Kreatywne programowanie

Zajęcia nr 3

- *Programowanie obiektowe*
- *typy danych, klasy konstrukcja*

Piotr Radzikowski
ptr.radzikowski@gmail.com
radzikowski.one.pl

Programowanie obiektowe - co to jest?

Programowanie obiektowe jest próbą przedstawienia świata rzeczywistego i relacji w nim zachodzących, za pomocą obiektów. Najważniejszymi cechami programowania obiektowego są:

- **Enkapsulacja/Hermetyzacja**
- **Abstrakcja**
- **Dziedziczenie**
- **Polimorfizm**

Implementacje są “identyczne” w każdym języku programowania

Program “witaj świecie” - Java

```java
public class HelloWorld {

    public static void main(String[] args) {
        // Prints "Witaj świecie!" to the terminal window.
        System.out.println("Witaj świecie!");
    }
}
```

Typy danych

W Javie podobnie jak w innych językach wyróżniamy wiele typów danych mogących przechowywać:

- liczby stało i zmiennoprzecinkowe,
- znaki, ciągi znaków,
- typ logiczny

Typy znakowe

Łączenie ciągów znakowych:

System.out.print(“pierwszy “+”drugi”);

char, czyli znak i służy do reprezentacji pojedynczych znaków kodu Unicode.

Istnieją również znaki specjalne, które muszą być poprzedzone znakiem backslash **** :

- \t - tab
- \n - nowa linia
- \r - powrót karetki
- \" - cudzysłów
- \' - apostrof
- \\ - backslash

Zadanie 1:

Napisz dane na temat książki w formacie:

Książka: “Czysty kod. Podręcznik dobrego programisty”
Autor: “Robert C. Martin”

Unicode - komputerowy zestaw znaków mający w zamierzeniu obejmować wszystkie pisma używane na świecie.

Typy proste - liczbowe

Liczby całkowite, wyróżniamy ich 4 rodzaje, ze względu na ilość zajmowanego miejsca.

- ***byte*** - 1 bajt - zakres od -128 do 127
- ***short*** - 2 bajty - zakres od -32 768 do 32 767
- ***int*** - 4 bajty - zakres od -2 147 483 648 do 2 147 483 647
- ***long*** - 8 bajtów - zakres od -2^63 do (2^63)-1

Liczby zmiennoprzecinkowe różnią się zakresem oraz ilością zajmowanego miejsca, a także precyzją

- ***float*** - 4 bajty - max ok 6-7 liczb po przecinku
- ***double*** - 8 bajtów - max ok 15 cyfr po przecinku

Uwaga! liczby zmiennoprzecinkowe w Java oddzielamy za pomocą kropki nie przecinka

Uwaga nr 2

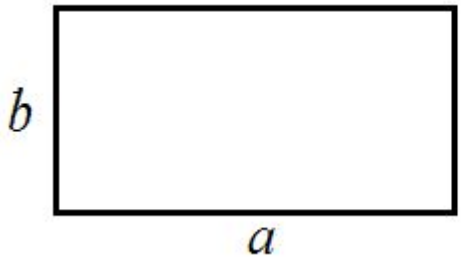
Należy też pamiętać, że liczby zmiennoprzecinkowe nie nadają się do obliczeń finansowych, w których liczy się dokładność. Spowodowane jest to tym, że w systemie dwójkowym nie da się przedstawić wszystkich liczb.

Z pomocą przychodzi tutaj specjalna klasa **BigDecimal**.

Istnieje również klasa **BigInteger**, która jest odpowiednikiem dla liczb całkowitych i może reprezentować w zasadzie nieograniczone co do wielkości liczby.

Zadanie 2

Napisz program który liczy pole prostokąta.



$$P = a \cdot b$$

```java
public class HelloWorld {

    public static void main(String[] args) {
        // Prints "Witaj świecie!" to the terminal window.
        System.out.println("Witaj świecie!");
    }
}
```

```java
// deklaracja zmiennej “a” o wartości całkowitej 3
int a = 3;
// deklaracja zmiennej “nazwa” o wartości typu łańcuch znaków o wartości ABC
string nazwa = “ABC”;

// Operatory:
+   - dodawanie, oraz łączenie
*   - mnożenie
/   - dzielenie
%   - reszta z dzielenia

_____

// wyświetlenie danych ze zmiennej “a”
int a = 3;
System.out.println(a);
```

Klasy osłonowe

Typy proste (int, double, float, czy boolean) nie są typami obiektowymi, przez co nie można ich na nich wywoływać żadnych metod.

Klasy osłonowe to odpowiednio:

- Boolean - boolean
- Character - char
- Integer - int
- Double - double
- Float - float
- Byte - byte
- Short - short
- Long - long

Klasy osłonowe

Dają one nam sporo możliwości - przykładowo w łatwy sposób możemy zmienić reprezentację liczb w systemie dziesiętnym na dwójkowy, czy szesnastkowy, lub zamienić je na String.

```
NazwaKlasy.nazwaMetody(parametr);
```

Przykład:

```
1. Integer.toString(5);
2. Integer.toBinaryString(10);
3. Integer.parseInt(“233”);
```

Tworzenie programu

Utworzymy program reprezentujący “książkę”.

Będzie posiadał zmienne:

- ilość stron (zmienna typu “int”)
- tytuł (zmienna typu “String”)
- autor (zmienna typu “String”)

Następnie wyświetli te dane na ekranie w formacie:

```
---------------------------------- Książka ----------------------------------
        Tytuł:      “Czysty kod. Podręcznik dobrego programisty”
        Autor:      “Robert C. Martin”
        Stron:      424
-----------------------------------------------------------------------------
```

Klasa

Klasa, z punktu widzenia programowania, jest to typ zmiennej.

W ujęciu projektowym to ogólna definicja pewnej grupy powiązanych ze sobą obiektów, które różnią się tożsamością.

Klasa definiuje metody, które są dostarczane przez obiekty. Poza tym definiuje również atrybuty, które są indywidualne (nie zawsze, ale do tego tematu wrócimy w przyszłości) dla konkretnych obiektów.

Czym jest obiekt? Jest to instancja danej klasy, czyli konkretna zmienna danego typu.

Przykład do rozjaśnienia:

Janek, Ania, Zosia to obiekty klasy Człowiek.

Każde z nich może spać, jeść, poruszać się - to są metody zdefiniowane w klasie Człowiek.

Każdy człowiek posiada imię oraz datę urodzenia, jednak są one indywidualne dla każdego, czyli nie są bezpośrednio powiązane z klasą, a z jej instancją (obiektem klasy).

```java
public class Human {

    private String name;
    private String birthDate;

    public Human(String name, String birthDate) { /*...*/ }
    public void eat() {/*...*/}
    public void sleep() {/*...*/}
    public void move() {/*...*/}
}
```

```java
    public static void main(String[] args) {
        Human janek = new Human("Janek", "1982-01-01");
        Human anna = new Human("Anna", "1950-01-01");
        Human zosia = new Human("Zosia", "2000-01-01");
    }
```

Konstruktor klasy

To specjalna metoda, która nie zwraca żadnego parametru. Określa ona jakie czynności zostaną wykonane w momencie tworzenia obiektu klasy (np. inicjalizuje zmienne – pola obiektu):

```
3 public class Human {
4
8     public Human(String name, String birthDate) { /*...*/ }
9     public void eat() {/*    */}
```

Zadanie - napisz klasę “książka”

zmienne:

- ilość stron (zmienna typu “int”)
- tytuł (zmienna typu “String”)
- autor (zmienna typu “String”)

```
public class Human {

    private String name;
    private String birthDate;

    public Human(String name, String birthDate) { /*...*/ }
    public void eat() {/*...*/}
    public void sleep() {/*...*/}
    public void move() {/*...*/}
}
```

funkcje:

- format - “zwraca” String w formacie:

---------- Książka ----------

Tytuł: “Czysty kod. Podręcznik dobrego programisty”
Autor: “Robert C. Martin”
Stron: 424

---

obiekt.toString()

Specjalna metoda toString() ma za zadanie zwrócić utworzony obiekt jako opis tekstowy.

```
@Override
public String toString() {
    return "Opis tekstowy obiektu";
}
```

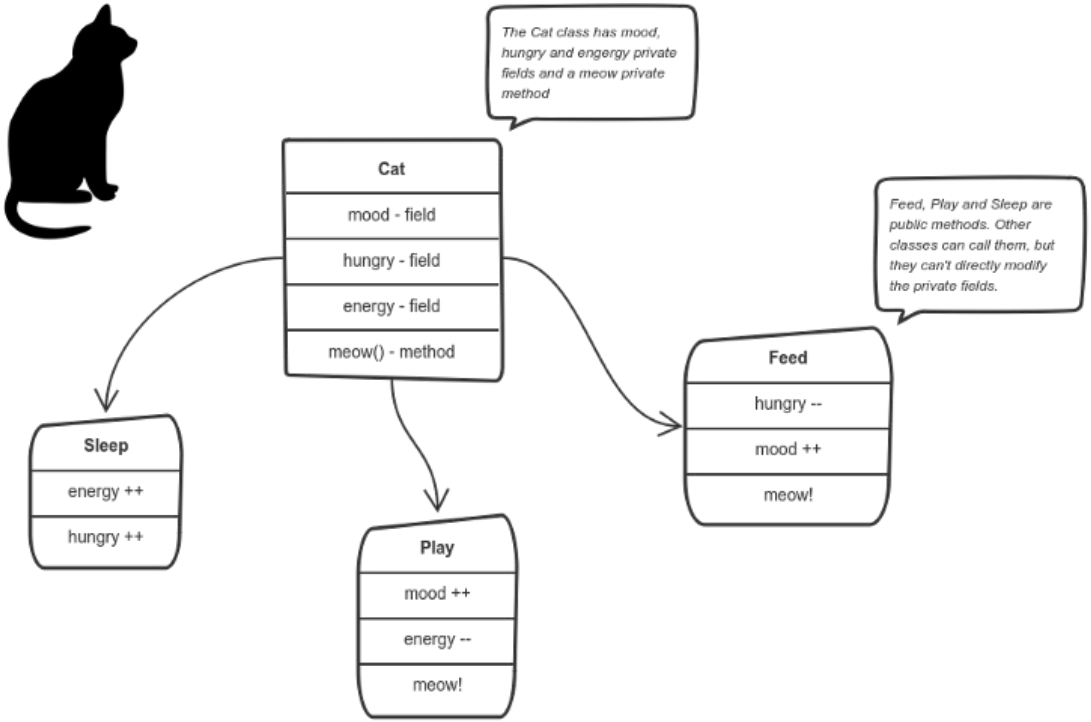
Programowanie obiektowe

Programowanie obiektowe jest próbą przedstawienia świata rzeczywistego i relacji w nim zachodzących, za pomocą obiektów (klas).

Najważniejszymi cechami programowania obiektowego są:

- **Enkapsulacja/Hermetyzacja**
- **Abstrakcja**
- **Dziedziczenie**
- **Polimorfizm**

Enkapsulacja/hermetyzacja



Enkapsulacja/hermetyzacja

Enkapsulacja (ang. *encapsulation*), czy inaczej *hermetyzacja* to sposób na ukrycie szczegółów implementacji klasy. Enkapsulacja to bardzo ważny element programowania obiektowego. Pozwala to na pełną kontrolę nad zachowaniem i stanem danego obiektu.

Modyfikatory dostępu

Modyfikator public

Słowo kluczowe **public** jest modyfikatorem dostępu, który pozwala na najbardziej swobodny dostęp do elementu, który poprzedza. **public** może być używane przed definicjami klas, pól w klasach, metod czy typów wewnętrznych.

Modyfikatory dostępu

Modyfikator protected

Modyfikator **protected** ma znaczenie w przypadku dziedziczenia. Elementy poprzedzone tym modyfikatorem dostępu są udostępnione dla danej klasy i jej podklas. Dodatkowo elementy oznaczone modyfikatorem **protected** dostępne są dla innych klas w tym samym pakiecie. Modyfikatora protected nie można stosować przed klasami.

Modyfikatory dostępu

Modyfikator private

Słowo kluczowe **private** jest najbardziej restrykcyjnym modyfikatorem dostępu. Może być stosowane wyłącznie przed elementami klasy, w tym przed klasami wewnętrznymi. Oznacza on tyle, że dany element (klasa, metoda, czy pole) widoczny jest tylko i wyłącznie wewnątrz klasy.

Brak modyfikatora dostępu

Brak modyfikatora dostępu również ma znaczenie. W przypadku gdy pominiemy modyfikator dostępu wówczas dana klasa czy element jest dostępna wyłącznie wewnątrz tego samego pakietu. Jest to podzbiór uprawnień, które nadaje modyfikator **protected**.

Modyfikator public w praktyce

```java
public class PublicVisitCounter {
    public int userCount = 0;

    public void increment() {
        userCount++;
    }
}
```

```java
public class MyWebsite {
    public void countMyVisit(PublicVisitCounter counter) {
        counter.increment();
        counter.userCount = -10;
    }
}
```

Enkapsulacja/hermetyzacja

Enkapsulacja (ang. encapsulation), czy inaczej hermetyzacja to sposób na ukrycie szczegółów implementacji klasy. Enkapsulacja to bardzo ważny element programowania obiektowego. Pozwala to na pełną kontrolę nad zachowaniem i stanem danego obiektu.

Gettery i Settery

```
public class EncapsulatedVisitCounter {
    private int userCount = 0;

    public void increment() {
        userCount++;
    }

    public int getUserCount() {
        return userCount;
    }

    public void setUserCount(int amount) {
        userCount = amount;
    }
}
```

Zadanie

Zmodyfikujcie klasę “książka” aby posiadał własne gettery i settery oraz poprawne modyfikatory dostępu.

Programowanie obiektowe - co to jest?

Programowanie obiektowe jest próbą przedstawienia świata rzeczywistego i relacji w nim zachodzących, za pomocą obiektów. Najważniejszymi cechami programowania obiektowego są:

- **Enkapsulacja/Hermetyzacja**
- **Abstrakcja**
- **Dziedziczenie**
- **Polimorfizm**

Implementacje są “identyczne” w każdym języku programowania

Konstrukcja klasy w różnych językach - Book.java

Book.java > ...

```java
/**
 * Book
 */
public class Book {

    private int pageAmount;
    private String title;
    private String author;

    public Book(String title, String author, int pageAmount) {
      this.title = title;
      this.author = author;
      this.pageAmount = pageAmount;
    }

    @Override
    public String toString() {
      return "------------------------------------ Książka ----------------------------------------\n"
          + "\tTytuł:\t\"" + title + "\"\n"
          + "\tAutor:\t\"" + author + "\"\n"
          + "\tStron:\t\"" + pageAmount + "\"\n"
          + "-------------------------------------------------------------------------------------\n";
    }
  }

```

Book.js

JS Book.js > ...

```
1   class Book {
2       pageAmount;
3       title;
4       author;
5
6       constructor (title, author, pageAmount) {
7           this.title = title;
8           this.author = author;
9           this.pageAmount = pageAmount
10      }
11
12      toString() {
13          return "------------------------------------ Książka ----------------------------------------\n"
14            + "\tTytuł:\t\"" + this.title + "\"\n"
15            + "\tAutor:\t\"" + this.author + "\"\n"
16            + "\tStron:\t\"" + this.pageAmount + "\"\n"
17            + "-------------------------------------------------------------------------------------\n";
18      }
19  }
20
```

Book.ts (type script)

TS book.ts › ...

```ts
class Book {
    private pageAmount :number;
    private title :string;
    private author :string;

    constructor(title: string, author: string, pageAmount: number) {
        this.title = title;
        this.author = author;
        this.pageAmount = pageAmount
    }

    public toString(): string {
        return "------------------------------------ Książka ----------------------------------------\n"
          + "\tTytuł:\t\"" + this.title + "\"\n"
          + "\tAutor:\t\"" + this.author + "\"\n"
          + "\tStron:\t\"" + this.pageAmount + "\"\n"
          + "-------------------------------------------------------------------------------------\n";
    }
}

```

Book.php

book.php

```php
<?php

class Book
{
    private $_title;
    private $_author;
    private $_pageAmount;

    function __constructor($title, $author, $pageAmount) {
        $this->_title = $title;
        $this->_author = $author;
        $this->_pageAmount = $pageAmount;
    }

    public function __toString() {
        return "------------------------------------ Książka ----------------------------------------\n"
          . "\tTytuł:\t\"{$this->_title}\"\n"
          . "\tAutor:\t\"{$this->_author}\"\n"
          . "\tStron:\t\"{$this->_pageAmount}\"\n"
          . "-------------------------------------------------------------------------------------\n";
    }
}

```

Book.cs (C#)

book.cs

```
/**
 * Book
 */
public class Book {

    private int PageAmount { get; set; }
    private string Title { get; set; }
    private string Author { get; set; }

    public Book(string title, string author, int pageAmount)
    {
      Title = title;
      Author = author;
      PageAmount = pageAmount;
    }

    public override string ToString()
    {
      return "------------------------------------ Książka ----------------------------------------\n" +
          "\tTytuł:\t\"" + Title + "\"\n" +
          "\tAutor:\t\"" + Author + "\"\n" +
          "\tStron:\t\"" + PageAmount + "\"\n" +
          "-------------------------------------------------------------------------------------\n";
    }
}

```

Dziękuje za uwagę!

Co na następnych zajęciach?

Piotr Radzikowski
ptr.radzikowski@gmail.com
radzikowski.one.pl