9. prednáška (22.11.2021)

Budujeme triedy

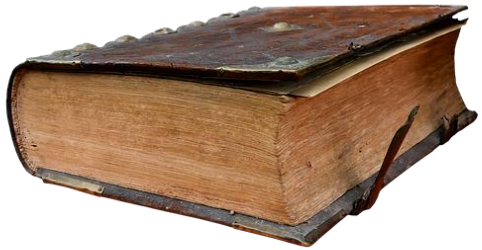
alebo

Murovanie v OOP

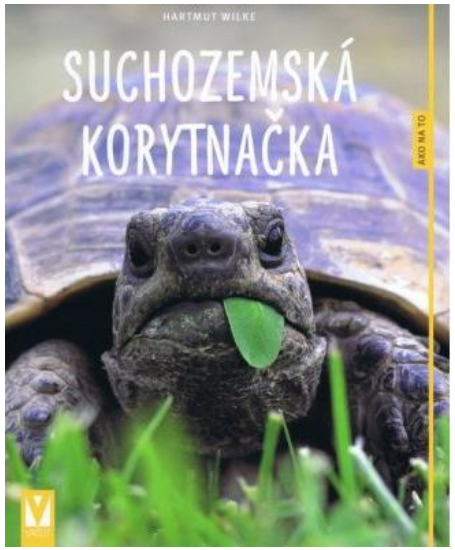
Knižnica

Zadanie

- Cieľ: pohodlná správa knižnice, resp. zbierky kníh.
- Vyžadovaná **funkcionalita**:
  - vieme vložiť info o novej knihe
  - odstrániť knihu zo zbierky
  - vypísať názvy všetkých kníh v knižnici
  - vypísať tie knihy, ktoré boli vydané v konkrétnom roku
  - vypísať tie knihy, ktoré napísal konkrétny autor
  - vypísať tie knihy, ktoré majú podľa nášho hodnotenia aspoň 8 bodov (na stupnici 0-10)
  - ...



Zadanie



- Dôležité informácie o každej knihe:
  - názov knihy
  - mená autorov
  - rok vydania knihy
  - žáner/tématika do ktorých spadá
    - kniha môže mať viac žánrov (napr. "western a dobrodružný")
  - hodnotenie kvality knihy: 0-10

Funkcionalita vs. dáta

- Dve kľúčové (základné) množiny požiadaviek:
  - **S akými dátami** bude program pracovať
  - **Aké služby** má poskytovať resp. **akú funkcionalitu** má program mať
- Pokiaľ nemáme jasno v ľubovoľnom z týchto bodov, ani nezačíname programovať!
- Princíp logického **rozdelenia na dáta a funkcionalitu** zachovávajú aj elementárne "súčiastky" **OOP - triedy**

Funkcionalita vs. dáta

- Princíp logického **rozdelenia na dáta a funkcionalitu** zachovávajú aj elementárne "súčiastky" **OOP - triedy**

```
public class PrepinaciaHra extends WinPane {
    private boolean[][] doska = new boolean[6][6];
    private boolean hraBezi = true;
    public PrepinaciaHra() {
    public void kresliMriezku() {
    public void kresliDosku() {
    public boolean dobraSuradnica(int r, int s) {
    public void tah(int r, int s) {
```

Dáta

Funkcionalita

Rozdeľuj a panuj

Ťažké problémy rozbiť na podproblémy.

V zložitom systéme (svete) identifikovať jednoduchšie časti – komponenty celku.

Realita vs. objekty a ich triedy

Zoznam/správca kníh

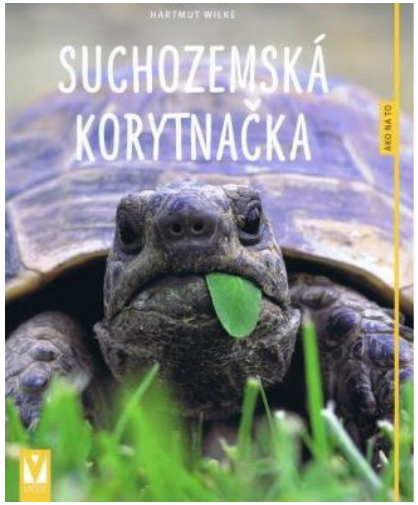
```
public class Library
```

Library **závisí od** triedy Book

```
public class Book
```

Kniha

Zmysel života?



Objekty

- Objekt:
  - poskytovateľ funkcionality
  - „obal“ na dáta, ktoré patria k sebe

`Color`

`String`

`Point2D`

`StringBuilder`

`Turtle`

`MouseEvent`

`WinPane`

Kniha a jej dáta

- Keď vieme s akými dátami budeme pracovať, musíme sa rozhodnúť, **ako budeme dáta uchovávať**.

- Zadanie v časti “dáta o jednom filme”:
  - názov knihy
  - mená autorov
  - rok vydania knihy
  - žáner/tématika do ktorých spadá
  - hodnotenie kvality knihy: 0-10

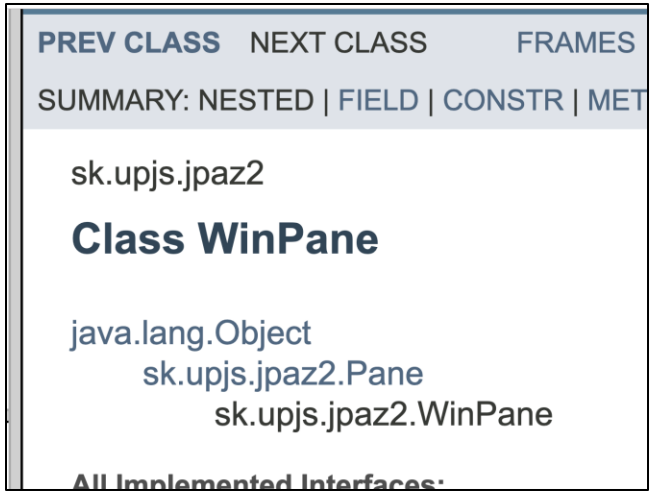
Ako budem dáta uchovávať?

Inštančné premenné objektov triedy Book?

- názov knihy
  - `String`
- mená autorov
  - `String[]`
- rok vydania
  - `Int`
- žánre do ktorých spadá, predpokladáme, že kniha môže mať viac žánrov
  - `String[]`
- hodnotenie kvality knihy na stupnici od nula do desať
  - `double`

Trieda Book

- Vytvoríme si triedu `Book`, ktorá bude **šablónou pre všetky knihy** a umožní im uchovávať si tieto informácie



- Akú triedu vylepšujeme?
  - nepotrebujeme funkcionalitu `Turtle` ani `WinPane`
  - rozšírime triedu `Object`
  - z triedy `Object` pochádzajú **všetky** triedy v Jave

Trieda ako obal pre viac premenných

```
public class Book extends Object {
  private String   title;
  private String[] authors;
  private int      publicationYear;
  private String[] genres;
  private double   rating;
}
```

Trieda ako obal pre viac premenných

```
public class Book extends Object {
  private String   title;
  private String[] authors;
  private int      publicationYear;
  private String[] genres;
  private double   rating;
}
```

Nemusíme písať

Použitie

- Môžeme vytvárať nové objeky triedy `Book`:

```
public class Launcher {
  public static void main(String[] args) {
    Book theLittlePrince = new Book();
    Book effectiveJava = new Book();
    ...
  }
}
```

Ale ako dostaneme dáta do objektov?

Ako dostať dáta do objektov?

Čo robia slovíčka **public** a **private**?

- Ako to robia iné objekty a triedy?
  - Konštruktory

```
String s = new String("Java");
Color c = new Color(100, 200, 100);
```

  - Metódy

```
franklin.setX(100);
franklin.getX();

sb.append('a');
```

Zapúzdrenie (Encapsulation)

- **Zabraňuje** priamemu prístupu k dátam (vnútorným častiam) objektu
- **Dáta a metódy**, ktoré s nimi pracujú, sú spolu.
- Každý objekt navonok sprístupňuje rozhranie (=metódy), pomocou ktorého (a nijako inak) sa s objektom pracuje.
  - objekty sú **zodpovedné** za konzistentný obsah svojich inštančných premenných
  - s objektami sa chceme rozprávať **iba cez ich metódy**

Setter

- **Setter** = metóda na nastavenie hodnoty inštančnej premennej
- Vie meniť hodnoty privátnym inštančným premenným
- Môže vykonávať kontroly
- Ak sa jej nová hodnota nepáči, môže zmenu odmietnuť, vypísať hlášku alebo hocičo iné..

```
private Typ premenná;

public void setPremenná(Typ premenná) {
   this.premenná = premenná;
}
```

Getter

- Getter = metóda na vrátenie hodnoty inštančnej premennej
- Vie čítať hodnoty privátnych inštančných premenných
- Nemusíme sprístupniť všetky

```
private Typ premenná;


public Typ getPremenná(){
    return this.premenná;
}
```

- Settery a gettery nám vie vygenerovať Eclipse:
  - Source -> Generate Getters and Setters

Setter, getter a referencie

```
public String[] getAuthors() {
      return authors;
}
```

Vrátenie referencie na pole **narušuje princíp** zapúzdrenia!

Obsah `String`-ov meniť nemožno, vrátenie referencie na `String` je OK.

```
public String[] getAuthors() {
      return authors.clone();
}
```

Konštruktor

- Doteraz známy aj ako „inicializačná metóda“
- Môže mať žiaden alebo viac parametrov
- Môžeme ich mať viac v jednej triede
  - musia sa líšiť počtom alebo typmi parametrov
- Volá sa cez **new**

```
File adresar = new File("C:/Windows");
File subor = new File(adresar, "system.ini");
Scanner sc = new Scanner(subor);
```

Konštruktor

- Každý konštruktor
  - vytvára objekt podľa šablóny - triedy
  - napĺňa inštančné premenné hodnotami
- Programátorom napísané konštruktory môžu nastaviť vhodnejšie inicializačné hodnoty ako default

Konštruktor

- Meno má rovnaké ako meno triedy v ktorej sa nachádza
- Nepíšeme návratový typ, nemá žiaden **`return`**

```
public class Book {

    public void Book(...parametre...) {
        ...
        return this;
    }
...
}
```

Ukážka

Pravidlá na zapamätanie

- Každá trieda **má aspoň jeden konštruktor**
- Ak nie je žiaden konštruktor napísaný programátorom, doplní sa neviditeľný implicitný konštruktor:

```
public class Book {

    public Book() {
    }
...
}
```

Takto by vyzeral implicitný konštruktor keby ho bolo vidieť

Konštruktor

- Vieme ho generovať z Eclipsu
  - Source -> Generate Constructor using fields
- Ak máme vytvorený konštruktor s parametrami, implicitný konštruktor sa nedopĺňa!
  - ak aj potom chceme používať konštruktor bez parametrov, musíme si ho vytvoriť explicitne!
- Konštruktor môže volať iný svoj konštruktor
  - musí to byť ale prvý príkaz konštruktora
  - `this(…parameter…)`

Napĺňanie priamym prístupom

- **Najhorší a neodporúčaný prístup**
  - zmažeme ochranu inštančných premenných: `private`
  - pristupujeme do vnútra objektu cez bodku nasledovanú názvom inštančnej premennej

Ukážka

- Kým na to nemáme pádne dôvody, **nikdy** to nerobíme pretože:
  - si objekty nedokážu ochrániť svoje premenné
  - vyladená trieda sa môže stať nestabilnou pri nevhodnom použití
  - používateľ metódy musí ovládať vnútornú logiku triedy, aby sa ju odvážil používať bez obavy, že utrpí jeho vlastný program

Vytvárame zoznam kníh

- Chceme uchovávať veľa kníh
- Pozor na prvoplánové riešenia:

```
public class BooksTest {
  public static void main(String[] args) {
    ...
    Book[] knihy = new Book[4];
    knihy[0] = theLittlePrince;
    knihy[1] = effectiveJava;
    knihy[2] = harryPotter3;
    knihy[3] = oxfordDictionary;
  }
}
```

Zadanie

- Chceme vedieť:
  - vložiť novú knihu
  - vymazať knihu
  - vypísať všetky knihy v knižnici
  - vypísať knihy, ktoré zodpovedajú danému žánru (napr. sci-fi)
  - vypísať tie knihy, ktoré vznikli pred viac ako 20 rokmi
  - vypísať všetky knihy, ktoré napísal daný autor
  - vypísať knihy, ktoré sú podľa nášho hodnotenia na stupnici od 8 do 10.

Realita vs. objekty a ich triedy

Zoznam/správca kníh

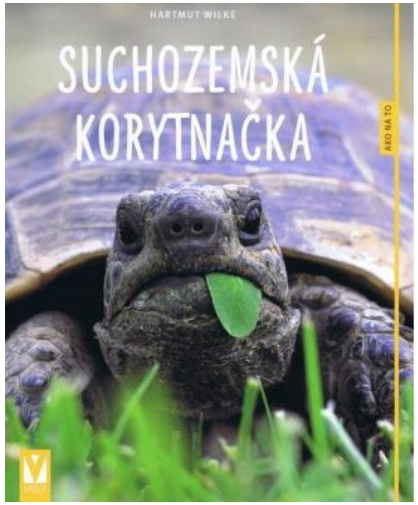
```
public class Library
```

Library **závisí od** triedy Book

```
public class Book
```

Kniha

Zmysel života?



Správca kníh vs. zapúzdrenosť

- Všetky dôležité dáta majú svojho „správcu“
- **Správca** = objekt vhodnej triedy
- **Dáta** = uložené v privátnych inštančných premenných tohto objektu
- Dáta môžeme spravovať len cez metódy objektu, ktorý dáta drží
- Správcom pre naše pole filmov bude objekt novej triedy `Library`

Kostra triedy Library

```
public class Library {
  private Book[] books;

  public Library() {
    books = new Book[0];
  }
  public void addNewBook(Book book) {
  }
  public void deleteBook(Book book) {
  }
  public void printAll() {
  }
  public void printByGenre(String genre) {
  }
  ...
}
```

Dopĺňame telá metód

- Pri vkladaní nafukujeme pole a pridávame novú knihu
- Pri mazaní nájdeme knihu a skracujeme pole
  - Hľadáme podľa inštancie
  - Hľadáme podľa názvu knihy (iná verzia tej istej metódy)

Preťaženie metód

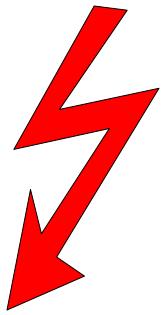
- Vhodné v prípade, že metódy robia to isté len sa líšia svojim vstupom
  - Počtom parametrov alebo
  - Aspoň jedným **typom** parametra

```
public void deleteBook(Book book) {
   ...
}

public void deleteBook(String title) {
   ...
}
```

Preťaženie metód

- Nestačí, že sa líšia názvom parametrov alebo návratovým typom

```
public int vypocet(int vstup1, double vstup2) {
    ...
}

public double vypocet(int prva, double druha) {
    ...
}
```



```
double vysledok = 3.0 * this.vypocet(5, 2.0);
```

Signatúra metódy

- Metóda nie je identifikovaná len názvom...
- **Signatúra metódy**:
  - názov metódy
  - (usporiadaný) zoznam typov parametrov metódy

```
public double vypocet(int prva, double druha)
```

Signatúra: [vypocet, int, double]

- Trieda nemôže mať 2 metódy s rovnakou signatúrou...

Dopĺňame telá metód

- Nasleduje plejáda metód na výpis tých kníh, ktoré spĺňajú nejakú požiadavku

```
public void printAll() {  }

public void printByGenre(String genre) {  }

public void printByPublicationYear(int year) {  }

public void printByAuthor(String author) {  }

public void printByRating(double from, double to) {  }
```

Library vs. Book

- Pri výpise by sme chceli vidieť nie len názov ale aj ostatné vlastnosti
  - práca s viacerými súkromnými premennými kníh
- Zoznam kníh nebudeme zaťažovať spracovaním týchto cudzích premenných, poprosíme príslušné knihy, nech nám vygenerujú sformátovaný výstup
  - vo všeobecnosti, každú rozumnú funkcionalitu čiastočne delegujeme na objekty triedy `Book`

Dopĺňame telá metód

- V zozname kníh už pohodlne využívame to, čo potrebujeme

```
public class Library {
  ...
  public void printAll() {
    for (int i=0; i < books.length; i++) {
      System.out.println(books[i].toString());
    }
  }
  ...
}
```

Dopĺňame telá metód

- Podobne hľadanie v súkromnom poli autorov a žánrov necháme na knihy

```
public class Library {
  ...
  public void printByGenre(String genre) {
    for (int i=0; i < books.length; i++) {
      if (books[i].hasGenre(genre))
        System.out.println(books[i].toString());
    }
  }
  ...
}
```

Zopakujme si

- Dáta a funkcionalita, premenné a metódy
- Trieda Object
- Private a public
- Zapúzdrenosť
- Gettre a settre
- Konštruktory
- Signatúra metódy
- Preťaženie metódy

Ďakujem za pozornosť !