

Obiektowe Języki Programowania

Wprowadzenie

Łukasz Rybka · Gdańsk 2015

0 **mnie**

0 mniej

Z WYKSZTAŁCENIA

0 mnie

Z WYKSZTAŁCENIA
fizyk

0 mniej

Z WYKSZTAŁCENIA
fizyk

Z ZAWODU

0 mniej

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

projektant-programista / kontrybutor Open Source / freelancer

0 mniej

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

projektant-programista / kontrybutor Open Source / freelancer

Z ZAMIŁOWANIA

0 mnie

Z WYKSZTAŁCENIA

fizyk

Z ZAWODU

projektant-programista / kontrybutor Open Source / freelancer

Z ZAMIŁOWANIA

wykładowca / prelegent / szkoleniowiec

Wykład - materiały

Literatura:

 <http://smoczysko.github.io/lectures/oopl>

Literatura:

Bentrand Meyer "Programowanie zorientowane obiektowo"

 <http://smoczysko.github.io/lectures/oopl>

Literatura:

Bentrand Meyer "Programowanie zorientowane obiektowo"

Bruce Eckel "Thinking in Java"

Literatura:

Bentrand Meyer "Programowanie zorientowane obiektowo"

Bruce Eckel "Thinking in Java"

Katy Sierra, Bert Bates "OCA/OCP Java SE 7 Programmer I & II Study Guide"

 <http://smoczysko.github.io/lectures/oopl>

Ustalenia

Ustalenia

Pytania mile widziane!

Pytania mile widziane!

**Konsultacje po zajęciach - informacja
odpowiednio przed (np. mailowo)**

Pytania mile widziane!

**Konsultacje po zajęciach - informacja
odpowiednio przed (np. mailowo)**

**Slajdy to tak naprawdę ściągawka dla
wykładowcy ;)**

Pytania mile widziane!

**Konsultacje po zajęciach - informacja
odpowiednio przed (np. mailowo)**

**Slajdy to tak naprawdę ściągawka dla
wykładowcy ;)**

Wykład nie jest obowiązkowy...

Organizacja zajęć

Organizacja zajęć

Wykład: 7 spotkań po 2h (14h)

Organizacja zajęć

Wykład: 7 spotkań po 2h (14h)

**Laboratoria: 15 spotkań po 4h tygodniowo
(60h)**

Organizacja zajęć

Wykład: 7 spotkań po 2h (14h)

Laboratoria: 15 spotkań po 4h tygodniowo (60h)

Egzamin: ostatni "poniedziałek" semestru o godzinie 16:00 w sali 221 GG

Zaliczenie laboratoriów

Zaliczenie laboratoriów

**5 mini-kolokwiów w trakcie laboratoriów
(każde za 6pkt)**

Zaliczenie laboratoriów

5 mini-kolokwiów w trakcie laboratoriów
(każde za 6pkt)

1 projekt za 30 pkt

Zaliczenie laboratoriów

**5 mini-kolokwiów w trakcie laboratoriów
(każde za 6pkt)**

1 projekt za 30 pkt

wymagane zdobycie przynajmniej 31 pkt

Zaliczenie przedmiotu

Zaliczenie przedmiotu

**zaliczenie laboratoriów (31/60 pkt) jest
wymogiem podejścia do egzaminu**

Zaliczenie przedmiotu

**zaliczenie laboratoriów (31/60 pkt) jest
wymogiem podejścia do egzaminu
egzamin w formie ustnej za 20 pkt**

Zaliczenie przedmiotu

**zaliczenie laboratoriów (31/60 pkt) jest
wymogiem podejścia do egzaminu
egzamin w formie ustnej za 20 pkt
zdobycie powyżej 50 punktów upoważnia do
zwolnienia z egzaminu – z oceną 3+**

Przelicznik punkty → ocena końcowa

00 – 40 pkt – niedostateczny

41 – 50 pkt – dostateczny

51 – 60 pkt – dostateczny plus

61 – 70 pkt – dobry

71 – 75 pkt – dobry plus

76 – 80 pkt – bardzo dobry

Zakres wykładów

Zakres wykładów

wprowadzenie do ekosystemu Java/JVM

Zakres wykładów

wprowadzenie do ekosystemu Java/JVM
klasy i obiekty

Zakres wykładów

wprowadzenie do ekosystemu Java/JVM

klasy i obiekty

dziedziczenie

Zakres wykładów

wprowadzenie do ekosystemu Java/JVM

klasy i obiekty

dziedziczenie

polimorfizm

Zakres wykładów

wprowadzenie do ekosystemu Java/JVM

klasy i obiekty

dziedziczenie

polimorfizm

kolekcje

Zakres wykładów

wprowadzenie do ekosystemu Java/JVM

klasy i obiekty

dziedziczenie

polimorfizm

kolekcje

obsługa wyjątków

O czym nie będziemy mówić

O czym nie będziemy mówić

**mechanizmy analogiczne do innych języków
(jak np. operacje bitowe)**

O czym nie będziemy mówić

**mechanizmy analogiczne do innych języków
(jak np. operacje bitowe)**

**zarządzanie pamięcią (Garbage Collector) - w
szczegółach**

O czym nie będziemy mówić

**mechanizmy analogiczne do innych języków
(jak np. operacje bitowe)**

**zarządzanie pamięcią (Garbage Collector) - w
szczegółach**

instrukcje sterujące, pętle itp.

O czym nie będziemy mówić

**mechanizmy analogiczne do innych języków
(jak np. operacje bitowe)**

**zarządzanie pamięcią (Garbage Collector) - w
szczegółach**

instrukcje sterujące, pętle itp.

...

Czego będziemy używać?

Czego będziemy używać?

Java Standard Edition Development Kit (Java SE JDK) - 8u60

Czego będziemy używać?

Java Standard Edition Development Kit (Java SE JDK) - 8u60

Eclipse IDE for Java Developers

Czego będziemy używać?

Java Standard Edition Development Kit (Java SE JDK) - 8u60

Eclipse IDE for Java Developers

Konsola + edytor tekstowy

Po co to wszystko?

Sep 2015	Sep 2014	Change	Programming Language	Ratings	Change
1	2	⬆	Java	19.565%	+5.43%
2	1	⬇	C	15.621%	-1.10%
3	4	⬆	C++	6.782%	+2.11%
4	5	⬆	C#	4.909%	+0.56%
5	8	⬆	Python	3.664%	+0.88%
6	7	⬆	PHP	2.530%	-0.59%
7	9	⬆	JavaScript	2.342%	-0.11%
8	11	⬆	Visual Basic .NET	2.062%	+0.53%
9	12	⬆	Perl	1.899%	+0.53%
10	3	⬇	Objective-C	1.821%	-8.11%

Go to jest JVM?

Co to jest JVM?

JAVA VIRTUAL MACHINE

Co to jest JVM?

JAVA VIRTUAL MACHINE

Środowisko uruchomieniowe

Co to jest JVM?

JAVA VIRTUAL MACHINE

Środowisko uruchomieniowe

JAVA BYTECODE

Co to jest JVM?

JAVA VIRTUAL MACHINE

Środowisko uruchomieniowe

JAVA BYTECODE

wiele języków kompilowanych do jednego kodu "maszynowego"

Co to jest JVM?

JAVA VIRTUAL MACHINE

Środowisko uruchomieniowe

JAVA BYTECODE

wiele języków kompilowanych do jednego kodu "maszynowego"

GARBAGE COLLECTOR

Co to jest JVM?

JAVA VIRTUAL MACHINE

Środowisko uruchomieniowe

JAVA BYTECODE

wiele języków kompilowanych do jednego kodu "maszynowego"

GARBAGE COLLECTOR

zarządzanie pamięcią

JVM languages



A word cloud of JVM languages. The words are arranged in a dense, overlapping manner. The colors are primarily green and grey. The word 'Java' is the largest and most prominent, located in the center. Other large words include 'Pascal' on the right, 'Scheme' and 'Groovy' in the upper right, 'Python' in the center, and 'Rexx' and 'Yeti' at the top. Smaller words include 'Scala', 'Flow', 'Lisp', 'Frink', 'Hec1', 'Ada', 'X10', 'Cobol', 'LOGO', 'Fantom', 'Jaskell', 'OCaml', 'Joy', 'Sleep', 'Judoscript', 'PHP', 'Fortress', 'Clojure', 'Script', 'ColdFusion', 'N.A.M.E', 'JavaScript', 'Lua', 'Common', 'Noop', 'BeanShell', 'AspectJ', 'Oberon-2', 'Ruby', 'NetLogo', 'Jelly', 'Basic', 'AWK', 'Tcl', 'Dubu', 'Forth', 'CAL', 'Join', 'Pizza', and 'Alef++'.

Scala Flow Rexx Yeti X10
Cobol Lisp Frink Hec1 Ada Joy
JavaFX LOGO Jaskell OCaml
Fantom Fortress Scheme
Sleep Judoscript PHP Clojure Groovy
Script Java Python
ColdFusion Pnuds N.A.M.E
Noop BeanShell Lua Common JavaScript
AspectJ Oberon-2 Ruby NetLogo
Jelly Basic AWK Tcl Dubu
CAL Join Pizza Forth Alef++
Pascal

Najprostszy program Java (Main.java)

```
public class Main {1  
    public static void main(String[] args) {2  
        System.out.println("Hello World!");3  
    }  
}
```

- ¹ Definicja głównej (i jedynej) klasy w aplikacji
- ² Definicja metody main, do której zostanie przekazane sterowanie aplikacją przy jej uruchomieniu
- ³ Instrukcja wyświetlenia na standardowe wyjście systemowe (konsolę) napisu

```
javac Main.java1
```

```
java Main2
```

- ¹ Kompilacja klasy (plik .java) do bytecode (plik .class)
- ² Uruchomienie skompilowanej klasy (programu)

Bytecode

Compiled from "Main.java"¹

```
public class Main {  
    public Main();  
        Code:  
            0: aload_0  
            1: invokespecial #1           // Method  
java/lang/Object."<init>":()V  
            4: return  
  
    public static void main(java.lang.String[]);  
        Code:  
            0: getstatic      #2           // Field  
java/lang/System.out:Ljava/io/PrintStream;  
            3: ldc            #3           // String Hello World!  
            5: invokevirtual #4           // Method  
java/io/PrintStream.println:(Ljava/lang/String;)V  
            8: return  
}
```

¹ javap -c Main

Kilka podstawowych informacji

Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa

Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa

Uruchamiamy program przez uruchomienie
klasy z funkcją `main(String[] args)`

Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa

Uruchamiamy program przez uruchomienie klasy z funkcją `main(String[] args)`

Pakiety - "nie wszystko w jednym worku"!

Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa

Uruchamiamy program przez uruchomienie klasy z funkcją `main(String[] args)`

Pakiety - "nie wszystko w jednym worku"!

Dokumentujemy kod w kodzie (javadoc)

Kilka podstawowych informacji

Plik ma tę samą nazwę co klasa

Uruchamiamy program przez uruchomienie klasy z funkcją `main(String[] args)`

Pakiety - "nie wszystko w jednym worku"!

Dokumentujemy kod w kodzie (javadoc)

Podział programu na klasy według (ich) odpowiedzialności

Klasa pomocnicza (DateUtils.java)

```
package pl.org.dragonia.helloapp.utils;

import java.util.Calendar;

public class DateUtils {
    private Calendar calendar;

    public DateUtils() {
        this.calendar = Calendar.getInstance();
    }

    public String sayHello() {
        String message = "Hello, today is ";

        message += calendar.get(Calendar.DAY_OF_YEAR);
        message += " day of year!";

        return message;
    }
}
```


Klasa pomocnicza (Main.java)

```
package pl.org.dragonia.helloapp;

import pl.org.dragonia.helloapp.utils.DateUtils;

public class Main {

    public static void main(String[] args) {
        DateUtils dateUtils = new DateUtils();

        System.out.println(dateUtils.sayHello());
    }
}
```

Dokumentacja kodu (Javadoc)

```
/**
 * Method saying hello and which day of year is today.
 *
 * @return {String} string telling which day of year is today
 */
public String sayHello() {
    String message = "Hello, today is ";

    int dayOfYear = calendar.get(Calendar.DAY_OF_YEAR);
    switch (dayOfYear) {
        case 1:
            message += dayOfYear + "st";
            break;
        // ...
        default:
            message += dayOfYear + "th";
            break;
    }

    message += " day of year!";

    return message;
}
```

ANY
QUESTIONS
?

Pytania?