9. prednáška (13.11.2017)

Budujeme triedy

alebo

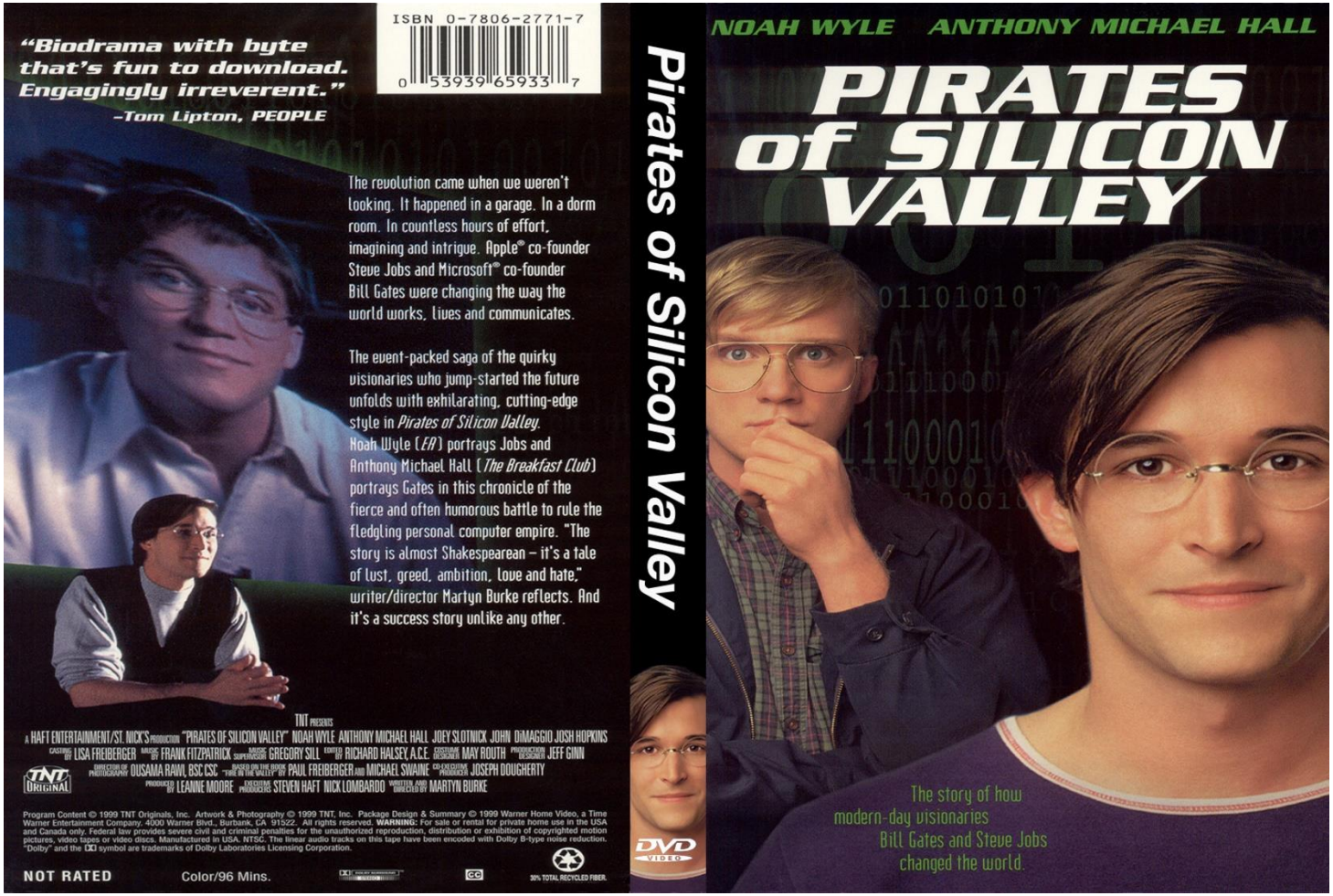
Murovanie v OOP

DVDčka

Zadanie

- Cieľ: pohodlná správu zbierky DVD-čiek.
- Vyžadovaná **funkcionalita**:
  - vieme vložiť info o novom DVD
  - odstrániť DVD (napríklad sa poškodilo alebo stratilo)
  - vypísať všetky filmy v 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 všetkých filmy, kde hral daný herec
  - vypísať filmy, ktoré sú podľa nášho hodnotenia na stupnici od 7 do 10.

DVDčka


"Biodrama with byte that's fun to download. Engagingly irreverent."
–Tom Lipton, PEOPLE
ISBN 0-7806-2771-7
0 53939 65933 7
The revolution came when we weren't looking. It happened in a garage. In a dorm room. In countless hours of effort, imagining and intrigue. Apple® co-founder Steve Jobs and Microsoft® co-founder Bill Gates were changing the way the world works, lives and communicates.
The event-packed saga of the quirky visionaries who jump-started the future unfolds with exhilarating, cutting-edge style in Pirates of Silicon Valley. Noah Wyle (ER) portrays Jobs and Anthony Michael Hall (The Breakfast Club) portrays Gates in this chronicle of the fierce and often humorous battle to rule the fledgling personal computer empire. "The story is almost Shakespearean – it's a tale of lust, greed, ambition, love and hate," writer/director Martyn Burke reflects. And it's a success story unlike any other.
TNT PRESENTS
A HAFT ENTERTAINMENT/ST. NICK'S PRODUCTION "PIRATES OF SILICON VALLEY" NOAH WYLE ANTHONY MICHAEL HALL JOEY SLOTNICK JOHN DIMAGGIO JOSH HOPKINS
CASTING BY LISA FREIBERGER MUSIC BY FRANK FITZPATRICK MUSIC SUPERVISOR GREGORY SILL EDITED BY RICHARD HALSEY, A.C.E. ASSISTANT DESIGNER MAY ROUTH PRODUCTION DESIGNER JEFF GINN
TNT ORIGINAL
DIRECTOR OF PHOTOGRAPHY OUSAMA RAWI, BSC, CSC BASED ON THE BOOK "FIRE IN THE VALLEY" BY PAUL FREIBERGER AND MICHAEL SWAINE CO-PRODUCER JOSEPH DOUGHERTY
PRODUCED BY LEANNE MOORE EXECUTIVE PRODUCERS STEVEN HAFT NICK LOMBARDO WRITTEN AND DIRECTED BY MARTYN BURKE
Program Content © 1999 TNT Originals, Inc. Artwork & Photography © 1999 TNT, Inc. Package Design & Summary © 1999 Warner Home Video, a Time Warner Entertainment Company. 4000 Warner Blvd., Burbank, CA 91522. All rights reserved. WARNING: For sale or rental for private home use in the USA and Canada only. Federal law provides severe civil and criminal penalties for the unauthorized reproduction, distribution or exhibition of copyrighted motion pictures, video tapes or video discs. Manufactured in USA. NTSC. The linear audio tracks on this tape have been encoded with Dolby B-type noise reduction. "Dolby" and the symbol are trademarks of Dolby Laboratories Licensing Corporation.
NOT RATED
Color/96 Mins.
30% TOTAL RECYCLED FIBER.
Pirates of Silicon Valley
DVD VIDEO
NOAH WYLE ANTHONY MICHAEL HALL
PIRATES of SILICON VALLEY
The story of how modern-day visionaries Bill Gates and Steve Jobs changed the world.

Zadanie

- Dôležité informácie o každom DVDčku:
  - názov filmu
  - mená hercov, ktorí v ňom hrali
  - žánre do ktorých spadá
    - film môže mať viac žánrov (napr. "kriminálka a thriller" alebo "romantika, komédia a rodinný")
  - dĺžku filmu
  - hodnotenie kvality filmu: 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 nemáte 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.

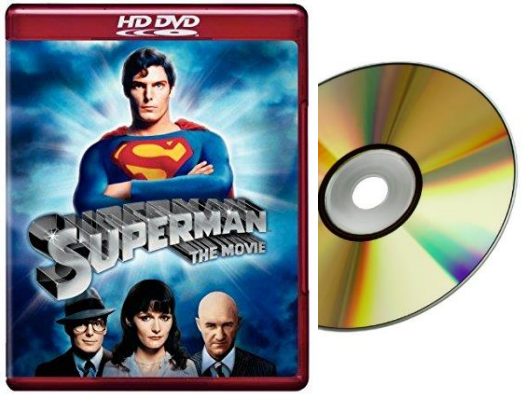
V zložitom systéme (svete) identifikovať jednoduchšie časti – komponenty celku.

Realita vs. objekty a ich triedy

Zoznam/Správca DVDčiek

```
public class ZoznamDvd
```

ZoznamDvd **závisí** od Dvd




```
public class Dvd
```

Zmysel života?

DVD

Objekty

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

`String`

`Color`

`Point2D`

`StringBuilder`

`Turtle`

`MouseEvent`

`WinPane`

DVD a jeho dáta

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

- 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?

Inštančné premenné objektov triedy Dvd?

- 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`
- hodnotenie kvality filmu na stupnici od nula do desať
  - `double`

Trieda Dvd

- Vytvoríme si triedu `Dvd`, ktorá bude **šablónou pre všetky DVD-čka** 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 Dvd extends Object {
   private String   nazovFilmu;
   private String[] herci;
   private String[] zanre;
   private int      dlzkaFilmu;
   private double   hodnotenie;
}
```

Trieda ako obal pre viac premenných

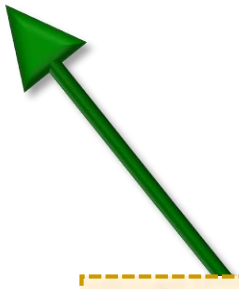
```
public class Dvd extends Object {
   private String   nazovFilmu;
   private String[] herci;
   private String[] zanre;
   private int      dlzkaFilmu;
   private double   hodnotenie;
}
```

Nemusíme písať

Použitie

- Môžeme vytvárať nové objeky triedy `Dvd`:

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



Ale ako dostaneme dáta do objektov?

Ako dostať dáta do objektov?

- 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[] getHerci() {
      return herci;
}
```

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

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

```
public String[] getHerci() {
      return herci.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 Dvd {

    public void Dvd(...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 Dvd {

    public Dvd() {
    }

...
}
```

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 DVD-čiek

- Chceme uchovávať veľa DVD-čiek
- Pozor na prvoplánové riešenia:

```
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] = pacho;
  }
}
```

Zadanie

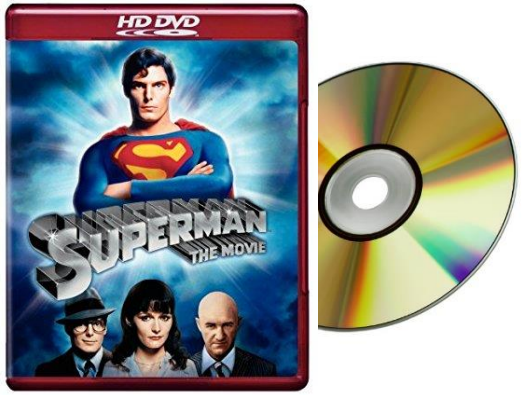
- Chceme vedieť:
  - vložiť nové DVD
  - vymazať DVD (napríklad sa poškodilo alebo stratilo)
  - vypísať všetky filmy vo vašej zbierke
  - vypísať tie filmy, ktoré zodpovedajú danému žánru (napr. komédie)
  - vypísať tie filmy, ktoré sa dajú pozrieť do nejakého času (napr. < 90 minút)
  - vypísať všetkých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

Realita vs. objekty a ich triedy

Zoznam/Správca DVDčiek

```
public class ZoznamDvd
```

ZoznamDvd **závisí** od Dvd




```
public class Dvd
```

Zmysel života?

DVD

Správca DVD 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 DVD-čiek bude objekt novej triedy `ZoznamDvd`

Kostra 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) {
  }
  ...
}
```

Dopĺňame 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

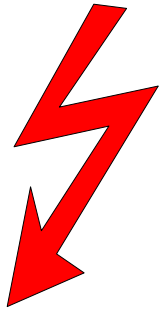
- Vhodné v prípade, že metódy robia to isté len sa líšia svojim vstupom
  - Počtom parametrov alebo
  - Aspoň jedným **typom** parametra

```
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);
```

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 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) {
}
```

ZoznamDvd vs. Dvd

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

Dopĺňame 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());
    }
  }
  ...
}
```

Dopĺňame 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 !