



# 8. prednáška (14.11.2016)



**Budujeme triedy,  
zapuzdrujeme triedy,  
pret'ažujeme metódy**



**alebo**

**Murovanie v OOP**





# Zadanie

- Chceme si vyrobiť program na pohodlnú správu našej zbierky DVD-čiek.
- Od nášho programu vyžadujeme nasledovnú funkcionálnosť:
  - vieme vložiť nové DVD
  - vymazať DVD (napríklad sa poškodilo alebo stratilo)
  - vypísať všetky filmy vo vašej zbierke
  - vypísať tie filmy, ktoré zodpovedajú danému žánru (napr. komédie)
  - vypísať tie filmy, ktoré sa dajú pozrieť do nejakého času (napr. do 90 minút)
  - vypísať všetky filmy, kde hral daný herec
  - vypísať filmy, ktoré sú podľa nášho hodnotenia na stupnici od 7 do 10.





# Zadanie

- O každom DVD si budeme uchovávať nasledovné informácie:
  - názov filmu
  - mená hercov, ktorí v ňom hrali
  - žánre do ktorých spadá; predpokladáme, že film môže mať viac žánrov (napr. "kriminálka a thriller" alebo "romantika, komédia a rodinný")
  - dĺžku filmu
  - vaše hodnotenie kvality filmu na stupnici od nula do desať.



# Zadania pre programy všeobecne

- V každom rozumnom zadaní sa špecifikujú dve kľúčové (základné) množiny požiadaviek:
  - **S akými dátami** bude program pracovať
  - **Aké služby** má poskytovať resp. **akú funkcionálnosť** má program mať
- Pokiaľ nemáte jasno v ľubovoľnom z týchto bodov, ani nezačínajte programovať!
- Princíp logického rozdelenia na dáta a funkcionálnosť zachovávajú aj elementárne “súčiastky” OOP - triedy



# Ako budem dáta uchovávať?

- Keď vieme s akými dátami budeme pracovať, musíme sa rozhodnúť, **ako budeme dáta uchovávať**.
- Pozrime na zadanie v časti “dáta o jednom DVD”:
  - názov filmu
  - mená hercov, ktorý v ňom hrali
  - žánre do ktorých spadá; predpokladáme, že film môže mať viac žánrov (napr. "kriminálka a thriller" alebo "romantika, komédia a rodinný")
  - dĺžku filmu
  - naše hodnotenie kvality filmu na stupnici od nula do desať.



# Ako budem dáta uchovávať?

- názov filmu
  - `String`
- mená hercov, ktorý v ňom hrali
  - `String[]`
- žánre do ktorých spadá, predpokladáme, že film môže mať viac žánrov
  - `String[]`
- dĺžku filmu
  - `int`
- naše hodnotenie kvality filmu na stupnici od nula do desať.
  - `double`



# *Trieda ako obal pre viac premenných*

- Všetky tieto informácie predstavujú vlastnosti jedného DVD, tak prečo pracovať paralelne s 5 premennými, keď chceme pracovať s celým DVD-čkom?
- Je lepšie, keď jedno DVD bude spravované z jednej premennej
- Vytvoríme si triedu `Dvd`, ktorá bude šablónou pre všetky DVD-čka a umožní im uchovávať si tieto informácie o sebe



# Trieda ako obal pre viac premenných

- Nepotrebujeme funkcionality `Turtle` ani `WinPane`
- Rozšírime triedu `Object`
- Z triedy `Object` pochádzajú všetky triedy v Java
- Pozrime sa cez `ObjectInspector`





# Trieda ako obal pre viac premenných

```
public class Dvd extends Object {  
    private String    nazovFilmu;  
    private String[]  herci;  
    private String[]  zanre;  
    private int       dlzkaFilmu;  
    private double    hodnotenie;  
}
```



# Trieda ako obal pre viac premenných

```
public class Dvd extends Object {  
    private String nazovFilmu;  
    private String[] herci;  
    private String[] zanre;  
    private int dlzkaFilmu;  
    private double hodnotenie;  
}
```

Nemusíme  
písať



# Použitie

- Môžeme vytvárať nové objekty triedy `Dvd`:

```
public class Spustac {  
    public static void main(String[] args) {  
        Dvd matrix = new Dvd();  
        Dvd shawshank = new Dvd();  
        ...  
    }  
}
```



# Napĺňanie priamym prístupom

- Naše DVD-čka ešte nemajú vo svojich inštančných premenných uložené požadované dáta
- **Najhorší a neodporúčaný prístup** je priame napĺňanie inštančných premenných zvonku cez bodkovú notáciu
  - zmažeme **private**
  - pristupujeme do vnútra objektu cez bodku nasledovanú názvom inštančnej premennej



# Zapuzdrenie

- Objekty sú zodpovedné za obsah svojich inštančných premenných
- S objektami sa chceme rozprávať iba cez ich metódy
- LEBO pri priamom prístupe :



- 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





# 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

- Privátne premenné nevieme z vonku ani čítať
- **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



# 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**

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





# Konštruktor

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

```
public class Dvd {
```

```
    public Dvd() {  
    }  
}
```

Takto by vyzeral  
implicitný konštruktor  
keby ho bolo vidieť

```
...  
}
```



# Konštruktor

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

```
public class Dvd {  
  
    public void Dvd() {  
        ...  
        return this;  
    }  
  
    ...  
}
```



# Konštruktor

- Každý konštruktor
  - Vytvára objekt
  - Vytvára mu inštančné premenné
  - Napĺňa inštančné premenné default hodnotami
- Tiež pomáha chrániť inštančné premenné ako settery
- Programátorom napísané konštruktory môžu nastaviť vhodnejšie inicializačné hodnoty ako default



# Konštruktor

- Vieme ho generovať z Eclipsu
  - Source -> Generate Constructor using fields
  - môžeme si povedať, ktoré inštančné premenné budú nastavované
- Ak máme vytvorený konštruktor s parametrami, implicitný konštruktor **sa nedopĺňa !**
  - ak aj potom chceme používať konštruktor bez parametrov, musíme si ho vytvoriť explicitne!



# Ako konštruktory fungujú

- Každá trieda v Jave pochádza z triedy `Object`
  - `MojaPlocha` rozširuje triedu `WinPane`
  - `WinPane` rozširuje triedu `Pane`
  - `Pane` rozširuje triedu `Object`
- Každé rozšírenie môže dodať nové premenné a nové metódy
- Rozšírená trieda vie automaticky všetko to, čo vie trieda, ktorú rozširuje



# Ako konštruktory fungujú

- Keďže každá trieda pochádza z triedy `Object`
  - Pri vytváraní objektu triedy `MojaPlocha` najprv vznikne objekt, ktorý má všetky inštančné premenné (vlastnosti) a metódy (schopnosti), čo mu definuje trieda `Object`
  - Potom tento objekt získa vlastnosti a schopnosti triedy `Pane`
  - Potom získa vlastnosti a schopnosti triedy `WinPane`
  - Nakoniec získa vlastnosti a schopnosti triedy `MojaPlocha`



# Ako konštruktory fungujú

- **Prvá vec**, čo **každý konštruktor** urobí je to, že zavolá konštruktor triedy, ktorú rozširuje
- Jedinou výnimkou je konštruktor triedy Object, ktorý ako jediný vie naozaj vytvoriť nový objekt
- Triedu, ktorú rozširujeme (rodiča), označujeme kľúčovým slovíčkom **super** (podobne ako triedu, v ktorej sa nachádzame označujeme slovíčkom **this**)
- Volanie rodičovského konštruktora vyzerá nasledovne:

```
public MojaTrieda(int mojVstup) {  
    super(); /*zavoláme rodičovský konštruktor  
             bez parametrov */  
    // ostatné príkazy konštruktora  
}
```



# Ako konštruktory fungujú

- Môžeme volať aj rodičovský konštruktor s parametrami, ak ho má definovaný, napríklad:

```
super (20, "Jožko");
```

- **super** (); bez parametrov písať nemusíme, lebo ak ako prvý príkaz konštruktora nevoláme konštruktor rodiča, vykoná sa implicitne
- Pozor na to, že prázdny konštruktor nemusí existovať, ak bol u rodiča vytvorený iba konštruktor s parametrami





# Ako konštruktory fungujú

- Po vykonaní rodičovského konštruktora
  - konštruktor vytvorí inštančné premenné svojej triedy
  - tieto inštančné premenné sa inicializujú (default hodnotami)
  - spustia sa ostatné príkazy konštruktora
- Vytvorme si triedu `DVDNaPredaj`, ktorá rozširuje triedu `DVD` a skúmame chovanie konštruktov



# Vytvárame zoznam DVD-čiek

- Už vieme jedno DVD-čko vyrobiť a naplniť mu inštančné premenné
- My ale chceme uchovávať veľa DVD-čiek
- Poviete si, ľahká pomoc:

```
public class SkusanieDvd {  
    public static void main(String[] args) {  
        ...  
        Dvd[] filmy = new Dvd[4];  
        filmy[0] = matrix;  
        filmy[1] = shawshank;  
        filmy[2] = fontana;  
        filmy[3] = pachos;  
    }  
}
```



# Vráťme sa k požiadavkám

- Zadanie sa však skladá z dvoch množín požiadaviek:
  - **S akými dátami** bude program pracovať
  - **Aké služby** má poskytovať resp. **akú funkcionality** má program mať



# Vráťme sa k požiadavkám

- Chceme vedieť:
  - vložiť nové DVD
  - vymazať DVD (napríklad sa poškodilo alebo stratilo)
  - vypísať všetky filmy vo vašej zbierke
  - vypísať tie filmy, ktoré zodpovedajú danému žánru (napr. komédie)
  - vypísať tie filmy, ktoré sa dajú pozrieť do nejakého času (napr. < 90 minút)
  - vypísať všetkých filmy, kde hral daný herec
  - vypísať filmy, ktoré sú podľa nášho hodnotenia na stupnici od 7 do 10.



# Vytvárame zoznam DVD-čiek

- Každá požiadavka na funkcionálnosť sa pretaví do metódy
- Kam dáme tieto metódy?
  - Do `main`?
  - Do `Dvd`?
- Tieto požiadavky nemá splňať `Dvd`, ale zoznam `DVD-čiek`





# *Spomnime na zapúzdrenosť*

- Všetky dôležité dáta majú svojho „správca“
- 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 DVD-čiek bude objekt novej triedy `ZoznamDvd`



# Vytvorme si kostru triedy ZoznamDvd

```
public class ZoznamDvd {  
    private Dvd[] filmy;  
  
    public ZoznamDvd() {  
        filmy = new Dvd[0];  
    }  
    public void vlozNoveDvd(Dvd dvd) {  
    }  
    public void vymazDvd(Dvd dvd) {  
    }  
    public void vypisVsetko() {  
    }  
    public void vypisPodlaZanru(String zaner) {  
    }  
    ...  
}
```



# *Dopíňame telá metód*

- Pri vkladaní nafukujeme pole a pridávame nové DVD-čko
- Pri mazaní nájdeme DVD-čko a skracujeme pole
  - Hľadáme podľa inštancie
  - Hľadáme podľa názvu filmu (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 vymazDvd(Dvd dvd) {  
    ...  
}
```

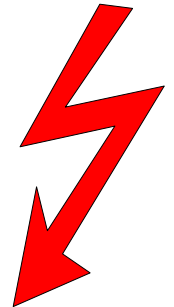
```
public void vymazDvd(String nazovFilmu) {  
    ...  
}
```



# 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);
```





# Dopíňame telá metód

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

```
public void vypisVsetko() {  
}  
public void vypisPodlaZanru(String zaner) {  
}  
public void vypisPodlaCasu(int maximalnyCas) {  
}  
public void vypisPodlaHerca(String menoHerca) {  
}  
public void vypisPodlaHodnotenia(double odHodnota,  
                                double doHodnota) {  
}
```



# Dopíňame telá metód

- Pri výpise by sme chceli vidieť nie len názov ale aj ostatné vlastnosti
- Ide o prácu s viacerými súkromnými premennými DVD-čiek
- Zoznam DVD-čiek nebudeme zat'azovať spracovaním týchto cudzích premenných, poprosíme príslušné DVD-čka, nech nám vygenerujú sformátovaný výstup



# Dopíňame telá metód

- V zozname DVD-čiek už pohodlne využívame to, čo potrebujeme

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



# Dopíňame telá metód

- Podobne hľadanie v súkromnom poli žánrov necháme na DVD-čka

```
public class ZoznamDvd {  
    ...  
    public void vypisPodlaZanru(String zaner) {  
        for (int i=0; i < filmy.length; i++) {  
            if (filmy[i].mamZaner(zaner))  
                System.out.println(filmy[i].toString());  
        }  
    }  
    ...  
}
```



**Ďakujem za pozornosť !**

