Podstawy Java SE

Hello

Tomasz Lisowski

Software developer, JIT Solutions

IT trainer

Agenda

- wprowadzenie
- klasy
- obiekty
- metody
- typy danych

Wprowadzenie

Wprowadzenie

- język maszynowy - **bytecode**
- zapis binarny - 01100101001l0
- Java - obiektowy język programowania
- kompilacja

Wprowadzenie

- **JVM – Java Virtual Machine**

  “procesor” wykonujący skompilowany kod Javy i zarządzający pamięcią
- **JRE – Java Runtime Environment**

  zawiera JVM oraz klasy niezbędne do uruchomienia programów Java
- **JDK – Java Development Kit**

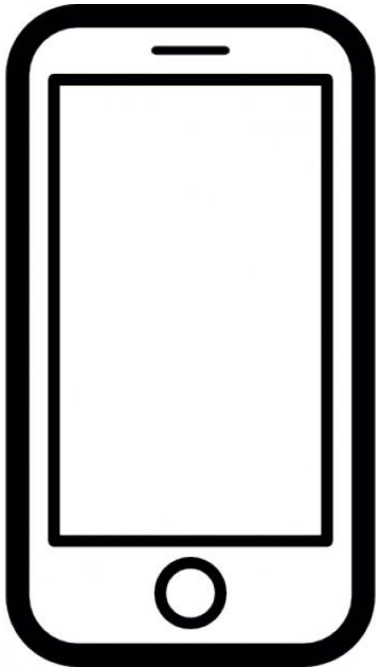
  zawiera JRE oraz narzędzia do implementacji i kompilacji

klasa

- podstawowy element składowy aplikacji
- typ danych
- konkretna definicja pewnego ‘bytu’ (instrukcja)
- zawiera pola i metody

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

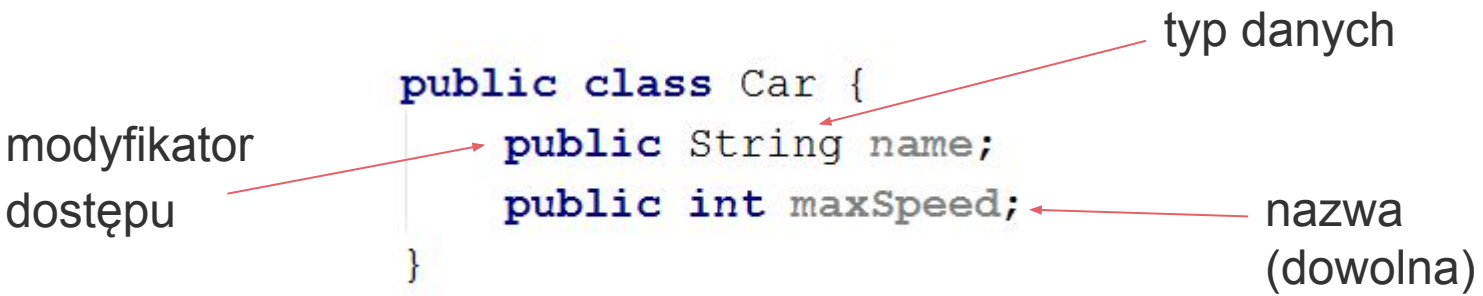
klasa



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

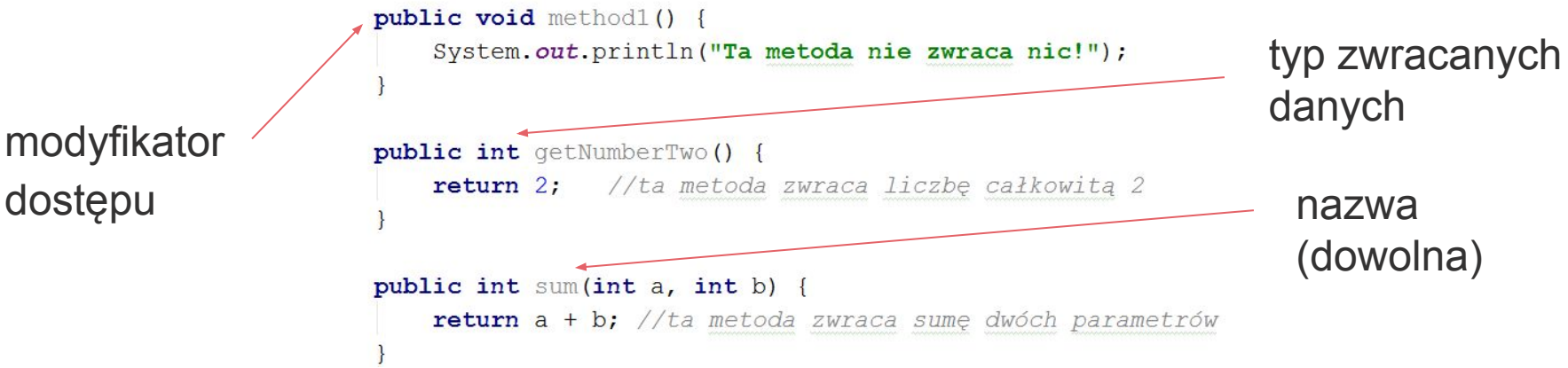
klasa - pole

- dana cecha naszej klasy
- może to być dowolny typ (klasa)
- może być ich wiele lub wcale



klasa - metoda

- funkcjonalność naszej klasy (tu logika - nie piszemy kodu poza metodami!)
- zwracają jakiś typ danych (lub nic - wtedy ‘zwracamy’ *void*)
- mogą przyjmować parametry



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");
    }
}
```

obiekt

- instancja klasy
- konkretny obiekt na podstawie definicji klasy

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

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

klasa *vs* obiekt

konstruktor

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

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

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

konstruktor

przypisanie do pola name (pole klasy) wartości parametru

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

    public Car() {
        name = "default name";
        maxSpeed = 150;
    }

    public Car(String name) {
        this.name = name;
    }
}
```

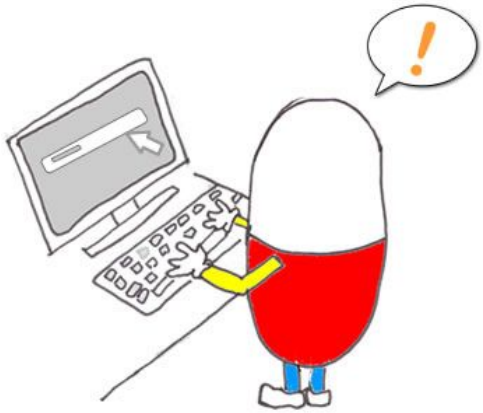
konstruktor

```
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;
}
```

Wprowadzenie

ćwiczenie 1

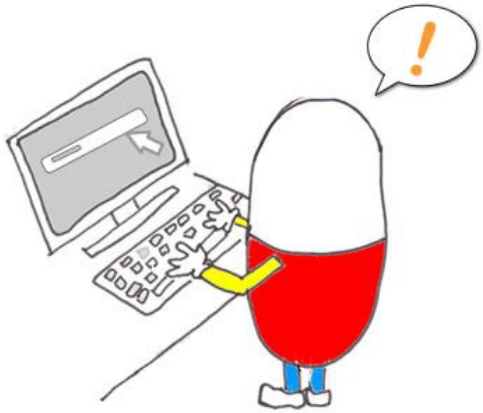
- stwórz klasę Engine, która ma pola typu Integer (nazwy: *power*, *capacity*)
- zmień klasę Car, dodaj pole typu Engine
- w metodzie main() stwórz 2 obiekty typu Car
- uzupełnij wszystkie pola wartościami



Wprowadzenie

ćwiczenie 2

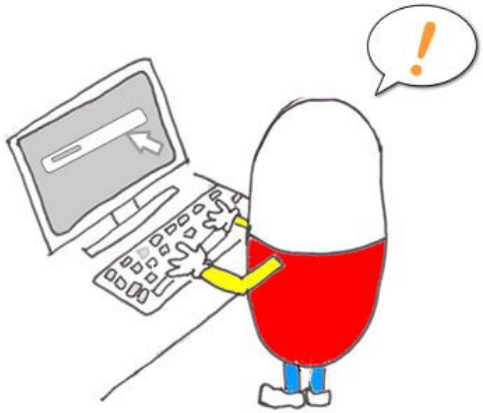
- w klasie Car stwórz metodę, która wypisze na ekran (*System.out.println()*) nazwę obiektu (pole *name*)
- wywołaj metodę na obiekcie powstałym w ćwiczeniu 1



Wprowadzenie

ćwiczenie 2b

- w klasie Engine stwórz metody, które wypiszą na ekran moc oraz pojemność danego obiektu (pola *power/capacity*)
- wywołaj metodę na obiekcie nowym obiekcie typu Engine



metody statyczne

- oznaczone słowem kluczowym ***static***
- można je wywołać bezpośrednio na klasie
- nie wymagają stworzenia instancji obiektu

metody statyczne

```
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!");
    }
```

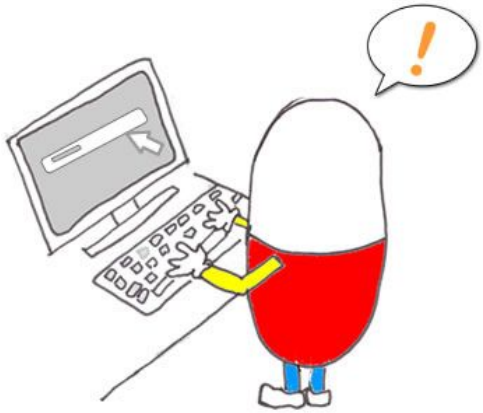
metody statyczne

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

    Menu.staticMethod();
```

Wprowadzenie

ćwiczenie 3

- utwórz nową klasę ***StaticExample***
- stwórz w niej dwie metody
- obydwie niech będą typu void
- jedna z nich niech będzie metodą statyczną
- każda z nich niech wypisze na ekran (System.out) czy jest statyczna czy nie
- wywołaj obydwie metody w klasie Main



zasięg widzenia pól

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

zasięg widzenia pól

```
int number;
String text;

public void method() {
    int otherNumber;

    number = 1;
    otherNumber = 2;
}

public void otherMethod() {
    number = 2;
    otherNumber = 3;
}
```

pola klasy

pole metody

błąd - *otherNumber* nie jest widoczne w tej metodzie

gettery i settery

- dostęp do wartości pól powinien odbywać się poprzez metody tzw. gettery i settery
- metody ***set(Type value)*** do ustawiania wartości
- metody ***get()*** do odczytania
- wszystkie pola powinny być ***private***
- gettery i/lub settery ***public***

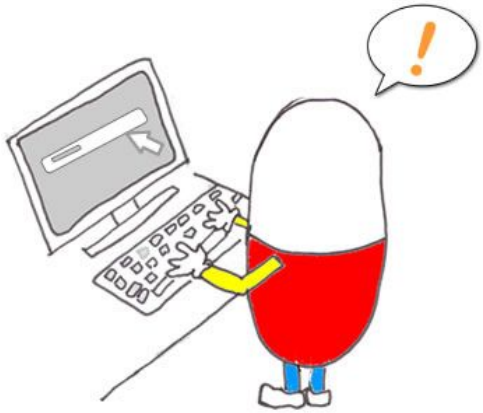
gettery i settery

```
private int number;

public int getNumber() {
    return number;
}

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

Wprowadzenie

ćwiczenie 4

- rozbuduj klasy Car i Engine o odpowiednie gettery i settery
- ustaw wartości obiektów za pomocą setterów
- wyświetl ustawione wartości za pomocą getterów



Typy danych

Typy danych

co to jest?

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

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;
```

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

Typy obiektowe

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

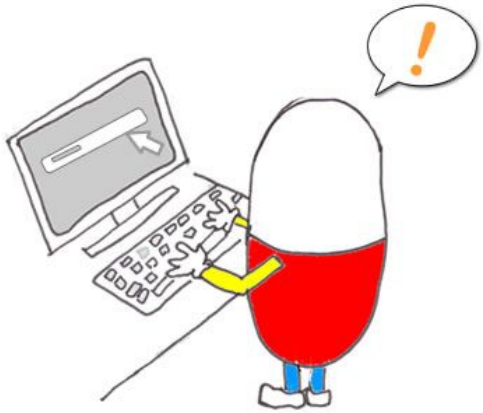
```
Integer liczbaCalkowita;
Long duzaLiczbaCalkowita;
Double liczbaZmiennoPrzecinkowa;
Float kolejnaLiczbaZmiennoPrzecinkowa;
Boolean prawdaFalsz;
String napis;
```

Typy danych
ćwiczenie 5

- stwórz dwie zmienne liczbowe
  - typu int
  - typu Integer
- wypisz ich domyślne wartości
- porównaj metody, które te dwie zmienne udostępniają:

zmiennaInt. ?

zmiennaInteger. ?



Rzutowanie

- zmiana typu danych na inny
- np. dzielenie dwóch liczb całkowitych

```
int liczbaA = 10;
int liczbaB = 3;
//wynik dzielenia nie jest liczbą całkowitą

int wynik = liczbaA / liczbaB;
// zmienna wynik = 3
```

Rzutowanie

```
int liczbaA = 10;
int liczbaB = 3;
//wynik dzielenia nie jest liczbą całkowitą

double wynikInt = liczbaA / liczbaB;
// zmienna wynik = 3.0
```

Rzutowanie

```
int liczbaA = 10;
int liczbaB = 3;

double liczbaC = (double) liczbaA;
//liczbaC = 10.0
double liczbaD = (double) liczbaB;
//liczbaD = 3.0

double wynikInt = liczbaA / liczbaB;
//wynikInt = 3.0
double wynikDouble = liczbaC / liczbaD;
//wynikDouble = ?
```

String

- obiektowy typ tekstowy
- **immutable** – nie można go zmienić, ‘zmiana’ powoduje utworzenie nowej instancji
- posiada zestaw metod do operacji na tekście
  np. *compare(), concat(), split(), length(), replace(), substring()*

```
char[] chars = {'i', 'n', 'f', 'o', 'S', 'h', 'a', 'r', 'e'};
String s = new String(chars);

String s2 = "infoShare";
```

String

- **immutable** – modyfikacje na Stringach wymagają przypisania

```
String s = "infoShare";
s.concat("Academy");
System.out.println(s);
```


infoShare

```
String s = "infoShare";
s = s.concat("Academy");
System.out.println(s);
```

infoShareAcademy

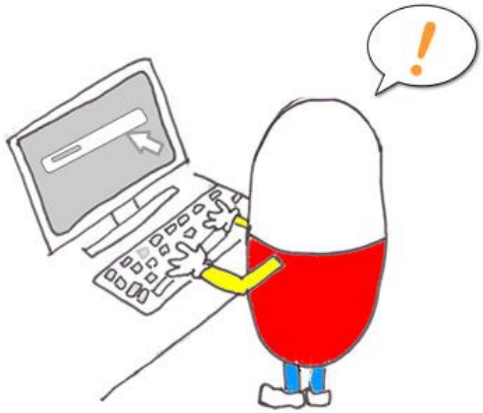
Operatory

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

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

Typy danych
ćwiczenie 6

- utwórz kilka obiektów różnych typów
- nadaj im wartości wykorzystując różne operatory
  np. *Integer i = 5 * 10;*
- wykorzystuj też inne obiekty do tworzenia kolejnych
  np. *Integer i = 2;*
  Integer j += i;



JAVA 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 ‘_’

JAVA konwencja

```
class MyClass {
    Integer myVariable;
    final Integer MY_CONSTANT = 2;

    void myMethod() {
        myVariable = MY_CONSTANT + 1;
    }
```

Podstawy JSE

materiały

- https://javastart.pl/baza-wiedzy/darmowy-kurs-java/podstawy-jezyka
- https://www.tutorialspoint.com/java/
- https://docs.oracle.com/javase/tutorial/

Pytania?

Thanks!

Q&A

tomasz.lisowski@protonmail.ch