

| Podstawy Java SE



Hello!

Tomasz Lisowski

Software developer, JIT Solutions
IT trainer

Agenda

- wprowadzenie
- klasy i metody
- typy danych
- instrukcje sterujące
- pętle
- równość obiektów



Java SE

Wprowadzenie

- aktualna wersja to Java10
- język obiektowy
- aplikacje pisane w Javie są kompilowane do bytecodu, a następnie uruchamiane na maszynie wirtualnej (JVM)
- Java jest multiplatformowa
- automatyczne zarządzanie pamięcią



Java SE

Wprowadzenie

Dystrybucje javy:

- **Java SE** – Standard Edition
 - podstawowa dystrybucja
- **Java EE** – Enterprise Edition
 - do wytwarzania aplikacji webowych

Java SE

Wprowadzenie

- **JVM** – wirtualna maszyna javy
 - “procesor” wykonujący skompilowany kod Javy
- **JRE** – Java Runtime Environment
 - zawiera JVM oraz klasy niezbędne do uruchomienia programów Java
- **JDK** – Java Development Kit
 - zawiera JRE oraz narzędzia do implementacji i kompilacji

Java SE

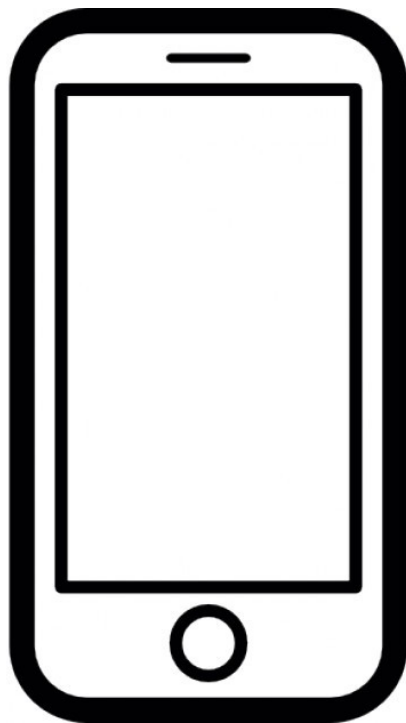
Klasa

- podstawowy element składowy aplikacji
- typ danych
- konkretna **definicja** pewnego 'bytu'
- .. zawierająca pola i metody

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("infoShareAcademy - Java SE");  
    }  
}
```

Java SE

Klasa



<i>właściwości / pola</i>	<i>czynności / metody</i>
marka	zadzwon()
model	odbierz()
numer	wyslijSMS()
lista kontaktów	odbierzSMS()
waga	zrobZdjecie()

Java SE

Składnia

Diagram illustrating the syntax of a Java SE class and its main method:

```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("infoShareAcademy - Java SE");  
    }  
}
```

Annotations:

- nazwa klasy (pliku)**: Points to `Main`.
- główna metoda**: Points to `main`.
- wnętrze klasy**: Points to the opening curly brace `{` of the class.
- ciało metody**: Points to the body of the `main` method, specifically the `System.out.println` statement.

Java SE

main()

- metoda *main()* to główna metoda, od której rozpoczyna się uruchamianie programu przez JVM


```
public class Main {  
    public static void main(String[] args) {  
        System.out.println("infoShareAcademy - Java SE");  
    }  
}
```

Java SE

Składnia

`int a,b;`  deklaracja
zmiennych

`a = 1;`  przypisanie
wartości
`b = 2;`

`int c = a + b;`  deklaracja z
jednoczesnym
przypisaniem

Java SE

Składnia

```
public class Menu {  
  
    int number;  
    String text;  
  
    Menu() {  
        number = 1;  
        text = "tekst";  
    }  
}
```

Java SE

Ćwiczenie 1

- stwórz klasę o nazwie Menu
- dodaj 2 pola
 - *int number;*
 - *String text;*



Java SE

Obiekt

- instancja klasy
- konkretny obiekt na podstawie definicji klasy

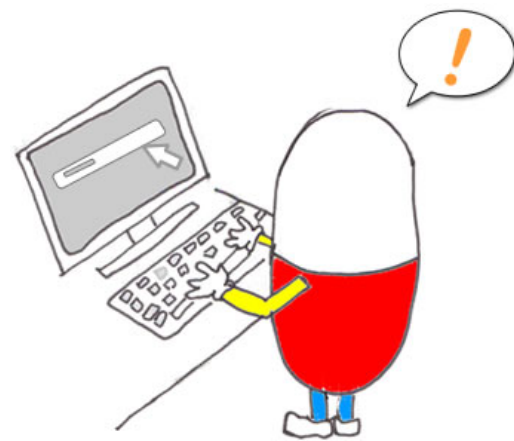
```
public class Car {  
    public String name;  
    public int maxSpeed;  
}
```

```
Car myCar = new Car();
```

Java SE

Ćwiczenie 2

- w klasie Main stwórz obiekt typu Menu
- wypisz wartości pól *number* oraz *text*



Java SE

Konstruktor

- metoda tworząca obiekt
- domyślny konstruktor
- konstruktor parametrowy
- słowo kluczowe *this*
 - zwraca aktualny obiekt


```
public class Car {  
    public String name;  
    public int maxSpeed;  
  
    public Car() {  
        name = "default name";  
        maxSpeed = 150;  
    }  
}
```


Java SE

Konstruktor

```
public class Car {  
    public String name;  
    public int maxSpeed;  
  
    public Car() {  
        name = "default name";  
        maxSpeed = 150;  
    }  
  
    public Car(String name) {  
        this.name = name;  
    }  
}
```

przypisanie do pola name
(pole klasy) wartości
parametru



Java SE

Ćwiczenie 3

- stwórz bezparametrowy konstruktor klasy Menu
- niech przypisze wartość dla wszystkich pól klasy
- wypisz wartości pól *number* oraz *text* dla obiektu stworzonego poprzez stworzony powyżej konstruktor



Java SE

Konstruktor

```
int number;  
String text;
```

```
//domyślny konstruktor, nie trzeba go pisać jawnie  
Menu() {  
}
```

```
//parametrowy konstruktor  
public Menu(int number, String text) {  
    this.number = number;  
    this.text = text;  
}
```

Java SE

Ćwiczenie 4

- stwórz parametrowy konstruktor klasy Menu
- niech przyjmuje 2 parametry tego typu co pola klasy
- przypisz wartości dla pól
- wypisz wartości pól *number* oraz *text* dla obiektu stworzonego poprzez parametryzowany konstruktor

np. `new Menu(number: 11, text: "tekst");`

- stwórz nowy obiekt wykorzystując domyślny konstruktor
- porównaj wartości obydwu obiektów



Java SE

Metody

- funkcje, które dana klasa udostępnia
- prywatne lub publiczne
- mogą posiadać parametry

```
public class Methods {  
  
    public void method1() {  
        System.out.println("Ta metoda nie zwraca nic!");  
    }  
  
    public int getNumberTwo() {  
        return 2; //ta metoda zwraca liczbę całkowitą 2  
    }  
  
    public int sum(int a, int b) {  
        return a + b; //ta metoda zwraca sumę dwóch parametrów  
    }  
}
```

Java SE

Ćwiczenie 5

- stwórz 2 metody w klasie Menu
 - pierwsza typu *void* (czyli nic nie zwraca), która wypisze tekst (metoda *println()*)
 - druga typu *int*, która zwróci wartość pola *number*
- wywołaj obydwie metody na obiekcie w klasie Main



Java SE

Metody statyczne

- mogą istnieć metody statyczne (*static*)
- można je wywołać bezpośrednio na klasie
- nie wymagają stworzenia instancji obiektu

```
public class Menu {  
    public static void staticMethod() {  
        System.out.println("This is static method!");  
    }  
  
    public void nonStaticMethod() {  
        System.out.println("This is NON static method!");  
    }  
}
```

Java SE

Metody statyczne

```
public static void main(String[] args) {  
    Menu menu = new Menu();  
    menu.nonStaticMethod();  
  
    Menu.staticMethod();  
}
```


Java SE

Ćwiczenie 6

- stwórz dodatkowe 2 metody w klasie Menu
- obydwie niech będą typu *void*
- jedna z nich niech będzie metodą statyczną
- każda z nich niech wypisze czy jest statyczna czy nie
- wywołaj obydwie metody w klasie Main



Java SE

Ćwiczenie 7

- stwórz 2 metody w klasie Menu:
 - jedna wypisująca na konsolę wartość w formacie “* *number* – *text* *”
 - druga powinna wypisać typ pola *text* (wykorzystanie metody getClass())



Java SE

Zasięg widzenia pól

- pola klasy dla wszystkich metod
- ..lub na zewnątrz
- pola w metodzie widoczne tylko w niej
- tworzony obiekt wewnątrz metody “żyje” tylko w niej

Java SE

Zasięg widzenia pól

```
int number;
```

```
String text;
```



pola klasy

```
public void method() {
```

```
    int otherNumber;
```



pole metody

```
    number = 1;
```

```
    otherNumber = 2;
```

```
}
```

```
public void otherMethod() {
```

```
    number = 2;
```

```
    otherNumber = 3;
```



błąd - *otherNumber* nie jest
widoczne w tej metodzie

```
}
```

Java SE

Gettery i settery

- dostęp do wartości pól powinien odbywać się poprzez metody
- metody *set(Type value)* do ustawiania wartości
- metody *get()* do odczytania

```
private int number;
```

```
public int getNumber() {  
    return number;  
}
```

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

Java SE

Ćwiczenie 8

- dodaj metody *get* i *set* do pól klasy Menu
- w metodzie *main* ustaw nowe wartości pól obiektów za pomocą *setter*'ów
- odczytaj zmienione wartości za pomocą *getter*'ów



Typy danych

Java SE

Co to są typy?

- *wszystko jest obiektem*
- reprezentują różne wartości, przechowywane w zmiennych
- typy tekstowe, liczbowe, zmiennoprzecinkowe, logiczne
- różne formaty dat
- każda klasa jest typem

Java SE

Typy proste

- typy proste (*primitive*) nie są instancjami obiektów
- reprezentują podstawowe typy danych
- zawsze mają jakąś wartość

```
int liczbaCalkowita;  
long duzaLiczbaCalkowita;  
double liczbaZmiennoPrzecinkowa; //64bit  
float kolejnaLiczbaZmiennoPrzecinkowa; //32bit  
boolean prawdaFalsz;  
char znak;
```

Java SE

Typy proste

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Java SE

Typy obiektowe

- konkretne instancje
- możemy tworzyć swoje typy
- mogą mieć dowolne zachowanie (metody)
- mogą nie mieć wartości -> NULL

```
Integer liczbaCalkowita;  
Long duzaLiczbaCalkowita;  
Double liczbaZmienniePrzecinkowa;  
Float kolejnaLiczbaZmienniePrzecinkowa;  
Boolean prawdaFalsz;  
String napis;
```

Java SE

Ćwiczenie 9

- stwórz dwie zmienne liczbowe
 - typu **int**
 - typu **Integer**
- wypisz ich domyślne wartości
- porównaj metody, które te dwie zmienne udostępniają:
zmiennaInt. ?
zmiennaInteger. ?



Java SE

Typy wyliczeniowe

- ENUM
- konkretny (ograniczony) zbiór możliwych wartości

```
public enum Colour {  
    RED,  
    GREEN,  
    BLUE  
}
```

```
Colour myColour = Colour.GREEN;
```

Java SE

Ćwiczenie 10

- stwórz własny typ wyliczeniowy
- dodaj pole tego typu do klasy Menu
- wypisz domyślną wartość pola
- dodaj nowy konstruktor Menu, który uwzględni nowe pole



Java SE

Operator

- działania, które można wykonywać na obiektach
- np. operacje matematyczne lub logiczne
- operatory porównania lub przypisania

Operator type	Example
Unary	<i>i++</i> , <i>i--</i> , <i>++i</i> , <i>--i</i> , <i>!</i>
Arithmetic	<i>*</i> , <i>/</i> , <i>%</i> , <i>+</i> , <i>-</i>
Relational	<i><</i> , <i>></i> , <i><=</i> , <i>>=</i> , <i>instanceof</i> , <i>==</i> , <i>!=</i>
Logical	<i>&&</i> , <i> </i>
Assignment	<i>=</i> , <i>+=</i> , <i>-=</i> , <i>*=</i> , <i>/=</i>

Java SE

konwencja

- każdy wyraz 'oddzielamy' dużą literą
- **KLASY** - rzeczownik, zaczynamy dużą literą
- **METODY** - czasownik, zaczynamy małą literą
- **ZMIENNE** - zaczynamy małą literą
- **STAŁE** - duże litery, wyrazy 'oddzielamy' znakiem '_'

```
class MyClass {  
    Integer myVariable;  
    final Integer MY_CONSTANT = 2;  
  
    void myMethod() {  
        myVariable = MY_CONSTANT + 1;  
    }  
}
```


Pobranie danych z klawiatury

Java SE

Scanner

- podstawowe pobranie danych od użytkownika
- obiekt korzysta ze strumienia wejściowego:

Scanner scanner = new Scanner(System.in);

- popularne metody:

- `nextLine()`
- `nextInt()`
- `nextDouble()`

Java SE

Ćwiczenie 11

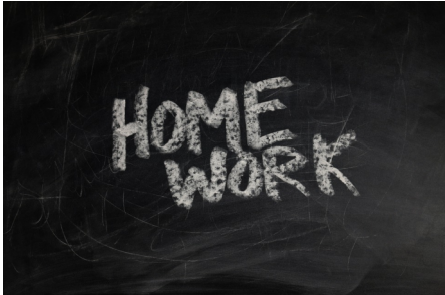
- pobierz za pomocą klasy Scanner dwie wartości:
 - tekst
 - liczbęi przypisz je do pól obiektu klasy Menu
- wyświetl wartości



Java SE

Zadanie domowe

- stwórz klasę o nazwie Card, która posiada 2 pola:
 - Suit
 - Rank
- dodaj dwie klasy enum określające kolor i figurę karty:
 - Suits (CLUBS, DIAMONDS, HEARTS, SPADES)
 - Ranks (ACE, KING, QUEEN, JACK)
- dodaj gettery i settery w klasie Card
- stwórz metodę getDescription, która zwraca opis w formacie “figura – kolor”, np. “Ace – Spades”
- ustaw przykładowe wartości dla kilku kart
- * kolor i figurę pobieraj z klawiatury
- uruchom i przetestuj aplikację

HOME
WORK



Thanks!



Q&A

tomasz.lisowski@protonmail.ch