



Podstawy programowania w Javie

Michał Idzik, Wojciech Frącz
AGH IEiT Informatyka

Czym jest Java?

Dlaczego Java?

Java

- Język wysokiego poziomu
 - Programowanie zorientowane obiektowo
 - Prosta składnia
 - Przenośność
 - Szeroki zestaw narzędzi w bibliotece standardowej
-

“Od dębu do kawy”

Krótką historia Javy

1991 - Sun tworzy projekt *Green*.

Jeden kod, wiele urządzeń - nowy język programowania: *Oak*

1996 - Java 1.0 dostępna publicznie, wsparcie w przeglądarkach

1998 - Java 2

...

2009 - Firma *Oracle* wykupuje *Sun*

2011 - Java 7

2014 - Java 8

2017 - Java 9

2018 - Java 10, Java 11

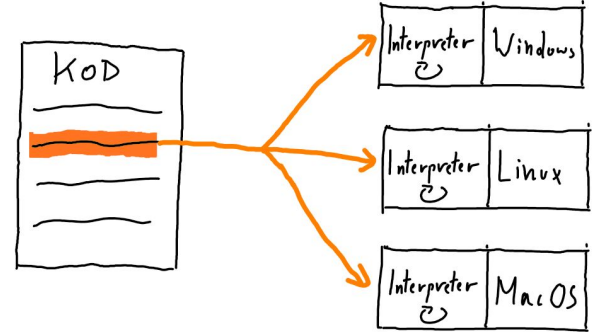
← Nowy, 6-miesięczny cykl, koniec darmowego *Long Term Support*

Java **Architektura**

Kompilatory i interpretery

Interpreter

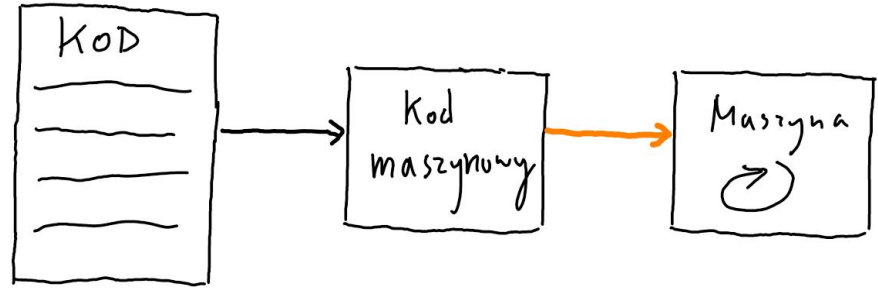
- Analizuje kod źródłowy
- **Wykonuje kod bezpośrednio** w trakcie działania
- Wolny, duży koszt zarządzania zasobami
- Duża elastyczność, wygodny do testów i procesu debugowania



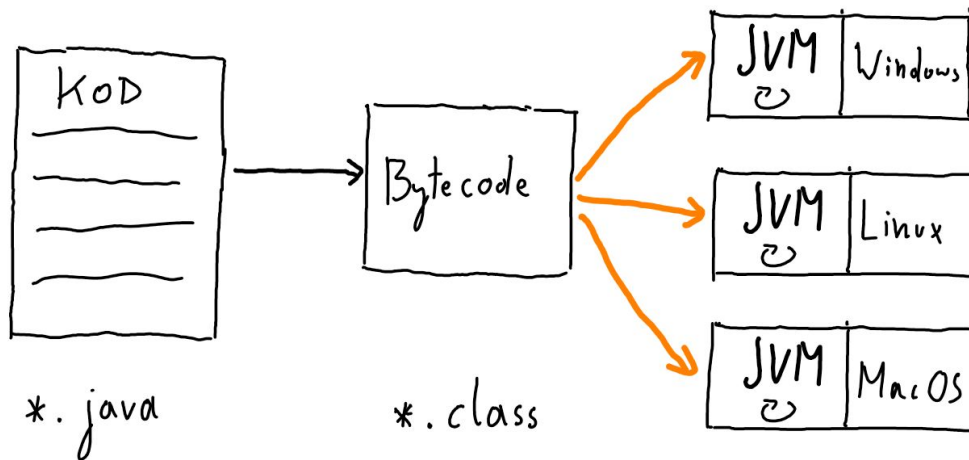
Kompilatory i interpretery

Kompilator

- Tłumaczy kod do postaci zapisanej w innym języku (najczęściej: kod maszynowy)
- **Nie wykonuje** kodu źródłowego
- Może dokonywać optymalizacji



JVM - Java Virtual Machine



- **Kompilacja** do kodu pośredniego (*bytecode*)
 - **Interpretacja** *bytecode* w emulatorze maszyny wirtualnej Javy (JVM)
-

JVM - Java Virtual Machine

- Przenośność: *“Write Once, Run Everywhere”*
 - Optymalizacje: JIT
 - Inne języki mogą być wykonywane na JVM:
 - Scala
 - Clojure
 - Kotlin
 - Jython (Python), JRuby (Ruby), itp.
-

Java Development Kit (JDK)

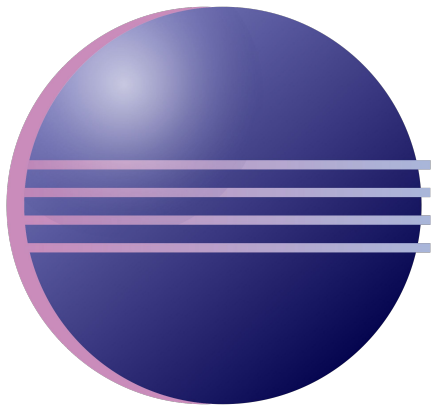
javac: plik.java → plik.class
(kompilacja)

java: plik.class → JVM (interpretacja)


HelloJava.java

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

Eclipse




- IDE - Integrated Development Environment
 - kompiluje, uruchamia, wspiera tworzenie programów
 - <http://www.eclipse.org>
 - `C:\eclipse\eclipse-neon`
-

 eclipse-workspace - Eclipse

File Edit Navigate Search Project Run Window Help


New Alt+Shift+N >


Open File...


 Open Projects from File System...

Close Ctrl+W











Close All Ctrl+Shift+W

 Save Ctrl+S

 Save As...

 Save All Ctrl+Shift+S

Revert









-  Maven Project
-  Enterprise Application Project
-  Dynamic Web Project
-  EJB Project
-  Connector Project
-  Application Client Project
-  Static Web Project
-  JPA Project
-  Project...
-  Servlet

Select a wizard

Create a Java project



Wizards:

-  Java Project
-  Java Project from Existing Ant Buildfile
-  Plug-in Project
- >  General
- >  Eclipse Modeling Framework
- >  EJB
- >  Gradle
- >  Java



< Back

Next >

Finish

Cancel

hello

> JRE System Library [JavaSE-1.8]

src

New

Open in New Window

Open Type Hierarchy

F4

Show In

Alt+Shift+W >

Show in Local Terminal

>



Copy

Ctrl+C



Copy Qualified Name



Paste

Ctrl+V



Delete

Delete



Remove from Context

Ctrl+Alt+Shift+Down

Build Path

>

Source

Alt+Shift+S >



Java Project



Project...



Package



Class



Interface



Enum



Annotation



Source Folder



Java Working Set



Folder



File



Untitled Text File

Java Class

Create a new Java class.



Source folder:

hello/src

Browse...

Package:

hello

Browse...

☐ Enclosing type:

Browse...

Name:

HelloJava

Modifiers:

☒ public ☐ package ☐ private ☐ protected☐ abstract ☐ final ☐ static

Superclass:

java.lang.Object

Browse...

Interfaces:

Add...

Which method stubs would you like to create?

☒ public static void main(String[] args)☐ Constructors from superclass☒ Inherited abstract methods

Finish

Cancel

hello
JRE System Library [JavaSE-1.8]
src
hello
HelloJava.java

New
Open F3
Open With
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
References
Declarations
Refresh F5
Assign Working Sets...
Coverage As
Run As
Debug As
Profile As
Validate

```
1 package hello;  
2  
3 public class HelloJava {
```

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

1 Run on Server Alt+Shift+X, R
2 Java Application Alt+Shift+X, J
Run Configurations...

Java

Składnia

Z czego składa się program?

“Algorytmy + Struktury Danych = Programy”



Instrukcje sterujące



Dane (zmienne, stałe)
o różnych typach

- N. Wirth

Typy danych w Javie

Wbudowane **typy proste (prymitywne)**:

- **int** - liczba całkowita
- **double** - liczba zmiennoprzecinkowa
- **boolean** - wartość logiczna
- **char** - pojedynczy znak

12 -10 0

3.14 1.0 2e10

true false

'a' 'z' 'ę'

Typy danych w Javie

Zmienna

- Reprezentuje dane określonego typu

`int numerButa;`

Typ
danych

Nazwa
zmiennej

Typy danych w Javie

Zmienna

- Reprezentuje dane określonego typu
- Przechowuje wartość danych

`int numerButa = 40;`

Typ danych Nazwa zmiennej Wartość danych

Typy danych w Javie

Zmienna

- Reprezentuje dane określonego typu
- Przechowuje wartość danych
- Wartość może się zmieniać

```
int numerButa = 40;
```

```
numerButa = 41;
```

Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość

```
int numer = 149;
```

*“Od teraz zmienna **numer** ma wartość 149”*

UWAGA: Znak równości nie oznacza porównania!



Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej



```
double twojWynik = 3.14;
```

```
double mojWynik = twojWynik;
```

“Od teraz zmienna **mojWynik** ma wartość 3.14”

Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej
- Użyć na niej operatora (np. **arytmetycznego**)

```
int rok = 2016;
```

```
rok = rok + 1;
```

*“Od teraz zmienna **rok** ma wartość 2017”*



Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej
- Użyć na niej operatora (np. arytmetycznego)
- Użyć jako argumentu funkcji (metody)

```
int kwota = 1000;
```

```
System.out.println(kwota);
```



Zadanie 1



Użyj języka Java do rozwiązania poniższego działania dla wartości $x=2$ i $y=0.5$

$$(2x + 3y) * 5x / 2$$

Typy danych w Javie

Typy złożone = **Klasy**

- składają się z zestawu atrybutów i funkcji (metod)
 - pozwalają definiować własne typy danych
 - wprowadzają programowanie obiektowe!
 - ...program napisany w Javie składa się z zestawu wzajemnie powiązanych klas
-

Typy danych w Javie

Klasy (przykłady):

- `String`
 - `System`
 - `HelloJava`
 - ...
 - `Samochód`
 - `Instrument`
-

Typy danych w Javie

Klasa `String`

- Reprezentuje łańcuch znaków (napis)

```
String inwokacja = "Litwo! Ojczyzna moja!";
```



Wartość napisu musi być umieszczona między " "

Typy danych w Javie

Klasa **String**

- Reprezentuje łańcuch znaków (napis)
- Oferuje zestaw operacji (metod) na napisach

```
String inwokacja = "Litwo! Ojczyzno moja!";
```

```
int rozmiar = inwokacja.length();
```



Wywołanie operacji (metody) oznaczamy kropką

Typy danych w Javie

Klasa `String`

- Reprezentuje łańcuch znaków (napis)
- Oferuje zestaw operacji (metod) na napisach

```
String inwokacja = "Litwo! Ojczyzno moja!";  
int rozmiar = inwokacja.length();  
String kraj = inwokacja.substring(0, 5);
```

Zadanie 2

Wypisz na konsolę *Hello World* używając zadeklarowanych zmiennych.

```
public class HelloJava {  
    public static void main(String[] args) {  
        String hello = "Hello";  
        String world = "world";  
    }  
}
```

Typy danych w Javie

Specjalny typ złożony : **Tablica**

- Ciąg elementów danego typu

```
int[] wynikiLotto;
```



Tablica wartości typu całkowitego

Typy danych w Javie

Specjalny typ złożony : **Tablica**

- Ciąg elementów danego typu
- Posiada określony rozmiar

```
int[] wynikiLotto = new int[6];
```



Tablica wartości typu całkowitego

Typy danych w Javie

Zmienna

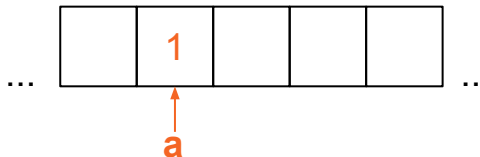


Tablica

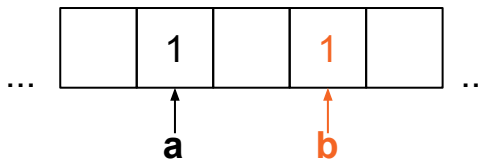


Zarządzanie pamięcią: typy proste

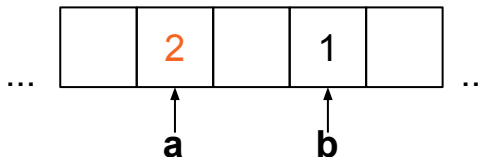
```
int a = 1;
```



```
int b = a;
```



```
a = a + 1;
```



W zmiennych typów prostych przypisujemy **wartości**.

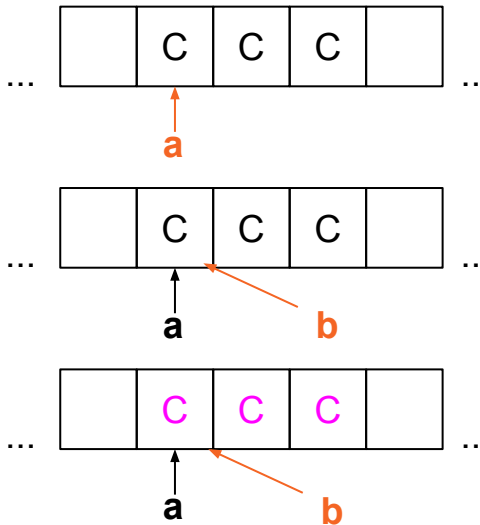
Zarządzanie pamięcią: typy złożone

```
Car a = new Car();
```

```
a.color = "black";
```

```
Car b = a;
```

```
a.color = "pink";
```



W zmiennych typów złożonych przypisujemy **referencje**.

Typy danych w Javie

- Java to język **statycznie typowany!**
- Każda zmienna musi explicitie deklarować swój typ
- Kompilator sprawdza poprawność operacji pod kątem typu danych

Python (dynamiczne typowanie):

```
liczba = 100
```

Java (statyczne typowanie):

```
int liczba = 100;
```

Typy danych w Javie

✓ `int a = 40;`

✗ `int b = 2.5;` → `int b = (int)2.5;`

✓ `double c = 3;` (bezstratne rzutowanie)

✗ `boolean d = 1;` → `boolean d = true;`

✓ `boolean e = a == 40;` (operator `==` zwraca `true` lub `false`)

Typy danych w Javie

Co można zrobić na zmiennej typu prostego?

- Przypisać wartość
- Skopiować do niej wartość innej zmiennej
- Użyć na niej operatora (np. arytmetycznego, **logicznego**)
- Użyć jako argumentu funkcji (metody)



```
int wagaZiemniakow = 2;
```

```
wagaZiemniakow > 1
```

“Czy ziemniaki ważą więcej niż 1?”

```
wagaZiemniakow == 2
```

“Czy ziemniaki ważą dokładnie 2?”

Instrukcje sterujące: if

```
if (wagaZiemniakow < 2) {  
    System.out.println("Dam rade");  
}  
else {  
    System.out.println("Nie dam rady");  
}
```

Instrukcje sterujące: if

```
if (wagaZiemniakow < 2) {  
    System.out.println("Dam rade");  
}  
else if (wagaZiemniakow < 5) {  
    System.out.println("Dam rade jak mi pomozesz");  
}  
else {  
    System.out.println("Nie dam rady");  
}
```



Zadanie 3

Napisz program który zadeklaruje zmienną `ocena` i w zależności od jej wartości:

- wypisze "Jest super" jeśli `ocena` jest równa 5
 - wypisze "Jest nieźle" jeśli `ocena` jest mniejsza od 5 ale większa lub równa 3
 - wypisze "Jakoś to będzie" jeśli `ocena` jest mniejsza od 3 ale większa lub równa 2
 - wypisze "Coś jest nie tak" w każdym innym przypadku
-

Pętle

- wykonują dany kod wiele razy
- dzięki modyfikacji zmiennych mogą w każdym przebiegu zachowywać się nieco inaczej



Pętla while

```
int zrobionePompki = 0;
while (zrobionePompki < 10) {
    System.out.println("Robię pompkę");
    zrobionePompki += 1;
}
```

Pętla do-while

```
int zrobionePompki = 1;  
do {  
    System.out.println("Robię pompkę");  
} while (zrobionePompki < 1);
```

→ Pomimo że warunek jest spełniony od początku, pętla do-while wykona się co najmniej raz!

Pętla for

```
for (int zrobionePompki = 0; zrobionePompki < 10; zrobionePompki++) {  
    System.out.println("Robię pompkę");  
}
```

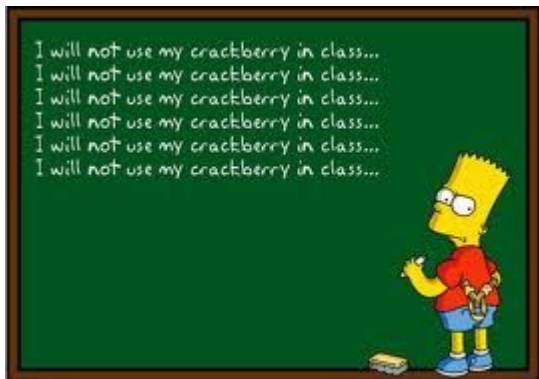
Pętla for each

```
String[] imiona = { "Antoni", "Zbigniew", "Mariusz" };
```

```
for (String imie : imiona) {  
    System.out.println(imie);  
}
```

→ Pętla for each
przechodzi po
kolejnych elementach
tzw. kolekcji

Zadanie 4



Wyobraź sobie, że Twój siostrzeniec dostał od nauczyciela karę. Ma on napisać 100 razy "Nie będę programować na lekcji".

Pomóż mu, pisząc odpowiedni program.

Weź pod uwagę, że nauczyciel patrzy tylko na początek i koniec zadania, więc nic nie stoi na przeszkodzie by od 10 do 90 napisu wydrukować "Ale dlaczego? Programowanie jest fajne!"
