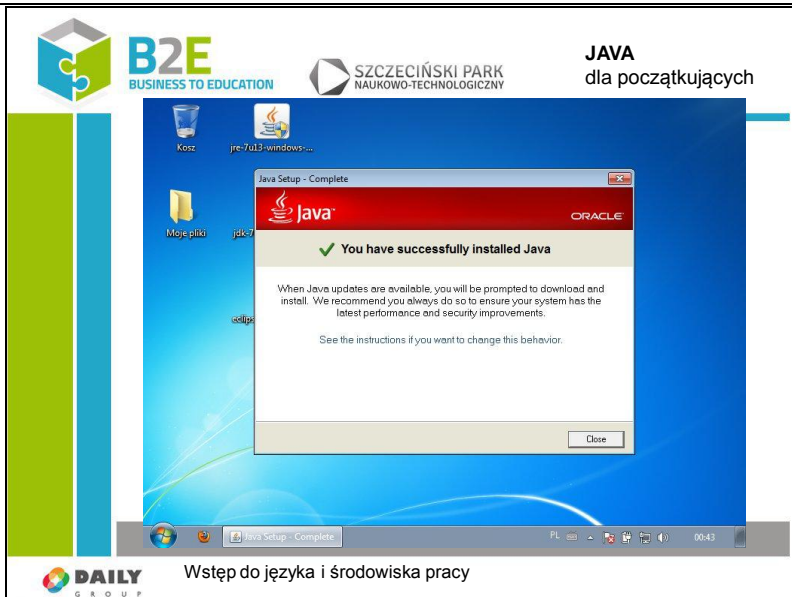
B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących



DAILY GROUP Wstęp do języka i środowiska pracy

Slajd
9

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących



DAILY GROUP Wstęp do języka i środowiska pracy

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

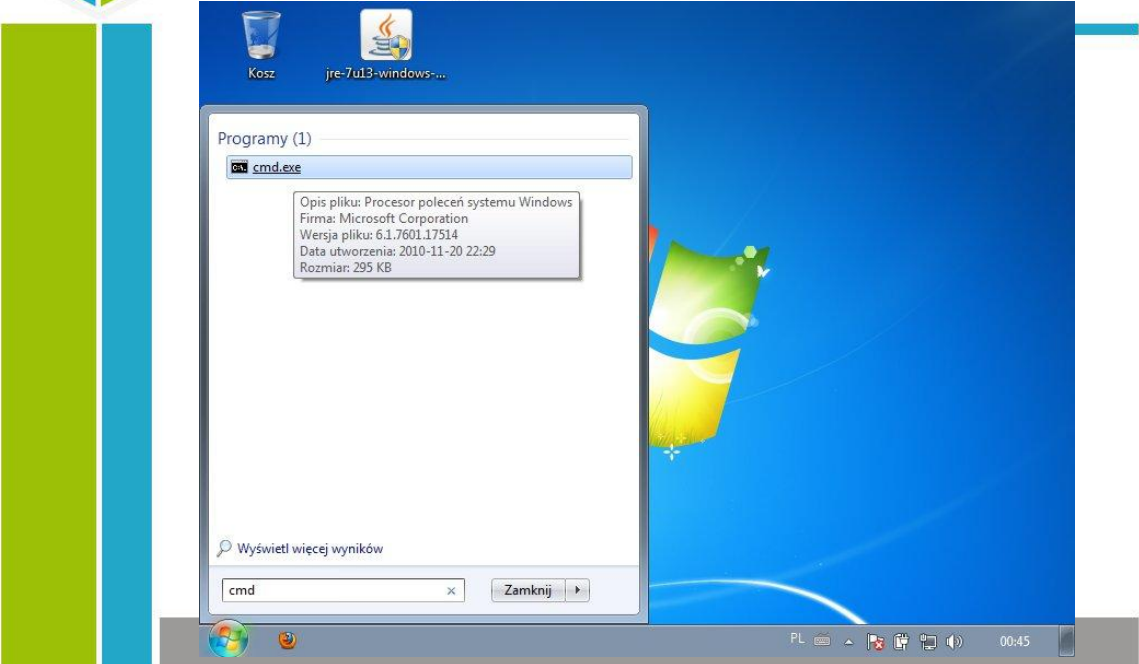
str. 29


Slajd
10

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Wstęp do języka i środowiska pracy

Po zakończeniu instalacji możemy korzystać z oprogramowania stworzonego w języku Java.

W dalszych slajdach pokazane będzie jak uruchomić aplikację. Dla zaprezentowania posiadam już gotowy plik wykonywalny. Proszę nie wykonywać tej części samodzielnie.

W tym celu należy uruchomić program cmd.exe.

Człowiek - najlepsza inwestycja

Slajd
11




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



```

C:\Windows\system32\cmd.exe
C:\tmp\szkolenie\lekcja01\przyklad1>dir *.class
Wolumin w stacji C nie ma etykiety.
Numer seryjny woluminu: 665F-1429

Katalog: C:\tmp\szkolenie\lekcja01\przyklad1
2013-02-18  23:28                419 MojaKlasa.class
               1 plik(ów)                419 bajtów
               0 katalog(ów)           8 644 915 200 bajtów wolnych

C:\tmp\szkolenie\lekcja01\przyklad1>

```



Wstęp do języka i środowiska pracy

Pliki wykonywalne dla maszyny wirtualnej Javy posiadają rozszerzenie class.

Człowiek - najlepsza inwestycja

Slajd
12



JAVA
dla początkujących



 **DAILY**
G R O U P

Wstęp do języka i środowiska pracy

Poleceniem java uruchamiamy maszynę podając jako parametr nazwę pliku bez rozszerzenia.

Człowiek - najlepsza inwestycja

Slajd
13





JAVA
dla początkujących




Wstęp do języka i środowiska pracy

W wyniku otrzymaliśmy napis „Działaj!”. Nie wyświetliła się jednak literka „ł”. Spowodowane jest to kodowaniem znaków w programie cmd.exe. Sama Java nie ma problemu z kodowaniem znaków, gdyż domyślnie korzysta z 16 bitów na symbol, więc bez żadnych ograniczeń można stosować znaki regionalne.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 33

Slajd
14





JAVA

dla początkujących



```

C:\Windows\system32\cmd.exe
C:\tmp\szkolenie\lekcja01\przyklad1>java MojaKlasa
Działa
C:\tmp\szkolenie\lekcja01\przyklad1>chcp
Aktywna strona kodowa: 852
C:\tmp\szkolenie\lekcja01\przyklad1>chcp 1250
Aktywna strona kodowa: 1250
C:\tmp\szkolenie\lekcja01\przyklad1>java MojaKlasa
Działa
C:\tmp\szkolenie\lekcja01\przyklad1>

```



Wstęp do języka i środowiska pracy

Problem kodowania polega na tym iż Java w systemach Windows domyślnie wypisuje używając kodowania CP-1250. Niestety konsola domyślnie jest skonfigurowana do używania kodowania CP-852. Zmieniając ustawienia konsoli można pozbyć się wcześniejszego błędu.

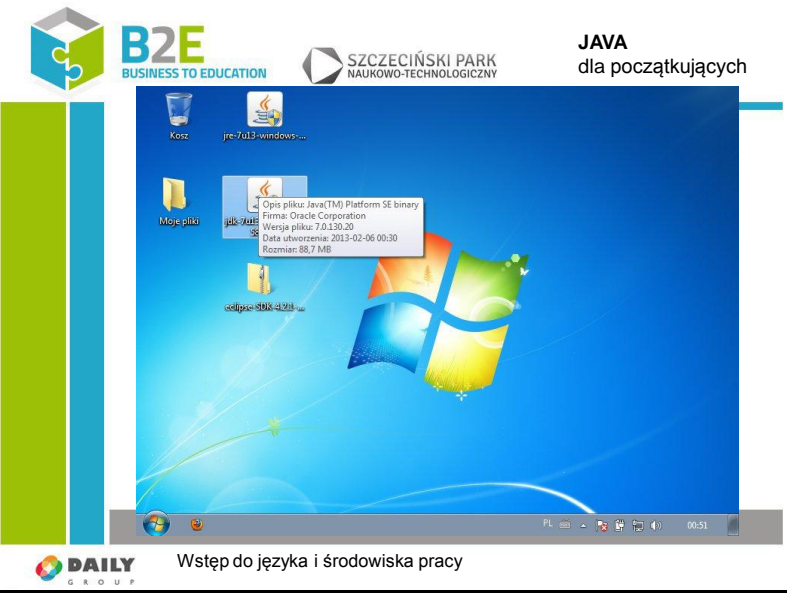
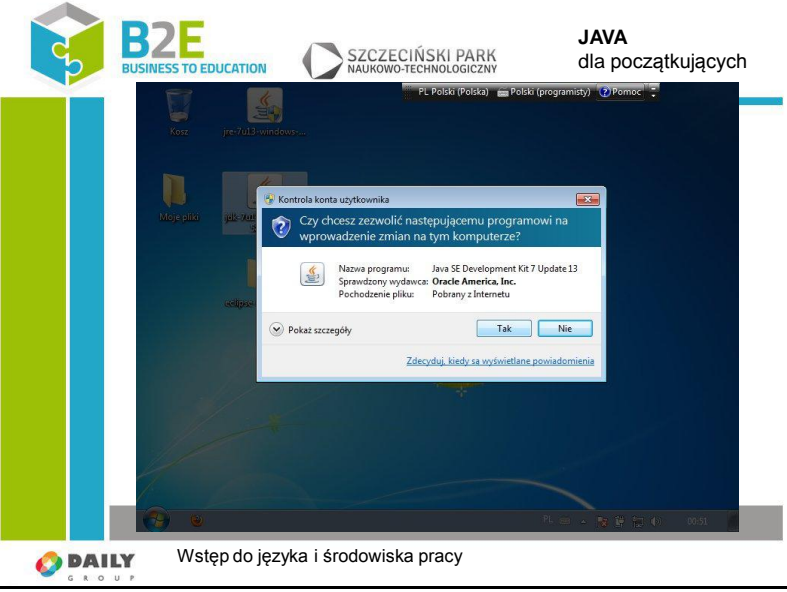
Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY




<p>Slajd 15</p>		<p>Aby móc zacząć programować w Javie, potrzebujemy kompilatora z pakietu Java JDK.</p>
<p>Slajd 16</p>		<p>Kolejno wykonujemy polecenia instalatora.</p>

Człowiek - najlepsza inwestycja

Slajd
17





B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących








DAILY
GROUP

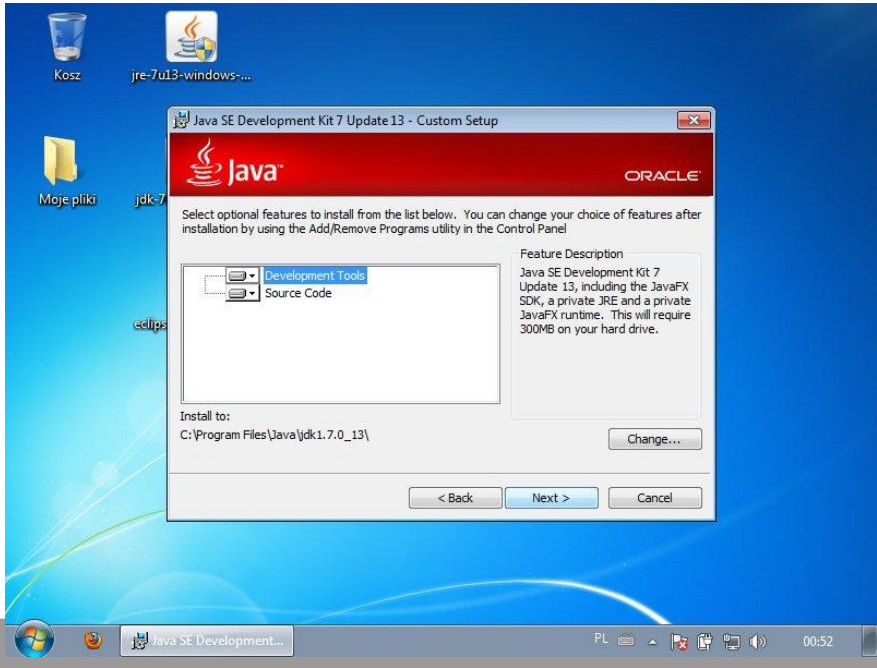
Wstęp do języka i środowiska pracy


Człowiek - najlepsza inwestycja

Slajd
18


JAVA
dla początkujących



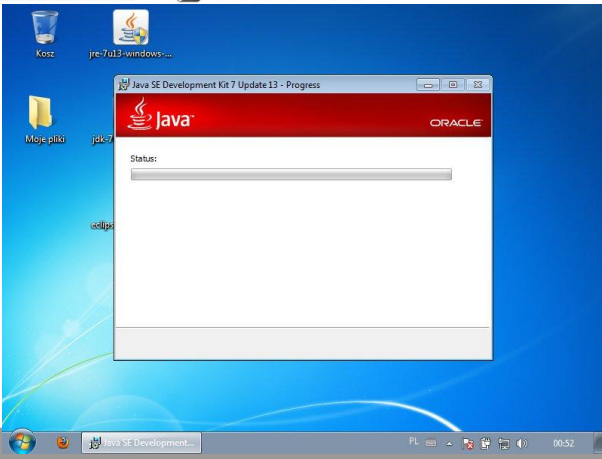

Wstęp do języka i środowiska pracy


Slajd
19





JAVA
dla początkujących




Wstęp do języka i środowiska pracy

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY




Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 37

Slajd
20





B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






DAILY
GROUP


Wstęp do języka i środowiska pracy

Człowiek - najlepsza inwestycja

Slajd
21

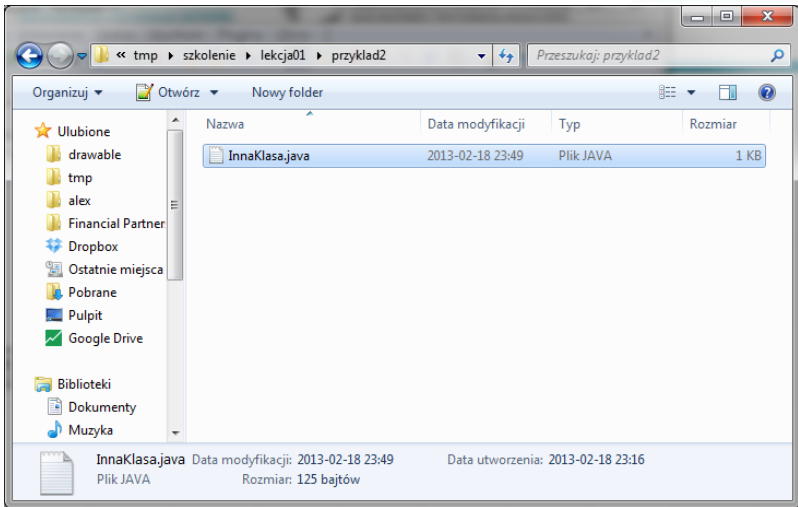



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Wstęp do języka i środowiska pracy


Po zakończonej instalacji Java JDK. Możemy napisać pierwszy program.

Możemy już wykonać pierwsze praktyczne ćwiczenie.

Stwórzmy plik o dowolnej nazwie, bez białych znaków i zaczynający się dużą literą, z rozszerzeniem *.java.

Człowiek - najlepsza inwestycja

Slajd
22

 **B2E**
BUSINESS TO EDUCATION
  **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



```

1 public class InnaKlasa {
2     public static void main(String[] args) {
3         System.out.println("Witaj, to ja InnaKlasa");
4     }
5 }
6
        
```

 **DAILY**
GROUP
 Wstęp do języka i środowiska pracy


Do pliku wstawiamy ten fragment kodu.

Uwaga!


Jeżeli użyta została inna nazwa pliku niż „InnaKlasa” to należy ją wpisać w pierwszym wersie (bez rozszerzenia) zamiast wyrażenia „InnaKlasa”.

Człowiek - najlepsza inwestycja

Slajd
23

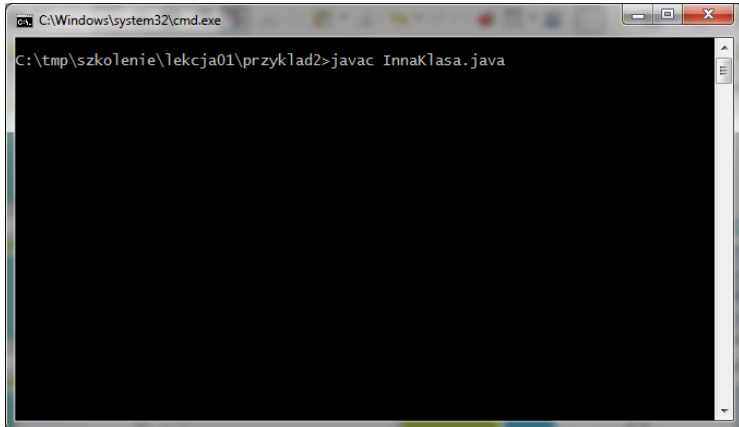




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY



B2E
BUSINESS TO EDUCATION

JAVA
dla początkujących





DAILY
GROUP

Wstęp do języka i środowiska pracy

Kompilujemy nasz kod źródłowy widocznym poleceniem.

Pierwsza część jest ścieżką do kompilatora Javy – javac.exe.

Argumentem przyjmowanym przez kompilator jest ścieżka do pliku z kodem źródłowym.

Człowiek - najlepsza inwestycja

Slajd
24




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY



B2E
BUSINESS TO EDUCATION

JAVA
dla początkujących



```

C:\Windows\system32\cmd.exe
C:\tmp\szkolenie\lekcja01\przyklad2>javac InnaKlasa.java
C:\tmp\szkolenie\lekcja01\przyklad2>
        
```




DAILY
GROUP

Wstęp do języka i środowiska pracy


Tak wygląda rezultat poprawnie wykonanego polecenia kompilacji (brak jakichkolwiek wiadomości i ponowne wyświetlenie znaku zachęty).

Człowiek - najlepsza inwestycja

Slajd
25

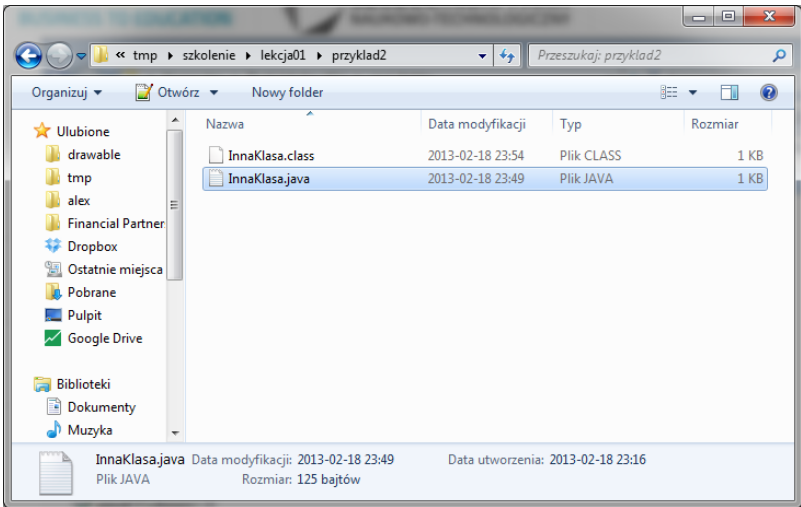



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Wstęp do języka i środowiska pracy

Jedyną widoczną zmianą jest pojawienie się pliku o takiej samej nazwie jaką wybraliśmy do pliku z rozszerzeniem „*.java”.

Nowy plik posiada rozszerzenie „*.class”. Zawiera on kod bajtowy przeznaczony do uruchamiania na każdej maszynie wirtualnej.

Człowiek - najlepsza inwestycja

Slajd
26



JAVA dla początkujących



Wstęp do języka i środowiska pracy

Możemy wyświetlić zawartość nowego pliku w edytorze tekstowym.

Jak widać nie jest to kod maszynowy. Wiele części można odczytać, np. „SourceFile” jest to plik źródłowy z którego powstał dany kod bajtowy, „java/lang/String” jest to adres obiektu, który przechowuje ciąg znaków.

Człowiek - najlepsza inwestycja



Slajd
27



JAVA dla początkujących

```
C:\Windows\system32\cmd.exe

C:\tmp\szkolenie\lekcja01\przyklad2>javac InnaKlasa.java
C:\tmp\szkolenie\lekcja01\przyklad2>java InnaKlasa
Witaj, to ja InnaKlasa
C:\tmp\szkolenie\lekcja01\przyklad2>
```



Wstęp do języka i środowiska pracy

Omawiany plik możemy uruchomić poleceniem pokazanym w poprzednim przykładzie.

Człowiek - najlepsza inwestycja




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY




Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd
28

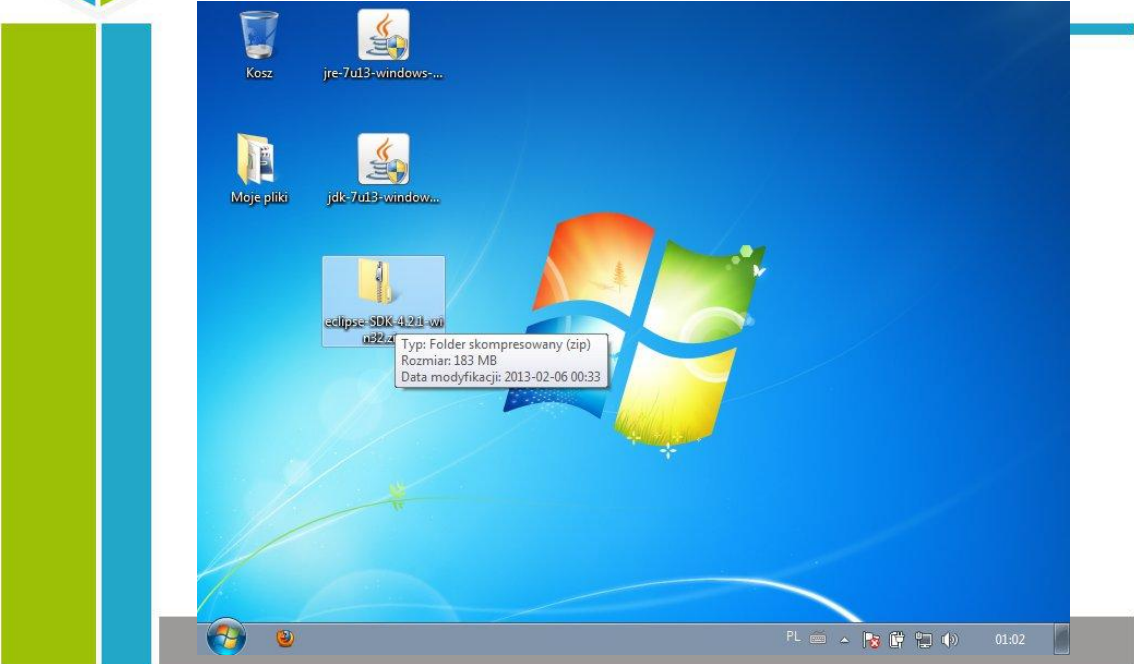



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






Wstęp do języka i środowiska pracy


Pisanie kodu w prostym edytorze tekstowym jest trudne. Będziemy korzystać ze środowiska Eclipse w celu usprawnienia pracy.

Człowiek - najlepsza inwestycja

Slajd
29

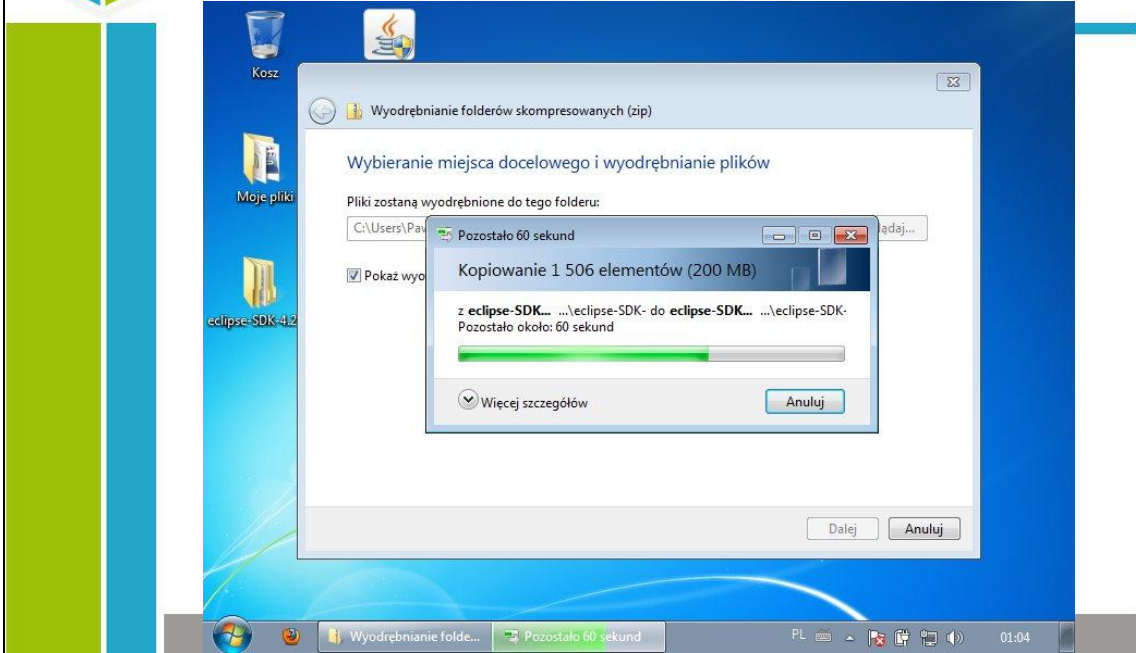



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






Wstęp do języka i środowiska pracy


Wypakowujemy pobrane archiwum.

Człowiek - najlepsza inwestycja

Slajd
30

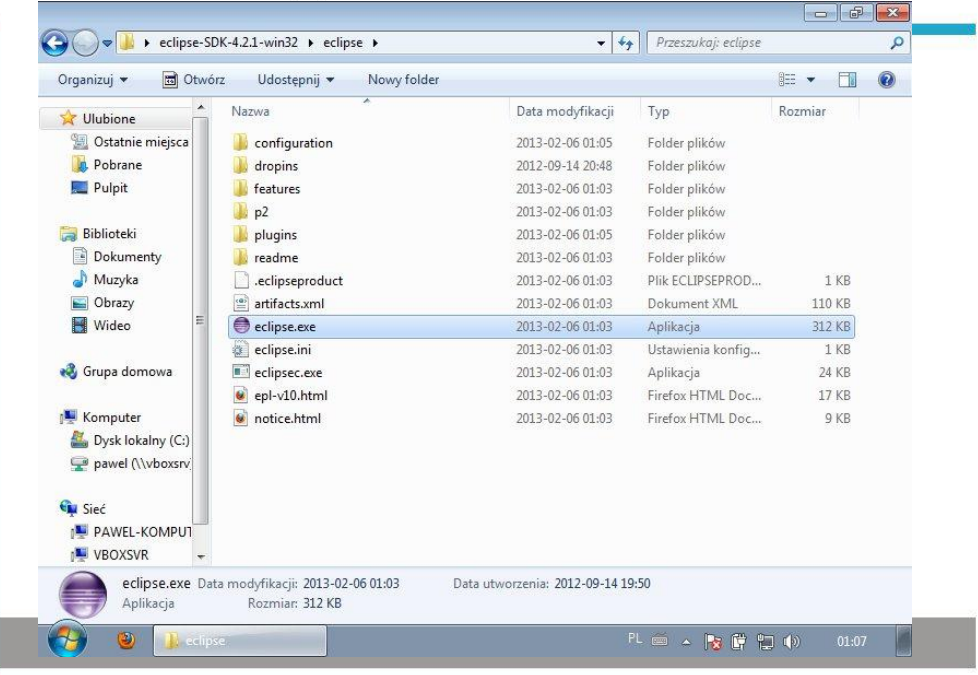



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Wstęp do języka i środowiska pracy

Środowisko uruchamiamy za pomocą pliku „eclipse.exe”.

Człowiek - najlepsza inwestycja




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Slajd
31

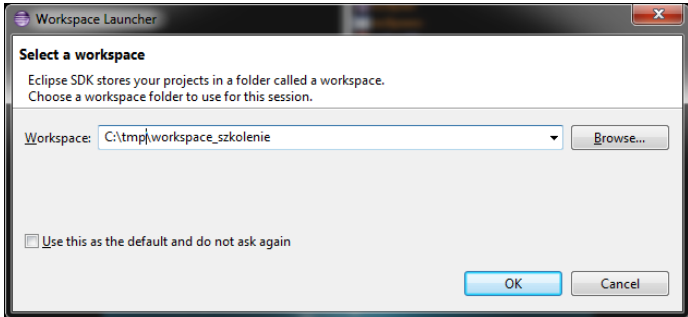



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących







Wstęp do języka i środowiska pracy

Przy pierwszym uruchomieniu trzeba wybrać folder pełniący funkcję „workspace”. Jest to lokalizacja w której znajdować się będą wszystkie nasze projekty

Slajd
32

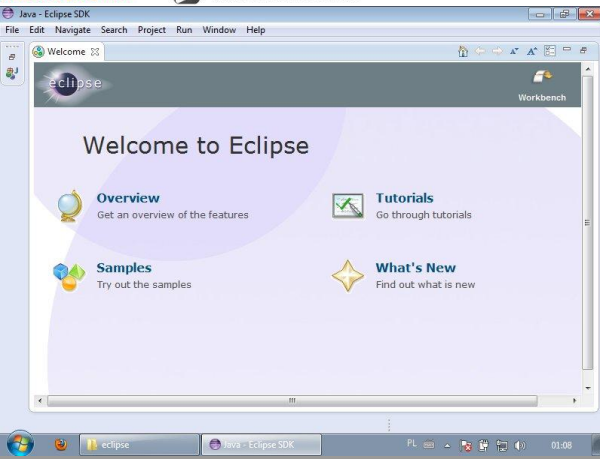



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

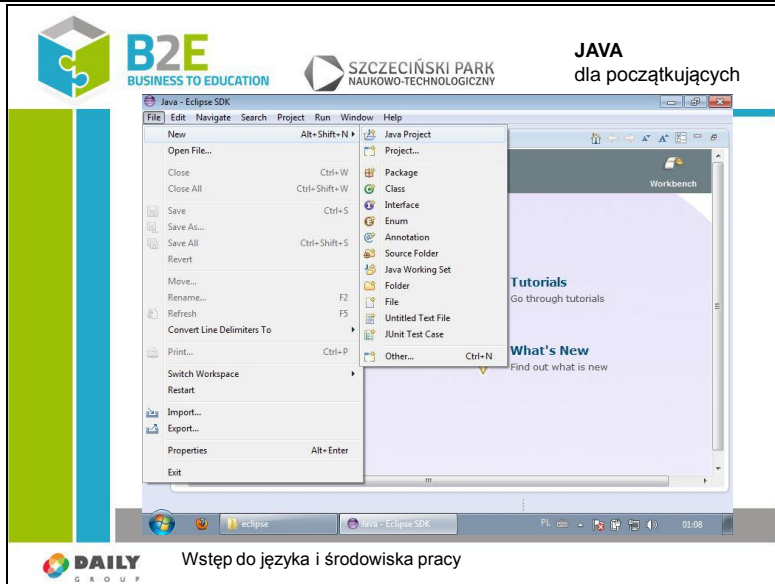




Wstęp do języka i środowiska pracy

Człowiek - najlepsza inwestycja

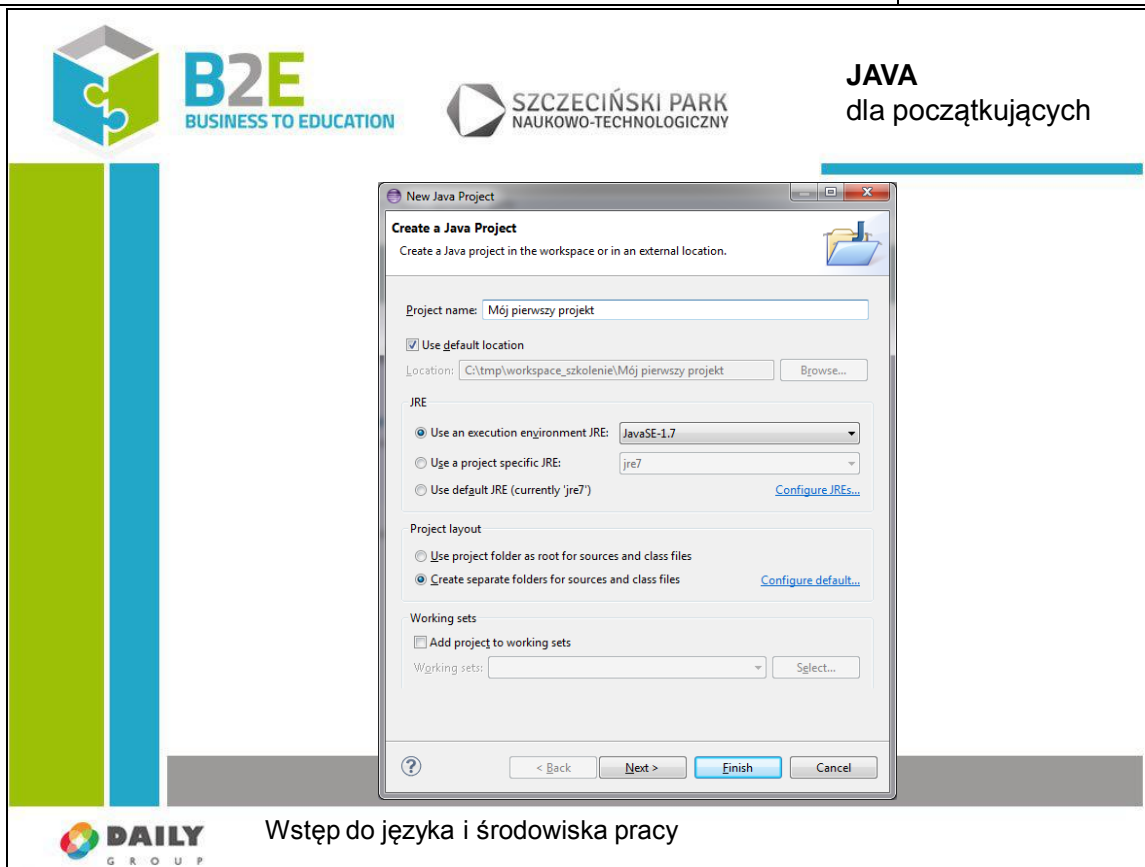
Slajd
33



Zanim przystąpimy do pisania kodu, musimy stworzyć projekt.

W tym celu wybieramy:
File/New/Java Project

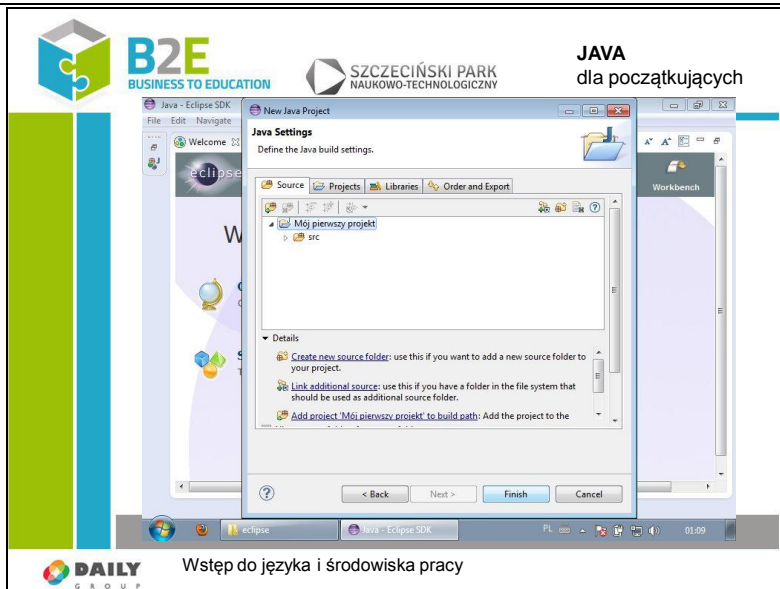
Slajd
34



Nazwa projektu (Project name) może być dowolna.
Pozostałe parametry najlepiej pozostawić z ustawieniami domyślnymi.
Klikamy „Next”.

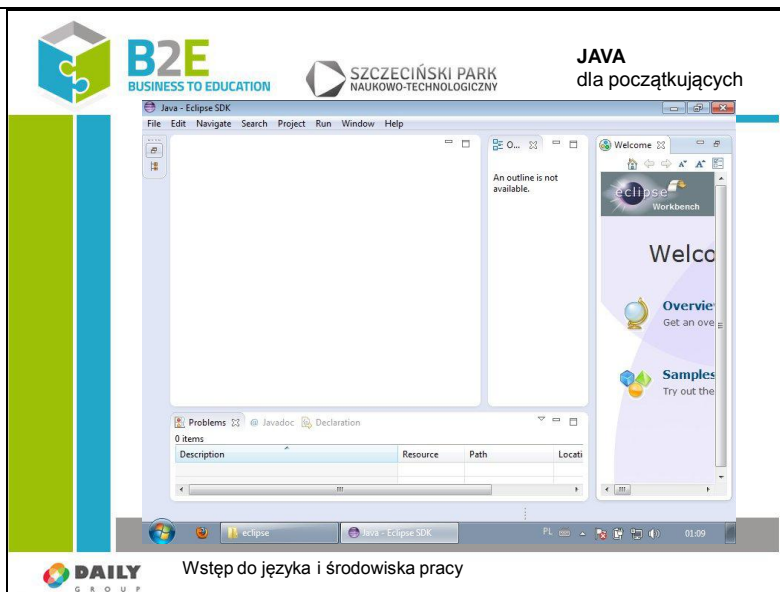
Człowiek - najlepsza inwestycja

Slajd
35



Klikamy „Finish”.

Slajd
36




Aby zobaczyć zawartość projektu klikamy na ikonę znajdującą się w lewym, górnym rogu (poniżej pozycji w menu o nazwie „File”).


Powinien wyświetlić się nam „Package Explorer”

Człowiek - najlepsza inwestycja

Slajd
37

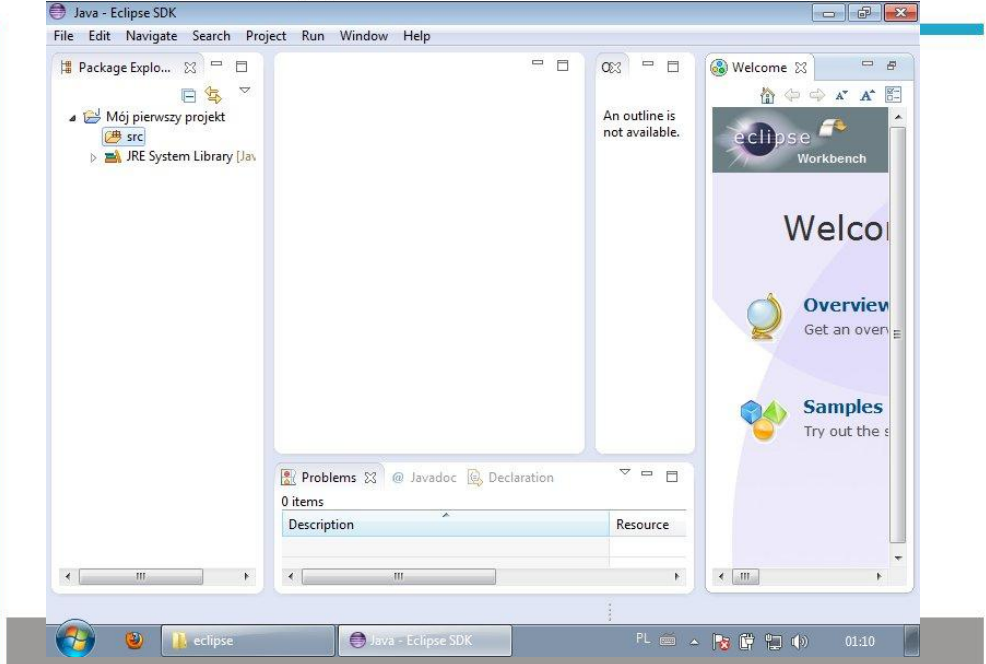



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





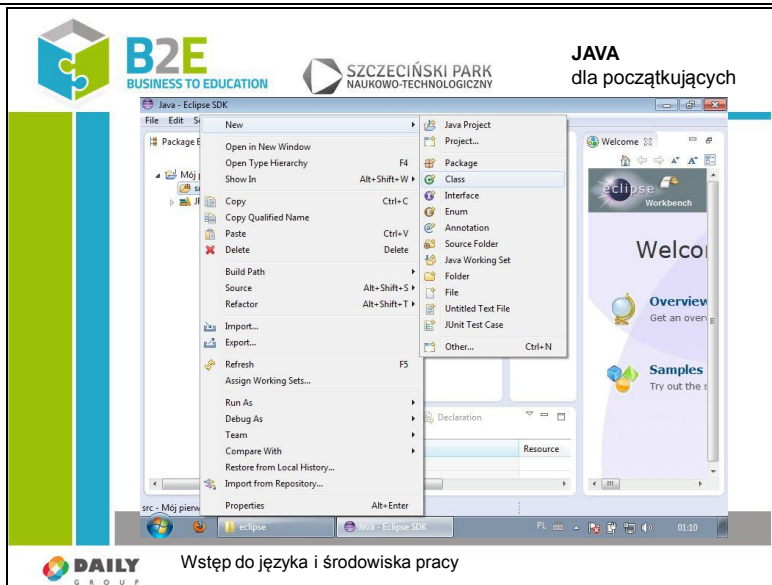
Wstęp do języka i środowiska pracy

Pliki kodu źródłowego przechowywane są w folderze „src”.

Nowy projekt nie posiada żadnych plików. W celu stworzenia nowego należy kliknąć na folderze „src” prawym przyciskiem myszy.

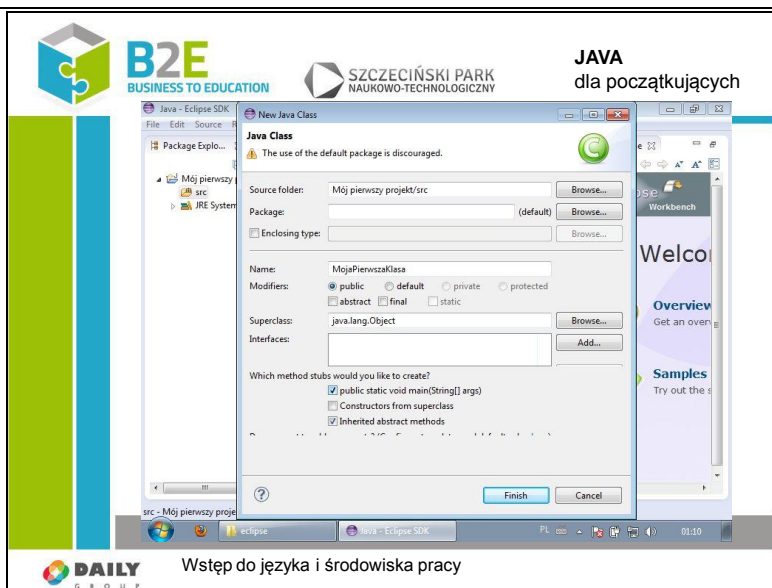
Człowiek - najlepsza inwestycja

Slajd
38



Następnie wybieramy:
New/Class.

Slajd
39



Podajmy nazwę klasy. Nie używamy spacji i zaczynamy dużą literą.


Zaznaczamy również opcję:

„public static void main(String [] args)”.


Całość zatwierdzamy przyciskiem „Finish”.

Człowiek - najlepsza inwestycja

Slajd
40

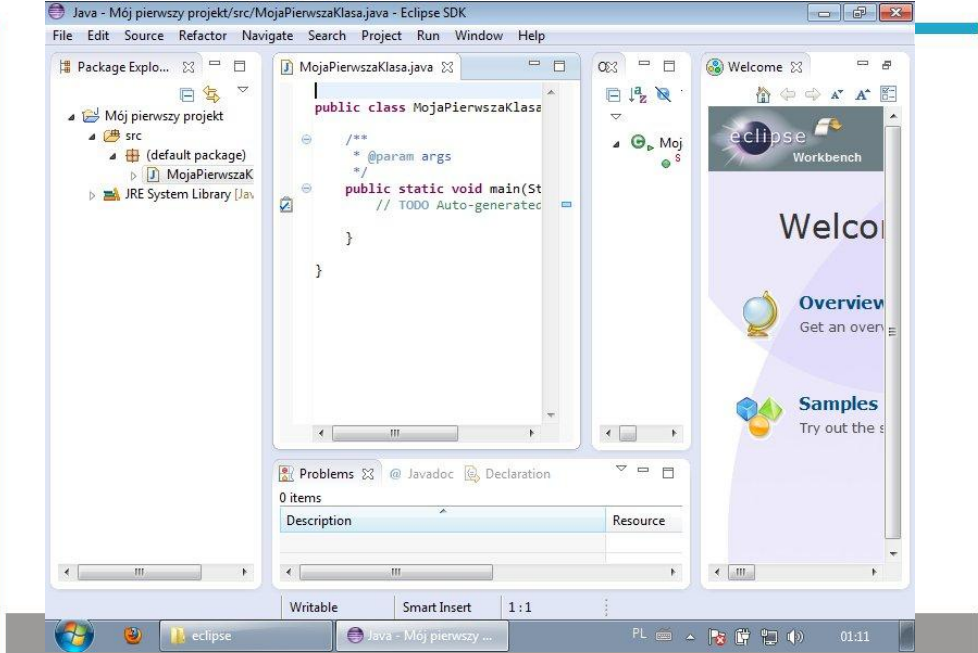



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących







Wstęp do języka i środowiska pracy

Na środku ekranu wyświetliła się zawartość pliku. Znajduje się on w folderze „src” w (default package)”.

Slajd
41

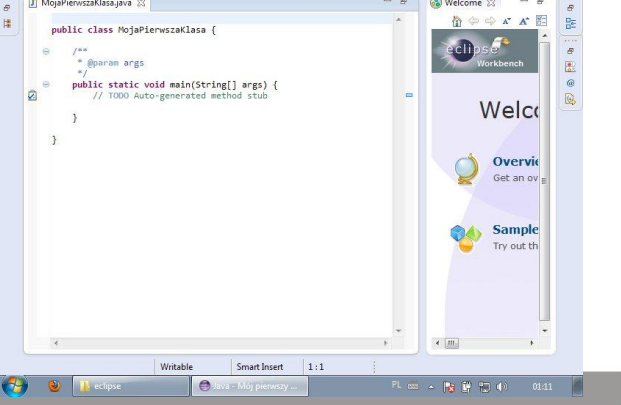



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






Wstęp do języka i środowiska pracy


Jeżeli chcemy wyświetlić plik na całej szerokości ekranu, klikamy podwójnie na zakładce z zawartością pliku.

Człowiek - najlepsza inwestycja

Slajd
42

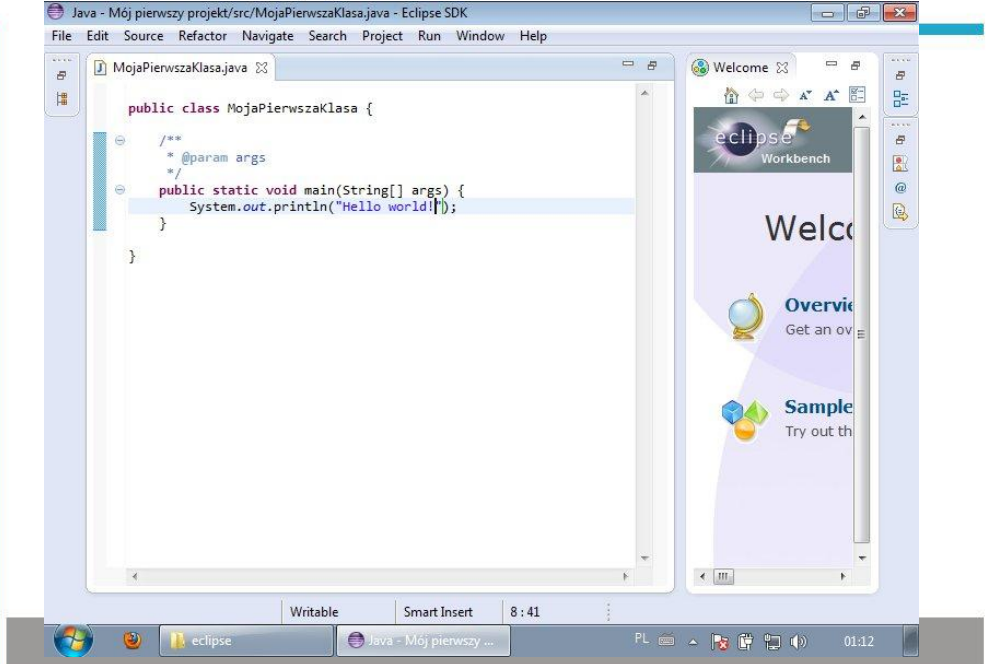



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Wstęp do języka i środowiska pracy

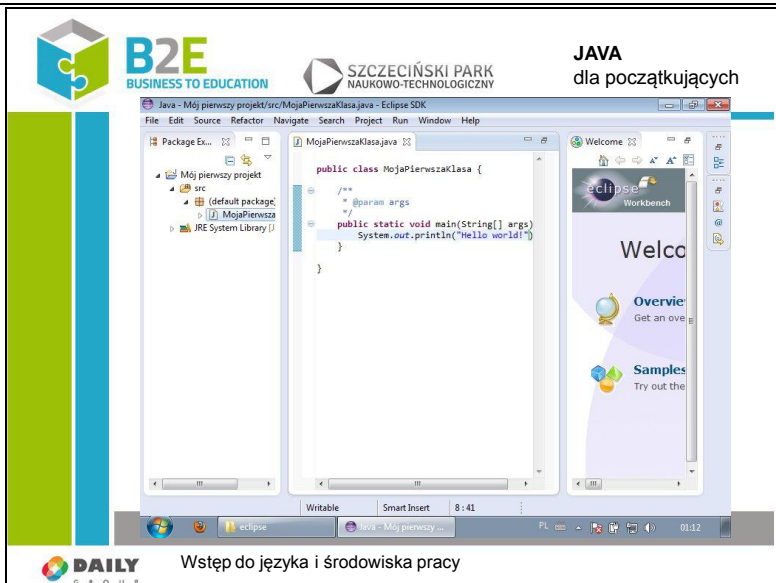
Wers zaczynający się na „//” zastępujemy wyrażeniem:

`System.out.println("Hello world!");`

Skrótem klawiszowym „Ctrl” + „S” zapisujemy zmiany.

Człowiek - najlepsza inwestycja

Slajd
43



JAVA dla początkujących

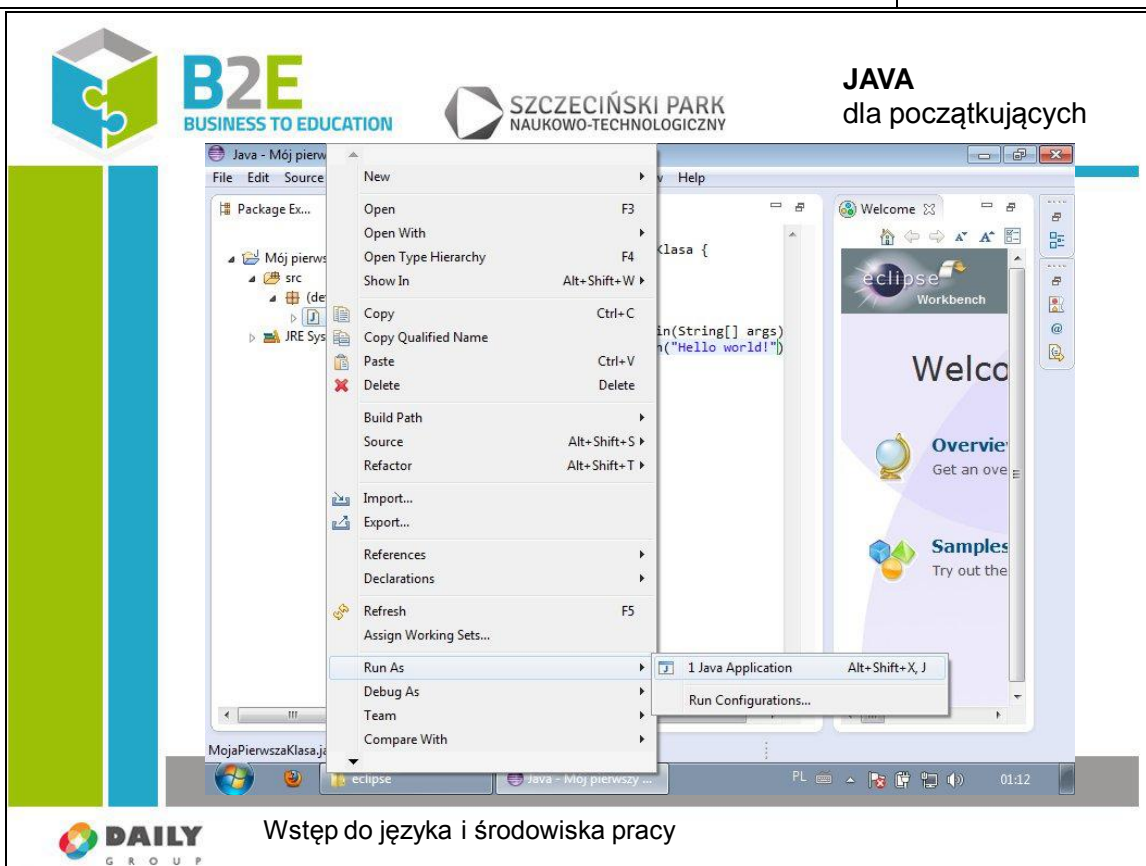
Wstęp do języka i środowiska pracy

Ponownie klikamy podwójnie na nazwie karty w celu jej zmniejszenia.

Dzięki temu ponownie widzimy strukturę plików w projekcie.

W celu uruchomienia pliku klikamy w obszarze „Package Explorer” na nim prawym klawiszem myszy.

Slajd
44



JAVA dla początkujących



Wstęp do języka i środowiska pracy

Wybieramy opcję:

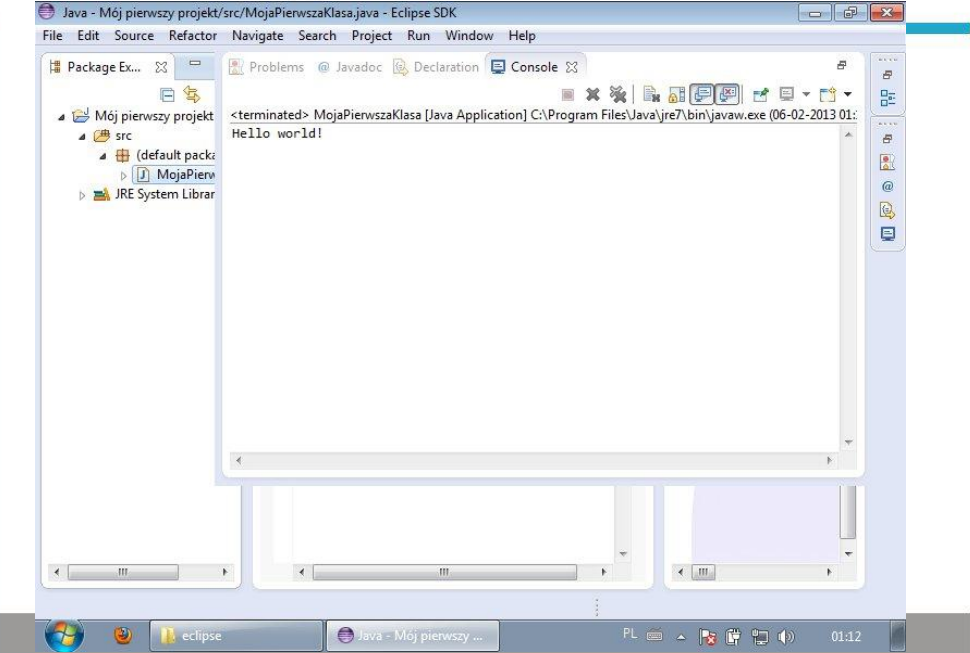
„Run As/1 Java Application”


Człowiek - najlepsza inwestycja

Slajd
45



JAVA
dla początkujących




Wstęp do języka i środowiska pracy

Jako rezultat operacji powinniśmy zobaczyć konsolę z widocznym napisem „Hello world!”.

Slajd
46

JAVA
dla początkujących

Zadanie:


Karty w Eclipse można zamykać i przemieszczać w zależności od indywidualnych preferencji programisty.

Dostosuj swoje środowisko tak aby widzieć strukturę projektu, plik tekstowy i konsolę jednocześnie, bez potrzeby przełączania kart.

Jeżeli chcesz ponownie wyświetlić zamkniętą kartę, wejdź:


Window/Show View/Other...

I wpisz nazwę karty, którą potrzebujesz.



Wstęp do języka i środowiska pracy

Człowiek - najlepsza inwestycja

Slajd
 47



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY


JAVA
 dla początkujących

```

public class MojaPierwszaKlasa {

    public static void main(String[] args) {
        System.out.println("Hello world!");
    }

}
            
```


Wstęp do języka i środowiska pracy

Przeanalizujemy kod pierwszego programu.

Wszystko w Javie jest obiektem. Obiekt jest instancją danej klasy, czyli pewnym tworem przechowywanym w pamięci, który przechowuje dane i nie tylko. Pojęcie obiektu zostanie dokładnie wytłumaczone na lekcji 3.

W Javie w każdym pliku znajduje się jedna klasa. Później przedstawię jak definiować klasy wewnątrz innych klas. Jednak po procesie kompilacji zawsze klasa jest osobnym plikiem.

Bardzo ważne jest aby klasa miała taką samą nazwę jak plik, w którym się znajduje. Pominięcie tej zasady traktowane jest jako błąd!

Wewnątrz klasy (pomiędzy nawiasami „{” i „}”) znajdują się pola i metody. W naszym przykładzie nie ma pól, jest tylko jedna metoda o nazwie „main”.

W późniejszych lekcjach zostaną również wytłumaczone znaczenia słów „public”, „static” i „void”. Na razie wszystkie polecenia będziemy pisać we wnętrzu metody „main”. Jest to szczególna metoda, ponieważ jest zawsze wywoływana przy próbie uruchomienia obiektu.

Obiekt przeznaczony do, bezpośredniego uruchomienia musi posiadać metodę:

„public static void main(String [] args)”.

Człowiek - najlepsza inwestycja

Slajd
 48

B2E
 BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojaPierwszaKlasa {

    /**
     * Pierwszy komentarz
     */
    public static void main(String[] args) {
        // Drugi komentarz
        System.out.println("Hello world!");
    } // koniec metody

    /**
     * Trzeci komentarz
     */
}
    
```



Wstęp do języka i środowiska pracy

W Javie, w prawie dowolnym miejscu, możemy pisać komentarze. Są one przeznaczone wyłącznie dla programistów.

Najczęściej spotykanymi komentarzami są te zaczynające się od symboli „//”. Możemy stosować je w tych samych liniach, w których są elementy kodu. Po wstawieniu symboli „//” kompilator pominie cały tekst, aż do końca linii.

Innym typem komentarzy są komentarze blokowe. Otwieramy je symbolem „/*” i zamykamy „*/”. Pomiędzy tymi znacznikami możemy wstawić dowolną liczbę wersów.

Dość szczególnym typem komentarzy blokowych są te zaczynające się symbolami „/**”. Zostawia się je przed definicją klasy i jej metodami. Nazywane są „Javadoc”. Pełnią ważną funkcję w tworzeniu dokumentacji. W późniejszych lekcjach zostanie pokazane jak je tworzyć, oraz jak korzystać z tej dokumentacji.

Pisanie komentarzy jest bardzo ważne. W dużych projektach, nad którymi pracuje wiele osób, są one praktyczną formą zostawiania wiadomości. Jeżeli napiszemy fragment kodu, o bardzo skomplikowanej logice, warto go skomentować. W takim przypadku bardzo przydatne mogą okazać się nawet lakoniczne informacje o podstawowych zasadach działania kodu. Bardzo ułatwią one rozwijanie aplikacji osobom,

Człowiek - najlepsza inwestycja


KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI



 UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY


które będą pracowały po nas.

Praca zespołowa nie jest jedynym powodem, dla którego pisze się komentarze. Jeżeli zechcemy po pewnym czasie powrócić do rozwijania swojej aplikacji, możemy nie pamiętać szczegółów dotyczących kodu. Oszczędzimy wiele czasu czytając komentarze, zamiast szczegółowo analizować kod.

Dla kompilatora bieżący kod jest identyczny jak ten na poprzednim slajdzie.

Slajd
49


JAVA
 dla początkujących

```
System.out.println("Hello
world!");

System.out.print("Hello");
System.out.println(" world!");

System.out.println(2+2);
```

Wynik:
 Hello world!
 Hello world!
 4



Wstęp do języka i środowiska pracy

Metoda:

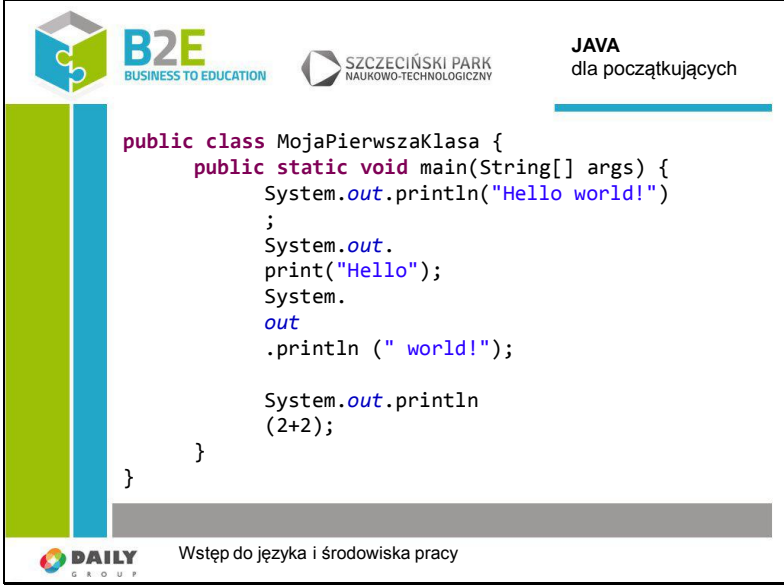
„System.out.println”

Wyświetla tekst podany w nawiasach (i w znakach „”) w konsoli i wstawia znak nowej linii.

W instancji klasy PrintStream „System.out” są również inne metody, np. „print” wstawia tekst bez znaku nowej linii.

Metoda „println” wyświetla nie tylko tekst. W Javie wszystkie obiekty mogą być wyświetlone w postaci tekstu. Będziemy wykorzystywać ją na najbliższych lekcjach do sprawdzania wyników przebiegu programów.

Człowiek - najlepsza inwestycja

	<p>Jest to jedna z najpowszechniej używanych metod. W środowisku Eclipse możemy wywołać ją automatycznie wpisując „sysou” i następnie wciskając „Ctrl” + „spacja”.</p> <p>Skrót klawiszowy „Ctrl” + „spacja” jest powszechnie wykorzystywany (nie tylko w Eclipse!) do podpowiadania elementów składni. Będziemy często go wykorzystywać.</p>	
Slajd 50		<p>W Javie każde polecenie musi kończyć się średnikiem. Nie oznacza to jednak, że średnik musi znajdować się na końcu każdej linii. Jeżeli nasze polecenia są zbyt długie (nie zaleca się stosować wersów o długości 200 znaków, są nieczytelne), możemy umieścić je w kilku wersach, pamiętając o zakończeniu całości średnikiem</p> <p>Na slajdzie znajduje się poprawny kod programu, wykonujący to co zostało przedstawione na slajdzie poprzednim. Należy zauważyć jednak, że polecenia zapisane w taki sposób nie są czytelne dla ludzi.</p> <p>Średnika nie wstawiamy, np. po nawiasach klamrowych.</p>

Człowiek - najlepsza inwestycja

Slajd
51



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Zadanie:

Napisz krótki program wypisujący na konsolę słowa:
„jeden dwa i trzy i cztery”;


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Człowiek - najlepsza inwestycja

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY


Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Wstęp do języka i środowiska pracy

```
public class Zolnierz {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("jeden dwa i trzy i cztery");
```

```
    }
```

```
}
```

8.1.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie poznają teoretyczną wiedzę na temat tego, czym jest Java. Będą umieli rozpocząć pracę w Javie w odpowiednim środowisku.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY




8.2 Lekcja 2 - Typy podstawowe i instrukcje sterujące

8.2.1 Cel lekcji



Celem lekcji jest poznanie podstawowych typów danych, wytłumaczenie jak bezpiecznie przechowywać dane w zmiennych. Poznanie pętli sterujących i zarządzanie przebiegiem wykonania programu za ich pomocą.

8.2.2 Treść – slajdy z opisem

Slajd 1		
------------	---	--

Człowiek - najlepsza inwestycja


Slajd
2

JAVA
 dla początkujących

Lista typów wbudowanych:

Nazwa typu	Rozmiar [bity]	Wartość minimalna	Wartość maksymalna	Odpowiednik obiektowy
boolean	-	-	-	Boolean
char	16	Unicode 0	Unicode 2 ¹⁶ -1	Character
byte	8	-128	+127	Byte
short	16	-2 ¹⁵	+2 ¹⁵ -1	Short
int	32	-2 ³¹	+2 ³¹ -1	Integer
long	64	-2 ⁶³	+2 ⁶³ -1	Long
float	32	IEEE754	IEEE754	Float
double	64	IEEE754	IEEE754	Double
void	-	-	-	Void


 Typy podstawowe i instrukcje sterujące

Typy wbudowane są szczególnym typem danych. Występują ponieważ traktowanie prostych danych jako obiekty nie jest szczególnie wydajne. Zmienne te przechowują proste wartości

Typ „boolean” służy do przechowywania wartości logicznej: prawda lub fałsz.

Typ „char” służy do przechowywania znaku (liter). Posiada 16 bitów, więc możemy swobodnie używać polskie znaki.

Typ „byte” służy do przechowywania bajtów danych.



Typy „short”, „int” i „long” służą do przechowywania liczb całkowitych. W zależności od zakresu na jakim chcemy operować wybieramy odpowiedni typ.

Typy „float” i „double” przechowują liczby zmiennoprzecinkowe zgodnie ze standardem IEEE754.

Typ „void” oznacza zerową informację. Nie można stworzyć zmiennej typu „void”, ale może on być zwracany przez metody.

Człowiek - najlepsza inwestycja

Slajd
3

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```


boolean x1 = true;
char x2 = 'Ż';
byte x3 = 102;
short x4 = 30000;
int x5 = 2000000000;
long x6 = 1000000000000000000L;
float x7 = 2.12f;
double x8 = 2.35;

System.out.println(x1);
System.out.println(x2);
System.out.println(x3);
System.out.println(x4);
System.out.println(x5);
System.out.println(x6);
System.out.println(x7);
System.out.println(x8);
        
```

Wynik:

```

true
Ż
102
30000
2000000000
1000000000000000000
2.12
2.35
        
```

 **DAILY**
G R O U P
 Typy podstawowe i instrukcje sterujące

Na slajdzie pokazane jest jak powinno się inicjalizować zmienne (nadawać wartości):

- boolean – używamy słów „true” lub „false”,
- char – podajemy pojedynczy znak w apostrofach,
- byte, short, int – liczba całkowita,
- long – liczba całkowita z literą „l” lub „L” na końcu,
- float – liczba ułamkowa z literą „f” lub „F” na końcu,
- double – liczba ułamkowa.

W celu wyświetlenia zainicjowanych wartości używamy wcześniej przedstawionej metody „println”.

Jak widać w wyniku nasze zmienne przechowują dokładnie takie wartości jak podaliśmy.

Człowiek - najlepsza inwestycja

Slajd
4




JAVA
 dla początkujących

```
float f = 3.1415926535897932384626433f;
double d = 3.1415926535897932384626433f;
System.out.println("3.1415926535897932384626433");
System.out.println(f);
System.out.println(d);
```

Wynik:

```
3.1415926535897932384626433
3.1415927
3.1415927410125732
```


 Typy podstawowe i instrukcje sterujące


Dokonując operacji na typach zmiennoprzecinkowych należy szczególnie pamiętać o dokładności wykonywanych obliczeń.

Przykład pokazuje dokładność z jaką typy „float” i „double” przechowują liczbę Pi.


Nie oznacza to jednak, że w Javie nie można przeprowadzać obliczeń o wyższej precyzji. W standardowych bibliotekach Javy znajdują się klasy takie jak „BigInteger” i „BigDecimal”, które mogą wykorzystać do zapamiętania liczby całą dostępną pamięć RAM komputera. Są to jednak obiekty. Jak korzystać z obiektów zostanie wytłumaczone w następnej lekcji.

Człowiek - najlepsza inwestycja

Slajd
5



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących


```

int a = 8, b = 2, c;
c = a + b;
System.out.println(c);
System.out.println(a+b);
System.out.println(a-b);
System.out.println(b-a);
System.out.println(a*b);
System.out.println(a/b);
System.out.println(c/a);
System.out.println(c%a);

a += a;
System.out.println(a);
a -= c;
System.out.println(a);
a *= a;
System.out.println(a);
a /= b;
System.out.println(a);
                
```

Wynik:

 10
 10
 6
 -6
 16
 4
 1
 2
 16
 6
 36
 18



Typy podstawowe i instrukcje sterujące

Na typach wbudowanych możemy wykonywać operacje matematyczne korzystając z dostępnych operatorów.

Operator „%” służy do otrzymywania reszty z dzielenia, pozostałe działają zgodnie z intuicją.

Przypisanie sumy liczb a i b do c nastąpiło poprzez użycie operatora podstawiania („=”).



Operatory „+”, „-” i „*” działają zgodnie z intuicją (suma, różnica, mnożenie).

Należy zwrócić szczególną uwagę na operator „/” dzielenia. W przypadku gdy dzielimy a przez b dostajemy poprawny wynik ($8/2=4$). Kiedy jednak dzielimy c przez a ($10/8=1,25$) otrzymaliśmy wynik 1 zamiast 1,25. Dzieje się tak, ponieważ typy te nie przechowują części ułamkowej. Nie następuje również zaokrąglenie wartości. Cyfry znajdujące się po przecinku są „odcinane”, więc zamiast np. 1,8 też dostaniemy 1.

Operatory użyte poniżej biorą argumenty z obu stron, przypisując wartość do zmiennej znajdującej się z lewej strony. Wykonują analogicznie operacje sumy, różnicy, mnożenia i dzielenia.

Człowiek - najlepsza inwestycja

Slajd
6

JAVA
 dla początkujących


```

int i = 1;
// preinkrementacja
System.out.println(++i);
// postinkrementacja
System.out.println(i++);
// predekrementacja
System.out.println(--i);
// postdekrementacja
System.out.println(i--);
    
```

Wynik:

```

2
2
2
2
    
```


 Typy podstawowe i instrukcje sterujące

Operator preinkrementacji najpierw zwiększa zmienną o jedną jednostkę, następnie zwraca jej wartość. 1 zostało zwiększone, a następnie wyświetlone. Po operacji i=2.



Operacja postinkrementacji najpierw zwraca wartość, a następnie ją zwiększa. 2 zostało najpierw wyświetlone, a następnie zwiększone. Po tej operacji i=3.

Operator predekrementacji najpierw zmniejsza zmienną o jednostkę, następnie zwraca jej wartość. 3 zostało zmniejszone, a następnie wyświetlone. Po operacji i=2.

Operacja postdekrementacji najpierw zwraca wartość, a następnie ją zmniejsza. 2 zostało najpierw wyświetlone, a następnie zmniejszone. Po tej operacji i=1.

Człowiek - najlepsza inwestycja

Slajd
 7

JAVA
 dla początkujących


```

int a = 8, b = 3;
System.out.println(a/b);
System.out.println((float)a/b);
System.out.println((double)a/b);

int x = 1500000000, y = 1500000000;
System.out.println(x+y);

Wynik:

2
2.6666667
2.6666666666666665
-1294967296
        
```


 Typy podstawowe i instrukcje sterujące

Fakt, że dla typów „int”, „short”, „long” wartości mniejsze od 1, są obcinane, nie oznacza, że nie możemy ich używać jeżeli w przyszłości może zająć potrzeba dokonania dokładniejszych obliczeń. Istnieje możliwość rzutowania jednych typów na drugie.

Na slajdzie mamy trzy możliwe wyniki dzielenia 8 przez 3 w zależności od dokładności wyniku jaki chcemy otrzymać.

Poniżej mamy dość szczególny przypadek. Zastanówmy się, jak to jest możliwe, że dodając dwie liczby dodatnie, otrzymaliśmy ujemny wynik.


Proszę wyobrazić sobie, że typy dla liczb całkowitych mogą się „przekręcać”.

Dobłą analogią do tego zjawiska jest przykład licznika samochodowego. Jeżeli wartość na nim pokazywana będzie cały czas rosnąć, to przekroczy wartość maksymalną i ponownie przekroczy zero. Z omawianymi typami dzieje się dokładnie tak samo.


Wartość maksymalna dla „int” to $2^{31}-1$ (około 2 mld). Jeżeli zechcemy zapisać 3 mld, to „licznik się przekręci” i do minimalnej wartości -2^{31} (około -2 mld) dodana zostanie wartość będąca różnicą 3 mld – 2mld = 1 mld. Otrzymaliśmy w ten sposób wynik około -1 mld.

Człowiek - najlepsza inwestycja

Slajd
8



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących


Operatory logiczne (wartości mogą być):

- „==” - równe,
- „!=” - różne,
- „<” - mniejsze,
- „>” - większe,
- „<=” - mniejsze lub równe,
- „>=” - większe lub równe.

```
int a = 3, b = 5;
System.out.println(a > b ? a : b);
```

Wynik:

5



Typy podstawowe i instrukcje sterujące


Przedstawione operatory służą do porównywania wartości. Działają zgodnie z tym, czego używaliśmy na lekcjach matematyki.

Poniżej mamy przykład jedynego trójargumentowego operatora w Javie. Składa się z dwóch znaków: „?” i „:”. W pierwszej kolejności podajemy wartość logiczną. Wynikiem każdej operacji logicznej jest wartość logiczna. Możemy w tym miejscu równie dobrze użyć zmiennej typu boolean. Następnie, po znaku „?” podajemy polecenie, które ma zostać wykonane w przypadku prawdy. Po znaku „:” podajemy polecenie dla fałszywej wartości logicznej.

Wynikiem użycia operatora „?” w przykładzie pokazanym na slajdzie jest drukowanie w konsoli wartości większej (a lub b).

Człowiek - najlepsza inwestycja

Slajd
9

JAVA
 dla początkujących

Instrukcja warunkowa if-else:

```

int a = 3, b = 5, c = 5;

if (a < b) {
    System.out.println("Przykład 1, przypadek 1.");
}

if (a > b) {
    System.out.println("Przykład 2, przypadek 1.");
}
else {
    System.out.println("Przykład 2, przypadek 2.");
}

if (a > b) {
    System.out.println("Przykład 3, przypadek 1.");
}
else if (b > c) {
    System.out.println("Przykład 3, przypadek 2.");
}
else {
    System.out.println("Przykład 3, przypadek 3.");
}
                
```

Wynik:

Przykład 1, przypadek 1.
 Przykład 2, przypadek 2.
 Przykład 3, przypadek 3.



Typy podstawowe i instrukcje sterujące

Na slajdzie przedstawione zostały różne warianty instrukcji warunkowej „if”. Korzystając z niej decydujemy, które polecenia mają być wykonane w zależności od zadanych kryteriów.

Instrukcja warunkowa „if” działa w następujący sposób. Warunek w nawiasie po słowie „if” musi być wyrażeniem logicznym (dawać wartość prawdę lub fałsz). W przypadku prawdy wykonają się instrukcje zawarte w nawiasach klamrowych, za wyrażeniem logicznym. Jeżeli później występuje słowo „else”, to w przypadku prawdy wykonają się instrukcje występujące po słowie else.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 71

Slajd
10

JAVA
 dla początkujących

```

if (true && true && false) {
    System.out.println("Prawda!");
}
else {
    System.out.println("Fałsz!");
}


if (true && true) {
    System.out.println("Prawda!");
}

if (false || true || false) {
    System.out.println("Prawda!");
}

if (true && true) {
    System.out.println("Prawda!");
}
        
```

Wynik:

 Fałsz!
 Prawda!
 Prawda!
 Prawda!




 Typy podstawowe i instrukcje sterujące

Dostępne mamy również inne operatory logiczne: koniunkcji („&&”) i alternatywy („||”).

Należy pamiętać, że kiedy użyjemy operatora „&&”, to kiedy przynajmniej jedna z wartości jest fałszywa, całe wyrażenie jest fałszywe. Dlatego wyrażenia sprawdzane są od lewej do prawej. Po napotkaniu pierwszej fałszywej wartości reszta nie jest już sprawdzana. Nie zachodzi taka potrzeba.

Człowiek - najlepsza inwestycja

Slajd
11

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących


```

int a = 5, b = 7, c = 0, y = 0;

while (c < b) {
    y += a;
    c++;
}

System.out.println(y);

Wynik:
35
    
```

 **DAILY**
G R O U P
 Typy podstawowe i instrukcje sterujące

Na slajdzie widzimy przykładową pętlę „while”. Wykonuje ona operację mnożenia a i b, zapisując wynik do c.

Po słowie „while” występuje wyrażenie logiczne, tak jak w instrukcji „if”. W przypadku prawdy instrukcje z nawiasów klamrowych zostaną wykonane. Następnie wyrażenie logiczne jest sprawdzane ponownie i w przypadku prawdy następuje kolejny krok pętli. Instrukcja „while” przestaje działać jeżeli warunek (wyrażenie logiczne w nawiasach) jest fałszywy.

Jeżeli źle skonstruujemy warunek logiczny, pętla może się nie zatrzymać! W takim przypadku proces w systemie operacyjnym, który odpowiada naszemu programowi, powinien być „zabity”.

Człowiek - najlepsza inwestycja

Slajd
 12

B2E
 BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Ćwiczenie:

Wykorzystując pętlę „while” napisz program obliczający 10 liczbę ciągu Fibonacciego.

Każda liczba ciągu powstaje poprzez dodanie dwóch poprzednich liczb, a dwie pierwsze liczby mają wartość 1.

Przykład:

Liczba 1 = 1

Liczba 2 = 1

Liczba 3 = 1 + 1 = 2

Liczba 4 = 1 + 2 = 3



Typy podstawowe i instrukcje sterujące

```
public class MojaPierwszaKlasa {

    public static void main(String[] args) {

        int krok = 0;

        int n1=1, n2, wynik=0;

        while (krok<10) {

            n2 = wynik;

            wynik += n1;

            n1 = n2;




            krok++;

            System.out.println(krok);
        }
    }
}
```

Człowiek - najlepsza inwestycja


KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

 UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY


	<pre> System.out.println(wynik); System.out.println(); } } } </pre>		
Slajd 13	<div>   <div>JAVA dla początkujących</div> <hr/> <pre> int a = 1, b = 1; do { a *= b++; System.out.println(a); } while (a<1000); </pre> <p>Wynik:</p> <pre> 1 2 6 24 120 720 5040 </pre> <div>  Typy podstawowe i instrukcje sterujące </div> </div>		
		<p>Pętla „do-while” działa tak samo jak pętla „while”, z tą różnicą, że zawsze występuje pierwszy przebieg pętli. Dopiero po nim sprawdzany jest warunek. W przypadku prawdy następuje kolejny przebieg.</p> <p>Pętla widoczna na slajdzie wypisuje kolejne wartości silni, aż do momentu, kiedy wartość będzie większa niż 1000.</p>	

Człowiek - najlepsza inwestycja

Slajd
14B2E
BUSINESS TO EDUCATIONSZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNYJAVA
dla początkujących

Ćwiczenie:

Wykorzystując pętlę „do-while” napisz program znajdujący najmniejszy dzielnik liczby 2629 (dzielnik musi być większy od 1).





Typy podstawowe i instrukcje sterujące

```
public class MojaPierwszaKlasa {  
  
    public static void main(String[] args) {  
  
        int x = 2639, i = 1;  
  
        do {  
  
            i++;  
  
        } while (x % i != 0);  
  
        System.out.println(i);  
  
    }  
  
}
```

Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCIUNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Slajd
 15

JAVA
 dla początkujących

```

for (instrukcja początkowa ; warunek ; krok) {}

int a = 2, b = 9, c = 1;


for (int i=0 ; i<b ; i++) {
    c *= a;
}
System.out.println(c);

Wynik:
512
                
```

```

// to działa tak samo
int i=0;
for (; i<b ;) {
    i++;
    c *= a;
}

// to jest poprawne
for (;;) {}
                
```


 Typy podstawowe i instrukcje sterujące

Pętla „for” jest bardziej złożona od poprzednich przykładów. W nawiasie po słowie „for” nie występuje tylko warunek. Wyrażanie ma 3 części:

Pierwsza część, instrukcja początkowa, wykonywana tylko raz na samym początku. W naszym przypadku zadeklarowaliśmy i zainicjowaliśmy zmienną i.

Druga część, warunek, jest sprawdzana w każdym kroku. Musi to być wyrażenie logiczne. Dopóki wartość jest prawdziwa, dopóty wykonuje się pętla.

Trzecia część, to instrukcja, która wykonuje się po zakończeniu każdego kroku.


Przedstawiony przykład oblicza wartość liczby a podniesioną do potęgi b.

Każda z trzech instrukcji pętli „for” może być pusta. Z prawej strony slajdu pokazany jest przykład, który działa identycznie jak omawiana pętla. Jednak takie zapisywanie pętli jest mało czytelne dla ludzi.


Pętla „for”, przedstawiona w prawym dolnym rogu, również jest poprawna. Powoduje ona zawieszenie się wykonywanego programu, z powodu braku instrukcji warunkowej. Jest to przykład pętli nieskończonej.

Człowiek - najlepsza inwestycja

Slajd
16



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących


```

for (int i=0, a=0 ; i<100 ; i++) {
    if (i>10) {
        break;
    }
    if (i%2 == 0) {
        continue;
    }
    a += i;
    System.out.println(a);
}
        
```

Wynik:

```

1
4
9
16
25
        
```



Typy podstawowe i instrukcje sterujące

Przy operowaniu na pętlach możemy skorzystać z dwóch instrukcji: „break” i „continue”.

Przy napotkaniu słowa „continue”, wykonanie bieżącego kroku pętli zostaje zatrzymane i zaczyna się wykonywanie kroku następnego.



Po wystąpieniu słowa „break” następuje całkowite wyjście z pętli „for”.

Powyższy przykład sumuje wszystkie liczby nieparzyste od 1 do 10. Instrukcja warunkowa w pętli „for” w naszym przypadku nie ma żadnego znaczenia. Polecenie „break” przerywa działanie pętli kiedy tylko i będzie większe od 10.

Kiedy reszta z dzielenia i przez 2 będzie równa 0 (liczba parzysta), nie nastąpi operacja sumy i wypisania na ekran.

Człowiek - najlepsza inwestycja

Slajd
17

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

```

int a=2;
switch (a) {
case 1:
    System.out.println("Wartość 1.");
    break;
case 2:
    System.out.println("Wartość 2.");
default:
    System.out.println("Inna wartość.");
}
    
```

Wynik:

Wartość 2.
Inna wartość.

 **DAILY**
G R O U P
 Typy podstawowe i instrukcje sterujące

Ostatnią instrukcją jest „switch”. Dzięki niej możemy decydować, które instrukcje wykonają się w danym przypadku.

Po słowie „switch” podajmy w nawiasie wartość typu „int”, „String” lub „Enum” (nie może to być wartość logiczna). Pojęcia „String” i „Enum” zostaną wytłumaczone w następnych lekcjach.

Następnie możemy zdefiniować dowolną ilość przypadków. Po słowie „case” podajmy wartość dla której dane instrukcje mają się wykonać. Nie otwieramy nawiasu klamrowego! Pamiętajmy, aby wstawić „break”, jeżeli nie chcemy, aby wykonały się instrukcje dla następnego przypadku. Po drugim „case” nie ma tego słowa, więc wykonały się również instrukcje dla przypadku „default”.

„default” jest przypadkiem szczególnym. Jeżeli wartość a nie występuje w żadnym z „case”, wówczas wykonają się instrukcje z tej części.

Człowiek - najlepsza inwestycja

Slajd
 18

B2E
 BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Ćwiczenie:

Korzystając z dwóch pętli „for” (jedna wewnątrz drugiej), wypisz w konsoli całą tabliczkę mnożenia dla liczb od 1 do 10.

Do wyrównania tabeli możesz skorzystać z różnic w wykonaniu poleceń „print” i „println”.

Aby uzyskać łańcuchu znaków efektu tabulacji użyj wyrażenia „\t”.



Typy podstawowe i instrukcje sterujące

```
public class MojaPierwszaKlasa {

    public static void main(String[] args) {

        for (int i=1 ; i<=10 ; i++) {

            for (int j=1 ; j<=10 ; j++) {

                System.out.print(i*j);

                System.out.print("\t");

            }

            System.out.println();

        }

    }

}
```

Człowiek - najlepsza inwestycja




KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY


	<p>}</p> <p>}</p>
--	-------------------

Człowiek - najlepsza inwestycja


Slajd
19

JAVA
dla początkujących

Ćwiczenie:

Korzystając z typu „float” i „double” dodaj 10 razy 0,1 do wartości 0. Sprawdź jaki jest wynik sumowania dla każdego z typów.



Typy podstawowe i instrukcje sterujące

```
public class MojaPierwszaKlasa {

    public static void main(String[] args) {

        float af = 0f, bf = 0.1f;

        double ad = 0, bd = 0.1;

        for (int i=0 ; i<10 ; i++) {

            af += bf;

            ad += bd;

        }

    }

}
```

Człowiek - najlepsza inwestycja

	<pre> System.out.println(af); System.out.println(ad); } }</pre>	
Slajd 20	 <p>Wnioski:</p> <p>Typy „float” i „double” zawsze obciążone są błędem obliczeniowym!</p> <p>Nie stosuj porównań typu:</p> <pre>If (liczba == 3.14) {...}</pre> <p>Aby robić to właściwie należy najpierw dobrze poznać arytmetykę zmiennopozycyjną.</p> <p><small>Człowiek - najlepsza inwestycja</small></p> <p><small>KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI</small></p> <p><small>UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY</small></p> <p><small>Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego</small></p> <p>DAILY GROUP Typy podstawowe i instrukcje sterujące</p>	

8.2.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą wiedzieli jak poprawnie przechowywać dane w zmiennych, oraz jak programować pętle do wykonywania odpowiednich operacji.



8.3 Lekcja 3 - Wstęp do programowania obiektowego

8.3.1 Cel lekcji

Celem lekcji jest wytłumaczenie, na czym polega idea programowania obiektowego. Opis budowy klasy w Javie, konstruktorów i metod.

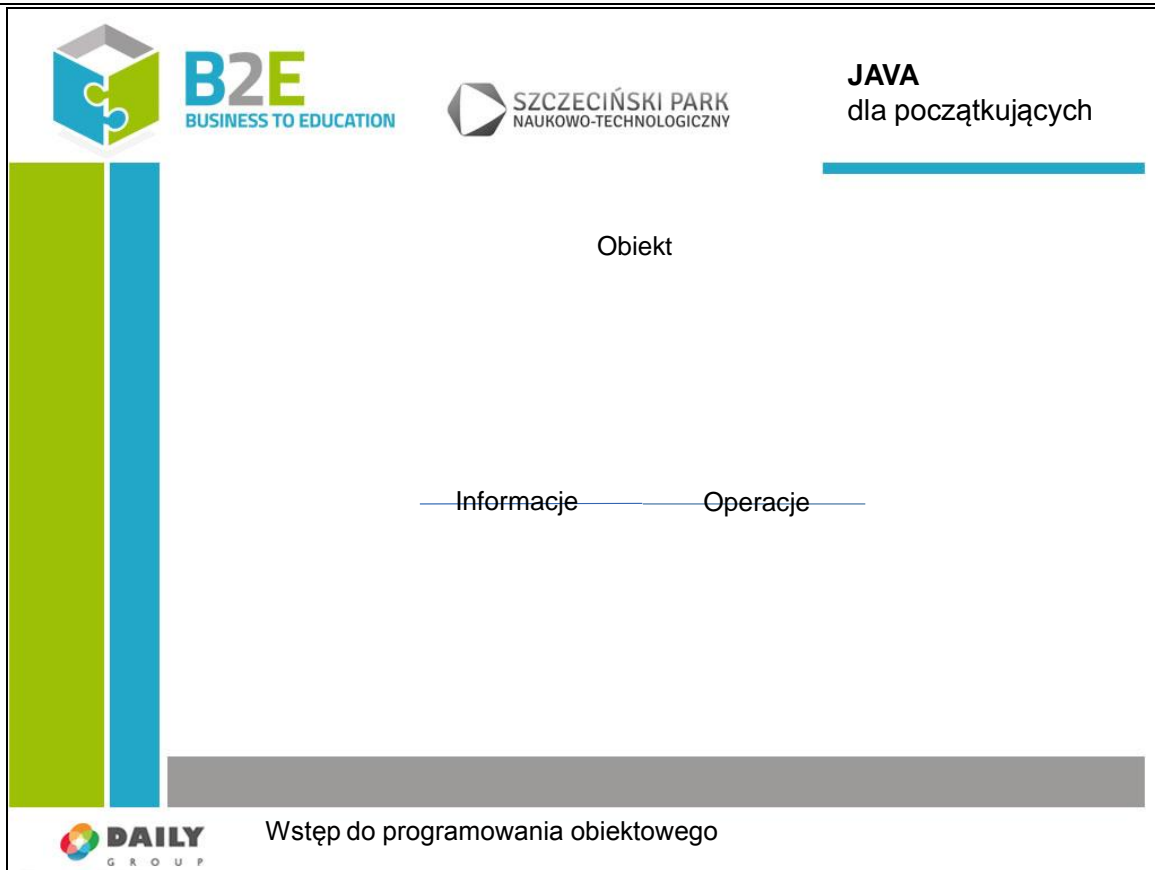
Człowiek - najlepsza inwestycja

8.3.2 Treść - slajdy z opisem

<p>Slajd 1</p>		
<p>Slajd 2</p>		<p>Niemal każdy element świata rzeczywistego możemy potraktować jako obiekt, niezależnie od tego, czy jest niepodzielnym elementem (np. kamień), czy składa się z wielu innych części (np. długopis).</p> <p>Jest to naturalny sposób postrzegania świata.</p>

Człowiek - najlepsza inwestycja

Slajd
3



O każdy obiekt ma typy właściwości:

Informacje:

- kolor długopisu,
- cena,
- prędkość maksymalna samochodu,
- masa;

Operacje:

- pisanie,
- zdolność do poruszania się,
- wydawanie dźwięków.

Zauważmy, że ciężko jest wyobrazić sobie temperaturę jako obiekt. Nie utożsamiamy obiektu z

Człowiek - najlepsza inwestycja

przedmiotem! Obiektem może być np. stan techniczny części samochodowej.

Myślę, że dobrym przykładem na to jest temperatura:

- ma swoją wartość,
- może maleć i wzrastać,
- nie może być niższa od zera absolutnego.

Jeżeli chcielibyśmy stworzyć model pomieszczenia, w którym się znajdujemy, obiektem powinno być przede wszystkim samo pomieszczenie, przedmioty znajdujące się w środku, jak i temperatura powietrza.

Byłby to model obiektowy. W jasny i czytelny sposób pozwoliłby zachować niezbędne informacje na temat stanu wszystkich przedmiotów, oraz przewidzieć możliwe zdarzenia. Jeżeli jest tablica i kreda, to możliwe, że któraś z osób coś na niej napisze. Jeżeli nie znalazłby się obiekt z operacją „pisz” (kreda lub długopis), to z obiektowego punktu widzenia niemożliwe jest, aby w tym pomieszczeniu zostało coś napisane.

Slajd
4



B2E
BUSINESS TO EDUCATION





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Pojazdy

Pojazdy spalinowe



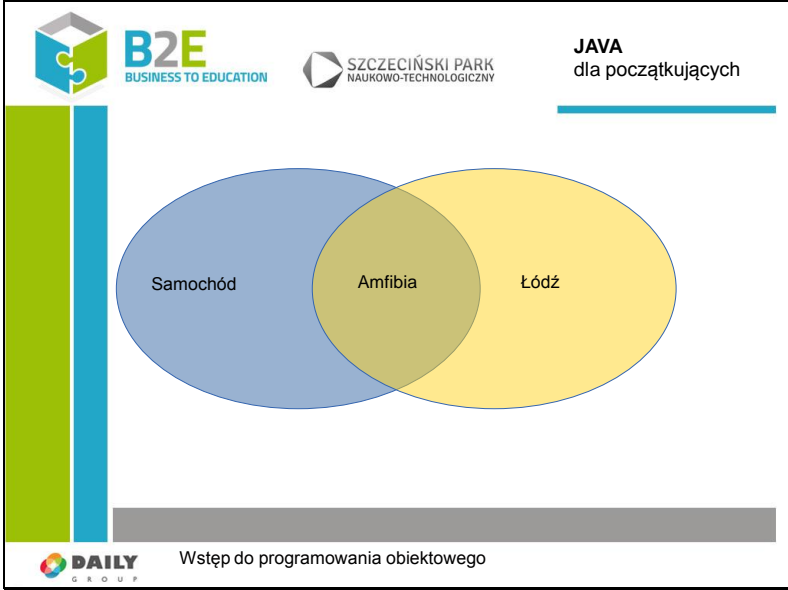





Wstęp do programowania obiektowego



Na obiektach możemy dokonywać generalizacji i specjalizacji. Są to kluczowe zjawiska, jeżeli chcemy

Człowiek - najlepsza inwestycja

	<p>skorzystać z mechanizmu dziedziczenia. Pojęcie dziedziczenia zostanie wyjaśnione na następnej lekcji.</p> <p>Samochód jest pojazdem spalinowym, jeżeli powiemy, że jest tylko pojazdem, wtedy następuje generalizacja.</p> <p>Jeżeli powiemy, że samochód jest pojazdem, a ponadto jest pojazdem spalinowym, to następuje specjalizacja.</p> <p>Zjawisko to jest bardzo przydatne. Wyobraźmy sobie, że piszemy grę, w której występuje ruch uliczny. Na skrzyżowaniu, na czerwonym świetle stoi grupa pojazdów (samochody, rowery, motocykle). Kiedy zapali się zielone światło, wywołamy na każdym z pojazdów metodę „jeźdź”. Nie interesuje nas wtedy jakiego typu jest poszczególny pojazd. Wszystkie one potrafią jeździć, zatrzymywać się i skręcać.</p> <p>W przypadku, kiedy ta sama grupa pojazdów dojedzie do wjazdu na autostradę, musimy zatrzymać wszystkie pojazdy, które nie są spalinowe. Jazda rowerem po autostradzie jest zakazana.</p>
Slajd 5	<div>  </div> <p>Pojedynczy obiekt może posiadać cechy, które są zbiorem cech, należących do dwóch różnych pojazdów.</p> <p>Np. amfibia posiada cechy zarówno łodzi, jak i samochodu.</p>


Człowiek - najlepsza inwestycja

Slajd
6

 **B2E**
BUSINESS TO EDUCATION
  **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:
Mając do dyspozycji terminy: koń, zwierzę, osioł, muł
Rozmieść je na kartce i zakreśl odpowiednie okręgi tak aby uwzględnić
relacje relacje między nimi

 **DAILY**
GROUP
 Wstęp do programowania obiektowego

Slajd
7

 **B2E**
BUSINESS TO EDUCATION
  **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Typy widoczności:

- public,
- package.

```
public class MojZnak {
}
```

 **DAILY**
GROUP
 Wstęp do programowania obiektowego

Klasę w Javie zapisujemy podając typ widoczności, słowo kluczowe „class”, nazwę klasy i nawiasy klamrowe, w których znajduje się ciało klasy.


Omówimy dwa typy widoczności: „public” i domyślny (Oficjalnie Oracle nazywa ten typ widoczności

Człowiek - najlepsza inwestycja


	<p>„package-private”, część programistów mówi o nim „package-protected” a jeszcze inni traktują go jako domyślny. W języku polskim mówi się czasem o „dostępie pakietowym”).</p> <p>„public” oznacza, że nasza klasa będzie widoczna z każdego miejsca w kodzie.</p> <p>Klasy w naszej aplikacji możemy trzymać w pakietach. Mechanizm działa tak, jak drzewo folderów w systemie operacyjnym. W Javie słowo kluczowe „package” nie opisuje typu widoczności, służy do deklarowania przynależności klasy do danego pakietu. Jeżeli chcemy skorzystać z typu widoczności domyślnej, to nie wpisujemy żadnego typu widoczności w definicji klasy. Wówczas nasza klasa będzie widoczna dla wszystkich klas znajdujących się w danym pakiecie.</p>
--	--

Człowiek - najlepsza inwestycja

Slajd
 8



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojZnak {

    private char znak;
    public int liczba;

}


public class MojaPierwszaKlasa {

    public static void main(String[] args) {

        MojZnak mojZnak = new MojZnak();
        mojZnak.liczba = 2;

    }

}
                
```



Wstęp do programowania obiektowego

Jeżeli chcemy przechowywać w klasie pewne informacje, tworzymy pola. W Javie nie można przechowywać danych, które nie należą do żadnego obiektu! (wyjątkiem od reguły są pola statyczne, które należą do definicji klasy a nie jej instancji).

Pola zawierają dodatkowo typ widoczności „private”. Oznacza to, że dane zmienna jest widoczna tylko w ciele danej klasy. Pola publiczne są widoczne w każdej klasie w aplikacji.

Na slajdzie w metodzie „main” po raz pierwszy świadomie tworzymy obiekt! Obiekt jest reprezentacją danej klasy (instancją) „powołaną do życia”. Przeanalizujmy jak odbywa się ten proces.

W pierwszej kolejności musimy stworzyć referencję. Referencja jest adresem na komórkę w pamięci, gdzie znajduje się dana instancja klasy. Referencja posiada również typ obiektu na który wskazuje.

Piszemy:

TypReferencji nazwaReferencji;



W naszym przypadku jest to referencja o nazwie „mojZnak”, typu „MojZnak”.

Człowiek - najlepsza inwestycja

	<p>Korzystając ze słowa kluczowego „new” alokujemy obszar pamięci operacyjnej komputera na przechowanie obiektu danego typu. Następnie znajduje się konstruktor danej klasy.</p> <p>Należy pamiętać, że nazwy klas piszemy dużą literą, a nazwy pól z małej.0</p> <p>Do pól danej klasy odwołujemy się pisząc referencję, wstawiając symbol kropki, następnie wstawiając nazwę pola (tak jak w przykładzie).</p> <p>Zauważmy, że do pola „znak” nie możemy się odwołać (jest prywatne).</p>
--	---

Człowiek - najlepsza inwestycja

Slajd
9


JAVA
dla początkujących

```

public class MojZnak {
    private char znak;
    public int liczba;
}

public class MojaPierwszaKlasa {
    public static void main(String[] args) {
        MojZnak mojZnak = new MojZnak();
        System.out.println(mojZnak.liczba);
    }
}

```


Wstęp do programowania obiektowego


W tym przykładzie chcemy wyświetlić pole „liczba” nie inicjalizując go. W tym przypadku otrzymamy 0, ale na ogół takie przypadki mogą być bardzo niebezpieczne i zagrażać stabilności naszego programu. Dlatego dobrym zwyczajem jest inicjalizacja zmiennych, mimo iż w Javie są dokładnie sprecyzowane reguły wartości domyślnych.

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false


Jeżeli tworzymy zaawansowane obiekty, musimy mieć kontrolę nad wartościami domyślnymi ich pól.

Człowiek - najlepsza inwestycja

Slajd
10



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojZnak {

    private char znak;
    public int liczba;


    public MojZnak() {
        liczba = 7;
    }

}

public class MojaPierwszaKlasa {

    public static void main(String[] args) {
        MojZnak mojZnak = new MojZnak();
        System.out.println(mojZnak.liczba);
    }

}
        
```



Wstęp do programowania obiektowego

Konstruktor jest metodą specjalną. Jeżeli nie zdefiniujemy własnego, to kompilator dołącza domyślny konstruktor bezparametrowy. Dlatego w poprzednich przykładach mogliśmy pisać „new MojZnak()”.

Tworzenie własnych konstruktorów rozwiązuje problem inicjalizowania pól.

W tym przypadku, po stworzeniu klasy pole „liczba” ma wartość 7.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY




Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 93

Slajd
 11



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojZnak {

    private char znak;
    public int liczba;


    public MojZnak(int liczba) {
        this.liczba = liczba;
    }

    public class MojaPierwszaKlasa {

        public static void main(String[] args) {
            // to już nie jest poprawne
            // MojZnak mojZnak = new MojZnak
            MojZnak mojZnak = new MojZnak(5)
            System.out.println(mojZnak.liczb

        }

    }
        
```



Wstęp do programowania obiektowego

Na slajdzie widoczny jest przykład konstruktora z jednym parametrem. Tworząc obiekt, użytkownik musi podać wartość początkową dla pola „liczba”. Kompilator nie dołączył już konstruktora domyślnego. Jest to jedyny sposób w jaki możemy stworzyć obiekt.

Słowo kluczowe „this” jest referencją na obiekt, w którym aktualnie się znajdujemy. Polecenie „this.liczba = liczba” może wyglądać dziwnie, ale jest całkowicie bezpieczne. Ponieważ argument ma taką samą nazwę jak pole przysłania „ukrywa” nam właśnie to pole. Dlatego sam zapis „liczba = liczba”, będąc poprawnym syntaktycznie byłby bezsensownym przypisaniem wartości argumentu do tego samego argumentu. Używając słowa kluczowego „this” możemy dostać się do przysłoniętego pola. Powoduje przypisanie wartości argumentu „liczba” do pola obiektu.

W środowisku Eclipse konstruktory możemy zbudować automatycznie korzystając z kreatora dostępnego w:

„Source/Generate Konstruktor using Fields...”

Człowiek - najlepsza inwestycja

Slajd
12



JAVA dla początkujących

Ćwiczenie:

Stwórz obiekt zawierający dwa pola liczbowe: a i b (dowolnego typu).
Stwórz dwa konstruktory. Jeden bez parametrów, wtedy zainicjalizuj liczby wartością 0. Drugi z dwoma parametrami i przypisz ich wartości polom.
Klasę nazwij „DwieLiczby”.



Wstęp do programowania obiektowego

```
public class DwieLiczby {

    private int a;

    private int b;

    public DwieLiczby(int a, int b) {

        this.a = a;

        this.b = b;

    }

    public DwieLiczby() {
```

Człowiek - najlepsza inwestycja






KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY




Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 95


	<pre> this.a = 0; this.b = 0; } } </pre>	
Slajd 13	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class MojZnak { private char znak; public int liczba; public MojZnak(int liczba) { this.liczba = liczba; } public MojZnak() { public class MojaPierwszaKlasa { public static void main(String[] args) { // teraz już jest poprawne MojZnak mojInnyZnak = new MojZnak(); MojZnak mojZnak = new MojZnak(5); System.out.println(mojZnak.liczba); } } } } </pre> <div>  Wstęp do programowania obiektowego </div>	<p>Możemy stworzyć dowolną liczbę konstruktorów. Muszą jednak różnić się listą argumentów, które przyjmują.</p>

Człowiek - najlepsza inwestycja

Slajd
 14



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojZnak {


    private char znak;
    private int liczba;

    public int getLiczba() {
        return liczba;
    }

    public void setLiczba(int liczba) {
        this.liczba = liczba;
    }

}

MojZnak mojZnak = new MojZnak();
mojZnak.setLiczba(13);
System.out.println(mojZnak.getLiczba());
            
```



Wstęp do programowania obiektowego

Pola nie powinny mieć widoczności publicznej. Dobrą praktyką jest enkapsulacja metod i pól, czyli ukrywania składowych obiektu przed światem zewnętrznym, dostępem z zewnątrz. Jest to dobra praktyka, dzięki której możemy zapewnić konsystentny stan naszego obiektu. Przykładowo mając definicję klasy pojazd i mając publiczne pole „przebieg”, narażamy się na to iż inny fragment kodu wprowadzi ujemną wartość. Jak więc na nich operować z poziomu innych klas?

Do tego używa się metod. Metoda posiada typ widoczności, typ który zwraca, nazwę oraz listę argumentów. Ogólnie przyjęta konwencja mówi że nazwa metody powinna być pisana małą literą.

Szczególnym typem zwracany przez metodę jest „void”. Oznacza to, że metoda wykonuje jedynie pewne operacje, nie zwracając rezultatu. Tak działa metoda „setLiczba”.

W innych przypadkach, jeżeli zadeklarujemy zwracanie typu „int”, musimy w metodzie użyć słowa „return” wpisując za nim tę wartość. Tak działa metoda „getLiczba”.

Uwaga, słowo „return” kończy wykonywanie się metody. Instrukcje, które podamy po tym słowie, nie wykonają się.

Metody zaczynające się od fraz „set”, „get” oraz „is” są charakterystyczne, służą właśnie do obsługi pól.

Człowiek - najlepsza inwestycja


Przedrostki te zdefiniowane są w konwencji JavaBeans.

Nie ma potrzeby ich ręcznego pisania. Można skorzystać z kreatora w Eclipse:


„Source/Generate Getters and Setters...”

Zwłaszcza metoda z przedrostkiem „set” jest bardzo istotna. Daje nam kontrolę nad wartością, którą użytkownik będzie chciał przypisać do pola. Np. jeżeli dane pole będzie przechowywało wartość temperatury, nie może mieć wartości mniejszej niż -273. Przy próbie zapisania wartości -400 metoda „set” powinna wpisać -273.

Slajd
15



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```


public class MojZnak {

    public static String getNapis() {
        return "Napis";
    }

    private static String getInnyNapis() {
        return "InnyNapis";
    }

}

System.out.println(MojZnak.getNapis());
// to jest nieprawidłowe
// System.out.println(MojZnak.getInnyNapis());
        
```



Wstęp do programowania obiektowego

Metody, tak samo jak pola, mogą być statyczne. Oznacza to, że są wspólne dla wszystkich instancji danej klasy. Nie ma też potrzeby tworzenia klasy, jeżeli chcemy skorzystać z jej elementu statycznego.


W przykładzie mamy podane prawidłowe wywołanie metody statycznej. Zauważmy, że nie mamy żadnej referencji. Metody statyczne wywoływane są bezpośrednio dla nazwy klasy.


Dlatego właśnie aplikacje w Javie uruchamia się za pomocą metody „main”. Metoda ta jest dostępna zanim zostanie stworzona instancja danego obiektu.

Oczywiście nigdy nie wywołamy metody prywatnej z poziomu innego obiektu, tak jak pokazane to jest w komentowanym kodzie.

Człowiek - najlepsza inwestycja

Slajd
16


B2E
 BUSINESS TO EDUCATION


SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojZnak {


    private static int liczba;
    private static char znak;

    public static void setValues(int mojaLiczba) {
        liczba = mojaLiczba;
    }

    public static void setValues(int mojaLiczba, char mojZnak) {
        liczba = mojaLiczba;
        znak = mojZnak;
    }

}

MojZnak.setValues(10);
MojZnak.setValues(13, 'X');
```


 Wstęp do programowania obiektowego

W tym przypadku nie ma sensu tworzenie instancji danej klasy, wszystko jest statyczne. Nie utworzymy również dwóch różnych obiektów typu „MojZnak”. Możemy zapisać:

MojZnak z1 = new MojZnak();

MojZnak z2 = new MojZnak();

W praktyce jest to bez sensu. I tak nie przechowamy w obiektach różnych wartości.

Oczywiście tak, jak w przypadku konstruktorów, może być kilka metod o tej samej nazwie, różniących się listą argumentów.

Człowiek - najlepsza inwestycja





KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Slajd
17

JAVA
 dla początkujących

```

public class MojZnak {


    private static int liczba = 0;
    private char znak;

    public MojZnak(char znak) {
        this.znak = znak;
        liczba++;
    }

    public static int getLiczba() {
        return liczba;
    }

}

MojZnak z1 = new MojZnak('A');
MojZnak z2 = new MojZnak('B');
System.out.println(MojZnak.getLiczba());
        
```


 Wstęp do programowania obiektowego

Jakie jest więc może być praktyczne zastosowanie pól statycznych? Możemy stworzyć klasę „StaleMatematyczne” i stworzyć publiczne, statyczne pole o nazwie pi i wartości „3,14”. We wszystkich klasach programu będziemy mogli korzystać z tej wartości wpisując „StaleMatematyczne.pi”.

Nieco ciekawszy jest przykład podany na slajdzie. Pole „liczba” jest statyczne, co oznacza, że jest wspólne dla wszystkich instancji danej klasy.

W każdym wywołaniu konstruktora następuje inkrementacja jego wartości. W prosty sposób możemy sprawdzić ile obiektów typu „MojZnak” zostało stworzonych.

Człowiek - najlepsza inwestycja




KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Slajd
18



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class MojZnak {


    private char znak1;
    private char znak2;

    public void setZnak1(char znak) {
        znak1 = znak;
        checkZnak(znak);
    }

    public void setZnak2(char znak) {
        znak2 = znak;
        checkZnak(znak);
    }

    private void checkZnak(char znak) {
        if (znak == 'X') {
            System.out.println("Wstawiles X!");
        }
    }
}

MojZnak mojZnak = new MojZnak();
mojZnak.setZnak1('A');
mojZnak.setZnak2('X');
```



Wstęp do programowania obiektowego

Z prywatnych metod korzystamy tylko w obrębie danej klasy. Można stosować je aby nie pisać dwa razy tego samego kodu.

W przykładzie, w każdej metodzie „set” następuje sprawdzenie, czy ustawiana wartość jest równa „X”.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 101

Slajd
 19


JAVA

dla początkujących

Ćwiczenie:

Stwórz obiekt zawierający dwie liczby: a i b (dowolnego typu). Nie pozwól aby, któraś z nich była ujemna (jeżeli użytkownik pod ujemną liczbę, zamień ją na 0). Nie pisz dwa razy tego samego kodu.

Udostępnij użytkownikowi dwie metody: suma i iloczyn. Mają zwracać sumę i iloczyn tych liczb.

Klasę nazwij „DwieLiczby”.



Wstęp do programowania obiektowego

```
public class DwieLiczby {

    private int a;

    private int b;

    public DwieLiczby(int a, int b) {

        this.a = validate(a);

        this.b = validate(b);

    }

    public int suma() {
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



```
        return a+b;

    }

    public int iloczyn() {

        return a*b;

    }

    private int validate(int x) {

        return x<0 ? 0 : x;

    }

}

public class MojaPierwszaKlasa {

    public static void main(String[] args) {

        DwieLiczby dwieLiczby = new DwieLiczby(10, -1);

        System.out.println(dwieLiczby.suma());

    }

}
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd
 20


JAVA dla początkujących

Ćwiczenie:

Stwórz obiekt (o dowolnej nazwie) z dwoma metodami statycznymi: suma i iloczyn.

Niech wynikiem metody suma będzie suma wartości, które zostały zwrócone przez metody suma, dla obiektów DwieLiczby.

Metoda iloczyn ma działać analogicznie.

Pamiętaj, że argumentem metody może być nie tylko typ wbudowany. Poniższy zapis jest poprawny:

```
void metoda(Obiekt a) {...}
```

Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Wstęp do programowania obiektowego

```
public class InnaKlasa {
```

```
    public static int suma(DwieLiczby x1, DwieLiczby x2) {
```

```
        return x1.suma() + x2.suma();
```

```
    }
```

```
    public static int iloczyn(DwieLiczby x1, DwieLiczby x2) {
```

```
        return x1.iloczyn() * x2.iloczyn();
```

```
    }
```

```
}
```

Człowiek - najlepsza inwestycja




```
public class MojaPierwszaKlasa {  
  
    public static void main(String[] args) {  
  
        DwieLiczby dwieLiczby1 = new DwieLiczby(10, -1);  
  
        DwieLiczby dwieLiczby2 = new DwieLiczby(2, 3);  
  
        System.out.println(InnaKlasa.suma(dwieLiczby1, dwieLiczby2));  
  
        System.out.println(InnaKlasa.iloczyn(dwieLiczby1, dwieLiczby2));  
  
    }  
  
}
```

8.3.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli tworzyć proste obiekty w Javie, odpowiednio inicjalizować pola klas w trakcie ich tworzenia.

8.4 Lekcja 4 – Interfejsy, klasy abstrakcyjne i dziedziczenie

8.4.1 Cel lekcji

Celem lekcji jest wyjaśnienie pojęcia interfejsu i klasy abstrakcyjnej. Wytłumaczone będzie jak zwiększyć efektywność i bezpieczeństwo pracy poprzez stosowanie dziedziczenia.

Człowiek - najlepsza inwestycja











KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

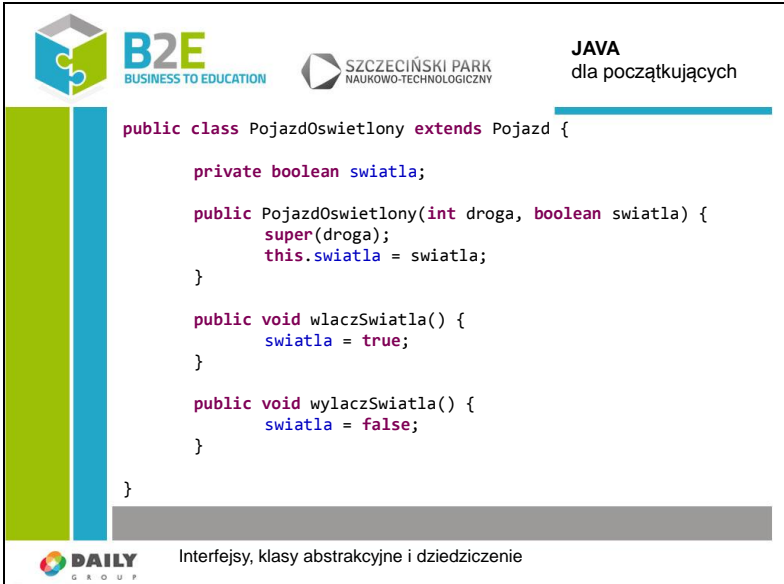
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY









8.4.2 Treść - slajdy z opisem

<p>Slajd 1</p>	  <div style="text-align: right;"> JAVA dla początkujących </div> <hr style="border: 1px solid #00AEEF; margin-top: 10px;"/> <h1 style="text-align: center;">Java - lekcja 4</h1> <p style="text-align: center;">Interfejsy, klasy abstrakcyjne i dziedziczenie</p> <div style="text-align: center; margin-top: 20px;">  <small>KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI</small> </div> <div style="text-align: center; margin-top: 5px;"> <small>Człowiek - najlepsza inwestycja</small> </div> <div style="text-align: center; margin-top: 5px;">  <small>UNIA EUROPEJSKA FUNDUSZ SPOŁECZNY</small> </div> <p style="text-align: center; font-size: small;">Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego</p> 	
<p>Slajd 2</p>	  <div style="text-align: right;"> JAVA dla początkujących </div> <hr style="border: 1px solid #00AEEF; margin-top: 10px;"/> <pre> public class Pojazd { private int droga; public Pojazd(int droga) { this.droga = droga; } public void doPrzodu(int odleglosc) { droga += odleglosc; } public void doTylu(int odleglosc) { droga -= odleglosc; } } </pre>  <div style="text-align: right;"> Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Klasa pokazana na przykładzie implementuj w prosty sposób niektóre właściwości pojazdu.</p> <p>Nowo stworzony pojazd ma licznik odległości ustawiony na zadaną wartość. Możemy poruszać się do</p>

Człowiek - najlepsza inwestycja

	<p>przodu i do tyłu.</p> <p>Może to być rower, motocykl , lub samochód.</p> <p>Zastanówmy się co możemy zrobić, jeżeli chcemy stworzyć klasę „PojazdOswietlony”, który posiada te same właściwości co „Pojazd”. Ponadto chcemy, aby „PojazdOswietlony”, miał metody „włączSwiatla” i „wylaczSwiatla”.</p> <p>Pisanie całości nowej klasy, przepisując część funkcjonalności z pojazdu jest niepotrzebne. W Javie (w innych językach również) nie powinno się pisać dwa razy tego fragmentu kodu. Możemy korzystać z mechanizmu dziedziczenia.</p>	
<p>Slajd 3</p>	 <pre> public class PojazdOswietlony extends Pojazd { private boolean swiatla; public PojazdOswietlony(int droga, boolean swiatla) { super(droga); this.swiatla = swiatla; } public void włączSwiatla() { swiatla = true; } public void wylaczSwiatla() { swiatla = false; } } </pre> <p>Interfejsy, klasy abstrakcyjne i dziedziczenie</p>	<p>Możemy stworzyć nową klasę która posiada wszystkie właściwości klasy „Pojazd”.</p> <p>Po nazwie klasy, używamy słowa „extends” i podajemy nazwę klasy, po której będziemy dziedziczyć.</p> <p>Klasa „Pojazd” posiadała tylko jeden konstruktor, z parametrem. Aby stworzyć obiekt „PojazdOswietlony” musimy najpierw wywołać konstruktor z klasy, którą dziedziczymy. Odpowiada za to instrukcja „super”. Jest to w pewnym sensie konstruktor klasy, z której dziedziczymy.</p>

Człowiek - najlepsza inwestycja

Slajd 4	<div>   <div> JAVA dla początkujących </div> </div> <pre> class MojaPierwszaKlasa { public static void main(String[] args) { PojazdOswietlony pojazdOswietlony = new PojazdOswietlony(0, false); pojazdOswietlony.wlaczSwiatla(); pojazdOswietlony.doPrzodu(10); } } </pre> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	Jak widać na slajdzie, na instancji obiektu „PojazdOswietlony” możemy wykonywać również metody z klasy „Pojazd”. W ten sposób uniknęliśmy ponownej implementacji niektórych funkcjonalności.
Slajd 5	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class Pojazd { private int droga; public Pojazd(int droga) { this.droga = droga; } public void doPrzodu(int odleglosc) { droga += odleglosc; } protected void zerojLicznik() { droga = 0; } } </pre> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Typ widoczności „protected” rozszerza domyślny typ widoczności. Metody i pola oznaczone w ten sposób widoczne są dla wszystkich klas znajdujących się w pakiecie.</p> <p>Rozszerzeniem jest to, iż pola i metody są widoczne dla wszystkich klas, które dziedziczą z danej klasy.</p>




Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



<p>Slajd 6</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class PojazdOswietlony extends Pojazd { private boolean swiatla; public PojazdOswietlony(int droga, boolean swiatla) { super(droga); this.swiatla = swiatla; } public void wlaczSwiatla() { swiatla = true; } public void zeroj() { zerojLicznik(); } } </pre> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Możemy skorzystać metody „zerojLicznik” z dwóch powodów.</p> <p>Po pierwsze, dziedziczymy po klasie „Pojzd”.</p> <p>Po drugie, klasa „PojazdOswietlony” znajduje się w tym samym pakiecie.</p>
<p>Slajd 7</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> class MojaPierwszaKlasa { public static void main(String[] args) { Pojazd pojazd = new Pojazd(0); pojazd.zerojLicznik(); PojazdOswietlony pojazdOswietlony = new PojazdOswietlony(0, false); pojazdOswietlony.zerojLicznik(); pojazdOswietlony.zeroj(); } } </pre> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Możemy skorzystać metod „zerojLicznik” tylko dlatego, że jesteśmy w tym samym pakiecie.</p> <p>W innych pakietach dostępna będzie tylko metoda „zeroj”.</p>

Człowiek - najlepsza inwestycja

Slajd
8JAVA
dla początkujących

Ćwiczenie:

Stwórz hierarchię klas dla następujących terminów: zwierzę, kot, pers, jaguar. Klasy nie muszą posiadać żadnych metod.









Interfejsy, klasy abstrakcyjne i dziedziczenie

```
public class Zwierze {};  
  
public class Kot extends Zwierze{};  
  
public class Pers extends Kot{};  
  
public class Jaguar extends Kot {};
```



Człowiek - najlepsza inwestycja

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCIUNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Slajd 9	<div>   <div> JAVA dla początkujących </div> </div> <div> <pre> ▲ pakiet1 │ ▸ ObiektA.java ▲ pakiet2 │ ▸ Main.java │ ▸ ObiektB.java package pakiet1; public class ObiektA { protected void pokazWiadomosc() { System.out.println("Wiadomość!"); } }</pre> </div> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Aby lepiej pokazać działanie typu widoczności stworzone zostały dwa pakiety.</p> <p>Klasa „ObiektA” znajduje się w osobnym pakiecie i zawiera metodę o właściwości „protected”.</p>
Slajd 10	<div>   <div> JAVA dla początkujących </div> </div> <div> <pre> package pakiet2; import pakiet1.ObiektA; public class ObiektB extends ObiektA { public void pokazOdziedziczonaWiadomosc() { pokazWiadomosc(); } }</pre> </div> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Pierwszy wers oznacza, że bieżąca klasa znajduje się w pakiecie „pakiet2”.</p> <p>„ObiektA” nie jest widoczny. Musi zostać zaimportowany z innego pakietu.</p> <p>„ObiektB” dziedziczy z „ObiektA”, więc ma dostęp do jego metody chronionej („protected”).</p>

Człowiek - najlepsza inwestycja

Slajd
11

JAVA
 dla początkujących

```


package pakiet2;

import pakiet1.ObiektA;

public class Main {

    public static void main(String[] args) {
        ObiektA obiektA = new ObiektA();
        // nie ma odpowiedniej metody dla obiektA

        ObiektB obiektB = new ObiektB();
        obiektB.pokazOdziedziczonaWiadomosc();
    }
}
        
```




 Interfejsy, klasy abstrakcyjne i dziedziczenie

Znajdujemy się cały czas w pakiecie „pakiet2”.

Nie mamy dostępu do metody chronionej obiektu „ObiektA”.

Możemy ją jednak wywołać korzystając z „ObiektB”.

Slajd
12

JAVA
 dla początkujących

```

import java.awt.Dimension;


class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Dimension dimension = new Dimension(2, 3);
        System.out.println(dimension.toString());

        System.out.println(new MojaPierwszaKlasa());
    }

    @Override
    public String toString() {
        return super.toString() + " - dziedziczę!";
    }
}
        
```

Wynik: java.awt.Dimension[width=2,height=3]
 MojaPierwszaKlasa@1e859c0 - dziedziczę!


 Interfejsy, klasy abstrakcyjne i dziedziczenie

Każda klasa w Javie dziedziczy po obiekcie „Object”, zawsze! „Object” posiada kilka metod i są one dostępne dla każdego stworzonego obiektu w Javie. Jedną z nich jest „toString”. Reprezentuje ona

Człowiek - najlepsza inwestycja

obiekt w postaci tekstu.

W metodzie „main” stworzony został obiekt typu „Dimension”. Obiekt ten przechowuje dwa podstawowe wymiary: wysokość i szerokość. Kiedy wywołamy na nim metodę „toString” otrzymamy tekstową reprezentację obiektu.

Klasa „MojPierwszaKlasa” zawiera już metodę „toString”. Chcę jednak aby zwracała wartość ustawioną przeze mnie. Wyrażenie „@Override” to adnotacja. Oznacza, że obiekt odziedziczył już taką metodę, ale chcę napisać ją we własny sposób. Używając słowa „super” wywołuję metodę „toString” dla obiektu, po którym dziedziczyłem.

Metoda „println” domyślnie wywołuje metodę „toString” dla każdego obiektu, dlatego ostatnia instrukcja metody „main” działa w ten sposób.

Człowiek - najlepsza inwestycja





KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Slajd
13

JAVA
 dla początkujących

```

Dimension dim1 = new Dimension(2, 3);
Dimension dim2 = new Dimension(2, 3);


if (dim1 == dim2) {
    System.out.println("Tak dla 1");
}

if (dim1.equals(dim2)) {
    System.out.println("Tak dla 2");
}

    public boolean equals(Object obj) {
        if (obj instanceof Dimension) {
            Dimension d = (Dimension)obj;
            return (width == d.width) && (height == d.height);
        }
        return false;
    }

```

Wynik:
Tak dla 2


 Interfejsy, klasy abstrakcyjne i dziedziczenie

Stworzone zostały dwa z pozoru takie same obiekty. Zastanówmy się jednak. Obiekty mają takie same dane, ale zostały utworzone w niezależny sposób.

W pierwszej instrukcji „if” nie sprawdzamy równości obiektów! Sprawdzamy równość referencji. Warunek byłby prawdziwy, jeżeli obie referencje wskazywałyby na ten sam obiekt, a tak nie jest.

Dla sprawdzenia równości obiektów wywołujemy na jednym z nich metodę „equals”. Metoda ta również jest dziedziczona po typie „Object”.

Jeżeli chcemy porównać dwie instancje stworzonej przez nas klasy, to pamiętajmy aby napisać dla niej własną wersję metody „equals”. Dla przykładu została podana Metoda equals zaimplementowana w klasie Dimension.

Człowiek - najlepsza inwestycja

Slajd
14



JAVA dla początkujących

Ćwiczenie:

Stwórz klasę „Animal” z metodami „eat” i „toString”. Niech „toString” zwraca napis „animal”. Stwórz klasy „Cat” i „Dog”. Dla tych klas niech metoda „toString” zwraca nazwę klasy.

Wywołaj „println” podając jako argument każdy z obiektów.



Interfejsy, klasy abstrakcyjne i dziedziczenie

```
public class Animal {

    public void eat() {System.out.println("eat");}

    @Override

    public String toString() {return "animal";}

}

public class Cat extends Animal {

    @Override

    public String toString() {return "cat";}

}
```




Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



	<pre> public class Dog extends Animal { @Override public String toString() {return "dog";} } class MojaPierwszaKlasa { public static void main(String[] args) { System.out.println(new Animal()); System.out.println(new Cat()); System.out.println(new Dog()); } } </pre>		
Slajd 15	<div>   <div> JAVA dla początkujących </div> </div> <div> <p>W Javie możemy dziedziczyć tylko po <u>jednej</u> klasie!</p> <p>Mam metody dla motorówki i dla samochodu. Jak stworzyć obiekt amfibia?</p> </div> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>		

Człowiek - najlepsza inwestycja

Slajd
16


 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public interface Boat {
    public void floatOnWater();
}

public interface Car {
    public void drive();
}
    
```







 **DAILY**
G R O U P
 Interfejsy, klasy abstrakcyjne i dziedziczenie

W tym celu stosuje się interfejsy. Interfejs zawiera jedynie metody bez ciał. Tak jak widzimy na slajdzie. Dodatkowo możemy wstawić listę argumentów.

Interfejs przedstawia listę operacji, które może wykonać dany obiekt.


W najnowszej wersji Javy 1.8 (8) zostały dodane metody domyślne do interfejsów. Są to pełnoprawne metody, które będą dostępne w klasach implementujących dany interfejs. Nie mniej jednak jest to temat bardziej zaawansowany więc tą informację należy potraktować jako zachętę do poszerzenia tematu we własnym zakresie.

Człowiek - najlepsza inwestycja


Slajd 17	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class Amphibian implements Boat, Car { @Override public void drive() {System.out.println("Jadę!");} @Override public void floatOnWater() {System.out.println("Płynę!");} } class MojaPierwszaKlasa { public static void main(String[] args) { Amphibian amphibian = new Amphibian(); amphibian.drive(); amphibian.floatOnWater(); } } </pre> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Interfejsów się nie dziedziczy. Interfejsy implementujemy, tak jak jest to pokazane w pierwszym wersie.</p> <p>Dopiero teraz możemy nadać ciałom metody.</p>
Slajd 18	<div>   <div> JAVA dla początkujących </div> </div> <pre> class MojaPierwszaKlasa { public static void main(String[] args) { Amphibian amphibian = new Amphibian(); printCar(amphibian); } private static void printCar(Car car) { System.out.println(car); } } </pre> <div>  Interfejsy, klasy abstrakcyjne i dziedziczenie </div>	<p>Bieżący przykład pokazuje jaki jest sens nadawania interfejsów obiektom.</p> <p>Metoda „printCar” przyjmuje jako argument obiekt typu „Car”. Obiekt „Amphibian” implementuje ten interfejs, dlatego wywołanie tej metody jest poprawne.</p>

Człowiek - najlepsza inwestycja

Slajd
 19



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public abstract class Ptak {


    private int waga;

    public int getWaga() {
        return waga;
    }

    public void setWaga(int waga) {
        this.waga = waga;
    }

    public abstract boolean isMiesozerny();

}
                
```


Interfejsy, klasy abstrakcyjne i dziedziczenie

Często spotykamy się z sytuacją gdy klasa bazowa zawiera wspólne pola i metody dla klas bardziej wyspecjalizowanych, ale zarazem jest zbyt ogólna aby móc sensownie wykorzystać instancje tej klasy. W powyższym przykładzie każdy Ptak ma swoją wagę, więc klasy pochodne np. „Wróbel” czy „Sokół” rozszerzając klasę bazową otrzymają jakby za darmo obsługę wagi. Niemniej jednak sam Ptak jest zbyt ogólny aby mógł reprezentować konkretny obiekt. W takim przypadku przychodzą nam z pomocą klasy abstrakcyjne. Za ich pomocą możemy zdefiniować wspólne składowe klasy (pole waga i metody), ukształtować wspólne API wszystkich klas wyspecjalizowanych (metoda isMiesozerny) oraz uniemożliwić stworzenie instancji obiektu tej klasy.

Aby stworzyć klasę abstrakcyjną należy użyć słowa „abstract” definiując klasę. Dodatkowo klasa może , ale nie musi zawierać metody abstrakcyjne. Metoda abstrakcyjna jest tylko definicją sygnatury metody i nie zawiera ciała metody. Metody abstrakcyjne mogą być zadeklarowane tylko w klasie abstrakcyjnej. Klasa dziedzicząca z klasy abstrakcyjnej musi implementować wszystkie abstrakcyjne metody klasy nadrzędnej, chyba że sama jest zadeklarowana jako abstrakcyjna. Nie mniej jednak pierwsze „konkretna” klasa w drzewie dziedziczenia będzie musiała zaimplementować wszystkie metody abstrakcyjne z klas w górze hierarchii dziedziczenia.

Klasie abstrakcyjnej jest trochę podobna do interfejsu, z tym że dodatkowo może zawierać pola i metody.

Człowiek - najlepsza inwestycja

Slajd
20


B2E
 BUSINESS TO EDUCATION


 SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

public class Bocian extends Ptak {
    public boolean isMiesozerny() {
        return true;
    }
}

// niepoprawne wywołanie
// Ptak ptak = new Ptak();
Ptak ptak = new Bocian();
Bocian bocian = new Bocian();
Object object = new Bocian();
ptak.setWaga(5);
bocian.setWaga(5);
// to już nie jest prawidłowe
// object.setWaga(5);
    
```


 Interfejsy, klasy abstrakcyjne i dziedziczenie

Dziedziczenie po klasie abstrakcyjnej odbywa się tak samo jak po zwykłej klasie. Nie można jednak stworzyć instancji klasy abstrakcyjnej, tak samo jak nie można stworzyć instancji interfejsu.

Na obiekt typu „Bocian” może wskazywać referencja typu: „Object”, „Bocian” lub „Ptak”. Należy pamiętać, że na referencji typu „Object” nie wywołamy metody „setWaga”.

Człowiek - najlepsza inwestycja

Slajd
21



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:

Stwórz interfejs „WiFi” z metodą „connect”.
Stwórz interfejs „Bluetooth” z metodą „connect”.
Sprawdź czy możesz stworzyć interfejs „Wireless” z taką metodą.
Może nie ma potrzeby pisania metody „connect” w dwóch pierwszych interfejsach?

Stwórz klasę abstrakcyjną „Phone”, niech posiada pole z numerem telefonu i metodę „call”.

Stwórz klasę „MobilePhone”, która dziedziczy po tych obiektach.

Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Interfejsy, klasy abstrakcyjne i dziedziczenie

```
public interface Wireless {  
    public void connect();  
}  
  
public interface WiFi extends Wireless {}  
  
public interface Bluetooth extends Wireless {}  
  
public abstract class Phone {  
    private long number = 123123123l;  
    public long getNumber() {return number;}  
}
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 121

```
public void setNumber(long number) {this.number = number;}

public void call() {}

}

public class MobilePhone extends Phone implements WiFi, Bluetooth {

    // ta metoda wystąpi tylko raz

    // Wniosek:

    // nie mamy osobnych metod dla WiFi i Bluetooth

    @Override

    public void connect() {}

}

class MojaPierwszaKlasa {

    public static void main(String[] args) {

        MobilePhone mobilePhone = new MobilePhone();

        mobilePhone.call();

        mobilePhone.connect();

    }

}
```

8.4.3 Opis założonych osiągnięć ucznia

Uczeń będzie umiał wykorzystać w praktyce interfejs i klasę abstrakcyjną, tworzyć nowe klasy wykorzystując wcześniej zaimplementowany kod.


Człowiek - najlepsza inwestycja

8.5 Lekcja 5 - Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

8.5.1 Cel lekcji



Celem lekcji jest wytłumaczenie pojęcia polimorfizmu, oraz jego praktycznego zastosowania w Javie. Pokazanie jak tworzyć klasy wewnętrzne, oraz efektywnie posługiwać się interfejsami.

8.5.2 Treść - slajdy z opisem

<p>Slajd 1</p>		
--------------------	---	--

Człowiek - najlepsza inwestycja

Slajd
2

JAVA
dla początkujących

```

public abstract class Ptak {


    private int waga;

    public int getWaga() {
        return waga;
    }

    public void setWaga(int waga) {
        this.waga = waga;
    }

    @Override
    public String toString() {
        return "Jestem Ptakiem!";
    }



}
    
```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Mamy prostą klasę "Ptak" z własną implementacją metody "toString".

Slajd
3


JAVA
dla początkujących

```

public interface MaImie {

    public String getName();






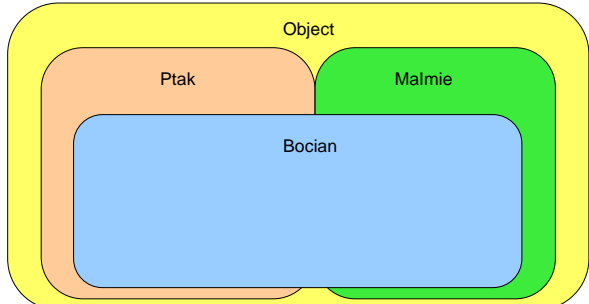

}
    
```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe



Istnieje również interfejs "MaImie" z metodą "getName".

Człowiek - najlepsza inwestycja

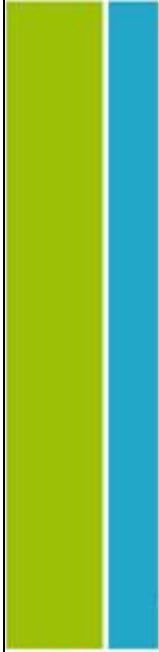
<p>Slajd 4</p>	<div>   </div> <div> <p>JAVA dla początkujących</p> <hr/> <pre> public class Bocian extends Ptak implements MaImie { private String imie = "Kajtek"; @Override public String getName() { return imie; } @Override public String toString() { return "Jestem bocianem o imieniu " + imie; } }</pre> </div> <div>  <p>Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe</p> </div>	<p>Stworzona jest klasa "Bocian", która rozszerza klasę "Ptak" i implementuje interfejs "MaImie".</p> <p>Oczywiście w przypadku implementacji interfejsu, musimy zaimplementować wszystkie metody, które do tego interfejsu należą.</p> <p>Bieżąca klasa ma własną wersję metody "toString".</p>
<p>Slajd 5</p>	<div>   </div> <div> <p>JAVA dla początkujących</p> <hr/>  </div> <div>  <p>Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe</p> </div>	

Człowiek - najlepsza inwestycja

Slajd
 6

JAVA
 dla początkujących




```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Bocian bocian = new Bocian();
        Ptak ptak = (Ptak) bocian;
        MaImie maImie = (MaImie) bocian;
        Object object = (Object) bocian;

        System.out.println(bocian);
        System.out.println(ptak);
        System.out.println(maImie);
        System.out.println(object);

        if (bocian == ptak) {
            System.out.println("Zachodzi równość!");
        }
    }
}
            
```


 Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

W wyniku działania programu dostaniemy cztery razy napis "Jestem bocianem o imieniu Kajtek". I napis "Zachodzi równość".

Jak widać możemy przypisać obiekt "bocian" do Referencji czterech różnych typów: "Bocian", "Ptak", "Malmie", "Object". Zjawisko to nazywamy polimorfizmem (inaczej wielopostaciowość).







Obiekt możemy traktować na różne sposoby w zależności od potrzeb.

Zastanówmy się dlaczego dla referencji typu "Ptak" nie pokazał się napis "Jestem ptakiem!"?

Odpowiedź może zasugerować nam porównanie. Pamiętajmy, że nie sprawdzamy równości obiektów, na które wskazują referencje! Sprawdzamy, czy wskazują na ten sam obiekt. Wszystkie referencje w metodzie "main" wskazują na obiekt stworzony przez konstruktor "Bocian()".

To, że metoda "toString" z klasy "Ptak" zwraca napis "Jestem ptakiem!", nie ma żadnego znaczenia. W klasie "Bocian" mamy zupełnie inną implementację tej metody. Klasa nie może posiadać dwóch metod o takiej samej nazwie i takiej samej liście argumentów.

Człowiek - najlepsza inwestycja

<p>Slajd 7</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class PolskiBocian extends Bocian { private String imie = "Polski bocian Bonifacy."; public static void main(String[] args) { Ptak ptak = new PolskiBocian(); System.out.println(ptak.toString()); } } </pre> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Proszę zgadnąć, co wyświetli powyższy program.</p> <p>Wynik: Jestem bocianem o imieniu Kajtek</p> <p>„Nadpisanie” pola imie nie skutkuje zmianą tekstu wypisywanego przez program.</p> <p>Przykład ten demonstruje, iż polimorfizm dotyczy tylko metod i nie ma analogicznego wpływu na pola.</p>
<p>Slajd 8</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> class MojaPierwszaKlasa { public static void main(String[] args) { Bocian bocian = new Bocian(); if (bocian instanceof Ptak) { System.out.println("Bocian jest ptakiem!"); } } } </pre> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Aby móc rozpoznawać typ obiektu możemy skorzystać z operatora "instanceof".</p> <p>W powyższym przykładzie widzimy jego użycie i jak można się domyślić wynikiem działania operatora jest wartość logiczna.</p>

Człowiek - najlepsza inwestycja

Slajd
 9

B2E
 BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA dla początkujących

Ćwiczenie:

Stwórz dwie klasy:

Liczba, niech przechowuje dowolną liczbę i posiada metodę „podajWartosc”.

Napis, niech przechowuje dowolny tekst i posiada metodę „podajTekst”.

W dowolnym obiekcie, który posiada metodę „main” stwórz metodę „wyswietl”, która będzie przyjmowała jako argument obiekt typu „Object”. W przypadku „Liczba” wywołaj metodę „podajWartosc” i analogicznie w przypadku „Napis”.



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

```
public class Liczba {

    private int liczba = 123;

    public int podajWartosc() {return liczba;}

}

public class Napis {

    private String napis = "Bla, bla, bla...";

    public String podajTekst() {return napis;}

}

class MojaPierwszaKlasa {
```

Człowiek - najlepsza inwestycja


KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

 UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



```
public static void main(String[] args) {  
  
    wyswietl(new Liczba());  
  
    wyswietl(new Napis());  
  
    wyswietl(new Object());  
  
}  
  
private static void wyswietl(Object object) {  
  
    if (object instanceof Liczba) {  
  
        System.out.println(((Liczba)object).podajWartosc());  
  
    }  
  
    else if (object instanceof Napis) {  
  
        System.out.println(((Napis)object).podajTekst());  
  
    }  
  
    else {  
  
        System.out.println("Nie rozpoznano obiektu!");  
  
    }  
  
}  
  
}
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd
 10

B2E
 BUSINESS TO EDUCATION

SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Ćwiczenie:

Stwórz klasę „Temperatura” z metodą „podajTemperature”. Wykorzystaj klasę „Liczba” z poprzedniego ćwiczenia do przechowywania wartości poprzez mechanizm dziedziczenia. Popraw metodę „wyswietl”, aby poprawnie wywoływała metodę w zależności od podanego typu obiektu.



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

```
public class Temperatura extends Liczba {

    private char jednostka = 'C';

    public String podajTemperature() {

        return Integer.toString(podajWartosc())

            + Character.toString(jednostka);

    }

}

class MojaPierwszaKlasa {

    public static void main(String[] args) {
```







Człowiek - najlepsza inwestycja


KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI







UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY


```
wyswietl(new Liczba());  
  
wyswietl(new Napis());  
  
wyswietl(new Object());  
  
wyswietl(new Temperatura());  
  
}  
  
private static void wyswietl(Object object) {  
  
    if (object instanceof Liczba) {  
  
        if (object instanceof Temperatura) {  
  
            System.out.println(((Temperatura)object).podajTemperature());  
  
        }  
  
        else {  
  
            System.out.println(((Liczba)object).podajWartosc());  
  
        }  
  
    }  
  
    else if (object instanceof Napis) {  
  
        System.out.println(((Napis)object).podajTekst());  
  
    }  
  
    else {  
  
        System.out.println("Nie rozpoznano obiektu!");  
  
    }  
  
}  
  
}
```

Człowiek - najlepsza inwestycja

<p>Slajd 11</p>	<div>   <div> JAVA dla początkujących </div> </div> <p>Ćwiczenie:</p> <p>Stwórz taką klasę:</p> <pre>public class BocianBialy extends Bocian { }</pre> <p>Następnie zmodyfikuj klasę „Bocian” dodając słowo „final” jak w przykładzie poniżej.</p> <pre>public final class Bocian extends Ptak implements MaImie {</pre> <p>Jaki problem wystąpił w aplikacji?</p> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Słowo "final" użyte w definicji klasy oznacza, że nie możemy po niej dziedziczyć.</p>
<p>Slajd 12</p>	<div>   <div> JAVA dla początkujących </div> </div> <p>Ćwiczenie:</p> <p>Dodaj „final” do metody „toString” w klasie „Bocian”. Czy „BocianBialy” może posiadać własną implementację tej metody?</p> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Pisanie własnych wersji metod nazywamy przesłanianiem. Nie można przesłonić metody oznaczonej jako final. Jest to ostateczna implementacja.</p>

Człowiek - najlepsza inwestycja

Slajd 13	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class Drzewo { private String gatunek = "Klon"; public class Lisc { @Override public String toString() { return "Jestem liściem drzewa " + gatunek; } } public Lisc getLisc() { return new Lisc(); } } </pre> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Java umożliwia tworzenie klas wewnętrznych. Klasa wewnętrzna ma dostęp do elementów prywatnych klasy, w której się znajduje.</p> <p>Klasa wewnętrzna może posiadać typy widoczności: prywatny, chroniony, publiczny i domyślny (pakietowy).</p>
Slajd 14	<div>   <div> JAVA dla początkujących </div> </div> <pre> class MojaPierwszaKlasa { public static void main(String[] args) { Drzewo drzewo = new Drzewo(); Drzewo.Lisc lisc = drzewo.new Lisc(); System.out.println(lisc); } } </pre> <p>Wynik:</p> <p>Jestem liściem drzewa Klon</p> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Tak wygląda prawidłowe wywołanie klasy wewnętrznej. Zwróćmy uwagę jak wygląda użycie operatora "new" dla obiektu "Lisc".</p> <p>Dlaczego nie można stworzyć obiektu typu "Lisc" bez użycia obiektu "Drzewo"?</p> <p>Obiekt typu "Lisc" może mieć dostęp do pól obiektu typu "Drzewo". Dlatego, aby stworzyć "lisc", musimy stworzyć najpierw obiekt "drzewo".</p>

Człowiek - najlepsza inwestycja









KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY




Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 133


Slajd 15	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class Miasto { public class Ulica { public class Dom { @Override public String toString() { return Ulica.this.toString() + " Dom"; } } @Override public String toString() { return Miasto.this.toString() + " Ulica"; } } @Override public String toString() { return "Miasto"; } } </pre> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Widoczny przypadek jest nieco bardziej skomplikowany. Jak widać każda z klas może mieć własną metodę "toString".</p> <p>Zwróćmy uwagę, jak następuje wywołanie metody z klas zewnętrznych.</p>
Slajd 16	<div>   <div> JAVA dla początkujących </div> </div> <pre> Miasto miasto = new Miasto(); Miasto.Ulica ulica = miasto.new Ulica(); Miasto.Ulica.Dom dom = ulica.new Dom(); System.out.println(dom); </pre> <p>Wynik:</p> <p>Miasto Ulica Dom</p> <div>  Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe </div>	<p>Obiekty tworzymy dokładnie tak, jak w poprzednim przykładzie.</p> <p>Aby zbudować wynik, wywołane zostały metody z wszystkich trzech obiektów.</p>

Człowiek - najlepsza inwestycja

Slajd
17



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public abstract class Ptak {

    private int waga;

    public int getWaga() {
        return waga;
    }

    public void setWaga(int waga) {
        this.waga = waga;
    }

    @Override
    public String toString() {
        return "Jestem Ptakiem!";
    }
}


```

```

public interface MaImie {

    public String getName();
}

```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

Skorzystajmy z klasy abstrakcyjnej i interfejsu, których używaliśmy wcześniej.

W poprzedniej lekcji podane było, że nie możemy stworzyć instancji danej klasy abstrakcyjnej lub interfejsu. Od tej reguły nie ma wyjątku, ale nie musimy deklarować wprost nowego typu klasy aby stworzyć instancję obiektu dziedziczącego z klasy abstrakcyjnej. W praktyce możemy stworzyć klasę anonimową, co zostanie pokazane na następnym slajdzie. Podobnie możemy postąpić z interfejsem.

Człowiek - najlepsza inwestycja

Slajd
 18



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        Ptak ptak = new Ptak() {
        };
        ptak.setWaga(10);

        MaImie maImie = new MaImie() {
            @Override
            public String getName() {
                return "Moje imie";
            }
        };
        System.out.println(maImie.getName());
    }
}
                
```



Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe


"Ptak" jest klasą abstrakcyjną. W nawiasach klamrowych możemy dodać dodatkowe metody i pola do klasy. W ten sposób stworzony został obiekt anonimowy. Nie możemy odwołać się do klasy, której obiekt jest instancją. Klasa ta nie ma nazwy, ale wiemy, że zgodnie z polimorfizmem, jest typu "Ptak".


W przypadku interfejsu sprawa jest nieco bardziej skomplikowana. Nie ma on ciała swoich metod, musimy więc je zaimplementować.

Klasy anonimowe są często wykorzystywane gdy istnieje potrzeba jednorazowego wykorzystania ich funkcjonalności. Interfejs graficzny użytkownika oraz obsługa zdarzeń zostaną omówione w późniejszej lekcji. Niemniej jednak jest to dobry przykład wykorzystania klas anonimowych. Powiedzmy że mamy dwa przyciski „Ok.” i „Cancel” obydwa z nich reagują na kliknięcie, ale ich zachowanie jest zupełnie odmienne. Wobec czego interfejs obsługujący kliknięcie jest ten sam (metoda „onClick”), ale implementacja będzie różna. Dodatkowo nie ma innych elementów GUI które powinny obsłużyć zdarzenie dokładnie w ten sam sposób, więc nie ma sensu tworzenia nowych klas np. OkClickListener i CancelClickListener. Dlatego lepiej jest użyć dwóch klas anonimowych, aby implementując ten sam interfejs wykonały różne zadania.

Człowiek - najlepsza inwestycja

Slajd
19

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących


Ćwiczenie:

Korzystając z klasy anonimowej stworzonej z interfejsu zaprojektuj mechanizm sprawdzający ile czasu trwa wykonanie dowolnej metody.
Czas w postaci long (milisekundy) zwróci Ci wywołanie:


```
System.currentTimeMillis();
```

Człowiek - najlepsza inwestycja

 **KAPITAŁ LUDZKI**
NARODOWA STRATEGIA SPÓJNOŚCI

 **UNIA EUROPEJSKA**
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

 **DAILY**
G R O U P

Polimorfizm, instrukcja „final”, klasy wewnętrzne i anonimowe

```
public interface CustomOperation {
```

```
    public void execute();
```

```
}
```

```
class MojaPierwszaKlasa {
```

```
    public static void main(String[] args) {
```

```
        System.out.println(checkTime(new CustomOperation() {
```

Człowiek - najlepsza inwestycja

	<pre>@Override public void execute() { myOperation(); } }); } private static long checkTime(CustomOperation customOperation) { long l = System.currentTimeMillis(); customOperation.execute(); return System.currentTimeMillis() - l; } private static void myOperation() { int i = 1; while (i>0) { i++; } } }</pre>
--	---

8.5.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą wiedzieć jak wykorzystuje się w praktyce polimorfizm i interfejsy.


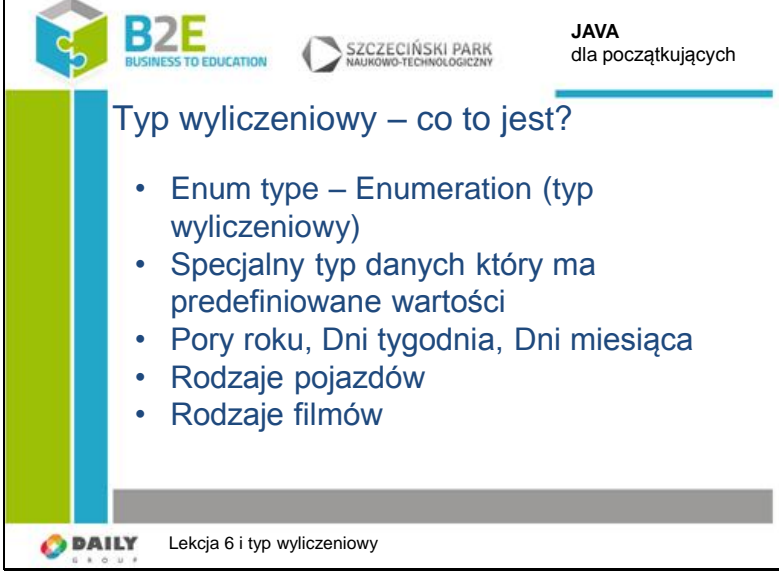
Człowiek - najlepsza inwestycja

8.6 Lekcja 6 - Typ wyliczeniowy, JavaDocs







8.6.1 Cel lekcji

Celem lekcji jest zapoznanie uczniów z typem wyliczeniowym i prostą metodą dokumentowania kodu za pomocą JavaDocs.

8.6.2 Treść - slajdy z opisem



<p>Slajd 1</p>		<p>Na lekcji poznamy co to jest typ wyliczeniowy, do czego znajduje zastosowanie i jak można go wykorzystać w programowaniu.</p> <p>Poznamy narzędzia do dokumentowania kodu.</p>
<p>Slajd 2</p>		<p>Typ wyliczeniowym jest specjalny typ danych który może przyjmować jedną z zadeklarowanych stałych.</p> <p>Przykładem może być pora roku. Załóżmy, że chcemy, aby zmienna poraRoku przyjmowała tylko ustalone przez nas wartości tj. wiosna, lato, jesień i zima.</p>

Człowiek - najlepsza inwestycja

Slajd 3	<div>   <div> JAVA dla początkujących </div> </div> <h2>Typ wyliczeniowy enum</h2> <pre>public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }</pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	<p>Przygotujmy nową uruchamialną klasę i zadeklarujmy następujący typ zmiennej:</p> <p>PoryRoku, który będzie przyjmować następujące wartości:</p> <pre>public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }</pre>
Slajd 4	<div>   <div> JAVA dla początkujących </div> </div> <h2>Typ wyliczeniowy enum</h2> <pre>public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA } public static void main(String[] args) { // TODO Auto-generated method stub PoryRoku poryRoku = PoryRoku.JESIEN; }</pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	<p>Przygotujmy nową uruchamialną klasę i zadeklarujmy następujący typ zmiennej: PoryRoku, który będzie przyjmować następujące wartości:</p> <pre>public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }</pre> <p>W metodzie main zadeklarujmy zmienną typu PoryRoku i przypiszmy jej wartość.</p> <p>Zauważmy, że wpisując wartość możemy posługiwać się zadeklarowanym typem PoryRoku.</p>

Człowiek - najlepsza inwestycja

Slajd
5

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Typ wyliczeniowy enum

```

public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }

public static void main(String[] args) {
    PoryRoku.
}
    
```

class : Class<Ptak.PoryRoku>

JESIEN : Ptak.PoryRoku - Ptak.PoryRoku

LATO : Ptak.PoryRoku - Ptak.PoryRoku


WIOSNA : Ptak.PoryRoku - Ptak.PoryRoku

ZIMA : Ptak.PoryRoku - Ptak.PoryRoku

valueOf(String arg0) : PoryRoku - PoryRoku

values() : PoryRoku[] - PoryRoku



valueOf(Class<T> enumType, String name) : T - Enum



Lekcja 6 i typ wyliczeniowy

Zauważmy, że wpisując wartość możemy posługiwać się zadeklarowanym typem PoryRoku. Eclipse podpowiada nam wartości mechanizmem intellisense.

Slajd
6





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Typ wyliczeniowy enum - zadanie

- Proszę napisać klasę która na podstawie przekazanej konstruktorowi pory roku ustali czy lubisz porę roku.
- Metoda czyLubie() niech wydrukuje / wyświetli informację czy zadana pora roku jest lubiana czy nie.



Lekcja 6 i typ wyliczeniowy


Proszę wykorzystać instrukcję switch/case. Dla uproszczenia klasę można zadeklarować jak zagnieżdżoną klasę programu zamiast nowego pliku dla klasy.

Inicjacja obiektu powinna następować wg przykładu:


LubiePoryRoku w =
LubiePoryRoku(PoryRoku.JESIEN);

Człowiek - najlepsza inwestycja

Slajd
 7



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących


Typ wyliczeniowy enum - rozwiązanie

```

public class LubiePoryRoku {
    PoryRoku poryRoku;

    LubiePoryRoku(PoryRoku p) {
        this.poryRoku = p;
    }

    public void czyLubie(){
        System.out.print("Cześć czy lubię porę roku " +
            poryRoku.toString()+"? ");
        switch (poryRoku) {
            case WIOSNA: System.out.println("Lubię wiosnę");break;
            case LATO: System.out.println("Bardzo lubię lato");break;
            case JESIEN: System.out.println("eech tak sobie");break;
            default: System.out.println("był padał śnieg");break;}
    }
}
```



Lekcja 6 i typ wyliczeniowy

Klasa nazywa się LubiePoryRoku

Jej konstruktor zapamiętuje zadany przy tworzeniu obiektu klasy zmienną typu PoraRoku w polu klasy.



W metodzie czyLubie zastosowano instrukcję switch / case.

Proszę zauważyć, jak łatwo można wykorzystać typ wyliczeniowy do obsługi poszczególnych przypadków tego typu.

Dodatkowo, proszę zwrócić uwagę na konwersję wartości nazw stałych w instrukcji wyświetlającej napis „Cześć czy lubię porę roku "+poryRoku.toString()+"? „

Człowiek - najlepsza inwestycja

Slajd
8

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Typ wyliczeniowy enum - rozwiązanie

```

public static void main(String[] args) {
    // TODO Auto-generated method stub


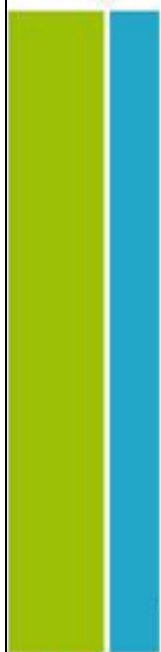

    LubiePoryRoku w = new LubiePoryRoku(PoryRoku.ZIMA);
    w.czyLubie();
}
    
```

 **DAILY**
G R O U P
 Lekcja 6 i typ wyliczeniowy

Wywołanie klasy zadeklarowanej jako klasę wewnętrzną można wykonać w zaprezentowany sposób.

Proszę zauważyć, że konstruktor wołany jest ze stałą ustawianą typem PoryRoku.

Człowiek - najlepsza inwestycja

Slajd
9 **B2E**
BUSINESS TO EDUCATION  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY **JAVA**
dla początkujących

JavaDocs – dokumentacja kodu

- Czy dokumentacja jest potrzebna?
- Jak dokumentować?
- JavaDocs – narzędzie do dokumentowania
- Korzyści



Lekcja 6 i typ wyliczeniowy

Czy dokumentacja kodu jest potrzebna?

Odp: Jest potrzebna. Wyobraźmy sobie że piszemy kod od dłuższego czasu. Powrót do napisanego kodu sprzed pół roku może okazać się trudny.

Krótki komentarz, opis zaraz przybliży nam do czego służy metoda, kto jaką wersję klasy i kiedy napisał

Jak dokumentować?

Oczywiście nie chodzi o pisanie wielostronicowych elaboratów. Opis ma być krótki, zrozumiały, zawierać tylko to co chcielibyśmy wiedzieć wykorzystując np. klasę. Najlepiej wg określonego szablonu

W lekcji pierwszej omówiliśmy komentarze w kodzie źródłowym:

```
// do końca linii
```

```
/* */ wieloliniowy
```

Człowiek - najlepsza inwestycja

**KAPITAŁ LUDZKI**
NARODOWA STRATEGIA SPÓJNOŚCI**UNIA EUROPEJSKA**
EUROPEJSKI
FUNDUSZ SPOŁECZNY

	<p>Ich zadaniem jest dokumentacja danego fragmentu kodu, ułatwienie zrozumienia algorytmu. Odbiorcami są developerzy czytający kod lub go modyfikujący.</p> <p>Zachodzi jednak czasem potrzeba udokumentowania API biblioteki, programu który stworzyliśmy. Często użytkownik, również developer może nie mieć dostępu do kodu źródłowego, lub po prostu chce użyć naszych narzędzi bez zagłębiania się w szczegóły. Do tworzenia tego rodzaju dokumentacji używa się JavaDocs.</p> <p>JavaDocs jest specjalnym rodzajem komentarza umieszczanym w kodzie źródłowym. Pozwala on na wygenerowanie dokumentacji technicznej naszego kodu. Dzięki tego rodzaju komentarzom czytelnik dokumentacji może dowiedzieć się jakie klasy, metody i pola są dostępne, opis każdego z nich. Ponieważ jest to ogólnoprzyjęty sposób dokumentacji środowiska dewelopersie wspierają go. I tak np. Eclipse pomaga nam w tworzeniu JavaDoc w kodzie, w popup'ach wyświetla informacje po najechaniu na interesujący element. Źródłem tych informacji może być nasz własny udokumentowany kod lub dokumentacja osobno ściągnięta do używanej biblioteki (nawet bez konieczności posiadania kodu źródłowego biblioteki).</p>
Slajd 10	<div data-bbox="279 981 1066 1563">  </div> <div data-bbox="1189 981 1511 1377"> <p>Wystarczy że najedziemy na metodę (tutaj println) i otrzymujemy szybką informację, jak dana metoda działa, jakie ma parametry.</p> <p>Dzięki temu nie musimy szukać informacji w dokumentacji.</p> </div>






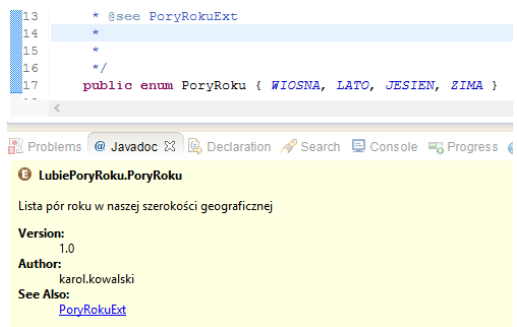

Człowiek - najlepsza inwestycja

<p>Slajd 11</p>	<p>JavaDocs – dokumentacja kodu</p> <p>public static void main(String[] args) { LubiePoryRoku lpr = new LubiePoryRoku(); }</p> <p>LubiePoryRoku</p> <p>Press 'F2' for focus</p> <p>@author karol.kowalski</p> <p>Seal</p> <p>DAILY GROUP Lekcja 6 i typ wyliczeniowy</p>	<p>A oto nasz ostatni kod (klasa LubiePoryRoku).</p> <p>Po najechnaniu na nazwę klasy nie wyświetlają się żadne podręczne informacje.</p>
<p>Slajd 12</p>	<p>JavaDocs – jak dokumentować</p> <ul style="list-style-type: none"> • Komentarz • Standard <pre>/** * Funkcja obliczająca kwadrat liczby * @author karol.kowalski * @version 1.1 */</pre> <ul style="list-style-type: none"> • Predefiniowane tagi @ <p>DAILY GROUP Lekcja 6 i typ wyliczeniowy</p>	<p>Dokumentacja kodu jest wykonywana w formie komentarza przed typem zmiennej, klasą, metodą.</p> <p>Aby odróżnić dokumentację od zwykłego komentarza stosuje się znaczniki</p> <pre>/** */</pre> <p>Ustalono standard opisu i tagi.</p>








Człowiek - najlepsza inwestycja

<p>Slajd 13</p>	<div>   </div> <div> JAVA dla początkujących </div> <hr/> <h2>JavaDocs – przykład</h2> <pre> /** * Lista pór roku w naszej szerokości geograficznej * * * @author karol.kowalski * @version 1.0 * @see PoryRokuExt * */ public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }</pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	<p>Proszę przyjrzeć się przykładowi, który dokumentuje ostatnio zdefiniowany typ wyliczeniowy PoryRoku.</p> <p>Dzięki dołączonej dokumentacji każde użycie typu PoryRoku powoduje podręczne wyświetlenie informacji.</p>
<p>Slajd 14</p>	<div>   </div> <div> JAVA dla początkujących </div> <hr/> <h2>JavaDocs – przykład</h2> <pre> /** * * @author manfred.poznanski * */ public { @author - author name @author @category @deprecated @see @serial @since @version (@code) (@docRoot) (@linkplain) (@link) (@literal) (@value) @ejb.bean @ejb.ejb-external-ref }</pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	<p>Narzędzie Eclipse wspiera przy pisaniu dokumentacji – lista dostępnych tagów.</p>






Człowiek - najlepsza inwestycja

Slajd 15	<div>   <div> JAVA dla początkujących </div> </div> <h2>JavaDocs – zasady</h2> <pre> /** * Krótki opis klasy LubiePoryRoku * <p> * Użyj {@link #LubiePoryRoku(PoryRoku)} aby ustawić porę roku. * * @param PoryRoku zmienna typu PoryRoku * @return konstruktor nic nie zwraca * * @author manfred.poznanski * @version 1.1 * @since 2013-02-02 */ </pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	<p>Przyjęto, że pierwszą linią dokumentującą będzie krótki opis, co dany element wykonuje.</p> <p>Następnie można dodać drugą linię (po znaczniku nowej linii <p>) zawierającą opis metody i jej parametrów, oraz co zwraca metoda.</p> <p>Potem, zależnie od uznania, tagi dot. autora, wersji daty wprowadzenia.</p>
Slajd 16	<div>   <div> JAVA dla początkujących </div> </div> <h2>JavaDocs – podgląd</h2> <div> <pre> 13 * @see PoryRokuExt 14 * 15 * 16 */ 17 public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA } </pre>  </div> <div>  Lekcja 6 i typ wyliczeniowy </div>	<p>Podgląd pisanej dokumentacji można śledzić na zakładce Javadoc.</p>

Człowiek - najlepsza inwestycja

<p>Slajd 17</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY </div> <div> JAVA dla początkujących </div> <hr/> <h3>JavaDocs – przykład</h3> <pre>public static void main(String[] args) { PoryRoku a = null; }</pre> <div>  LubiePoryRoku.PoryRoku Lista pór roku w naszej szerokości geograficznej Version: 1.0 Author: karol.kowalski See Also: PoryRokuExt </div> <div>  DAILY G R O U P Lekcja 6 i typ wyliczeniowy </div>	<p>Oto widok po najechniu na nazwę typu PoryRoku.</p>
<p>Slajd 18</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY </div> <div> JAVA dla początkujących </div> <hr/> <h3>JavaDocs – zadanie</h3> <ul style="list-style-type: none"> • Proszę napisać dokumentację do ostatnio dodanej klasy LubiePoryRoku • Proszę opisać konstruktor • Oraz metodę czyLubie() • Proszę sprawdzić czy przy pisaniu oprogramowania wyświetla się dokumentacja <div>  DAILY G R O U P Lekcja 6 i typ wyliczeniowy </div>	<p>Oto widok po najechniu na nazwę typu PoryRoku.</p>

Człowiek - najlepsza inwestycja

Slajd 19	<div>   <div> JAVA dla początkujących </div> </div> <h2>JavaDocs – rozwiązanie zadania</h2> <pre> /** * Klasa LubiePoryRoku * <p> * Implementuje zaawansowany algorytm ekspercki do ustalania * preferencji pór roku użytkownika.<p> * Konstruktor {@link #LubiePoryRoku()} * ustawia wybraną porę roku.<p> * Metoda {@link #czyLubie()} zwraca preferencje * * @author manfred.poznanski * @version 1.1 * @since 2013-02-02 */ public class LubiePoryRoku </pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	Oto przykładowa dokumentacja klasy. Proszę zwrócić uwagę na linki do poszczególnych metod klasy.
Slajd 20	<div>   <div> JAVA dla początkujących </div> </div> <h2>JavaDocs – rozwiązanie zadania</h2> <pre> public static void main(String[] args) { LubiePoryRoku lpr = new LubiePoryRoku(PoryRoku.JESIEN); } </pre> <div>  Lekcja 6 i typ wyliczeniowy </div>	Oto przykładowa dokumentacja klasy. Proszę zwrócić uwagę na linki do poszczególnych metod klasy.

Człowiek - najlepsza inwestycja

Slajd
21

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

JavaDocs – rozwiązanie zadania

```

/**
 * Krótki opis konstruktora LubiePoryRoku
 * <p>
 * Użyj {@link #LubiePoryRoku(PoryRoku)}
 * aby ustawić porę roku.
 *
 * @param PoryRoku zmienna typu PoryRoku
 * @return konstruktor nic nie zwraca
 *
 * @author manfred.poznanski
 * @version 1.1
 * @since 2013-02-02
 */
LubiePoryRoku(PoryRoku p) {
    
```

 **DAILY**
G R O U P
 Lekcja 6 i typ wyliczeniowy

Slajd
22

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

JavaDocs – rozwiązanie zadania

```

/**
 * Metoda wyświetlająca preferencje pór roku użytkownika
 * <p>
 * Metoda jest bezparametrowa (wykorzystuje pole klasy
 * PoryRoku
 *
 * @return przesyła na standardowe urządzenie wyjściowe
 * (console) raport z preferencjami
 *
 * @author manfred.poznanski
 * @version 1.1
 * @since 2013-02-02
 */
public void czyLubie(){
    
```

 **DAILY**
G R O U P
 Lekcja 6 i typ wyliczeniowy

Człowiek - najlepsza inwestycja

Slajd
23

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

JavaDocs – rozwiązanie zadania

```

20 * Klasa LubiePoryRoku
14 public class LubiePoryRoku {
15
16     private PoryRoku poraRoku;
17
18     * Lista pór roku w naszej szerokości geograficznej
28 public enum PoryRoku { WIOSNA, LATO, JESIEN, ZIMA }
29
30 * Krótki opis konstruktora LubiePoryRoku
31 public LubiePoryRoku(PoryRoku poraRoku) {
44     this.poraRoku = poraRoku;
45 }
46
47
48 public static void main(String[] args) {
49     LubiePoryRoku lpr = new LubiePoryRoku(PoryRoku.JESIEN);
50 }
51
52 }
    
```

DAILY Lekcja 6 i typ wyliczeniowy

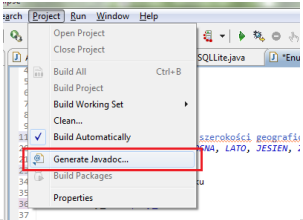
Dokumentacja nie musi zaśmieszać kodu – można ją zwinąć do jednej linii, żeby nie przeszkadzała.

Slajd
24

B2E BUSINESS TO EDUCATION **SZCZECIŃSKI PARK** NAUKOWO-TECHNOLOGICZNY **JAVA** dla początkujących

JavaDocs – dokumentacja html

- Cały projekt wraz z dokumentacją można zapisać w HTML-u
- W tym celu należy wybrać opcję „Generate Javadoc”





DAILY Lekcja 6 i typ wyliczeniowy

Po uruchomieniu kreatora, wybrać folder docelowy, gdzie będzie zapisana strona z dokumentacją (Destination) i nacisnąć Finish.

Człowiek - najlepsza inwestycja

Slajd
25

JAVA
dla początkujących

JavaDocs – dokumentacja html

All Classes

- EnumTest
- EnumTest.Pory_roku

Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Enum Constants | Field | Method Detail: Enum Constants | Field

Enum EnumTest.Pory_roku

java.lang.Object
java.lang.Enum<EnumTest.Pory_roku>
EnumTest.Pory_roku

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable<EnumTest.Pory_roku>

Enclosing class:
EnumTest


```
public static enum EnumTest.Pory_roku
extends java.lang.Enum<EnumTest.Pory_roku>
```

Lista pór roku w naszej szerokości geograficznej

Version:
1.0

Author:
karol.kowalski

See Also:
Pory_roku_ext





Lekcja 6 i typ wyliczeniowy

Oto rezultat generowania.

Automatycznie powstaje dokument HTML który przedstawia hierarchię klas, dla każdej klasy opis, listę metod, drzewo relacji i dokumentację, którą wpisaliśmy.

Slajd
26






JAVA
dla początkujących


Pytania

1. Co to jest typ wyliczeniowy?
2. Do czego stosujemy typ enum?
3. Jakie zalety w programowaniu ma typ enum?
4. Co to jest JavaDoc?
5. Jak należy dokumentować kod w JavaDoc?
6. Dlaczego dokumentowanie jest potrzebne?

Człowiek - najlepsza inwestycja

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Lekcja 6 i typ wyliczeniowy

Człowiek - najlepsza inwestycja

8.6.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli poznawać informacje na temat wykorzystywanych obiektów i ich metod z dokumentacji. W łatwy sposób operować typami wyliczeniowymi.

8.7 Lekcja 7 - Tablice jednowymiarowe i wielowymiarowe, kolekcje

8.7.1 Cel lekcji


Celem lekcji jest przedstawienie podstawowych możliwości przechowywania zbiorów obiektów w tablicach, lub kolekcjach. Wy tłumaczone będzie pojęcie listy.


8.7.2 Treść - slajdy z opisem

Slajd 1		
------------	---	--

Człowiek - najlepsza inwestycja

Slajd
2

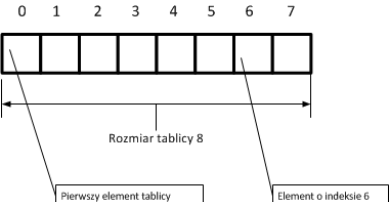
 **B2E**
BUSINESS TO EDUCATION


 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Co to jest tablica?

- Obiekt
- Określony typ
- Długość





 Lekcja 7 – tablice, kolekcje

Tablicą nazywamy obiekt zawierający zestaw elementów określonego typu. Długość i typ tablicy są określone przy jej deklarowaniu

Struktura taka pozwala na zarządzanie w programie zbiorem elementów tego samego typu. Np. zestaw ostatnich 10 pomiarów temperatury.

Slajd
3

 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

Deklaracja tablicy jednowymiarowej

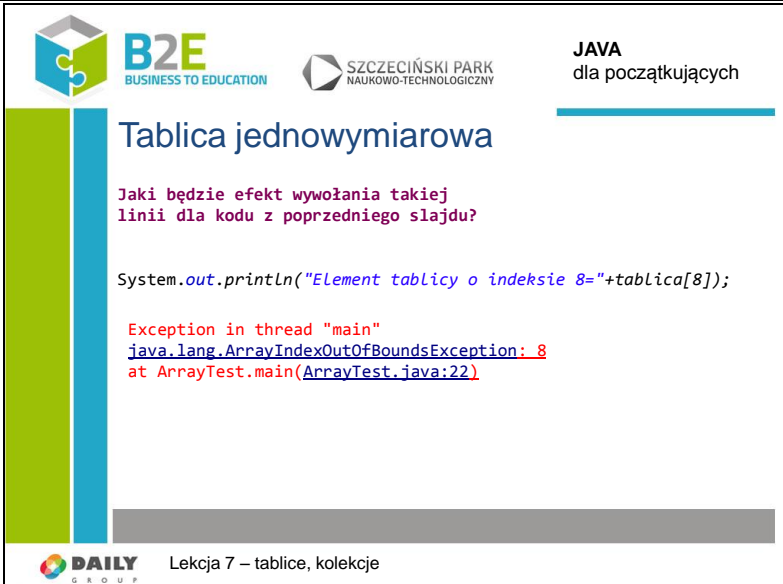
```
int[] tablica; // deklaracja - tablica będzie zawierać typu int
tablica = new int[8]; // inicjacja - długość tablicy jest 8

tablica[0] = 100; // pierwszy element tablicy
tablica[1] = 220;
tablica[2] = 250;
tablica[3] = 280;
tablica[4] = 300;
tablica[5] = 310;
tablica[6] = 400;
tablica[7] = 405; // ostatni element tablicy







System.out.println("Element tablicy o indeksie 3=" + tablica[3]);
```

 Lekcja 7 – tablice, kolekcje







Człowiek - najlepsza inwestycja

	<p><code>int[] tablica;</code></p> <p>W pierwszej linii wykonywana jest deklaracja tablicy. Mówi ona że:</p> <ul style="list-style-type: none"> • tablica będzie zawierać typ całkowity (<code>int</code>), • nazwa obiektu tablicy to ... <code>tablica</code> <p><code>tablica = new int[8];</code></p> <p>Inicjowana jest tablica i ustawiana jest jej długość</p> <p>Linie zawierające „<code>tablica[1] = 220;</code>” ustawiają kolejne wartości tablicy</p> <p>Co robi ostatnia linia kodu?</p> <p>Odp. Wyświetla napis „Element tablicy o indeksie 3=280”</p> <p>Co zrobić aby wyświetlić pierwszą wartość tablicy?</p> <p>Odp. Ustawić indeks <code>tablica[0]</code></p>	
Slajd 4		<p>Zgłoszony zostanie błąd.</p> <p>Array Index Out Of Bounds Exception – Wyjątek: Indeks tablicy poza zakresem.</p> <p>Oznacza to, że nie wolno wychodzić poza zakres tablicy, bo spowoduje to wyjątek i przerwanie działania programu.</p>







Człowiek - najlepsza inwestycja

<p>Slajd 5</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY </div> <div> JAVA dla początkujących </div> <hr/> <h2>Tablica jednowymiarowa - zadanie</h2> <p>Proszę napisać program który:</p> <ul style="list-style-type: none"> •zadeklaruje tablicę 10 elementów typu int •wypełni ją dowolnymi wartościami •wyświetli wszystkie wartości w konsoli <div>  DAILY G R O U P Lekcja 7 – tablice, kolekcje </div>	<p>Sugestia: wykorzystać pętlę for.</p>
<p>Slajd 6</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY </div> <div> JAVA dla początkujących </div> <hr/> <h2>Rozwiązanie zadania</h2> <pre> int[] tablica;// deklaracja tablica = new int[10];// inicjacja for (int i=0;i<10;i++)// iteruj od 0 do 9 tablica[i] = i*100; for (int i=0;i<10;i++)// iteruj od 0 do 9 System.out.println("Element tablicy o indeksie "+i+"="+tablica[i]); </pre> <p> Element tablicy o indeksie 0=0 Element tablicy o indeksie 1=100 Element tablicy o indeksie 2=200 Element tablicy o indeksie 3=300 Element tablicy o indeksie 4=400 ... </p> <div>  DAILY G R O U P Lekcja 7 – tablice, kolekcje </div>	<p>Zastosowanie pętli for i zmiennej „i” do iterowania po elementach tablicy.</p> <p>Proszę zwrócić uwagę na:</p> <p>Zmienna „i” jest wykorzystana do ustalenia wartości tablicy = i*100;</p> <p>Zmienna „i” jest wykorzystana do wyświetlenia indeksu przy wyświetlaniu wyników.</p>

Człowiek - najlepsza inwestycja


<p>Slajd 7</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY <div> JAVA dla początkujących </div> </div> <h2>Tablica jednowymiarowa - zadanie</h2> <p>Proszę zmodyfikować poprzedni program tak aby:</p> <ul style="list-style-type: none"> •wyświetlał sumę wartości w konsoli <div>  DAILY G R O U P Lekcja 7 – tablice, kolekcje </div>	<p>Sugestia: wykorzystać pętlę for dla wszystkich elementów tablicy</p> <p>for (int x:tablica)</p>
<p>Slajd 8</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY <div> JAVA dla początkujących </div> </div> <h2>Rozwiązanie zadania</h2> <pre>int suma =0; for (int x:tablica) suma+=x; System.out.println("Suma elementów tablicy wynosi "+suma);</pre> <p>Suma elementów tablicy wynosi 4500</p> <div>  DAILY G R O U P Lekcja 7 – tablice, kolekcje </div>	<p>Zastosowanie pętli for dla wszystkich elementów tablicy nie wymaga iteratora (zmiennej „i”).</p> <p>Proszę zwrócić uwagę na formułę suma+=x;</p> <p>Co ona oznacza?</p> <p>Odp: suma = suma + x;</p>


Człowiek - najlepsza inwestycja

<p>Slajd 9</p>	<div>   <div> <p>JAVA dla początkujących</p> </div> </div> <h2>Tablica – inny sposób deklaracji</h2> <pre>int[] tablica = {12,23,45,67,89,0,1,2,3,4};</pre> <p> Element tablicy o indeksie 0=12 Element tablicy o indeksie 1=23 Element tablicy o indeksie 2=45 Element tablicy o indeksie 3=67 Element tablicy o indeksie 4=89 Element tablicy o indeksie 5=0 Element tablicy o indeksie 6=1 Element tablicy o indeksie 7=2 Element tablicy o indeksie 8=3 Element tablicy o indeksie 9=4 Suma elementów tablicy wynosi 246 </p> <div>  <p>Lekcja 7 – tablice, kolekcje</p> </div>	<p>Inny sposób deklarowania i inicjowania tablicy – przyda się tylko wówczas, gdy wartości tablicy będą w programie stałe.</p>
<p>Slajd 10</p>	<div>   <div> <p>JAVA dla początkujących</p> </div> </div> <h2>Łańcuch znaków - string</h2> <pre>String nazwisko = "Kowalski"; System.out.println("Nazwisko =" + nazwisko);</pre> <div>  <p>Lekcja 7 – tablice, kolekcje</p> </div>	<p>Łańcuch znaków String jest tablicą jednowymiarową zawierającą elementy typu znak (char).</p> <p>Proszę, sprawdźcie, jak będzie działał powyższy program. Odp. Jak się można spodziewać, program wyświetli nazwisko „Kowalski”.</p>

Człowiek - najlepsza inwestycja

Slajd
11

 **B2E**
BUSINESS TO EDUCATION


 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Rozmiar tablicy

```
System.out.println("Liczba  
elementów tablicy wynosi "+tablica.length);

System.out.println("Długość nazwiska =" +nazwisko.length());
```


 **DAILY**
GROUP


Lekcja 7 – tablice, kolekcje

Typ String jest specyficznym typem tablicowym i posiada metody do wykonywania operacji na znakach. Metoda `length()` zwraca długość napisu zapamiętanego w zmiennej.

Typ tablicowy posiada licznik `.length`, który zwraca liczbę elementów w tablicy.

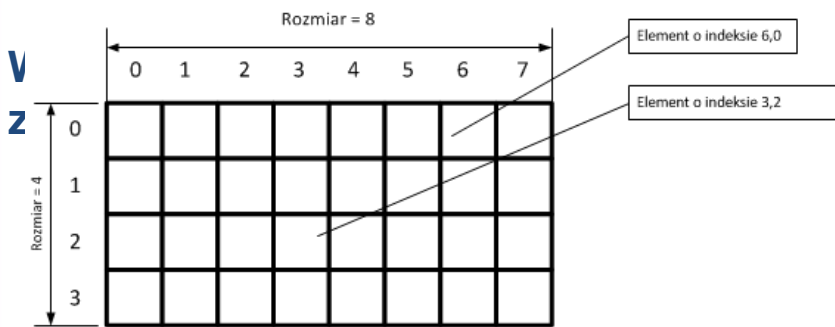
Slajd
12


 **B2E**
BUSINESS TO EDUCATION

 SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Tablica dwuwymiarowa









 **DAILY**
GROUP







Lekcja 7 – tablice, kolekcje

Rozważmy tablicę dwuwymiarową 8x4
Jak będzie wyglądała deklaracja takiej tablicy?

Człowiek - najlepsza inwestycja

<p>Slajd 13</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY <div>JAVA dla początkujących</div> </div> <h2>Deklaracja tablicy dwuwymiarowej</h2> <pre>int[][] tablica2; tablica2 = new int[8][4]; int[][][] tablica3; tablica3 = new int[8][4][3];</pre> <div>  DAILY G R O U P Lekcja 7 – tablice, kolekcje </div>	<p>Jak zadeklarować tablicę trójwymiarową 8x4x3?</p>
<p>Slajd 14</p>	<div>  B2E BUSINESS TO EDUCATION  SZCZECIŃSKI PARK NAUKOWO-TECHNOLOGICZNY <div>JAVA dla początkujących</div> </div> <h2>Tablica dwuwymiarowa - zadanie</h2> <p>Proszę program który:</p> <ul style="list-style-type: none"> •Zadeklaruje tablicę dwuwymiarową 8x4 •Ustawi jako wartość tablicy iloczyn indeksów np. dla indeksu 2,2 wartość będzie 4 •Wyświetli wszystkie elementy tablicy dwuwymiarowej •Obliczy i wyświetli sumę wszystkich elementów tablicy <div>  DAILY G R O U P Lekcja 7 – tablice, kolekcje </div>	<p>Sugestia: wykorzystać 2x pętlę for dla wszystkich elementów tablicy w wymiarze x i y (8,4).</p> <p>Kto poda pierwszy wynik sumy?</p> <p>Odp: 168</p>

Człowiek - najlepsza inwestycja

<p>Slajd 15</p>	<div>   </div> <div> JAVA dla początkujących </div> <hr/> <h2>Rozwiązanie zadania</h2> <pre> int[][] tablica2; tablica2 = new int[8][4]; int suma2 = 0; for (int i=0;i<8;i++) for (int j=0;j<4;j++) tablica2[i][j] = i*j; for (int i=0;i<8;i++) for (int j=0;j<4;j++) { suma2+=tablica2[j][i]; System.out.println("Element tablicy o indeksie "+i+ ", "+j+"="+tablica2[i][j]); } System.out.println("Suma elementów tablicy dwuwymiarowej wynosi "+suma2); </pre> <div>  Lekcja 7 – tablice, kolekcje </div>	<p>W rozwiązaniu zadania jest błąd.</p> <p>Kto wskaże, w którym miejscu, na czym błąd polega i co stałoby się przy uruchomieniu tego programu.</p> <p>Należy bardzo uważać, żeby nie pomylić kolejności iteratorów.</p>
<p>Slajd 16</p>	<div>   </div> <div> JAVA dla początkujących </div> <hr/> <h2>Inne sposoby zarządzania tablicami</h2> <p>Predefiniowane struktury:</p> <ul style="list-style-type: none"> •Collection •LinkedList •ArrayList •HashMap <div>  Lekcja 7 – tablice, kolekcje </div>	<p>Zarządzanie tablicami z przykładów jest na dłuższą metę dość uciążliwe.</p> <p>Wady:</p> <ul style="list-style-type: none"> •Musimy pilnować zakresów. •Musimy deklarować typy obiektów. •Aby wyszukać dany element, musimy napisać parę linii kodu. •... <p>W celu wyeliminowania tych, ale również innych wad, środowisko programistów przygotowało sobie specjalne narzędzia. Są to struktury pozwalające na łatwe zarządzanie danymi.</p>

Człowiek - najlepsza inwestycja

Slajd
17

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

ArrayList

Narzędzia do zarządzania listami danych:

- Collection
- LinkedList
- ArrayList
- HashMap

 **DAILY**
G R O U P
 Lekcja 7 – tablice, kolekcje

Zarządzanie tablicami z przykładów jest na dłuższą metę dość uciążliwe.

Wady:

- Musimy pilnować zakresów.
- Musimy deklarować typy obiektów.
- Aby wyszukać dany element, musimy napisać parę linii kodu.
- ...

W celu wyeliminowania tych ale również innych wad środowisko programistów przygotowało sobie specjalne narzędzia. Są to struktury pozwalające na łatwe zarządzanie danymi.

Człowiek - najlepsza inwestycja

Slajd
 18




JAVA
 dla początkujących


Przykład listy

```
import java.util.*; // potrzebna biblioteka narzędziowa
...
{
    List l = new ArrayList(); // deklaracja i inicjacja listy

    for (int i=0; i<10; i++)
        l.add(new Integer(i)); // dodawanie obiektów do listy

    Iterator i = l.iterator(); // powołanie iteratora

    while (i.hasNext()) // pętla po wszystkich elementach
        System.out.println("Element listy = "+i.next());
    // wyświetlenie kolejnych elementów
}
```


 Lekcja 7 – tablice, kolekcje

Pierwsza uwaga do przykładu – należy wykorzystać bibliotekę narzędzi java.util.

Dla uproszczenia dodamy wszystkie narzędzia w domenę util poprzez instrukcję

import java.util.*; - gwiazdka oznacza, że wszystkie biblioteki w danej bibliotece będą dołączone do programu.

Alternatywnie można dodać następujące linie:

import java.util.ArrayList;

import java.util.List;

import java.util.Iterator;







Deklaracja listy o nazwie „l” jest prosta: **List l = new ArrayList();**

Warto zauważyć, że nie deklarowaliśmy typu tablicy.

Następnie dodawane są **obiekty** do listy (10 elementów). Aby powołać obiekt, stosujemy instrukcję new Integer(<wartość>) – nowy obiekt typu Integer.



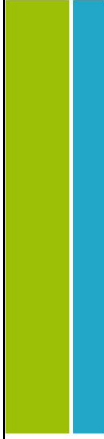

Aby wyświetlić wszystkie elementy z listy, należy powołać iterator. Jest to obiekt

Człowiek - najlepsza inwestycja

	<p>pozwalający na dostęp do kolejnych elementów listy.</p> <p>Zastosowano pętlę <code>Chile</code>, której sprawdza, czy jest na iteratorze kolejny element. Jeśli tak, instrukcja <code>i.next()</code> zwraca obiekt i jednocześnie przechodzi do kolejnego elementu listy.</p>	
Slajd 19	  <div style="float: right;"> JAVA dla początkujących </div> <hr style="width: 100px; margin-left: auto;"/> <div style="clear: both;"></div> <h2 style="color: #0070C0;">Lista - zadanie</h2> <p>Proszę przerobić poprzedni program tak aby:</p> <ul style="list-style-type: none"> • Zarządzał łańcuchami znaków zamiast liczbami • Dodanych zostało 20 elementów zamiast 10 • Wyświetlone zostały wszystkie elementy listy <div style="text-align: right; margin-top: 20px;">  Lekcja 7 – tablice, kolekcje </div>	<p>Sugestia: wykorzystać obiekt <code>String</code> analogicznie jak <code>Integer</code>.</p>
Slajd 20	  <div style="float: right;"> JAVA dla początkujących </div> <hr style="width: 100px; margin-left: auto;"/> <div style="clear: both;"></div> <h2 style="color: #0070C0;">Rozwiązanie</h2> <pre> for (int i=1;i<20;i++) l.add(new String("test "+i)); // dodawanie obiektów String do listy for (int i=1;i<20;i++) l.add(new Integer(i+20)); // dodawanie obiektów Integer do listy </pre> <p style="margin-left: 40px;">Element listy =test 19 Element listy =21</p> <div style="text-align: right; margin-top: 20px;">  Lekcja 7 – tablice, kolekcje </div>	

Wystarczy w jednym miejscu zmienić instrukcję powołując nowe obiekty.
 Z `new Integer(i)` na np. `new String(„tekst”+i)`

Człowiek - najlepsza inwestycja

	<p>Co warto zauważyć:</p> <ul style="list-style-type: none"> •Przerobienie programu jest dużo prostsze niż w przypadku tablic [][] Jedno miejsce. •Nie musimy zajmować się kontrolą liczby elementów. •Nic nie stoi na przeszkodzie aby lista zarządzała różnymi obiektami np. String i Integer. •Mamy do dyspozycji całą paletę narzędzi do zarządzania listą – następny slajd.
Slajd 21	<div>   <div> JAVA dla początkujących </div> <div>  <h2>Lista – dodatkowe funkcje</h2> </div> <div>  Lekcja 7 – tablice, kolekcje </div> </div> <div> <p>Są wśród nich:</p> <ul style="list-style-type: none"> •Dodawanie i wstawianie obiektów (add), na pozycję, dodawanie całej kolekcji. •Czyszczenie listy. •Wyszukiwanie obiektów. •Pobieranie obiektów wg nr. •Usuwanie wg indeksów, wg obiektów, wszystkich. •Tworzenie podlist. •Ustawianie obiektu na pozycji. •Sprawdzanie rozmiaru. •Iterowanie. </div>

Człowiek - najlepsza inwestycja

Klasy generyczne






```
//Do tej pory:
List imiona = new ArrayList();
for(int i = 0; i < imiona.size(); i++) {
    String imie = (String) imiona.get(i) ;
}

//Generycznie:
List<String> imiona = new ArrayList<String>();
for(int i = 0; i < imiona.size(); i++) {
    String imie = imiona.get(i) ;
}
```

Klasy generyczne zostały dodane do Javy 1.5. Generyczność pozwala na tworzenie klas posiadających pełną implementację, jednakże bez deklarowania typów wykorzystywanych przez implementację.

Na poprzednich slajdach nie deklarowaliśmy jaki typ danych przetrzymuje kolekcja. Przez co możliwe było dodanie dowolnego typu do tej samej kolekcji. Dodatkowo podczas wyciągania elementów z kolekcji trzeba było rzutować je na odpowiedni typ danych. Jeśli przez pomyłkę umieściliśmy przypadkowy obiekt w kolekcji to o błędzie dowiadaliśmy się dopiero podczas działania programu w momencie rzutowania.

Generics – tak określane w języku angielskim nie dotyczą tylko i wyłącznie kolekcji, można je stosować w własnych klasach. Jednak ich wykorzystanie wykracza poza ramy tego programu nauczania. Niemniej jednak kolekcje są bardzo dobrym przykładem ich użycia. Kolekcja przetrzymuje dane, zazwyczaj tego samego typu. Wobec czego logika zarządzania kolekcją (dodawanie, usuwanie, pobieranie elementów) nie zależy od typu jaki kolekcja przechowuje. Dopiero podczas użycia kolekcji programista deklaruje iż zamierza przetrzymywać dany typ w kolekcji. Na naszym przykładzie String. Taka deklaracja pozwala kompilatorowi na zgłaszanie prób wstawienia do kolekcji nie kompatybilnego typu i wczesne zauważenie błędu. Dodatkowo nie musimy już rzutować pobranych elementów, gdyż z góry wiadomo jaki typ danych przetrzymuje kolekcja.

	Jest to dość pobieżne omówienie typów generycznych, więc zachęcam do nauki we własnym zakresie. W dalszej części kursu będziemy czasem używać generics'ów, jednak nie zakładamy umiejętności tworzenia klas generycznych przez uczniów.	
Slajd 23	  <div style="text-align: right;">JAVA dla początkujących</div> <hr style="width: 100px; margin-left: auto;"/> <h2>Pytania</h2> <ol style="list-style-type: none"> 1. Jak zadeklarować listę 1-dno wymiarową? 2. Jak wypełnić listę wartościami – zaproponuj krótki program? 3. Co się stanie przy przekroczeniu zakresu tablicy? 4. Jak sprawdzić długość tablicy? 5. Jak zadeklarować listę trójwymiarową? 6. Co to jest ArrayList i do czego służy? 7. Do czego służy obiekt Iterator? 8. W jakiej domenie znajdują się narzędzia do obsługi list? <div style="text-align: center;">  <small>Człowiek - najlepsza inwestycja</small>  </div> <p style="text-align: center; font-size: small;">Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego</p> <div style="display: flex; justify-content: space-between; align-items: center;">  Lekcja 7 – tablice, kolekcje </div>	

8.7.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli gromadzić obiekty w tablicach i kolekcjach. Posiadą również umiejętność przeglądanie tych struktur za pomocą pętli.


8.8 Lekcja 8 - Operacje wejścia – wyjścia i wyjątki

8.8.1 Cel lekcji

Celem lekcji prezentacja możliwości zabezpieczenia aplikacji przed błędami, oraz umiejętność wymiany danych z zasobami z poza programu (sieć, pliki lokalne).

Człowiek - najlepsza inwestycja

8.8.2 Treść - slajdy z opisem



<p>Slajd 1</p>		
--------------------	--	--

Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd
2


SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

IO

Co to jest?

Zdarzenia losowe?

JAVA
dla początkujących





Operacje wejścia – wyjścia i wyjątki

Operacje wejścia/wyjścia - w informatyce taką nazwę nadano zadaniom związanym z komunikacją ze światem zewnętrznym. Światem zewnętrznym z punktu widzenia pisanej właśnie aplikacji. I tak oczekiwanie na interakcję użytkownika (wpisanie czegoś w konsoli, kliknięcie myszką), odczyt z lokalnego dysku twardego, zapis do sieciowego dysku, pobranie zasobu z Internetu, wydrukowanie czegoś, użycie kamery, odczyt z bazy danych itp. – to wszystko można zaklasyfikować jako komunikację ze światem zewnętrznym.

Ponieważ operacje IO odbywają się pomiędzy różnymi systemami (światami) mogą wystąpić różne problemy komunikacyjne. Zdalny serwer się zrestartuje lub po prostu przerwie połączenie, kabel od drukarki się odłączy (użytkownik akurat przesunął nogę i zahaczył o niego), dysk twardy będzie miał uszkodzoną powierzchnię i nie da się odczytać pliku, lub po prostu się przepełni i nie damy rady nic więcej zapisać. Ogólnie wiele nieprzewidzianych sytuacji może się wydarzyć.

Człowiek - najlepsza inwestycja

Slajd
3

JAVA
dla początkujących

wyjątki

Sytuacje wyjątków!

Potrzebne jest ubezpieczenie (zabezpieczenie)



Operacje wejścia – wyjścia i wyjątki


W związku z czym musi istnieć jakiś mechanizm obsługi takich sytuacji. Z pomocą przychodzą nam wyjątki (exceptions).

Wyjątki:


Podczas obsługi IO w Javie często korzysta się z wyjątków, w celu naprawy stanu programu po nieoczekiwanym zdarzeniu. Jednak wyjątki w Javie nie ograniczają się tylko do obsługi błędów IO. Wyjątki są uniwersalnym mechanizmem obsługi nieprzewidzianych zdarzeń, również pochodzących z wnętrza naszej aplikacji.

Człowiek - najlepsza inwestycja

Slajd
 4



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY


JAVA
 dla początkujących

```

public void printWaga(Ptak p) {
    System.out.println(p.getWaga());
}

...

public static void main(String [] args) {
    ....
    myPrinter.printWaga(null);
}
                
```



Operacje wejścia – wyjścia i wyjątki

Przykładowo próba odwołania się do metody na referencji której wartość jest równa NULL;

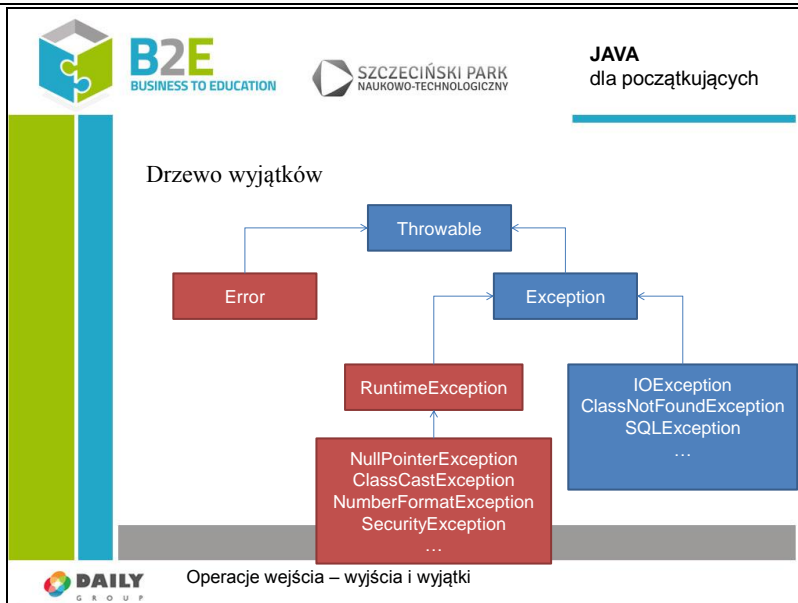
JVM chętnie wykona nasze rozkazy, ale jak ma rozumieć to wywołanie?

Maszyna wirtualna nie jest w stanie wywołać metody dla nie zdefiniowanego obiektu, dlatego napotykając na taką sytuację wyrzuca wyjątek `NullPointerException`.

Programista został poinformowany, że wydarzyła się sytuacja, której nikt się nie spodziewał, z której nie ma prostego logicznego i domyślnego wyjścia. Co niby ma zrobić program, wypisać losową wagę? Udać, że nic się nie stało? A co, jeśli ktoś czeka, aż na ekranie pojawi się waga ptaka? No właśnie, wyjątki muszą zostać obsłużone w należyty sposób. Brak obsługi wyjątku spowoduje zakończenie programu, aby tego uniknąć to programista musi zdecydować, co w takim przypadku zrobić. Każdy problem powinien mieć jakieś rozsądne rozwiązanie, dla naszego przykładu można przechwycić wyjątek i wypisać na konsolę „Niestety ptak niedostępny, wobec czego nie mogę podać wagi”.

Człowiek - najlepsza inwestycja

Slajd
5



Wyjątki w Javie tworzą drzewo dziedziczenia. Rozróżniamy dwa typy wyjątków. Checked i Unchecked exceptions. Na slajdzie na czerwono zaznaczone są wyjątki nieweryfikowalne, a na zielono weryfikowalne.


Checked exceptions – ich obsługa jest obowiązkowa, wiążą się z sytuacjami, które ze swojej natury zazwyczaj można jeszcze naprawić i zmuszają programistę do podjęcia takiej próby.

Unchecked Exceptions – ich obsługa nie jest wymuszona, gdyż z swej natury sytuacja jest na tyle krytyczna, że zazwyczaj nie ma łatwego sposobu jej naprawy.


Oczywiście sposób naprawy sytuacji zależy od decyzji programisty. I dlatego każdy z tych wyjątków można przechwycić i spróbować poradzić sobie z sytuacją.

Człowiek - najlepsza inwestycja

Slajd
 6



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY


JAVA
 dla początkujących

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.zip.GZIPInputStream;

...

FileInputStream fis = new FileInputStream("/ksiazka.txt.gz");
BufferedReader bis = new BufferedReader(fis);
GZIPInputStream gis = new GZIPInputStream(fis);
InputStreamReader isr = new InputStreamReader(gis);
BufferedReader br = new BufferedReader(isr);
br.readLine();
            
```



Operacje wejścia – wyjścia i wyjątki

Operacje wejścia-wyjścia w Javie polegają na obsłudze strumieni. Maszyna wirtualna udostępnia obiekty, które, np. odczytują znaki z klawiatury.

W architekturze obsługi strumieni w Javie zastosowano wzorzec dekoratora. Podstawowym obiektem jest strumień, który to w zależności od potrzeb możemy opakować w bardziej specjalizowane klasy dekorujące. Klasa dekoratora dostarcza dodatkowych bardziej wyspecjalizowanych operacji.


W przykładzie odczytujemy pierwszą linię książki przechowywanej w spakowanym pliku tekstowym.

Najpierw otwieramy plik, później aby przyspieszyć odczyt buforujemy go w pamięci, później dekompresujemy, następnie zamieniamy strumień bajtów na strumień znaków, na koniec opakowujemy cały czas ten sam strumień w klasę dającą łatwiejszy dostęp do operacji na ciągach znaków. W tym przypadku zależało nam na metodzie `readLine()`.


Dzięki strumieniowi nie musimy w pamięci przechowywać osobnego obiektu dla każdego z kroków. Przetwarzanie odbywa się płynnie i odczytywane są kolejne bajty na żądanie, w związku z czym możliwe jest przetwarzanie dużych plików.

Człowiek - najlepsza inwestycja

Slajd
 7



B2E
 BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        InputStreamReader isr = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(isr);
        String line = "";
        try {
            line = br.readLine();
            System.out.println("no exception was thrown");
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            br.close();
        }
        System.out.println("Wczytałem: \"" + line + "\"");
    }
}
                
```



Operacje wejścia – wyjścia i wyjątki

Przykład podany na slajdzie odczytuje cały wers wpisany do konsoli.

Pojawił się nowy element: bloki "try-catch". Jest to mechanizm obsługi błędów. W przypadku gdybyśmy utracili połączenie z urządzeniem z którego czytamy, program jest w stanie obsłużyć błąd i zareagować w odpowiedni sposób. Uchroni to naszą aplikację od nieprzewidzianego zachowania.

Zasada działania bloku try-catch jest intuicyjna. Wszystko co znajduje się pomiędzy try i catch jest podejrzane o możliwość wyrzucenia wyjątku. Jako argument do instrukcji catch podajemy klasę wyjątku jaką chcemy złapać. Istnieje możliwość dodania więcej niż jednej instrukcji catch (przykład będzie na dalszych slajdach), w związku z czym można mieć oddzielną obsługę różnych typów nieprzewidzianych sytuacji. Jeśli w bloku try mamy kilka linii kodu i w pierwszej zostanie rzucony wyjątek to pozostałe linie nie zostaną wykonane i program wznowi działanie od zawartości bloku Catch, który złapał zadany wyjątek. W naszym przykładzie w przypadku wyjątku rzuconego z metody readLine() na ekranie nie pojawi się napis „no exception was thrown”. Jeśli żadne z bloków catch nie przechwyci wyjątku, zostanie on przekazany w górę stosu wywołań. Jeśli nic go nie przechwyci program się zatrzyma. Opcjonalną częścią bloku try-catch jest blok finally – blok ten gwarantuje że kod zawarty w nim zawsze się wykona, niezależnie czy był wyjątek czy nie i czy został obsłużony. Jeśli wystąpi wyjątek wewnątrz bloku finalny, to pozostałe linie kodu w tym bloku nie będą wykonane.

Człowiek - najlepsza inwestycja

Slajd
8



JAVA dla początkujących

Ćwiczenie:

Sprawdź co wyświetli się na konsoli po wykonaniu poniższego kodu.
 Sprawdź na własnych przykładach działanie wyrażień: „\n”, „\t”, „\\”, „\”.

```
class MojaPierwszaKlasa {

    public static void main(String[] args) {
        System.out.println("\");
        System.out.println("\ntekst");
        System.out.println("\\");
        System.out.println("\ttekst");
    }
}
```



Operacje wejścia – wyjścia i wyjątki

\ " - cudzysłów

\n – znak nowej linii

\\ - znak \

\t – tabulator

Więcej o formatowaniu tekstu można dowiedzieć się w dokumentacji:

<http://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html#syntax>

Człowiek - najlepsza inwestycja





KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Slajd
 9




SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY


JAVA
 dla początkujących

```

class MojaPierwszaKlasa {

    public static void main(String[] args) {
        String str = "sto dwa";

        try {
            int i = Integer.parseInt(str);
        } catch (NumberFormatException e) {
            e.printStackTrace();
        }
        System.out.println("Jestem tutaj!");
    }
}
                
```



Operacje wejścia – wyjścia i wyjątki

W przypadku próby stworzenia instancji klasy "Integer", korzystając z nieprawidłowego łańcucha znaków, program zakończy się niepowodzeniem.

Możemy tego uniknąć stosując blok "try-catch", tak jak na przykładzie. W bloku "try" wykonują się polecenia potencjalnie "niebezpieczne". Oczywiście łańcuch znaków "sto dwa" jest nieprawidłowy i przejdziemy do bloku "catch". Problem zostanie przechwycony i wykonają się polecenie z tego bloku. Na ekranie zobaczymy cały stack-trace programu. W przyszłości, w dużych aplikacjach, będziemy mogli łatwo odszukać klasę, metodę, a nawet wers, w którym pojawił się błąd, wraz z opisem błędu.

Fakt, że w konsoli pokazał się raport o błędzie, nie wpływa na niestabilność programu. Napis "Jestem tutaj!" również będzie widoczny. Uniknęliśmy niespodziewanego przerwania programu.

Człowiek - najlepsza inwestycja

Slajd
 10


JAVA dla początkujących

Ćwiczenie:

Wykorzystując przykład z poprzedniego slajdu wczytaj z konsoli dwie liczby dowolnego typu. Przechwyć ewentualne wyjątki. Program powinien zakończyć swoje działanie dopiero po wczytaniu dwóch liczb.

Wykorzystaj fakt, że każda referencja może wskazywać na pusty obiekt „null”, to znaczy, że możesz zapisać:

```
Integer i = null;
```

Następnie możesz wykonać porównanie, aby sprawdzić, czy nadal jest „pusty”:

```
i == null
```



Operacje wejścia – wyjścia i wyjątki

```
import java.io.BufferedReader;
```

```
import java.io.IOException;
```

```
import java.io.InputStreamReader;
```

```
class MojaPierwszaKlasa {
```

```
    public static void main(String[] args) {
```

```
        int a = read();
```

```
        int b = read();
```

```
        System.out.println("Wczytałem: "
                             + a + " i " + b + ".");
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



```
}

private static int read() {

    InputStreamReader isr = new InputStreamReader(System.in);

    BufferedReader br = new BufferedReader(isr);

    Integer i = null;

    while (i == null) {

        try {

            i = Integer.parseInt(br.readLine());

        } catch (IOException e) {

            System.err.println("Błąd odczytu z konsoli. Prawdopodobnie nie da się  
z tym już nic zrobić, propaguję wyjątek wyżej!");

            throw new RuntimeException(e);

        } catch (NumberFormatException e) {

            System.out.println("Podany ciąg znaków nie jest liczbą. Spróbuj  
ponownie.");

        }

    }

    return i;

}

}
```

Człowiek - najlepsza inwestycja





KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Slajd
11





SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class ZeroParameterException extends Exception {
    public ZeroParameterException(String message) {
        super(message);
    }
}
    
```




Operacje wejścia – wyjścia i wyjątki

Stworzone przez nas klasy mogą powodować charakterystyczne tylko dla nich błędy. Warto, aby w takim przypadku rzucały wyjątek stworzony specjalnie dla nich.


Stworzenie takiego wyjątku jest bardzo proste. Wystarczy dziedziczyć po klasie "Exception" i zaimplementować przynajmniej jeden konstruktor.

Człowiek - najlepsza inwestycja

Slajd
12



B2E
 BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        int a = 2, b = 0;
        try {calculate(a, b);} catch
        (ZeroParameterException e) {
            e.printStackTrace();
        }
    }
    private static int calculate(int a, int b)
        throws ZeroParameterException {
        if (a == 0) {throw new ZeroParameterException(
            "Argument a jest zerem!");}
        if (b == 0) {throw new ZeroParameterException(
            "Argument b jest zerem!");}
        return a + b;
    }
}
    
```


 Operacje wejścia – wyjścia i wyjątki

Jeżeli metoda może wyjątek wyrzucić, należy zadeklarować to za listą argumentów, tak jak na slajdzie.

Kiedy zajdzie niepożądana sytuacja (w tym przypadku jeden z argumentów będzie zerem), wyjątek powinien być wyrzucony. Użycie "throw" kończy wywołanie metody. Nie zwróci ona żadnej wartości.

Człowiek - najlepsza inwestycja

Slajd
13



JAVA dla początkujących

Ćwiczenie:

Napisz program liczący pierwiastki trójkątnu kwadratowego. W przypadku parametru a równego 0 rzuć wyjątkiem. W przypadku ujemnej delty również. Pierwiastek jest metodą statyczną „sqrt” w klasie „Math”.



Operacje wejścia – wyjścia i wyjątki

```
public class AZeroException extends Exception {

    public AZeroException() {

        super("A nie może być zerem!");

    }

}

public class DeltaUjemnaException extends Exception {

    public DeltaUjemnaException() {

        super("Delta jest ujemna");

    }

}
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



```
public class Trojmian {

    public static void licz(int a, int b, int c)

        throws AZeroweException, DeltaUjemnaException {

        if (a == 0)

            throw new AZeroweException();

        int delta = b * b - 4 * a * c;

        if (delta < 0)

            throw new DeltaUjemnaException();

        // ...

    }

}

class MojaPierwszaKlasa {

    public static void main(String[] args) {

        try {

            Trojmian.licz(2, 20, -1000);

        } catch (AZeroweException | DeltaUjemnaException e) {

            e.printStackTrace();



        }

    }

}
```

Człowiek - najlepsza inwestycja

Slajd
14

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY


JAVA
dla początkujących

```

import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        try {
            URL url = new URL("http://google.com");
            URLConnection con = url.openConnection();
            InputStream is = con.getInputStream();

            is.close();
        } catch (MalformedURLException e) {
            System.out.println("Niepoprawny URL");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
    
```

 **DAILY**
G R O U P
 Operacje wejścia – wyjścia i wyjątki

Strumienie są bardzo uniwersalne. Java umożliwia w łatwy sposób obsługę sieci za ich pomocą.

Spójrz na wyjątki i spróbuj odgadnąć jakie problemy mogą wystąpić w trakcie połączenia z Internetem.

Nieprawidłowy adres URL i błąd odczytu (zerwane połączenie).

Odpowiednie nazewnictwo czyni pracę programisty łatwiejszą.

Człowiek - najlepsza inwestycja

Slajd
15



JAVA dla początkujących

Ćwiczenie:

Wykorzystując kod z poprzedniego przykładu połącz się z dowolną stroną internetową i wyświetl w konsoli jej kod źródłowy.

Sprawdź jakimi metodami możesz operować na obiekcie typu „InputStream”. Każdy bajt odczytany ze strumienia możesz rzutować na typ „char” i wyświetlić w konsoli jako znak.



Operacje wejścia – wyjścia i wyjątki

```
import java.io.IOException;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLConnection;

class MojaPierwszaKlasa {

    public static void main(String[] args) {

        try {

            URL url = new URL("http://google.com");
```

Człowiek - najlepsza inwestycja



	<pre>URLConnection con = url.openConnection(); InputStream is = con.getInputStream(); byte [] b = new byte[512]; while (is.read(b) > -1) { for (int i=0 ; i<512 ; i++) System.out.print((char)b[i]); } is.close(); } catch (MalformedURLException e) { e.printStackTrace(); } catch (IOException e) { e.printStackTrace(); } }</pre>
--	---

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Slajd
16



B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.FileInputStream;
import java.io.InputStreamReader;

class MojaPierwszaKlasa {
    public static void main(String[] args) {
        try(BufferedReader br = new BufferedReader(
            new InputStreamReader(
                new DataInputStream(new FileInputStream("plik.txt"))
            ))) {
            String strLine;
            while ((strLine = br.readLine()) != null) {
                System.out.println (strLine);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```


Operacje wejścia – wyjścia i wyjątki

Podstawową operacją na strumieniach jest obsługa plików. Przykład pokazuje, jak odczytać plik.

Domyślna lokalizacja pliku jest w folderze naszego projektu.

Dodatkowo proszę zwrócić uwagę na nową konstrukcję bloku try. Od Javy 1.7 mamy możliwość skorzystania z automatycznego zamykania zasobów. Konstrukcja jest dość prosta w nawiasach półokrągłych następujących zaraz po try inicjalizujemy zasoby, które mają być automatycznie obsługiwane przez maszynę wirtualną Javy. Warunkiem jest to aby zasoby otwierane implementowały interfejs AutoClosable. Po zakończeniu wykonywania kodu z bloku try wszystkie zasoby zostaną zwolnione.

Człowiek - najlepsza inwestycja

Slajd
17**JAVA**
dla początkujących**Ćwiczenie:**

Korzystając z klas „FileWriter” i „BufferedWriter” utwórz plik i zapisz w nim własny tekst.



Operacje wejścia – wyjścia i wyjątki

```
import java.io.BufferedWriter;
```

```
import java.io.FileWriter;
```

```
class MojaPierwszaKlasa {
```

```
    public static void main(String[] args) {
```

```
        try {
```

```
            FileWriter fstream = new FileWriter("out.txt");
```

```
            BufferedWriter out = new BufferedWriter(fstream);
```

```
            out.write("Mój tekst");
```

```
            out.close();
```

```
        } catch (Exception e) {
```


Człowiek - najlepsza inwestycja

**KAPITAŁ LUDZKI**
NARODOWA STRATEGIA SPÓJNOŚCIUNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

	<pre>e.printStackTrace(); } } }</pre>
--	--

Człowiek - najlepsza inwestycja

Slajd
18


 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



Ćwiczenie:

Zapisz do pliku źródło dowolnej strony internetowej. Niech plik ma rozszerzenie „*.html”. Co mogło spowodować taki efekt?

 **KAPITAŁ LUDZKI**
NARODOWA STRATEGIA SPÓJNOŚCI
 Człowiek - najlepsza inwestycja
 **UNIA EUROPEJSKA**
EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

 **DAILY**
G R O U P
 Operacje wejścia – wyjścia i wyjątki

```
import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLConnection;

class MojaPierwszaKlasa {

    public static void main(String[] args) {

        try {
```

Człowiek - najlepsza inwestycja

	<pre>URL url = new URL("http://google.com"); URLConnection con = url.openConnection(); InputStream is = con.getInputStream(); FileWriter fstream = new FileWriter("strona.html"); BufferedWriter out = new BufferedWriter(fstream); out.write("strona.txt"); // byte zajmuje 1 bajt, a char 2 bajty (można błędnie zapisać znaki) byte [] b = new byte[512]; while (is.read(b) > -1) { for (int i=0 ; i<512 ; i++) out.write((char)b[i]); } out.close(); is.close(); } catch (MalformedURLException e) { e.printStackTrace(); } catch (IOException e) { e.printStackTrace(); } }</pre>
--	--

8.8.3 Opis założonych osiągnięć ucznia

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego


Po tej lekcji uczniowie będą umieli skorzystać z połączenia sieciowego i plików lokalnych w komputerze. Ponadto będą umieli zabezpieczyć swój program przed potencjalnymi błędami.

8.9 Lekcja 9 - Graficzny interfejs użytkownika - Swing

8.9.1 Cel lekcji


Celem lekcji jest pokazanie możliwości graficznego pakietu Swing. Wytłumaczenie jak poprawnie korzystać z tych elementów, oraz jak zarządzać ich rozkładem w oknie.

8.9.2 Treść - slajdy z opisem


Slajd 1	 <p>Slide content: The slide is titled 'Java - lekcja 9' with the subtitle 'Graficzny interfejs użytkownika - Swing'. It features logos for B2E, Szczeciński Park Naukowo-Technologiczny, DAILY, and the European Union. It also includes the text 'Człowiek - najlepsza inwestycja' and 'Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego'.</p>	
------------	---	--

Człowiek - najlepsza inwestycja

Slajd
2



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```
import javax.swing.JFrame;


public class Start {

    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 500, 300);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CL
OSE);

        mainFrame.setVisible(true);
    }
}
```





Graficzny interfejs użytkownika - Swing

Okno możemy stworzyć korzystając z klasy "Jframe".

Łańcuch znaków podany w konstruktorze jest tytułem okna.

Metoda "setBounds" posiada 4 argumenty:



- odległość w poziomie od lewego, górnego rogu,
- odległość w pionie od lewego, górnego rogu,
- szerokość okna,
- wysokość okna.

Jeżeli nie ustawimy metody "setDefaultCloseOperation", to kliknięcie przycisku zamknięcia w prawym, górnym rogu nie spowoduje wyłączenia programu. Okno zniknie, ale aplikacja nadal będzie chodziła w tle!

Ostatnie polecenie powoduje wyświetlenie się okna.

Człowiek - najlepsza inwestycja

Slajd
3

JAVA
dla początkujących

```

public class Start {

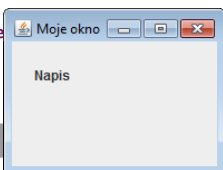
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 200, 150);


        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(null);

        JLabel label = new JLabel("Napis");
        label.setBounds(20, 20, 50, 20);

        mainFrame.add(label);
        mainFrame.setVisible(true);
    }
}

```





Graficzny interfejs użytkownika - Swing

Do dyspozycji mamy wiele gotowych komponentów. Jeżeli chcemy wyświetlić tekst możemy skorzystać z klasy "JLabel".

Skorzystajmy z metody "setLayout" z argumentem "null". Będziemy wtedy mogli skorzystać z absolutnego pozycjonowania komponentów widoku. W późniejszych slajdach zostanie wytłumaczone, jak układać elementy korzystając z zarządców rozkładu (layoutów).

Nie zapomnijmy dodać obiektu "label" do okna.

Slajd
4




JAVA
dla początkujących

```

public class Start {

    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 200, 150);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(null);

        JTextField jTextField = new JTextField("Napis");
        jTextField.setBounds(20, 20, 100, 40);

        mainFrame.add(jTextField);
        mainFrame.setVisible(true);
    }
}

```



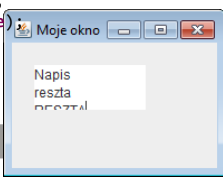



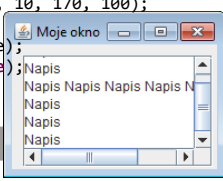







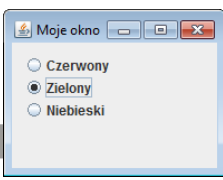



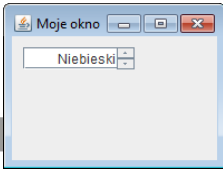

Graficzny interfejs użytkownika - Swing

Klasa "JtextField" umożliwia wyświetlenie pola tekstowego z możliwością edycji. Ma ono jednak ograniczenie. Tekst może mieć tylko jeden wers.



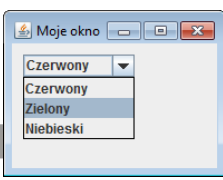



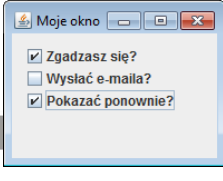

Człowiek - najlepsza inwestycja

<p>Slajd 5</p>	<div>   <div> <p>JAVA dla początkujących</p> <hr/> </div> </div> <pre> public class Start { public static void main(String [] args) { JFrame mainFrame = new JFrame("Moje okno"); mainFrame.setBounds(200, 200, 200, 150); mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); mainFrame.setLayout(null); JTextArea jTextArea = new JTextArea("Napis"); jTextArea.setBounds(20, 20, 100, 40); mainFrame.add(jTextArea); mainFrame.setVisible(true); } } </pre>  <div>  <p>Graficzny interfejs użytkownika - Swing</p> </div>	<p>"JTextArea" umożliwia wyświetlenie tekstu składającego się z wielu wersów. Zauważmy jednak, że tekst nie mieści się w polu.</p>
<p>Slajd 6</p>	<div>   <div> <p>JAVA dla początkujących</p> <hr/> </div> </div> <pre> public class Start { public static void main(String [] args) { JFrame mainFrame = new JFrame("Moje okno"); mainFrame.setBounds(200, 200, 200, 150); mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); mainFrame.setLayout(null); JTextArea jTextArea = new JTextArea("Napis"); JScrollPane jScrollPane = new JScrollPane(jTextArea); jScrollPane.setBounds(10, 10, 170, 100); mainFrame.add(jScrollPane); mainFrame.setVisible(true); } } </pre>  <div>  <p>Graficzny interfejs użytkownika - Swing</p> </div>	<p>Problem z widocznością tekstu możemy rozwiązać za pomocą "JScrollPane". Obiekt "JTextArea" może teraz zmieniać swój rozmiar dynamicznie. Przeglądanie ułatwiają paski przewijania.</p>

Człowiek - najlepsza inwestycja

<p>Slajd 7</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> JRadioButton button1 = new JRadioButton("Czerwony"); button1.setBounds(10, 10, 100, 20); JRadioButton button2 = new JRadioButton("Zielony"); button2.setBounds(10, 30, 100, 20); JRadioButton button3 = new JRadioButton("Niebieski"); button3.setBounds(10, 50, 100, 20); ButtonGroup colorButtonGroup = new ButtonGroup(); colorButtonGroup.add(button1); colorButtonGroup.add(button2); colorButtonGroup.add(button3); mainFrame.add(button1); mainFrame.add(button2); mainFrame.add(button3); </pre>  <div>  Graficzny interfejs użytkownika - Swing </div>	<p>Obiekty "JRadioButton" umożliwiają wybór pojedynczej opcji. Zaznaczony może być tylko jeden element dlatego wszystkie muszą tworzyć grupę. Obiekt "ButtonGroup" kontroluje wszystkie komponenty "JRadioButton". Jeżeli nie użylibyśmy go, opisywana zasada nie zachodziłaby.</p>
<p>Slajd 8</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> String [] dane = {"Czerwony", "Zielony", "Niebieski"}; SpinnerModel model = new SpinnerListModel(dane); JSpinner spinner = new JSpinner(model); spinner.setBounds(10, 10, 100, 20); mainFrame.add(spinner); </pre>  <div>  Graficzny interfejs użytkownika - Swing </div>	<p>Obiekt typu "JSpinner" również umożliwia nam wybór tylko jednej opcji, ale w nieco inny sposób. Zwróćmy uwagę na sposób, w jaki wstawiamy dane do obiektu. Korzystamy z modelu. Wynika to z zastosowania Wzorca projektowego MVC przez twórców Swingu. Wzorzec ten rozdziela warstwę widoku od warstwy danych.</p>

Człowiek - najlepsza inwestycja

<p>Slajd 9</p>	<div>   <div> <p>JAVA dla początkujących</p> <hr/> </div> <pre> JComboBox<String> jComboBox = new JComboBox<String>(); jComboBox.addItem("Czerwony"); jComboBox.addItem("Zielony"); jComboBox.addItem("Niebieski"); jComboBox.setBounds(10, 10, 100, 20); mainFrame.add(jComboBox); </pre>  <div>  <p>Graficzny interfejs użytkownika - Swing</p> </div> </div>	<p>"JComboBox", też pozwala wybrać tylko jedną opcję. Jest on typem generycznym. W nawiasach "<>" podajemy typ obiektu jaki będzie przechowywany w "JComboBox".</p>
<p>Slajd 10</p>	<div>   <div> <p>JAVA dla początkujących</p> <hr/> </div> <pre> JCheckBox jCheckBox1 = new JCheckBox("Zgadzasz się?"); jCheckBox1.setBounds(10, 10, 150, 20); JCheckBox jCheckBox2 = new JCheckBox("Wysłać e-maila?"); jCheckBox2.setBounds(10, 30, 150, 20); JCheckBox jCheckBox3 = new JCheckBox("Pokazać ponownie?"); jCheckBox3.setBounds(10, 50, 150, 20); mainFrame.add(jCheckBox1); mainFrame.add(jCheckBox2); mainFrame.add(jCheckBox3); </pre>  <div>  <p>Graficzny interfejs użytkownika - Swing</p> </div> </div>	<p>"JCheckBox" umożliwia wybór kilku opcji. Elementy te są całkowicie niezależne. Nie dodajemy ich do żadnej grupy, tak jak to było w przypadku "JRadioButton".</p>

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

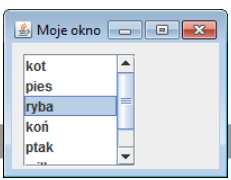



Slajd
11




JAVA
dla początkujących



```
String[] selections = { "kot", "pies", "ryba", "koń", "ptak",
    "wilk", "lis"};
JList<String> list = new JList<String>(selections);
JScrollPane jScrollPane = new JScrollPane(list);
jScrollPane.setBounds(10, 10, 100, 100);
mainFrame.add(jScrollPane);
```




 Graficzny interfejs użytkownika - Swing

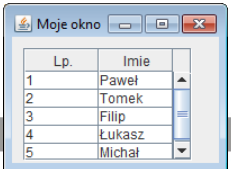
"JList" jest również typem generycznym. Pozwala on wyświetlić dane w bardziej czytelny sposób. Można ustawiać możliwość zaznaczania pojedynczego lub wielokrotnego.


Slajd
12

JAVA
dla początkujących







```
String rowData[][] = { { "1", "Paweł"},
    { "2", "Tomek"},
    { "3", "Filip"},
    { "4", "Łukasz"},
    { "5", "Michał"} };
String columnNames[] = { "Lp.", "Imie"};
JTable table = new JTable(rowData, columnNames);
JScrollPane jScrollPane = new JScrollPane(table);
jScrollPane.setBounds(10, 10, 150, 100);
mainFrame.add(jScrollPane);
```




 Graficzny interfejs użytkownika - Swing

"Jtable" działa podobnie jak "Jlist", z tą różnicą, że umożliwia wyświetlenie dodatkowych kolumn.

Człowiek - najlepsza inwestycja

<p>Slajd 13</p>	<div>   </div> <p>JAVA dla początkujących</p> <hr/> <p>Ćwiczenie:</p> <p>Zbuduj formularz do wpisania adresu zamieszkania.</p> <div> <p>Imię: <input type="text"/></p> <p>Nazwisko: <input type="text"/></p> <p>Ulica: <input type="text"/></p> <p>Numer domu: <input type="text"/></p> <p>Numer Lokalu: <input type="text"/></p> <p>Kod pocztowy: <input type="text"/> - <input type="text"/></p> <p>Miasto: <input type="text"/></p> </div> <div>  <p>Graficzny interfejs użytkownika - Swing</p> </div>	
<p>Slajd 14</p>	<div>   </div> <p>JAVA dla początkujących</p> <hr/> <p>Ćwiczenie:</p> <p>Stwórz plik o zawartości:</p> <p>Jan Kowalski Wiejska 12 4 12-345 Warszawa</p> <p>Plik może znajdować się w dowolnej lokalizacji.</p> <div>  <p>Graficzny interfejs użytkownika - Swing</p> </div>	<p>Zapisując dane aplikacji w plikach można stworzyć coś w rodzaju prymitywnej bazy danych. Jest to dobre rozwiązanie w przypadku małych i prostych aplikacji.</p>

Człowiek - najlepsza inwestycja

Slajd
 15


JAVA dla początkujących

Ćwiczenie:

Odczytaj dane z pliku i wprowadź do stworzonego formularza.

Aby ułatwić sobie odszukanie pliku skorzystaj z gotowego okna dialogowego z pakietu Swing.

```
JFileChooser chooser = new JFileChooser();
chooser.showOpenDialog(null);
File curFile = chooser.getSelectedFile();
```



Graficzny interfejs użytkownika - Swing

```
import java.io.BufferedReader;

import java.io.DataInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;


import javax.swing.JFileChooser;

import javax.swing.JFrame;

import javax.swing.JTextField;
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY




```
public class Start {
```

```
    public static void main(String [] args) {
```

```
        JFrame mainFrame = new JFrame("Moje okno");
```

```
        mainFrame.setBounds(200, 200, 200, 150);
```

```
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
        mainFrame.setLayout(null);
```

```
        JFileChooser chooser = new JFileChooser();
```

```
        chooser.showOpenDialog(null);
```

```
        File curFile = chooser.getSelectedFile();
```

```
        JTextField jTextField = new JTextField();
```

```
        jTextField.setBounds(10, 10, 150, 20);
```

```
        try {
```

```
            FileInputStream fis = new FileInputStream(curFile);
```

```
            DataInputStream in = new DataInputStream(fis);
```

```
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
```

```
            jTextField.setText(br.readLine());
```

```
            fis.close();
```

```
        } catch (FileNotFoundException e) {
```

Człowiek - najlepsza inwestycja








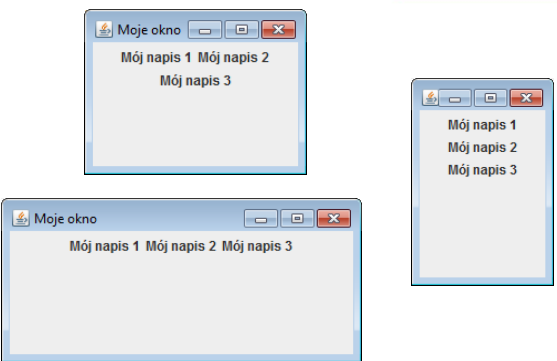

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY





	<pre> e.printStackTrace(); } catch (IOException e) { e.printStackTrace(); } mainFrame.add(jTextField); mainFrame.setVisible(true); } }</pre>		
Slajd 16	<div>   <div> JAVA dla początkujących </div> </div> <div> <p>Ćwiczenie:</p> <p>Spróbuj zmienić obraz okna z formularzem. Czy komponenty zawsze są widoczne?</p> <p>Wywołaj metodę:</p> <pre>mainFrame.setResizable(false);</pre> <p>Sprawdź ponownie zachowanie okna.</p> </div> <div>  Graficzny interfejs użytkownika - Swing </div>		

Człowiek - najlepsza inwestycja

<p>Slajd 17</p>	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class Start { public static void main(String [] args) { JFrame mainFrame = new JFrame("Moje okno"); mainFrame.setBounds(200, 200, 200, 150); mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); mainFrame.setLayout(new FlowLayout()); JLabel label1 = new JLabel("Mój napis 1"); mainFrame.add(label1); JLabel label2 = new JLabel("Mój napis 2"); mainFrame.add(label2); JLabel label3 = new JLabel("Mój napis 3"); mainFrame.add(label3); mainFrame.setVisible(true); } } </pre> <div>  Graficzny interfejs użytkownika - Swing </div>	<p>We wcześniejszych przykładach do metody "setLayout" wstawialiśmy obiekt "null". Teraz skorzystamy z zarządcy rozkładu o nazwie "FlowLayout". Nie musimy już układać każdego elementu osobno. Co więcej, nawet gdy użyjemy metody "setBounds", nie uzyskamy oczekiwanego efektu. "FlowLayout" decyduje o położeniu komponentów.</p>
<p>Slajd 18</p>	<div>   <div> JAVA dla początkujących </div> </div> <div>  </div> <div>  Graficzny interfejs użytkownika - Swing </div>	<p>Możemy uruchomić poprzedni przykład i zobaczyć, jak "FlowLayout" zarządza widokiem.</p>

Człowiek - najlepsza inwestycja

Slajd
19


JAVA
dla początkujących

```

JLabel label1 = new JLabel("Mój napis 1");
mainFrame.add(label1);
JLabel label2 = new JLabel("Mój napis 2");
mainFrame.add(label2);
JLabel label3 = new JLabel("Mój napis 3");
mainFrame.add(label3);

JPanel jPanel = new JPanel();
jPanel.setBackground(Color.YELLOW);
JLabel label4 = new JLabel("Mój napis 4");
jPanel.add(label4);
JLabel label5 = new JLabel("Mój napis 5");
jPanel.add(label5);



mainFrame.add(jPanel);
mainFrame.setVisible(true);
        
```



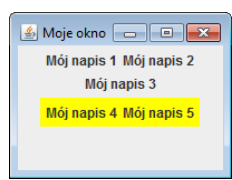
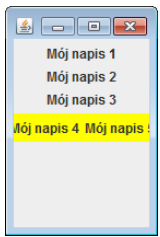
Graficzny interfejs użytkownika - Swing


Korzystając z klasy "JPanel" możemy grupować elementy. "Mój napis 4" i "Mój napis 5" będą znajdować się wewnątrz panelu.

Slajd
20

JAVA
dla początkujących





Graficzny interfejs użytkownika - Swing

Możemy zaobserwować, że przemieszczany jest cały panel. Dzieje się tak, ponieważ "JPanel" ma własnego zarządcę rozkładu i ustawiony "FlowLayout" nie zarządza komponentami z żółtego pola.

Człowiek - najlepsza inwestycja

Slajd
21

JAVA
 dla początkujących

```

public class Start {
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 200, 150);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(new BorderLayout());

        JLabel label1 = new JLabel("Mój napis 1");
        mainFrame.add(label1, BorderLayout.NORTH);
        JLabel label2 = new JLabel("Mój napis 2");
        mainFrame.add(label2, BorderLayout.WEST);
        JLabel label3 = new JLabel("Mój napis 3");
        mainFrame.add(label3, BorderLayout.EAST);
        JLabel label4 = new JLabel("Mój napis 4");
        mainFrame.add(label4, BorderLayout.SOUTH);
        JLabel label5 = new JLabel("Mój napis 5");
        mainFrame.add(label5, BorderLayout.CENTER);

        mainFrame.setVisible(true);
    }
}
    
```

Graficzny interfejs użytkownika - Swing

Kolejnym przykładem jest "BorderLayout". Jeżeli nie wywołamy metody "setLayout", jest on domyślnym zarządcą. Dodając każdy element podajemy dodatkowo jeden z parametrów: "NORTH", "SOUTH", "WEST", "EAST" i "CENTER".

Slajd
22








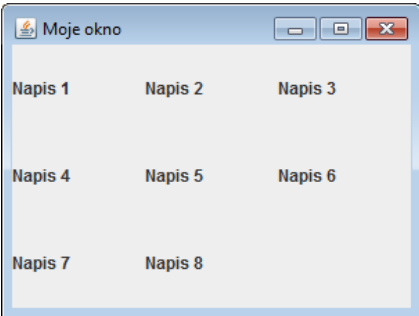


JAVA
 dla początkujących



Graficzny interfejs użytkownika - Swing



Zmieniając rozmiar okna możemy zaobserwować gdzie poszczególne obszary się znajdują.

Człowiek - najlepsza inwestycja

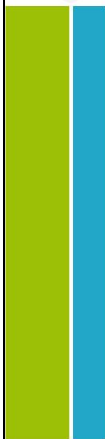
Slajd 23	<div>   <div> JAVA dla początkujących </div> </div> <pre> public class Start { public static void main(String [] args) { JFrame mainFrame = new JFrame("Moje okno"); mainFrame.setBounds(200, 200, 200, 150); mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); mainFrame.setLayout(new GridLayout(3, 2)); mainFrame.add(new JLabel("Napis 1")); mainFrame.add(new JLabel("Napis 2")); mainFrame.add(new JLabel("Napis 3")); mainFrame.add(new JLabel("Napis 4")); mainFrame.add(new JLabel("Napis 5")); mainFrame.add(new JLabel("Napis 6")); mainFrame.add(new JLabel("Napis 7")); mainFrame.add(new JLabel("Napis 8")); mainFrame.setVisible(true); } } </pre> <div>  Graficzny interfejs użytkownika - Swing </div>	"GridLayout" układa komponenty podobnie jak tabela. Tworząc go należy podać liczbę kolumn i wiersów.
Slajd 24	<div>   <div> JAVA dla początkujących </div> </div>  <div>  Graficzny interfejs użytkownika - Swing </div>	

Człowiek - najlepsza inwestycja

Slajd
25

 **B2E**
BUSINESS TO EDUCATION
  SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Ćwiczenie:

Popraw formularz tak, aby dostosowywał się do rozmiaru okna. Możesz określić minimalny rozmiar korzystając z metody:

```
mainFrame.setMinimumSize(new Dimension(200, 150));
```

W ten sposób będziemy mieli pewność, że użytkownik nie zmniejszy okna zbyt mocno i zawsze wszystkie elementy będą widoczne.



 **DAILY**
G R O U P
 Graficzny interfejs użytkownika - Swing

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 207

Slajd
 26


JAVA
 dla początkujących

Co siedzi w środku?

Interfejs graficzny użytkownika jest renderowany przy użyciu tylko jednego wątku.

EDT – event dispatching thread – jest odpowiedzialny za rysowanie i uaktualnianie widoku, jest to jedyny wątek który powinien zajmować się renderowaniem.

SwingUtilities.invokeLater(Runnable)
 SwingUtilities.invokeAndWait(Runnable)
 SwingWorker




Graficzny interfejs użytkownika - Swing

GUI w Swing'u jest renderowane i uaktualniane tylko przez 1 wątek – EDT. Model ten ułatwia architekturę framework'u, ale za razem wymaga ostrożnego obchodzenia się z nim. Przypuśćmy iż po naciśnięciu klawisza chcielibyśmy rozpocząć pobieranie dużego pliku z Internetu. Wątek EDT rozpoczyna wykonywać kod obsługi zdarzenia kliknięcia przycisku, rozpoczyna pobieranie i czeka, i czeka... W międzyczasie GUI się „zawiesza”, nie ma żadnego wolnego wątku który by mógł obsłużyć zdarzenia takie jak ruch myszą, kliknięcie innego przycisku czy wpisywanie tekstu. Dlatego dobrym pomysłem jest uruchomienie czasochłonnych zadań w tle, w osobnym wątku. Oczywiście możemy stworzyć nowy wątek i przekazać mu czasochłonne zadanie, wtedy GUI będzie nadal obsługiwane płynnie przez EDT. Jednak nadejdzie czas kiedy zadanie się skończy i wypadłoby zasygnalizować to jakoś użytkownikowi. Np. poprzez dodanie informacji do listy że plik został już pobrany. I tu pojawia się problem, gdyż nie powinniśmy z naszego dodatkowego wątku ingerować w komponenty GUI. Z pomocą przychodzą nam metody z klasy SwingUtilities. Przekazujemy tam obiekt implementujący Runnable, którego metoda run() zostanie wywołana już w ramach EDT. Uwaga ciało metody run() powinno się szybko wykonywać aby nie zamrozić GUI. W ten sposób możemy uaktualniać bezpiecznie widok z dowolnego wątku.


Drugim rozwiązaniem jest użycie klasy SwingWorker, gdzie tworzenie dodatkowego wątku do wykonania czasochłonnej operacji, informowanie o postępie procentowym tego zadania oraz zakończenie działania już w wątku EDT zostały znacznie ułatwione.

Człowiek - najlepsza inwestycja

Slajd
27



B2E
BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```

public class DiagonalLayout implements LayoutManager {
    @Override
    public void addLayoutComponent(String name, Component comp) {}
    @Override
    public void layoutContainer(Container parent) {
        Component[] tab = parent.getComponents();
        Dimension dim = parent.getSize();
        int h = dim.height / (tab.length + 6);
        int w = dim.width / (tab.length + 2);
        for (int i=0 ; i<tab.length ; i++) {
            tab[i].setBounds(tab[i].getWidth(), tab[i].getHeight(),
                             w * (i+1), h * (i+1));
        }
    }
    @Override
    public Dimension minimumLayoutSize(Container parent) {
        return null;
    }
    @Override
    public Dimension preferredLayoutSize(Container parent) {
        return null;
    }
    @Override
    public void removeLayoutComponent(Component comp) {}
}
                
```




Graficzny interfejs użytkownika - Swing

Swing umożliwia nam napisanie własnego zarządcy rozkładu. Wystarczy zaimplementować "LayoutManager". Zaimplementujemy przynajmniej metodę "layoutContainer". Odpowiada ona za ponowne rozmieszczenie komponentów po zmianie rozmiaru okna.


Powyższy przykład układa elementy wzdłuż przekątnej okna.

Człowiek - najlepsza inwestycja

Slajd
28



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

```


public class Start {
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 200, 150);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(new DiagonalLayout());

        mainFrame.add(new JLabel("Napis 1"));
        mainFrame.add(new JLabel("Napis 2"));
        mainFrame.add(new JLabel("Napis 3"));
        mainFrame.add(new JLabel("Napis 4"));
        mainFrame.add(new JLabel("Napis 5"));
        mainFrame.add(new JLabel("Napis 6"));
        mainFrame.add(new JLabel("Napis 7"));
        mainFrame.add(new JLabel("Napis 8"));


        mainFrame.setMinimumSize(new Dimension(200, 150));
        mainFrame.setVisible(true);
    }
}

```



Graficzny interfejs użytkownika - Swing

Slajd
29

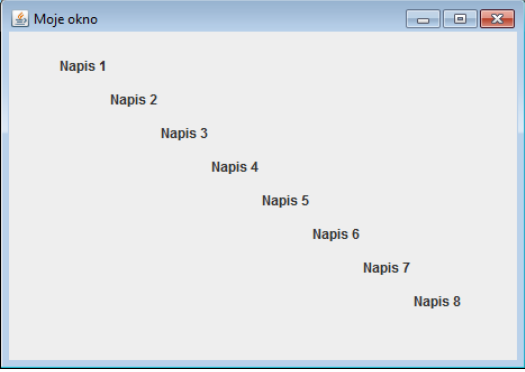



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących







Graficzny interfejs użytkownika - Swing

Człowiek - najlepsza inwestycja

Slajd
30



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Ćwiczenie:

Napisz własnego zarządcę rozkładu („Layout”), aby uzyskać taki efekt:

Tekst 1


Tekst 1

Tekst 1

Tekst 1


Tekst 1

Tekst 1




KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Człowiek - najlepsza inwestycja



UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Graficzny interfejs użytkownika - Swing

```
import java.awt.Component;
```

```
import java.awt.Container;
```

```
import java.awt.Dimension;
```

```
import java.awt.LayoutManager;
```

```
public class DiagonalLayout implements LayoutManager {
```

```
    @Override
```

```
    public void addLayoutComponent(String name, Component comp) {}
```

Człowiek - najlepsza inwestycja

@Override

```
public void layoutContainer(Container parent) {  
  
    Component[] tab = parent.getComponents();  
  
    Dimension dim = parent.getSize();  
  
    int h = dim.height;  
  
    int w = dim.width;  
  
  
    tab[0].setBounds(tab[0].getWidth(), tab[0].getHeight(), w/2, h/5);  
  
  
  
    tab[1].setBounds(tab[1].getWidth(), tab[1].getHeight(), w/3, (h*2)/5);  
    tab[2].setBounds(tab[2].getWidth(), tab[2].getHeight(), (w*2)/3, (h*2)/5);  
  
  
  
    tab[3].setBounds(tab[3].getWidth(), tab[3].getHeight(), w/4, (h*3)/5);  
    tab[4].setBounds(tab[4].getWidth(), tab[4].getHeight(), (w*2)/4, (h*3)/5);  
    tab[5].setBounds(tab[5].getWidth(), tab[5].getHeight(), (w*3)/4, (h*3)/5);  
  
}
```

@Override

```
public Dimension minimumLayoutSize(Container parent) {  
  
    return null;  
  
}
```

@Override

```
public Dimension preferredLayoutSize(Container parent) {
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



	<pre> return null; } @Override public void removeLayoutComponent(Component comp) {} } </pre>
--	--

8.9.3 Opis założonych osiągnięć ucznia

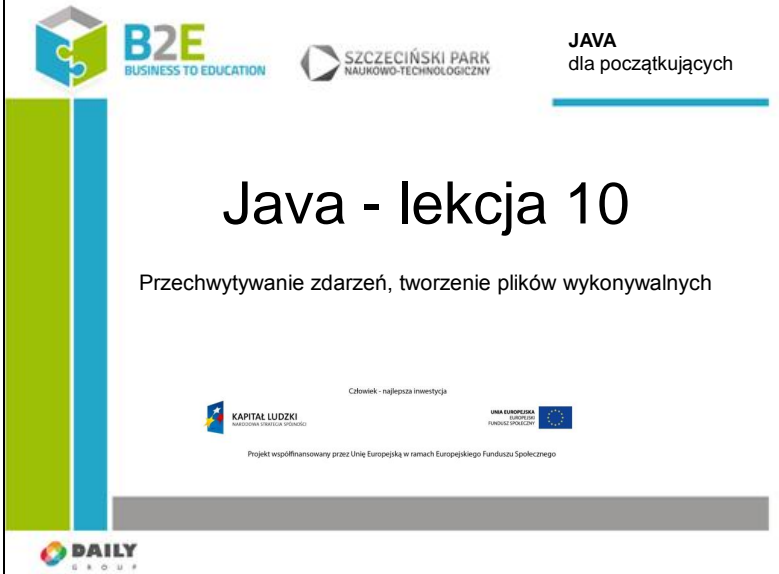
Po tej lekcji uczniowie będą potrafili stworzyć proste okno ze skonfigurowanymi komponentami. Obsługa zmiany rozmiaru okna.

8.10 Lekcja 10 - Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

8.10.1 Cel lekcji

Celem lekcji jest pokazanie jak prawidłowo obsługiwać zdarzenia takie jak: ruchy kursora, starowania klawiatura. Omówiona zostanie budowa plików wykonywalnych jar.

8.10.2 Treść - slajdy z opisem

Slajd 1		
------------	--	--

Człowiek - najlepsza inwestycja

Slajd
2



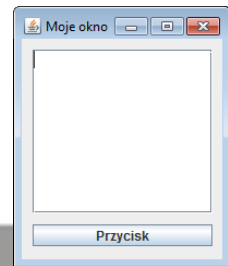
JAVA dla początkujących

```
public class Start {
    public static void main(String [] args) {
        JFrame mainFrame = new JFrame("Moje okno");
        mainFrame.setBounds(200, 200, 300, 240);
        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        mainFrame.setLayout(null);

        JTextArea jTextArea = new JTextArea();
        JScrollPane jScrollPane = new JScrollPane(jTextArea);
        jScrollPane.setBounds(10, 10, 265, 150);

        JButton jButton = new JButton("Przycisk");
        jButton.setBounds(10, 170, 265, 20);

        mainFrame.add(jScrollPane);
        mainFrame.add(jButton);
        mainFrame.setVisible(true);
    }
}
```



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Człowiek - najlepsza inwestycja

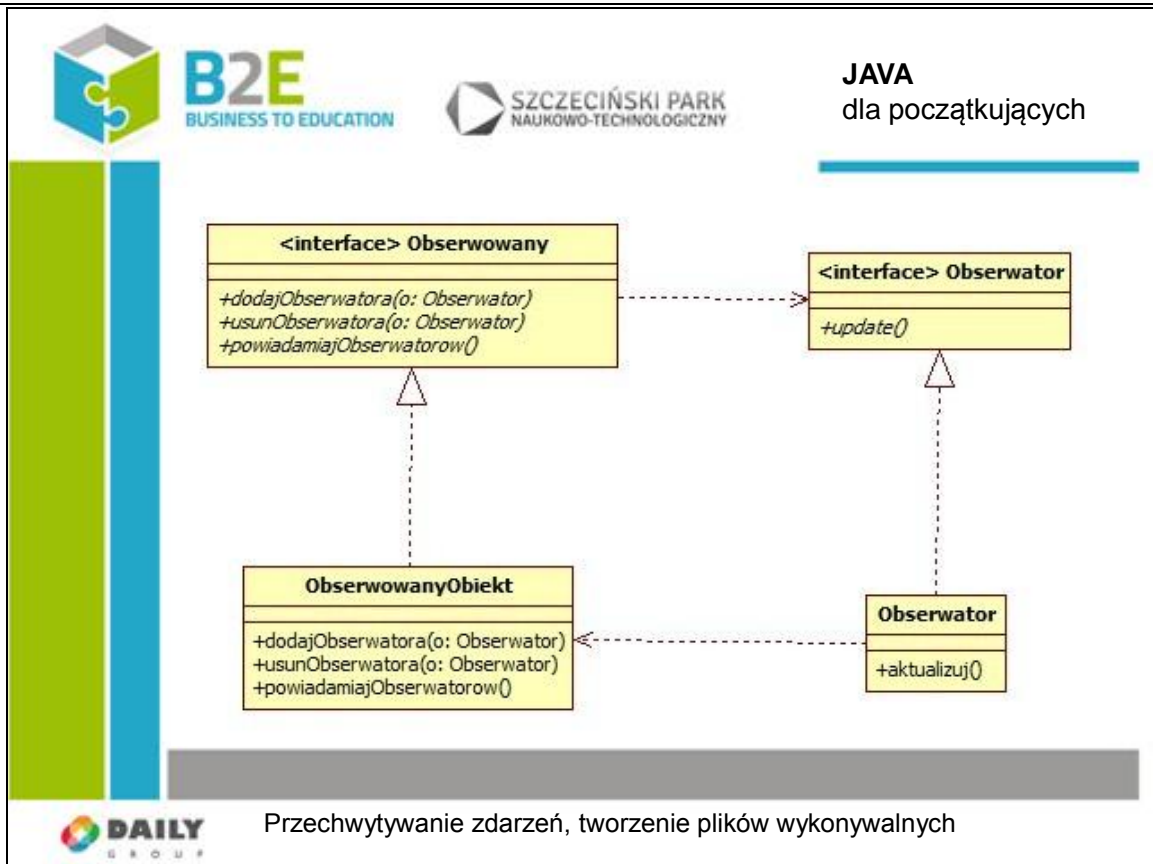


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Slajd
3



W programowaniu obiektowym obiekty posiadają pewien stan, tj. zbiór aktualnych wartości pól obiektu, który w wyniku wykonywania na nich operacji może ulegać zmianie. Od bieżącego stanu mogą być zależne inne obiekty, dlatego musi istnieć możliwość ich powiadomienia o jego zmianie tak, aby mogły one się do niej dostosować. Możemy także żądać, aby inne obiekty były powiadamiane o tym, że inny obiekt próbuje wykonać konkretną czynność, np. ponownie nawiązywać utracone połączenie z bazą danych. Pragniemy zaimplementować ogólny mechanizm, który umożliwi nam osiągnięcie tych celów.

We wzorcu obserwator wyróżniamy dwa podstawowe typy obiektów:

obserwowany (ang. observable, subject) – obiekt, o którym chcemy uzyskiwać informacje,
 obserwator (ang. observer, listener) – obiekty oczekujące na powiadomienie o zmianie stanu obiektu obserwowanego.

Kiedy stan obiektu obserwowanego się zmienia, wywołuje on metodę "powiadomObserwatorow()", która wysyła powiadomienia do wszystkich zarejestrowanych obserwatorów.

Obserwator jest stosowany w aplikacjach z graficznym interfejsem użytkownika.

Człowiek - najlepsza inwestycja

Slajd
4



JAVA dla początkujących

```
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import javax.swing.JTextArea;

public class MyListener implements MouseListener {
    private JTextArea jTextArea;
    public MyListener(JTextArea jTextArea) {
        this.jTextArea = jTextArea;
    }
    @Override
    public void mouseClicked(MouseEvent arg0) {
        jTextArea.setText(jTextArea.getText()
            + "\nnastąpiło kliknięcie");
    }
    @Override
    public void mouseEntered(MouseEvent arg0) {
        jTextArea.setText(jTextArea.getText()
            + "\nkursor wszedł w obszar przycisku");
    }
}
```



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Człowiek - najlepsza inwestycja





KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Slajd
5


JAVA
dla początkujących

```

@Override
public void mouseExited(MouseEvent arg0) {
    JTextArea.setText(JTextArea.getText()
        + "\nkursor wyszedł z obszaru przycisku");}



@Override
public void mousePressed(MouseEvent arg0) {
    JTextArea.setText(JTextArea.getText()
        + "\nprzycisk naciśnięty");}

@Override
public void mouseReleased(MouseEvent arg0) {
    JTextArea.setText(JTextArea.getText()
        + "\nprzycisk zwolniony");}
    
```

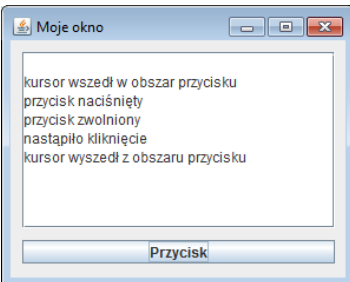



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Slajd
6

JAVA
dla początkujących





Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 217

```
import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.JTextArea;

public class MyKeyboardListener implements KeyListener {

    private JTextArea jTextArea;

    public MyKeyboardListener(JTextArea jTextArea) {

        this.jTextArea = jTextArea;
    }
}
```

```

    }

    @Override
    public void keyPressed(KeyEvent e) {

        JTextArea.setText(JTextArea.getText()

            + "\nnastąpiło wciśnięcie klawisza "

            + e.getKeyChar());

    }

    @Override
    public void keyReleased(KeyEvent e) {

        JTextArea.setText(JTextArea.getText()

            + "\nnastąpiło zwolnienie klawisza "

            + e.getKeyChar());

    }

    @Override
    public void keyTyped(KeyEvent e) {

        JTextArea.setText(JTextArea.getText()

            + "\nzostał wpisany znak "

            + e.getKeyChar());

    }


}

jButton.addKeyListener(new MyKeyboardListener(jTextArea));


```

Człowiek - najlepsza inwestycja

Slajd
8



B2E
 BUSINESS TO EDUCATION




SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących

Ćwiczenie:

Stwórz okno z komponentem „JLabel”. Niech komponent porusza się po oknie za pomocą strzałek.
 Jeżeli zmienisz pozycję elementu, wywołaj metodę „repaint”.



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

```
public class Start {

    public static void main(String [] args) {

        JFrame mainFrame = new JFrame("Moje okno");

        mainFrame.setBounds(200, 200, 300, 240);

        mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        mainFrame.setLayout(null);

        JLabel jLabel = new JLabel("Tekst");

        jLabel.setBounds(50, 50, 50, 20);

        mainFrame.addKeyListener(new MyKeyListener(jLabel, mainFrame));

        mainFrame.add(jLabel);

        mainFrame.setVisible(true);

    }

}
```

Człowiek - najlepsza inwestycja

```

    }

}

public class MyKeyListener implements KeyListener {

    private JLabel jLabel;

    private JFrame jFrame;

    private int x = 50;

    private int y = 50;

    public MyKeyListener(JLabel jLabel, JFrame jFrame) {

        this.jLabel = jLabel;

        this.jFrame = jFrame;}

    @Override

    public void keyPressed(KeyEvent e) {

        int d = 10;

        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {

            x += x>100 ? 0 : d;

            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());

        } else if (e.getKeyCode() == KeyEvent.VK_LEFT) {

            x -= x<10 ? 0 : d;

            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());

        } else if (e.getKeyCode() == KeyEvent.VK_UP) {

            y -= y<10 ? 0 : d;

            jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight());

        } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {

            y += y>100 ? 0 : d;
    
```




Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



	<pre>jLabel.setBounds(x, y, jLabel.getWidth(), jLabel.getHeight()); } JFrame.repaint(); } @Override public void keyReleased(KeyEvent e) {} @Override public void keyTyped(KeyEvent e) {} }</pre>	
Slajd 9	<div><div><div>JAVA dla początkujących</div></div><div><pre>public class Start { public class MyInnerClass{ } public static void main(String [] args) { JFrame mainFrame = new JFrame("Moje okno"); mainFrame.setBounds(200, 200, 300, 240); mainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); } }</pre></div><div>Przechwytywanie zdarzeń, tworzenie plików wykonywalnych</div></div>	Do przykładu z początku lekcji dodaj klasę wewnętrzną i zapisz zmiany. Dla pewności uruchom program.

Człowiek - najlepsza inwestycja

Slajd
10



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Foldery i pliki:

- 📁 .settings
- 📁 bin
- 📁 src
- 📄 .classpath
- 📄 .project

Pliki źródłowe:

- 📄 MyKeyboardListener.java
- 📄 MyListener.java
- 📄 Start.java

Pliki klas:

- 📄 MyKeyboardListener.class
- 📄 MyListener.class
- 📄 Start\$MyInnerClass.class
- 📄 Start.class



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

W projekcie znajdują się trzy pliki źródłowe:

- MyKeyboardListener.java,
- MyListener.java,
- Start.java.

Wejdź do folderu workspace, następnie do folderu z tym projektem. W środku znajdują się dwa foldery "src" i "bin".

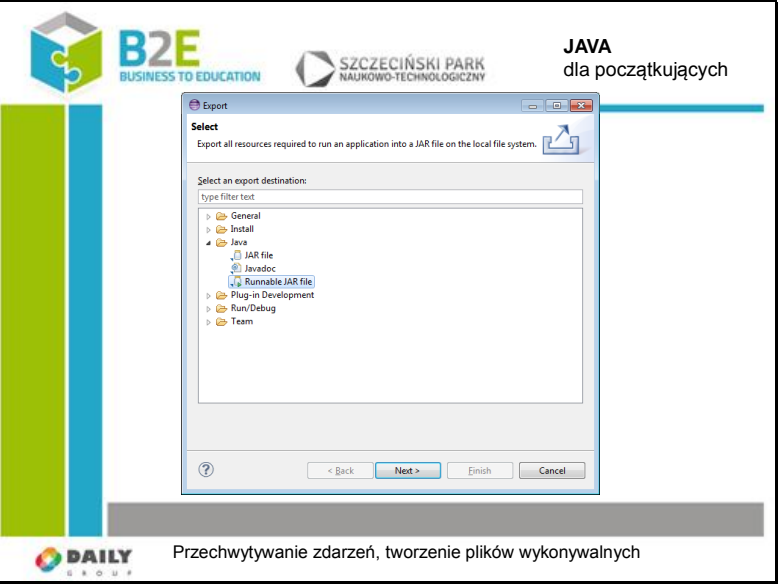
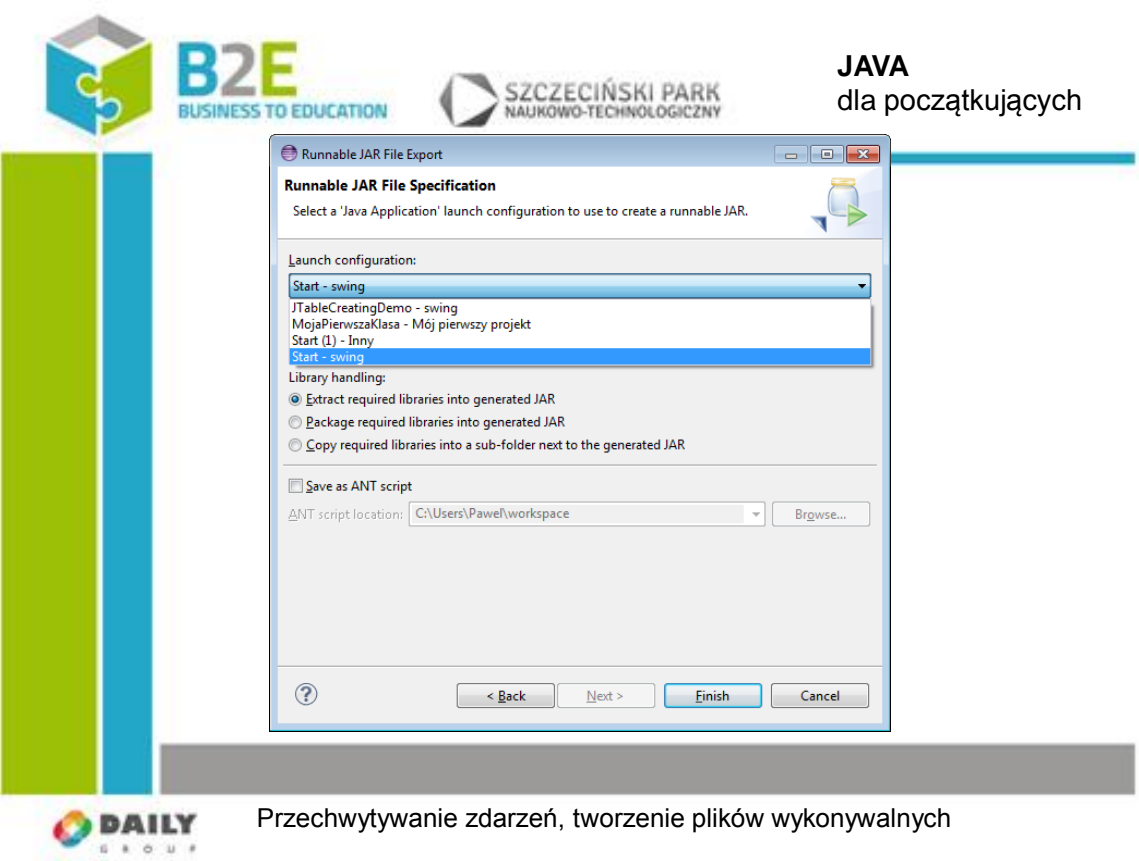
Sprawdź ich zawartość. W folderze "src" znajdują się pliki źródłowe i są trzy.

Dlaczego w folderze są cztery pliki?

Na jednej z pierwszych lekcji powiedziane było, że w jednym pliku znajdować się może tylko jedna klasa.



W plikach źródłowych mogą być klasy wewnętrzne, ale po procesie kompilacji każda klasa stanowi osobny plik.

Człowiek - najlepsza inwestycja

	W klasie "Start" jest klasa wewnętrzna "MyInnerClass". Po skompilowaniu jest osobnym plikiem. Po nazwie można łatwo rozpoznać, z której klasy została wyodrębniona.	
Slajd 11	 <p>Przechwytywanie zdarzeń, tworzenie plików wykonywalnych</p>	<p>W środowisku Eclipse kliknij prawym przyciskiem na projekt i użyj opcji eksport.</p> <p>Wybierz opcję "Java/Runnable JAR file".</p>
Slajd 12	 <p>Przechwytywanie zdarzeń, tworzenie plików wykonywalnych</p>	<p>Wybierz plik, który służy do uruchamiania aplikacji. Następnie podaj nazwę i lokalizację pliku.</p>

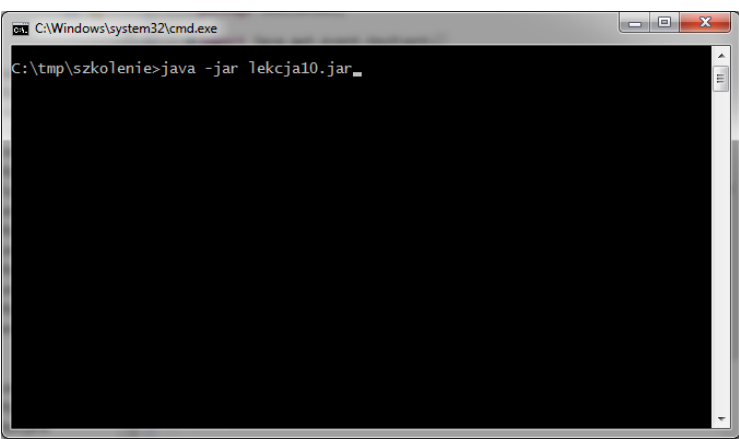
Człowiek - najlepsza inwestycja


Slajd
13

SHCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

Uruchom konsolę i wywołaj polecenie tak, jak na slajdzie.

Powinno nastąpić prawidłowe uruchomienie programu.

Zmień rozszerzenie pliku z "jar" na "zip" i rozpakuj go.

Zauważ, że jest to zwykłe archiwum zip. Znajdują się w nim wszystkie pliki bajtkodu, należące do danego projektu.

W folderze "META-INF" znajduje się plik manifestu. Zobacz jego zawartość.

Manifest-Version: 1.0



Class-Path: .

Main-Class: Start

W ostatnim wersie jest nazwa klasy, z której wywoływany jest program.

Człowiek - najlepsza inwestycja


Slajd
14


JAVA
 dla początkujących

Ćwiczenie:


Zbuduj plik „*.jar” samodzielnie. Umieść w środku plik o nazwie „Hello”. Niech efektem będzie wyświetlenie w konsoli napisu „Hello world!”.



Człowiek - najlepsza inwestycja



Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego



Przechwytywanie zdarzeń, tworzenie plików wykonywalnych

8.10.3 Opis założonych osiągnięć ucznia

Po tej lekcji uczniowie będą umieli obsługiwać zdarzenia systemowe związane z kursorem i klawiaturą. Będą rozumieć jak wygląda tworzenie plików wykonywalnych.


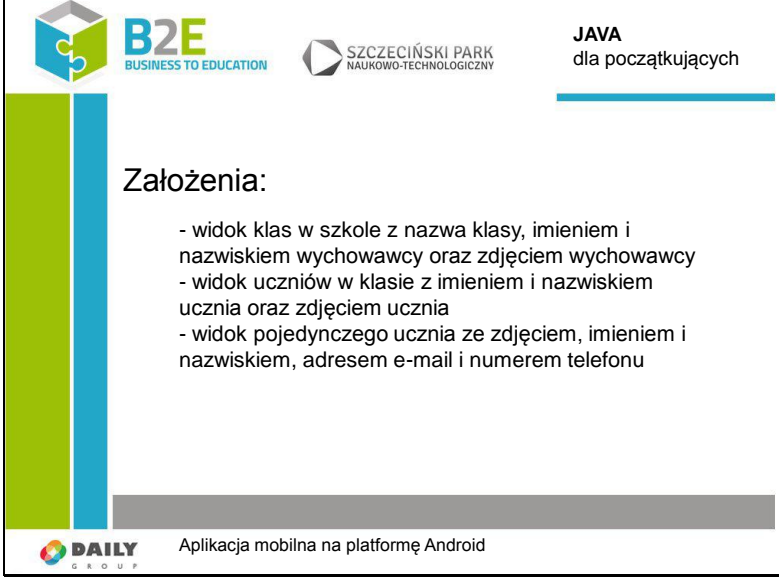
8.11 Lekcja 11 – Projekt Interdyscyplinarny

8.11.1 Cel lekcji



Celem lekcji jest zdobycie umiejętności wykorzystania dotychczasowej wiedzy podczas programowania aplikacji mobilnych na platformie Android.

Człowiek - najlepsza inwestycja

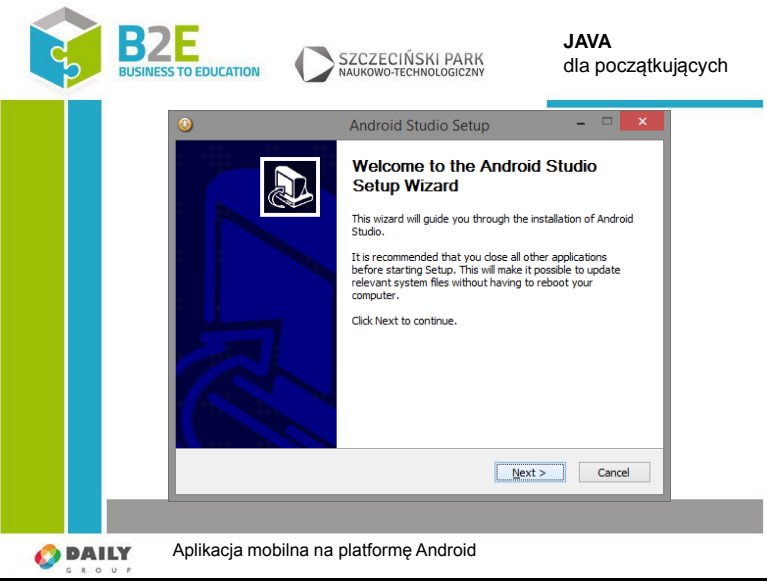
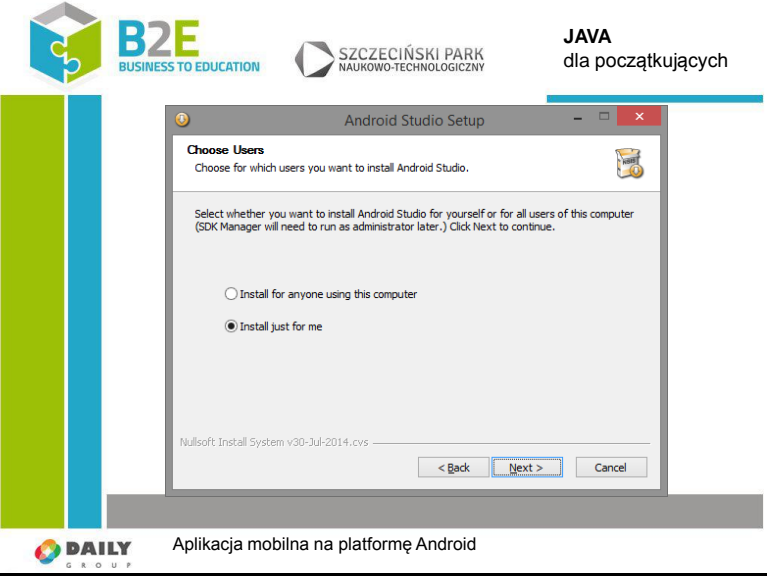
8.11.2 Treść – slajdy z opisem

<p>Slajd 1</p>		
<p>Slajd 2</p>		<p>Aplikacja będzie służyć do prezentacji danych klas oraz uczniów w szkole. Jako dodatkowa funkcjonalność będzie pozwalać na szybkie wybranie numeru ucznia i wykonanie telefonu lub wysłanie e-mail'a.</p>

Człowiek - najlepsza inwestycja



<p>Slajd 3</p>	 <p>JAVA dla początkujących</p> <p>Przygotowanie środowiska pracy:</p> <p>www.oracle.com</p> <p>Java JRE (Java Runtime Environment)</p> <p>Java JDK (Java Development Kit)</p> <p>https://developer.android.com/sdk/installing/studio.html</p> <p>Android Studio</p> <p>Aplikacja mobilna na platformę Android</p>	<p>Środowisko uruchomieniowe Javy, jak i pakiet developerski, mamy już zainstalowane.</p> <p>Na tej lekcji będziemy używali innego środowiska developerskiego. O ile możliwe jest tworzenie aplikacji na platformę Android w Eclipse, o tyle aktualnie zalecanym środowiskiem jest Android Studio. Wobec czego skorzystamy z doskonalszego IDE podczas tej lekcji.</p>
<p>Slajd 4</p>	 <p>JAVA dla początkujących</p> <p>Android Studio BETA</p> <p>Android Studio is a new Android development environment based on IntelliJ IDEA. It provides new features and improvements over Eclipse ADT and will be the official Android IDE once it's ready. On top of the capabilities you expect from IntelliJ, Android Studio offers:</p> <ul style="list-style-type: none"> • Flexible Gradle-based build system. • Build variants and multiple APK generation. • Expanded template support for Google Services and various device types. • Rich layout editor with support for theme editing. • List tools to catch performance, usability, version compatibility, and other problems. • ProGuard and app-signing capabilities. • Built-in support for Google Cloud Platform (GCP), making it easy to integrate Google Cloud Messaging and App Engine. <p>Caution: Android Studio is currently in beta. Some features are not yet implemented and you may encounter bugs. If you are not comfortable using an unfinished product, you may want to instead download (or continue to use) Eclipse with ADT.</p> <p>VIEW ALL DOWNLOADS AND SIZES</p> <p>SYSTEM REQUIREMENTS</p> <p>Aplikacja mobilna na platformę Android</p>	<p>Pobieramy oprogramowanie, lub korzystamy z dostarczonej wersji.</p>

Człowiek - najlepsza inwestycja

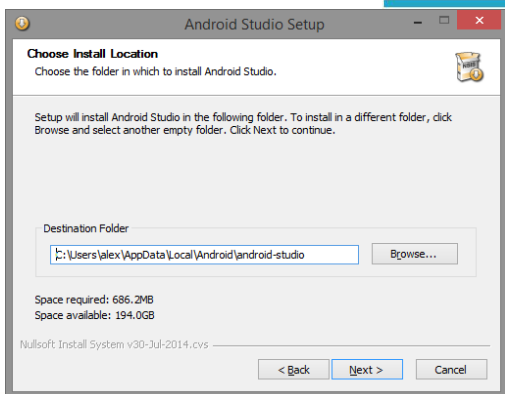
<p>Slajd 5</p>		<p>Rozpoczynamy instalację IDE</p>
<p>Slajd 6</p>		


Człowiek - najlepsza inwestycja

Slajd
7






JAVA
dla początkujących

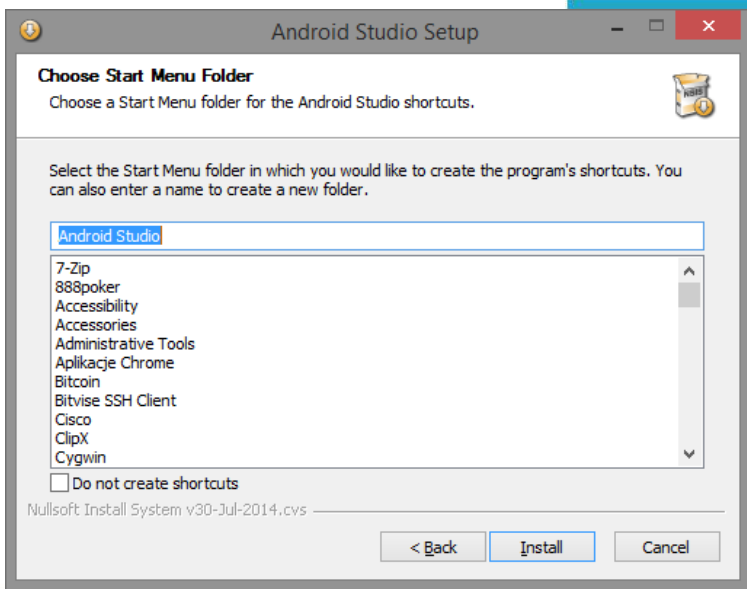




Aplikacja mobilna na platformę Android

Slajd
8



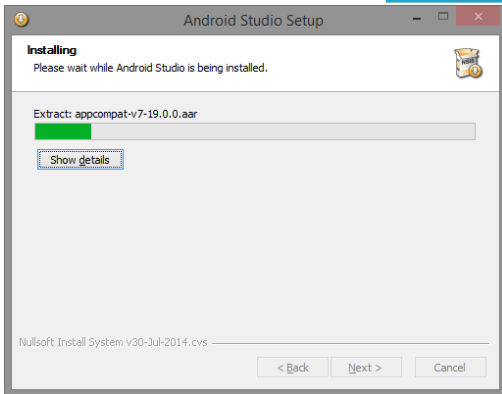



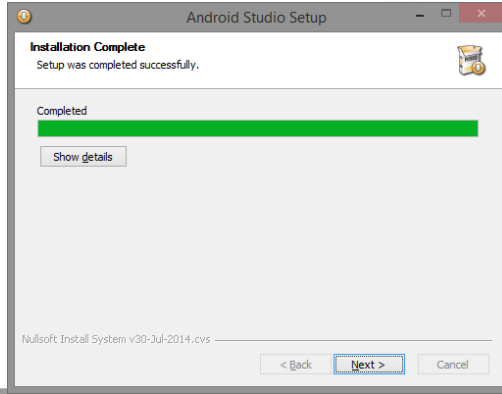




JAVA
dla początkujących




Aplikacja mobilna na platformę Android

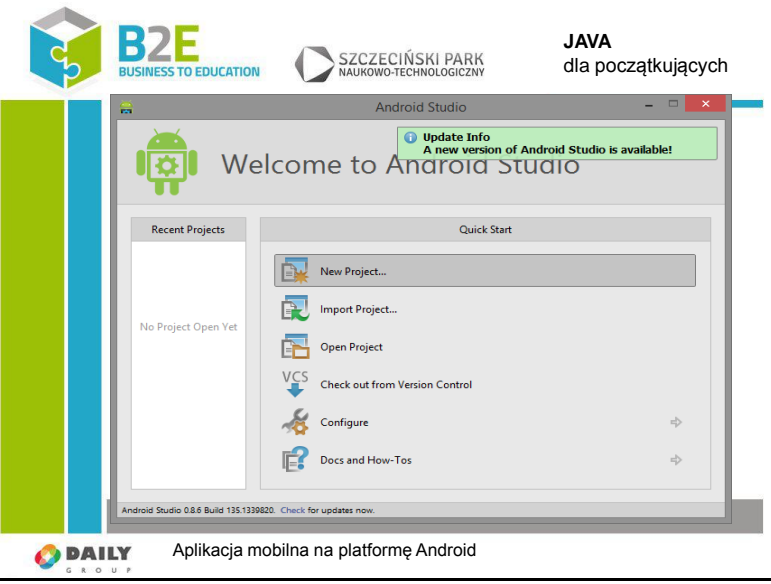
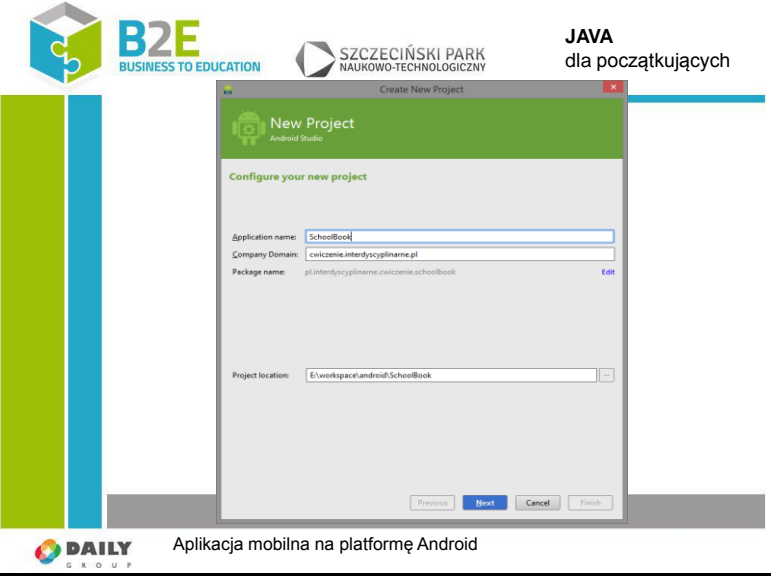
Człowiek - najlepsza inwestycja

<p>Slajd 9</p>	<div>   <div> JAVA dla początkujących </div> </div>  <div>  Aplikacja mobilna na platformę Android </div>	
<p>Slajd 10</p>	<div>   <div> JAVA dla początkujących </div> </div>  <div>  Aplikacja mobilna na platformę Android </div>	

Człowiek - najlepsza inwestycja


<p>Slajd 11</p>	 <p>Applikacja mobilna na platformę Android</p>	<p>Kończymy instalację i uruchamiamy Android Studio.</p>
<p>Slajd 12</p>	 <p>Applikacja mobilna na platformę Android</p>	<p>Jeśli pojawi się opcja importu ustawień, to odrzucamy to udogodnienie. W przeciwnym razie moglibyśmy zacząć pracować na niestandardowych ustawieniach poprzedniego użytkownika komputera.</p>

Człowiek - najlepsza inwestycja


<p>Slajd 13</p>	 <p>Aplicacja mobilna na platformę Android</p>	<p>Tworzymy nowy projekt.</p>
<p>Slajd 14</p>	 <p>Aplicacja mobilna na platformę Android</p>	<p>Nadajemy projektowi nazwę, domyślny pakiet oraz lokalizację na dysku twardym.</p>

Człowiek - najlepsza inwestycja

Slajd
15

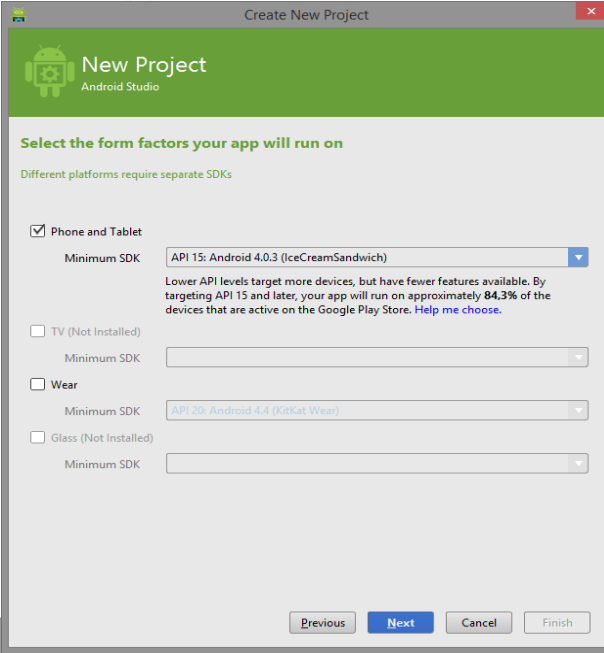



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących






Aplikacja mobilna na platformę Android

Wybieramy minimalną wersję API Androida od której nasza aplikacja będzie mogła zostać uruchomiona. Należy zwrócić uwagę że czym niższa wersja zostanie wybrana, tym więcej urządzeń będzie mogło uruchomić naszą aplikację. Jednak oznacza to również utratę nowych funkcjonalności dostępnych tylko w wyższych wersjach systemu. Istnieje możliwość emulacji nowych funkcji na starszych urządzeniach ale niestety wykracza to poza zakres tej lekcji.


Wybieramy w miarę nowe API na potrzeby tej lekcji.

Człowiek - najlepsza inwestycja

Slajd
16





B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



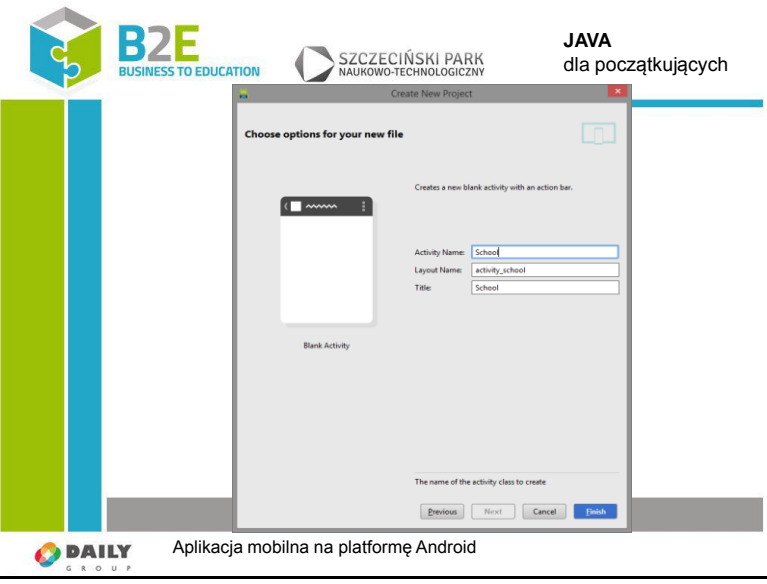



Aplikacja mobilna na platformę Android

Kolejnym krokiem jest wybór pustego „activity”. Aplikacje w Androidzie składają się z widoków, w nomenklaturze Androida jest to właśnie „activity”. Dla przykładu, nasza aplikacja będzie składać się z trzech widoków:

- listy klas w szkole,
- listy uczniów w klasie,
- szczegółów danego ucznia.

Człowiek - najlepsza inwestycja

<p>Slajd 17</p>	 <p>Aplikacja mobilna na platformę Android</p>	<p>Nadajemy nazwę nowemu widokowi i kończymy działanie kreatora. Musimy chwilę poczekać aż IDE wygeneruje nasz nowy projekt.</p>
<p>Slajd 18</p>	 <p>Aplikacja mobilna na platformę Android</p>	<p>Naszym oczom ukazuje się interfejs IDE i kod przykładowego pustego activity. Jak widać widok od razu jest renderowany aby developer mógł jak najszybciej widzieć wprowadzone zmiany. Istnieje również możliwość edycji wizualnej widoków, w tym celu trzeba się przełączyć na inną zakładkę „Design”.</p>

Człowiek - najlepsza inwestycja

Slajd
19



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





Aplikacja mobilna na platformę Android

Podczas tworzenia aplikacji możemy ją uruchamiać w emulatorze lub na prawdziwym sprzęcie. W ramach zajęć będziemy korzystać z emulatora. W tym celu należy zainstalować pliki emulujące interesujące nas urządzenie.

Uruchamiamy Android SDK manager.

Człowiek - najlepsza inwestycja

Slajd
20

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Aplikacja mobilna na platformę Android

Zaznaczamy „ARM EABI v7a System Image” i instalujemy. Trzeba chwilę poczekać.

Slajd
21

SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY



JAVA
dla początkujących

Aplikacja mobilna na platformę Android

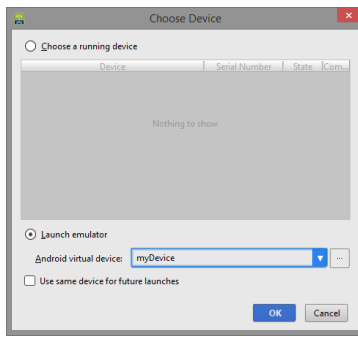
Po instalacji czas uruchomić naszą pierwszą aplikację na Androida. Naciskamy zielony przycisk strzałki w górnej części IDE.


Człowiek - najlepsza inwestycja

Slajd
22

JAVA
dla początkujących




Aplikacja mobilna na platformę Android

Pojawi się okno wyboru urządzenia. Powinno być puste, jako że nie dysponujemy jeszcze wirtualnymi urządzeniami. Zaznaczamy „launch emulator” i klikamy na przycisk z kropkami.

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI


UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY




Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

str. 239

Slajd
23

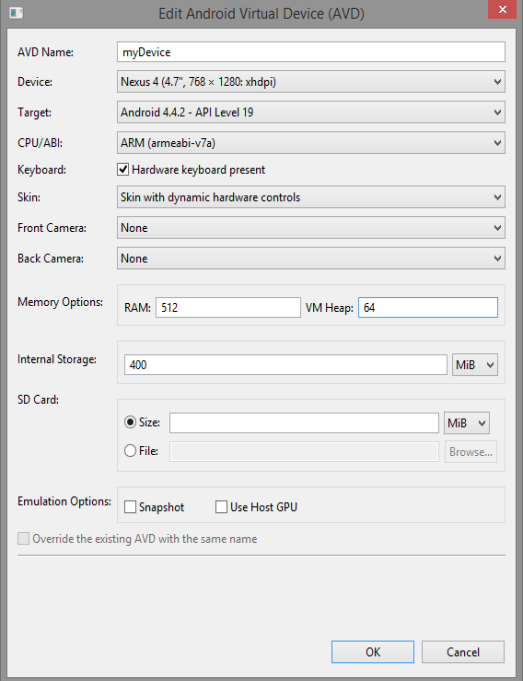



B2E
BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących





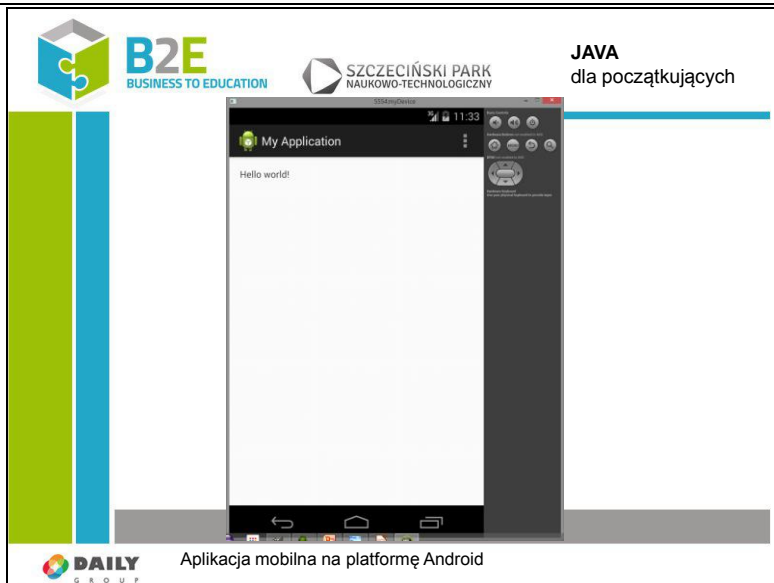
Aplikacja mobilna na platformę Android

Startuje AVD – Android Virtual Device Manager. Wciskamy przycisk „Create” i wypełniamy specyfikację techniczną wirtualnego urządzenia. Po czym klikamy OK i czekamy na utworzenie urządzenia.

Teraz z listy wybieramy nasze urządzenie i klikamy OK. Nasza aplikacja zostanie uruchomiona w emulatorze.

Człowiek - najlepsza inwestycja

Slajd
24




Aplikacja się uruchomiła. Proszę nie wyłączać emulatora, gdyż strasznie długo się uruchamia, za to można go wykorzystać wielokrotnie podczas kolejnych uruchomień naszej aplikacji.


Teraz przejrzymy najważniejsze pliki w projekcie i zaczniemy budować naszą aplikację.

Człowiek - najlepsza inwestycja

Slajd
25

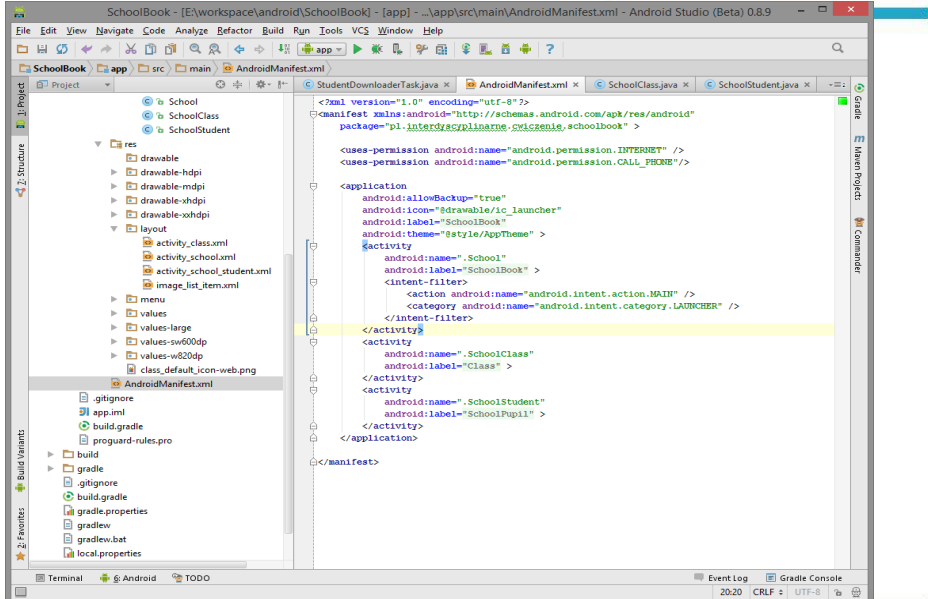


B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących





Aplikacja mobilna na platformę Android

Jednym z ważniejszych plików jest plik „AndroidManifest.xml”. Jest to plik konfiguracyjny aplikacji, gdzie można ustawić wiele opcji takich jak nazwę, czy ikonę programu. Zarejestrowane są tu też wszystkie widoki aplikacji, ale proszę ich nie dodawać ręcznie gdyż IDE zrobi to za nas w swoim czasie.

Kolejnym istotnym elementem są uprawnienia aplikacji. Aplikacja musi jasno zadeklarować z jakich funkcji systemu chce korzystać. Dzięki temu podczas instalowania aplikacji użytkownik może zdecydować czy aplikacja jest bezpieczna i czy chce ją uruchomić na swoim sprzęcie. My poprosimy o dwa przywileje: dostęp do Internetu – stamtąd będziemy pobierać informacje do wyświetlenia oraz o możliwość inicjowania połączeń głosowych. Proszę na początku pliku dodać:

```
<uses-permission android:name="android.permission.INTERNET" />
```

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY



Slajd
26



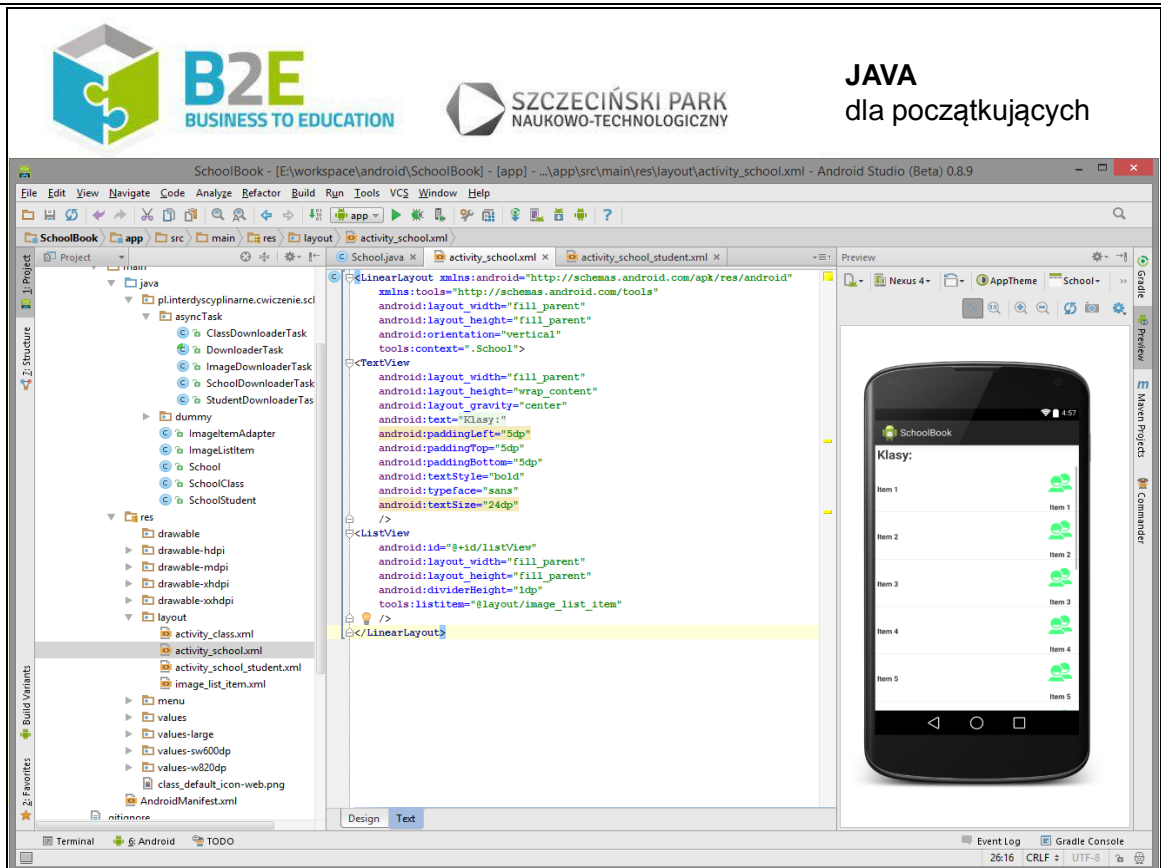
Activity jest plikiem xml'a definiującym strukturę widoku. W skład tej struktury mogą wchodzić różne komponenty, widgety, layouts i inne.

Do naszego widoku dodamy nagłówek oraz kontener ListView. Kontener ten służy do wyświetlania listy, czyli kolejnych wierszy z danymi tego samego typu. Niestety sposób reprezentacji danych może być różny w związku z czym to na nas spoczywa obowiązek zdefiniowania wyglądu wiersza listy. W następnym kroku stworzymy plik layoutu dla wiersza. Na slajdzie tym już w jednej z ostatnich linii wskazałem o który widok elementu listy mi chodzi:

`tools:listitem="@layout/image_list_item"`

Człowiek - najlepsza inwestycja

Slajd
27



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical"
```

```
    tools:context=".School">
```

```
<TextView
```

```
    android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content"
```

```
    android:layout_gravity="center"
```

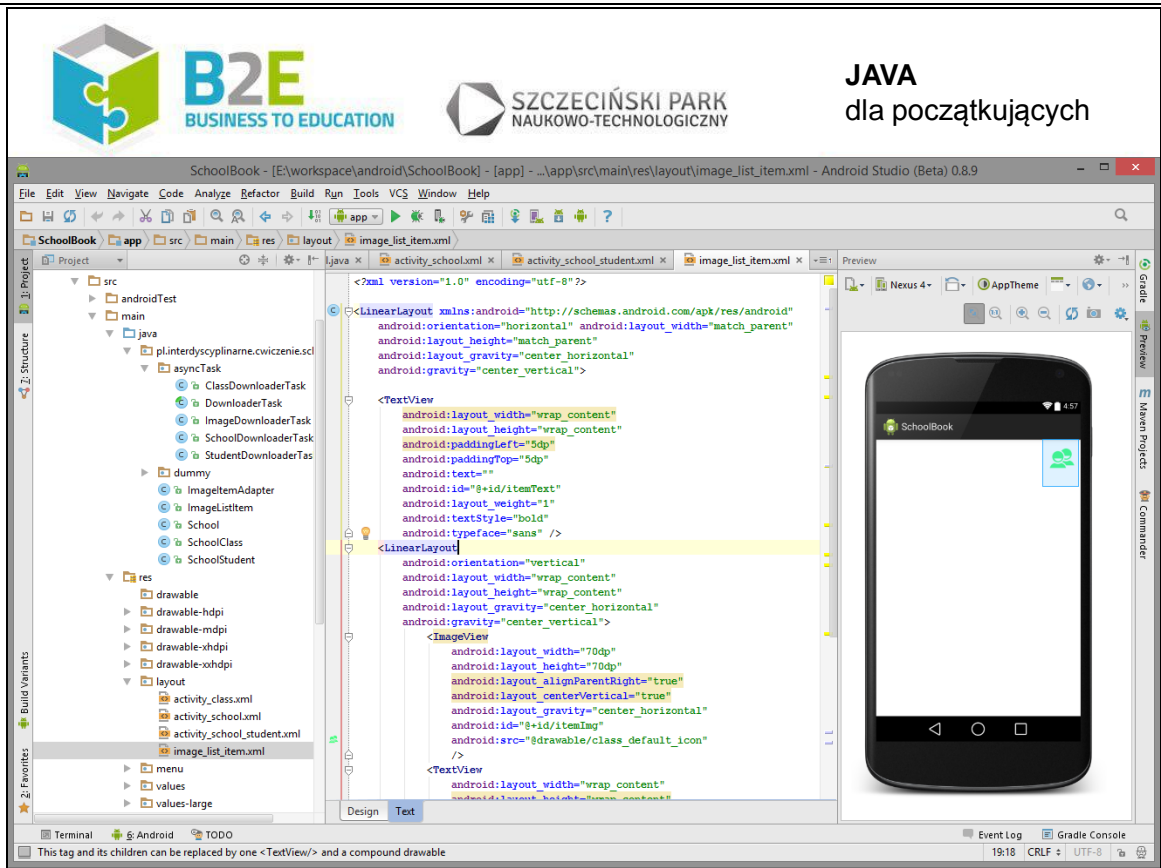
```
    android:text="@string/TitleSchool"
```

Człowiek - najlepsza inwestycja

	<pre> android:paddingLeft="5dp" android:paddingTop="5dp" android:paddingBottom="5dp" android:textStyle="bold" android:typeface="sans" android:textSize="24dp" /> <ListView android:id="@+id/listView" android:layout_width="fill_parent" android:layout_height="fill_parent" android:dividerHeight="1dp" tools:listitem="@layout/image_list_item" /> </LinearLayout></pre>
--	--

Człowiek - najlepsza inwestycja

Slajd
28



Klikamy na folderze layoutu prawym klawiszem myszy i wybieramy „new->Layout resource file” w celu zdefiniowania layoutu wiersza.

<LinearLayout

android:orientation="vertical"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

xmlns:android="http://schemas.android.com/apk/res/android"

android:clickable="false">

<RelativeLayout

android:layout_width="fill_parent"

Człowiek - najlepsza inwestycja

```

android:layout_height="100dp">

<LinearLayout

    android:orientation="vertical"

    android:layout_width="wrap_content"

    android:layout_height="fill_parent"

    android:layout_alignParentLeft="true"

    >

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_weight="1"

        android:gravity="center_vertical"

        android:text=""

        android:id="@+id/firstName"

        android:padding="5dp"

        android:textSize="15dp"

        android:textStyle="bold"

        android:typeface="sans" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_weight="1"

```

Człowiek - najlepsza inwestycja

```
        android:gravity="center_vertical"

        android:text=""

        android:id="@+id/lastName"

        android:padding="5dp"

        android:textSize="15dp"

        android:textStyle="bold"

        android:typeface="sans" />

</LinearLayout>

<ImageView

    android:id="@+id/studentPicture"

    android:src="@drawable/class_default_icon"

    android:layout_width="100dp"

    android:layout_height="100dp"

    android:layout_alignParentRight="true"/>

</RelativeLayout>

<RelativeLayout

    android:layout_width="fill_parent"

    android:layout_height="50dp">

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="fill_parent"
```

Człowiek - najlepsza inwestycja


```
android:gravity="center_vertical"
```

```
android:text=""
```

```
android:id="@+id/email"
```

```
android:padding="5dp"
```

```
android:textSize="15dp"
```

```
android:textStyle="bold"
```

```
android:typeface="sans" />
```

```
<ImageButton
```

```
android:layout_width="100dp"
```

```
android:layout_height="wrap_content"
```

```
android:id="@+id/btnMail"
```

```
android:layout_alignParentRight="true"
```

```
android:src="@android:drawable/sym_action_email"
```

```
android:background="@android:color/background_light"
```

```
android:padding="5dp" />
```

```
</RelativeLayout>
```

```
<RelativeLayout
```

```
android:layout_width="fill_parent"
```

```
android:layout_height="50dp">
```

```
<TextView
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



```
android:layout_width="wrap_content"
```

```
android:layout_height="fill_parent"
```

```
android:gravity="center_vertical"
```

```
android:text=""
```

```
android:id="@+id/phoneNumber"
```

```
android:padding="5dp"
```

```
android:textSize="15dp"
```

```
android:textStyle="bold"
```

```
android:typeface="sans" />
```

```
<ImageButton
```

```
android:layout_width="100dp"
```

```
android:layout_height="wrap_content"
```

```
android:id="@+id/btnCall"
```

```
android:layout_alignParentRight="true"
```

```
android:src="@android:drawable/ic_menu_call"
```

```
android:background="@android:color/background_light"
```

```
android:padding="5dp" />
```

```
</RelativeLayout>
```

```
</LinearLayout>
```

Człowiek - najlepsza inwestycja

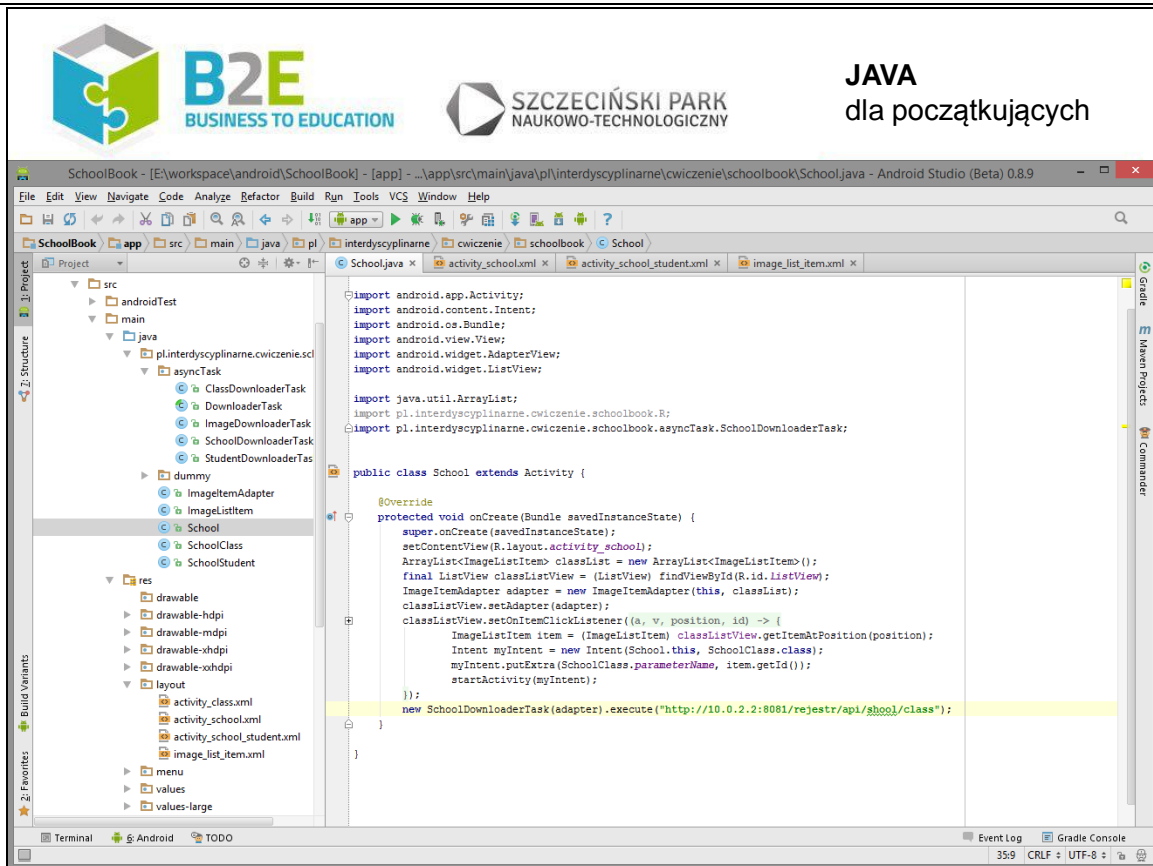


KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



Slajd
29



W ten sposób zakończyliśmy pierwszy etap dodawania nowego widoku. Kolejnym krokiem jest dodanie zachowania/funkcjonalności do widoku. Programowanie jak można było przypuszczać odbywa się w języku Java. Będziemy chcieli wyświetlić listę klas, dlatego za chwilę stworzymy pomocniczą klasę, która jest w stanie przechowywać wszystkie niezbędne informacje dla jednego wiersza. Klasę nazwiemy ImageListItem, gdyż będzie zawierać również informacje o zdjęciu wychowawcy.

Obiekty tej klasy przetrzymywać będziemy w ArrayList.

Następnym elementem układanki jest funkcja findViewById umożliwiająca pobranie referencji do elementu widoku. W naszym przypadku odszukujemy ListView.

Mając model danych i widok czas połączyć je razem. Do tego celu stworzymy kolejną klasę ImageListAdapter, zapełni nam ona widok wierszami pokazującymi dane z listy.

Ustawiamy adapter na naszym ListView.

Dodanie obsługi kliknięcia możemy na tym etapie pominąć.

Na samym dole jest jeszcze jedno ciekawe wywołanie. Ponieważ activity jest elementem GUI,

Człowiek - najlepsza inwestycja

a pobieranie listy klas z Internetu może długo potrwać, to z pewnością chcielibyśmy uniknąć efektu zamrożenia interfejsu użytkownika. W związku z czym pobieranie danych zlecimy do osobnego wątku.

Czas pokazać wymienione elementy szczegółowo.

```
package pl.interdyscyplinarne.cwiczenie.schoolbook;
```

```
import android.app.Activity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.AdapterView;
```

```
import android.widget.ListView;
```

```
import java.util.ArrayList;
```

```
import pl.interdyscyplinarne.cwiczenie.schoolbook.R;
```

```
import pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask.SchoolDownloaderTask;
```

```
public class School extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_school);
```

```
        ArrayList<ImageListItem> classList = new ArrayList<ImageListItem>();
```

```
        final ListView classListView = (ListView) findViewById(R.id.listView);
```

```
        ImageItemAdapter adapter = new ImageItemAdapter(this, classList);
```

```
        classListView.setAdapter(adapter);
```

Człowiek - najlepsza inwestycja

```
classListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
  
    @Override  
  
    public void onItemClick(AdapterView<?> a, View v, int position, long id) {  
  
        ImageListItem item = (ImageListItem) classListView.getItemAtPosition(position);  
  
        Intent myIntent = new Intent(School.this, SchoolClass.class);  
  
        myIntent.putExtra(SchoolClass.parameterName, item.getId());  
  
        startActivity(myIntent);  
  
    }  
  
});  
  
new SchoolDownloaderTask(adapter).execute();  
}  
  
}
```

Człowiek - najlepsza inwestycja



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY



JAVA

dla początkujących

```
{"_id":1,"nazwa_klasy":"1A","wychowawca":"Micha\u0142 Kowalski","zdjecie":"http://localhost:8081/rejestr/upload/nauczyciel2.jpg"}
```

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

```
package pl.interdyscyplinarne.cwiczenie.schoolbook;

import android.util.Log;

import org.json.JSONException;
import org.json.JSONObject;

/**
 * Created by alex on 2014-10-03.
 */
public class ImageListItem {

    private String itemText;
    private String itemText2;
    private String imgUrl;
    private String id;

    public ImageListItem(String text, String text2, String imgUrl, String id) {
        this.itemText = text;
        this.itemText2 = text2;
        this.imgUrl = imgUrl;
        this.id = id;
    }
}
```

Człowiek - najlepsza inwestycja

```
public static ImageListItem classItemFromJSON(JSONObject jsonObject) {  
    try {  
        return new ImageListItem(  
            jsonObject.getString("nazwa_klasy"),  
            jsonObject.getString("wychowawca"),  
            jsonObject.getString("zdjecie"),  
            jsonObject.getString("_id")  
        );  
    } catch (JSONException e) {  
        Log.w("ImageListItem", "Error while parsing klass image item");  
    }  
    return null;  
}  
  
public static ImageListItem studentItemFromJSON(JSONObject jsonObject) {  
    try {  
        return new ImageListItem(  
            jsonObject.getString("imie")+" "+jsonObject.getString("nazwisko"),  
            "",  
            jsonObject.getString("zdjecie"),  
            jsonObject.getString("_id")  
        );  
    } catch (JSONException e) {
```

Człowiek - najlepsza inwestycja


```
Log.w("ImageListItem", "Error while parsing student image item");

}

return null;

}

public String getItemText() {

    return itemText;

}

public void setItemText(String itemText) {

    this.itemText = itemText;

}

public String getImgUrl() {

    return imgUrl;

}

public void setImgUrl(String imgUrl) {

    this.imgUrl = imgUrl;

}

public String getItemText2() {

    return itemText2;

}
```

Człowiek - najlepsza inwestycja

```
public void setItemText2(String itemText2) {  
    this.itemText2 = itemText2;  
}  
  
public String getId() {  
    return id;  
}  
  
public void setId(String id) {  
    this.id = id;  
}  
  
}
```

Człowiek - najlepsza inwestycja





KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

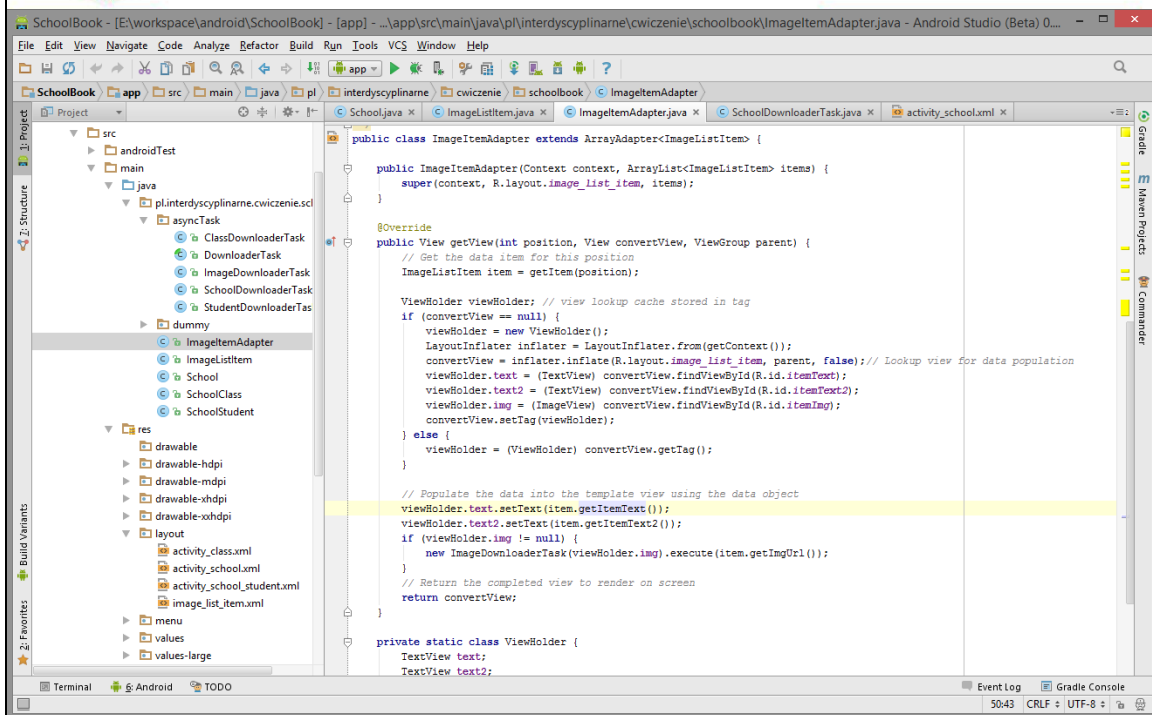


Slajd
31


B2E
 BUSINESS TO EDUCATION


SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących



ListView chcąc wyrenderować kolejny element listy, odwoła się do naszej klasy ImageItemAdapter w celu uzyskania widoku. Nasz adapter rozszerza domyślny adapter dla list, w związku z czym musimy tylko zaimplementować jedną metodę tworzącą nasz specjalizowany widok. Najpierw w instrukcji if sprawdzamy, czy przypadkiem wcześniej już nie stworzyliśmy widoku dla danego wiersza. Jeśli nie, to przy pomocy metody LayoutInflater.inflate „dmuchamy” widok z naszego xml'a i wydobywamy z niego interesujące nas elementy przy użyciu findViewById. Jeśli jednak mieliśmy już stworzony wcześniej widok wiersza, to nie tworzymy nowego, tylko uaktualnimy danymi już istniejący.

Poniżej instrukcji if ustawiamy dwa teksty, a operację ściągnięcia obrazka znów zlecamy osobnemu wątkowi.

```
package pl.interdyscyplinarne.cwiczenie.schoolbook;
```

```
import android.content.Context;
```

```
import android.graphics.Bitmap;
```

```
import android.graphics.BitmapFactory;
```

Człowiek - najlepsza inwestycja

```
import android.os.AsyncTask;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ImageView;

import android.widget.TextView;


import java.io.InputStream;

import java.util.ArrayList;


import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageListItem;

import pl.interdyscyplinarne.cwiczenie.schoolbook.R;

import pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask.ImageDownloaderTask;


/**
 * Created by alex on 2014-10-03.
 */
public class ImageItemAdapter extends ArrayAdapter<ImageListItem> {

    public ImageItemAdapter(Context context, ArrayList<ImageListItem> items) {

        super(context, R.layout.image_list_item, items);

    }
}
```

Człowiek - najlepsza inwestycja

```
@Override

public View getView(int position, View convertView, ViewGroup parent) {

    // Get the data item for this position

    ImageListItem item = getItem(position);

    ViewHolder viewHolder; // view lookup cache stored in tag

    if (convertView == null) {

        viewHolder = new ViewHolder();

        LayoutInflater inflater = LayoutInflater.from(getContext());

        convertView = inflater.inflate(R.layout.image_list_item, parent, false); // Lookup view for data
        population

        viewHolder.text = (TextView) convertView.findViewById(R.id.itemText);

        viewHolder.text2 = (TextView) convertView.findViewById(R.id.itemText2);

        viewHolder.img = (ImageView) convertView.findViewById(R.id.itemImg);

        convertView.setTag(viewHolder);

    } else {

        viewHolder = (ViewHolder) convertView.getTag();

    }

    // Populate the data into the template view using the data object

    viewHolder.text.setText(item.getItemText());

    viewHolder.text2.setText(item.getItemText2());

    if (viewHolder.img != null) {

        new ImageDownloaderTask(viewHolder.img).execute(item.getImgUrl());

    }

}
```

Człowiek - najlepsza inwestycja

```

    }

    // Return the completed view to render on screen

    return convertView;

}

private static class ViewHolder {

    TextView text;


    TextView text2;

    ImageView img;


}

}
    
```

Slajd
32

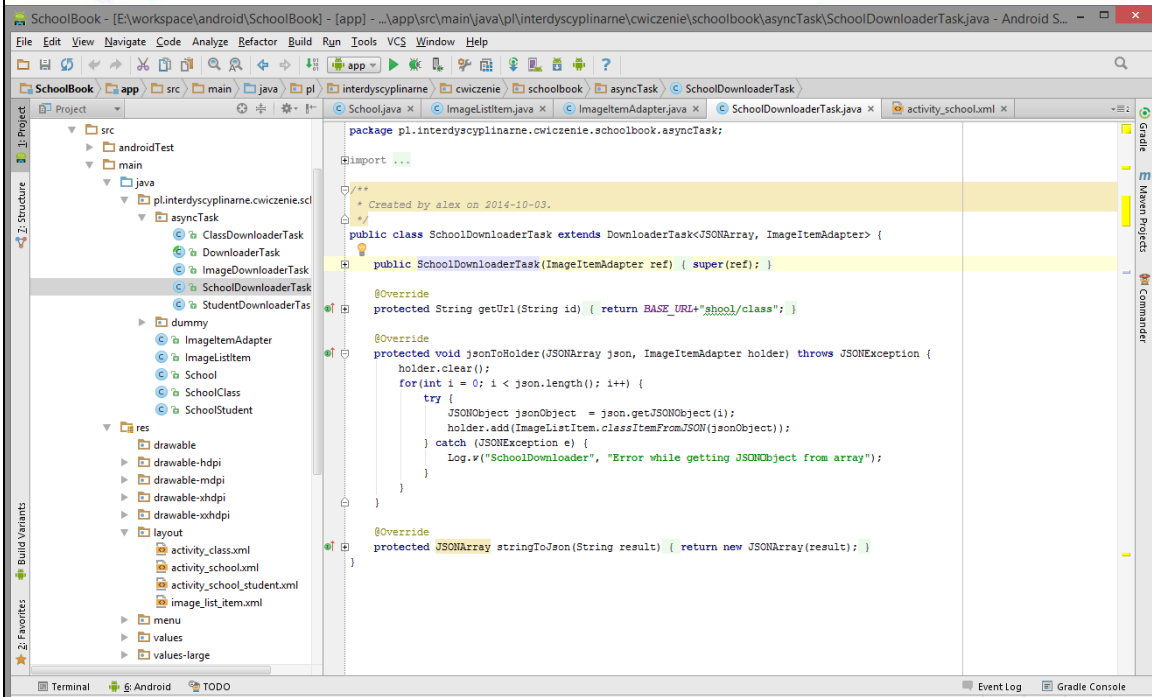


B2E
 BUSINESS TO EDUCATION



SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY

JAVA
 dla początkujących



Człowiek - najlepsza inwestycja

DownloaderTask – wykonuje całą pracę związaną z połączeniem z webserwisem oraz odebraniem odpowiedzi i zamienienia jej na obiektową reprezentację JSON'a, a później modelu danych. Wszystkie operacje wspólne zaimplementowane są właśnie w tej klasie, a specyficzne oddelegowane do klas podrzędnych przy użyciu metod abstrakcyjnych.

```
package pl.interdyscyplinarne.cwiczenie.schoolbook.asyncTask;
```

```
import android.net.http.AndroidHttpClient;
```

```
import android.os.AsyncTask;
```

```
import android.util.Log;
```

```
import org.apache.http.HttpEntity;
```

```
import org.apache.http.HttpResponse;
```

```
import org.apache.http.HttpStatus;
```

```
import org.apache.http.client.methods.HttpGet;
```

```
import org.json.JSONArray;
```

```
import org.json.JSONException;
```

```
import org.json.JSONObject;
```

```
import java.io.BufferedReader;
```

```
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
```

```
import java.lang.ref.WeakReference;
```

```
import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageItemAdapter;
```

Człowiek - najlepsza inwestycja

```
import pl.interdyscyplinarne.cwiczenie.schoolbook.ImageListItem;

/**
 * Created by alex on 2014-10-07.
 */
public abstract class DownloaderTask<T, J> extends AsyncTask<String, Void, T> {

    protected final static String BASE_URL = "http://10.0.2.2:8081/rejestr/api/";

    protected final WeakReference<J> holderRef;

    public DownloaderTask(J ref) {

        holderRef = new WeakReference<J>(ref);
    }

    @Override

    // Actual download method, run in the task thread
    protected T doInBackground(String... params) {

        // params comes from the execute() call: params[0] is the url.

        return downloadJSON(params[0]);
    }

    @Override

    // Once the image is downloaded, associates it to the imageView
    protected void onPostExecute(T json) {

        if (isCancelled()) {
```

Człowiek - najlepsza inwestycja


```
        json = null;

    }

    if (holderRef != null) {

        J holder = holderRef.get();

        if (holder != null) {

            if (json != null) {

                try {

                    jsonToHolder(json, holder);

                } catch (JSONException e) {

                    Log.w("DownloaderTask", "Error while parsing JSON");

                }

            }

        }

    }

}

}

}

private T downloadJSON(String id) {

    final AndroidHttpClient client = AndroidHttpClient.newInstance("Android");

    final HttpGet getRequest = new HttpGet(getUrl(id));

    try {

        HttpResponse response = client.execute(getRequest);
```

Człowiek - najlepsza inwestycja

```

final int statusCode = response.getStatusLine().getStatusCode();

if (statusCode != HttpStatus.SC_OK) {

    Log.w("DownloaderTask", "Error " + statusCode + " while retrieving response from " +
    getUrl(id));

    return null;

}

final HttpEntity entity = response.getEntity();

if (entity != null) {

    InputStream inputStream = null;

    try {

        inputStream = entity.getContent();

        BufferedReader bufferedReader = new BufferedReader(new
        InputStreamReader(inputStream));

        String line = "";

        String result = "";

        while ((line = bufferedReader.readLine()) != null) {

            result += line;

        }

        return stringToJson(result);

    } finally {

        if (inputStream != null) {

            inputStream.close();

        }


        entity.consumeContent();
    }
}
    
```


Człowiek - najlepsza inwestycja

```
}  
  
}  
  
} catch (Exception e) {  
  
    // Could provide a more explicit error message for IOException or  
  
    // IllegalStateException  
  
    getRequest.abort();  
  
    Log.w("DownloaderTask", "Error while retrieving result from " + getUrl(id));  
  
} finally {  
  
    if (client != null) {  
  
        client.close();  
  
    }  
  
}  
  
return null;  
  
}  
  
  
protected abstract String getUrl(String id);  
  
protected abstract <T> T stringToJson(String result) throws JSONException;  
  
protected abstract void jsonToHolder(T json, J holder) throws JSONException;  
  
}
```

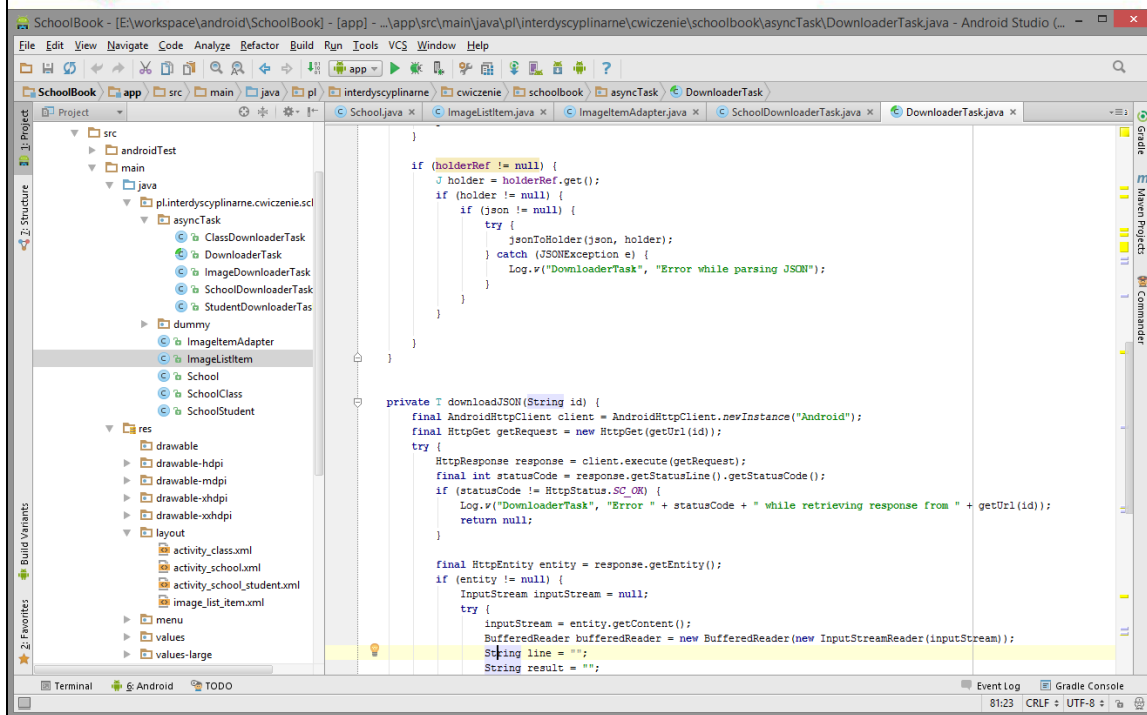
Człowiek - najlepsza inwestycja

Slajd
33


B2E
 BUSINESS TO EDUCATION


SZCZECIŃSKI PARK
 NAUKOWO-TECHNOLOGICZNY


JAVA
 dla początkujących




Jeśli na komputerze developera jest uruchomiony serwer www z web serwisami, to mimo iż adresy do wywołania webserwisów zaczynają się od localhost, to w Androidzie trzeba wpisać inny adres widoczny na slajdzie. Spowodowane jest to tym, że aplikacja odpalana w emulatorze działa w maszynie wirtualnej, w związku z czym adres localhost odnosiłby się do adresu maszyny wirtualnej, a nie komputera developer'a. Adres podany w przykładzie jest standardowym dressem dla emulatorów Androida, jeśli by jednak nie zadziałał, to można podać rzeczywisty adres IP komputera developera.

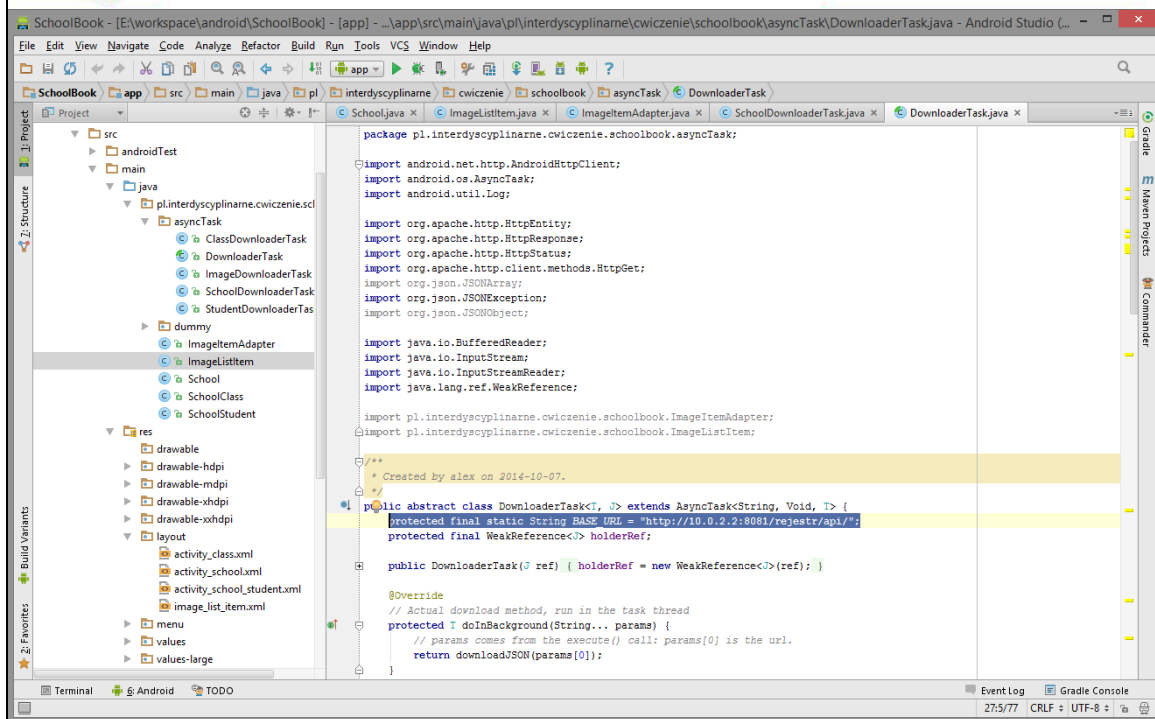
Człowiek - najlepsza inwestycja

Slajd
34


 **B2E**
BUSINESS TO EDUCATION


 **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących



Slajd
35


 **B2E**
BUSINESS TO EDUCATION

 **SZCZECIŃSKI PARK**
NAUKOWO-TECHNOLOGICZNY

JAVA
dla początkujących

Zadanie:

Wzorując się na implementacji widoku listy klas szkoły stwórz widok dla listy uczniów w klasie. W tym zadaniu nie odpytuj webserwisu, a listę możesz zapisać ręcznie przykładowymi danymi.



Aplikacja mobilna na platformę Android

Człowiek - najlepsza inwestycja

Slajd
 36

JAVA
 dla początkujących

Zadanie:

Dodaj nawigację między widokiem listy klas, a widokiem uczniów w klasie za pomocą obsługi zdarzenia kliknięcia wiersza danej klasy. Uruchom aplikację w emulatorze i przetestuj działanie.

Podpowiedź: Kod obsługi zdarzenia kliknięcia:

```
classListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> a, View v, int position, long id) {
        ImageListItem item = (ImageListItem) classListView.getItemAtPosition(position);
        Intent myIntent = new Intent(School.this, SchoolClass.class);
        myIntent.putExtra(SchoolClass.parameterName, item.getId());
        startActivity(myIntent);
    }
});
```



Aplikacja mobilna na platformę Android

 Slajd
 37

JAVA
 dla początkujących

Zadanie:

Wiedząc iż adres webserwisu do pobierania listy uczniów w klasie to `http://localhost/rejestr/api/shool/class/ + {id ucznia}`
 Dodaj implementację zachowania widoku listy uczniów w klasie. Stwórz nową klasę `ClassDownloader` rozszerzającą `DownloaderTask` i użyj jej w implementacji „activity”.

Podpowiedź: Sprawdź rezultaty zwracane przez webserwis wpisując adresy w przeglądarce internetowej:

```
http://localhost/rejestr/api/shool/class
http://localhost/rejestr/api/shool/class/2
http://localhost/rejestr/api/person/1
```





Aplikacja mobilna na platformę Android

Człowiek - najlepsza inwestycja


KAPITAŁ LUDZKI
 NARODOWA STRATEGIA SPÓJNOŚCI

 UNIA EUROPEJSKA
 EUROPEJSKI
 FUNDUSZ SPOŁECZNY


Slajd
38





JAVA
 dla początkujących



Zadanie:

Znając już adresy webserwisów z poprzedniego zadania stwórz „activity” dla ostatniego widoku wraz z pełną funkcjonalnością i komunikacją z webserwisem oraz nawigacją między widokami.

Dodaj przyciski służące do inicjowania rozmowy telefonicznej oraz wysyłania e-maila. Kod do obsługi interakcji z systemem widoczny jest na następnym slajdzie:


 Aplikacja mobilna na platformę Android


Slajd
39






JAVA
 dla początkujących

```
holder.btnemail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent emailIntent = new Intent(Intent.ACTION_SENDTO, Uri.fromParts("mailto",email, null));
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Hello "+name);
        holder.caller.startActivity(Intent.createChooser(emailIntent, "Send email..."));
    }
});

holder.btnCall.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent callIntent = new Intent(Intent.ACTION_CALL);
        callIntent.setData(Uri.parse("tel:" + tel));
        holder.caller.startActivity(callIntent);
    }
});
```


 Aplikacja mobilna na platformę Android


 KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Człowiek - najlepsza inwestycja

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

8.11.3 Opis założonych osiągnięć ucznia

Człowiek - najlepsza inwestycja

Uczeń nauczy się obsługiwać nowe IDE i tworzyć proste aplikacje mobilne. Nawiązywanie połączenia z zewnętrznym serwerem oraz komunikacja za pomocą formatu JSON powinny być znane uczniowi.

Człowiek - najlepsza inwestycja