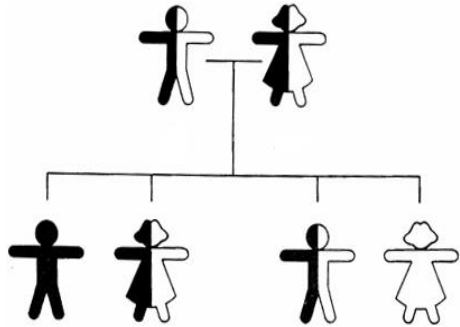




# 10. prednáška (26.11.2018)



## Dedičnosť'

a

## polymorfizmus

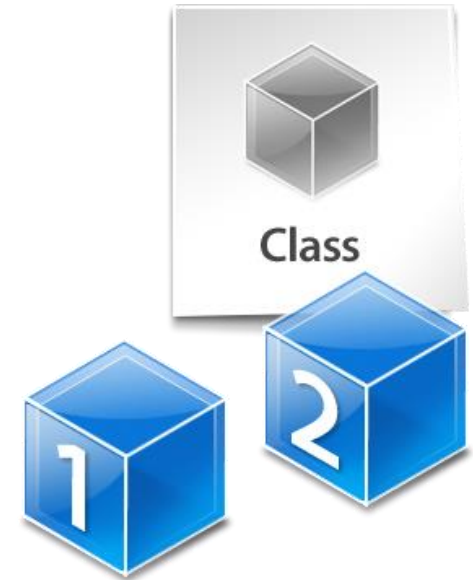


*New Rules!*



# Čo je to trieda?

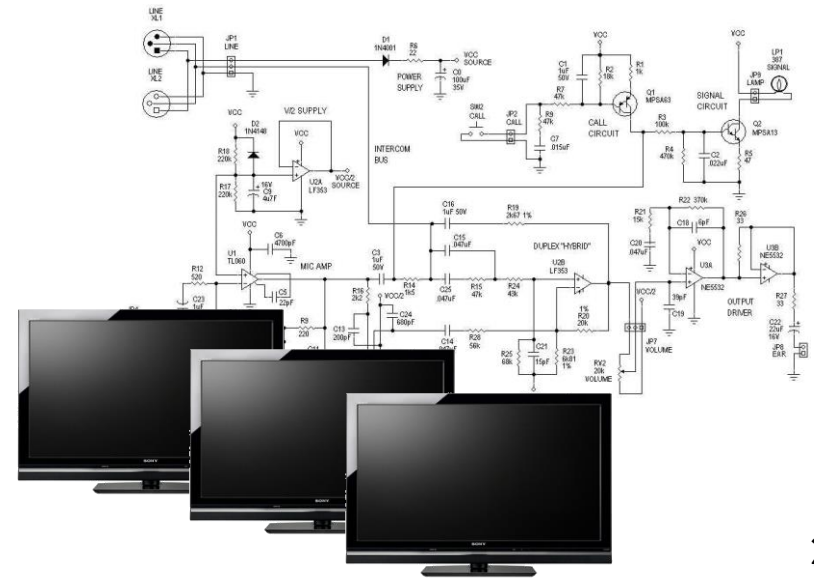
- Trieda je **šablóna** (vzor), ktorý **predpisuje** aké **inštančné premenné** a aké **metódy** majú objekty danej triedy a čo sa udeje pri zavolaní týchto metód



**class**  
Turtle

inštančné  
premenne

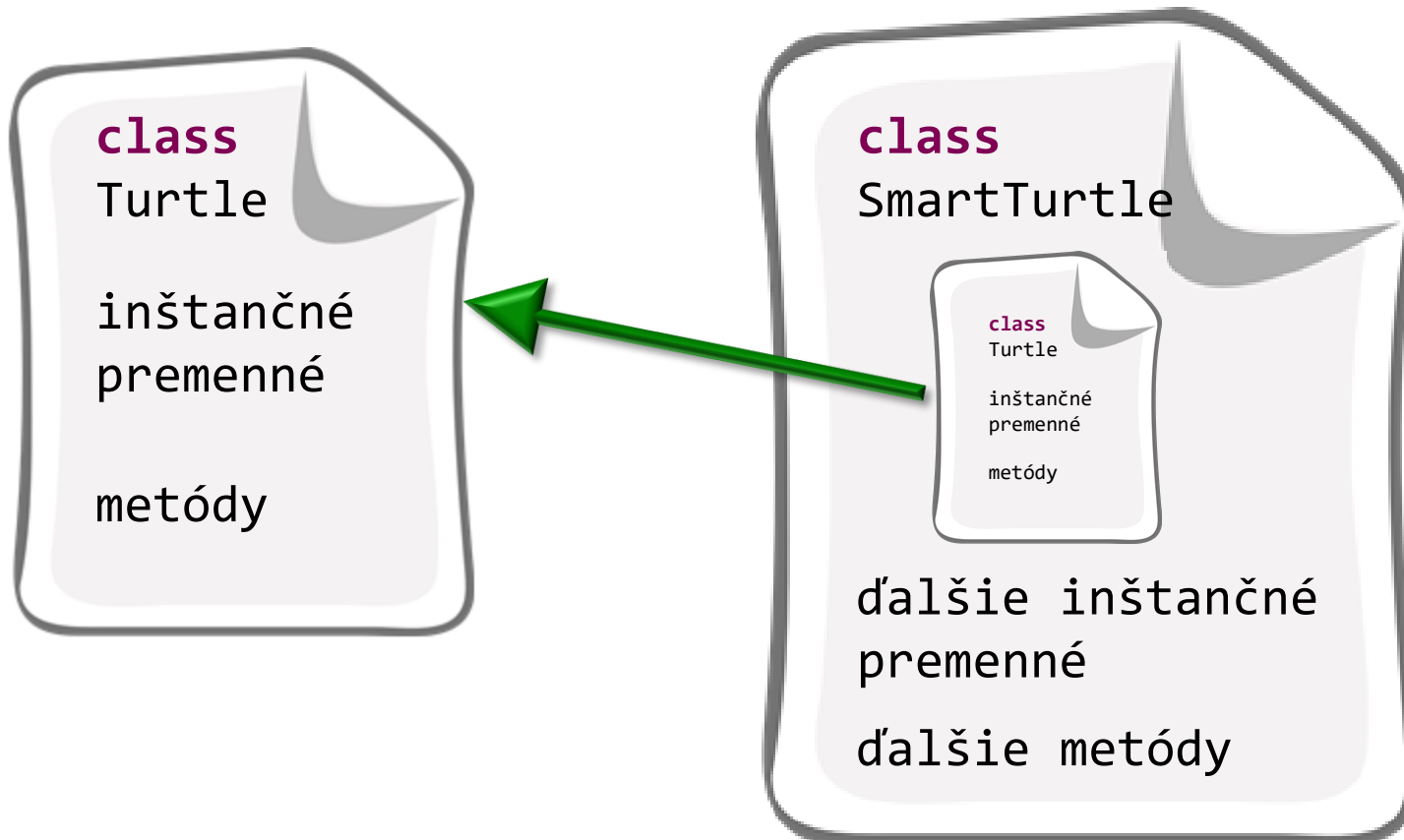
metódy





# Rozširovanie tried

```
public class SmartTurtle extends Turtle
```





# Premenné referenčného typu

```
Turtle franklin;
```

Referencie na objekty  
akej triedy môžu byť  
uložené v premennej

Názov premennej

- Premenná `franklin` môže referencovať len objekty triedy `Turtle` („uchovávať ich rodné čísla“)
- Špeciálna hodnota **`null`** určujúca, že premenná neuchováva referenciu na objekt.



# Nočná mora: Duplicita kódu

- Duplicita kódu je **nežiadúca** z mnohých dôvodov:
  - chyby
  - údržba
  - optimalizácia
  - budúce zmeny



- Riešenie:
  - metódy
  - triedy
  - „knižnice“





# Zadanie (z minula)

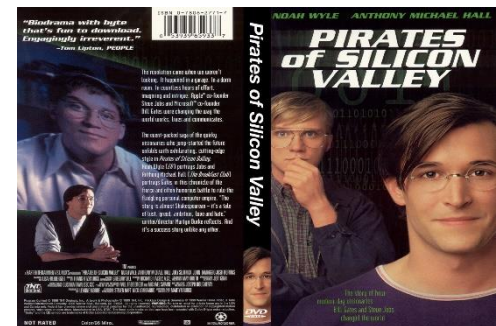
- Ciel': pohodlná správu zbierky DVD-čiek.
- Vyžadovaná **funkcionalita**:
  - vieme vložit' info o novom DVD
  - odstrániť DVD
  - vypísať všetky filmy v zbierke
  - vypísať tie filmy, ktoré zodpovedajú danému žánru (napr. komédie)
  - vypísať tie filmy, ktoré sa dajú pozrieť do nejakého času (napr. do 90 minút)
  - vypísať všetkých filmy, kde hral daný herec
  - vypísať filmy, ktoré sú podľa nášho hodnotenia na stupnici od 7 do 10.





# Zadanie (z minula)

- 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





# Zadania pre programy

- V každom rozumnom zadaní sa špecifikujú dve kľúčové (základné) množiny požiadaviek:
  - **s akými dátami** bude program pracovať
    - ... inštančné premenné
  - **aké služby** má poskytovať resp. **akú funkcionality** má program (objekty triedy) mať
    - ... metódy





# Zapúzdzrenie (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**
- Konštruktory, metódy (getter, setter)



# Konštruktor

- Inicializuje inštančné premenné (aj na základe hodnôt parametrov)

```
public class Dvd {
```

```
    public Dvd(String nazovFilmu) {  
        this(nazovFilmu, 0, 0);  
    }
```

Volanie iného konštruktora  
(ak sa použije, musí to byť  
prvý príkaz konštruktora)

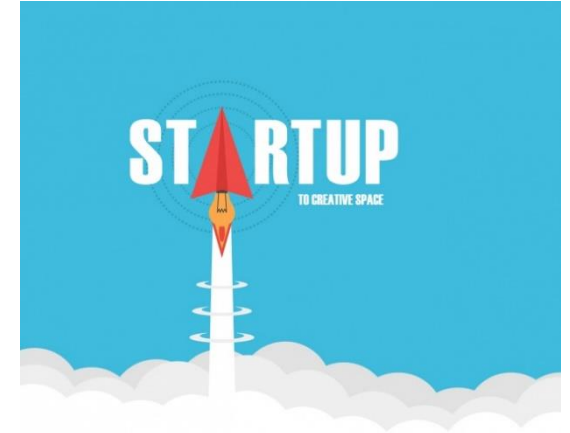


```
    public Dvd(String nazovFilmu, int dlzkaFilmu,  
                double hodnotenie) {  
        this.nazovFilmu = nazovFilmu;  
        this.dlzkaFilmu = dlzkaFilmu;  
        this.hodnotenie = hodnotenie;  
    }  
}
```



# Rozširujeme zadanie

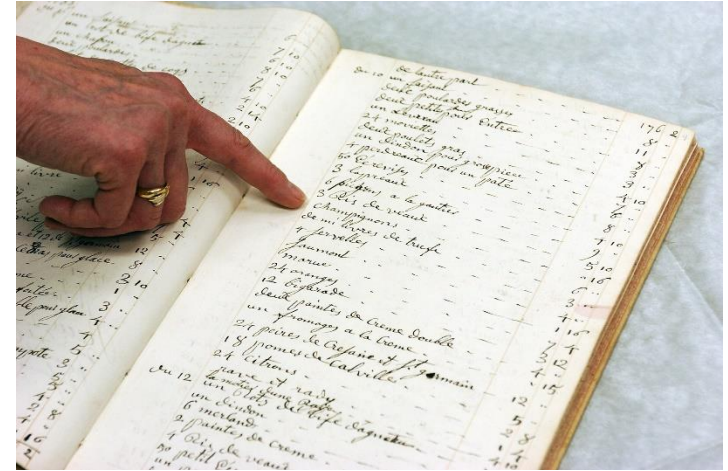
- správa DVD → správa **filmov**
- rôzne umiestnenia filmov:
  - DVD, VHS, počítač, ...





# Rozširujeme zadanie

- Na DVD
  - očíslované
- Na páske
  - očíslované
  - chceme vedieť aj začiatočnú minútu (kópie z TV)
- V súbore v počítači
  - názov počítača
  - cesta k súboru
  - veľkosť súboru





# Aké triedy?



**class**  
FilmNaDvd

inštančné  
premenné

metódy

**class**  
FilmNaPaske

inštančné  
premenné

metódy

**class**  
FilmVPocitaci

inštančné  
premenné

metódy



# Veľa rozdielneho aj spoločného...

- Filmy na ľubovoľnom médiu majú niektoré **rozdielne dáta**:
  - identifikácia média
    - očíslovanie, meno počítača a cesta k súboru
  - doplňujúce údaje
    - začiatočná minúta, veľkosť súboru
- Filmy na ľubovoľnom médiu majú aj **rozdielne chovanie funkčných schopností** :
  - výpis umiestnenia filmu
  - spôsob uloženia do súboru a načítania z neho
  - poskytovanie dodatočných informácií



# Veľa rozdielneho aj spoločného...

- Filmy na ľubovoľnom médiu majú ale aj **spoločné dáta**:
  - 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
  - hodnotenie kvality filmu na stupnici od nula do desať.
- Filmy na ľubovoľnom médiu majú aj **spoločnú funkcionality**:
  - **boolean** mamZaner()
  - **boolean** mamHerca()
  - `String toString()` - vypisuje (zatiaľ) len spoločné dáta



# Aké triedy?



**class**  
FilmNaDvd

[Green bar]

[Yellow bar]

**class**  
FilmNaPaske

[Green bar]

[Blue bar]

**class**  
FilmVPocitaci

[Green bar]

[Red bar]





# Nočná mora: Duplicita kódu

- Duplicita kódu je **nežiadúca** z mnohých dôvodov:
  - chyby
  - údržba
  - optimalizácia
  - budúce zmeny

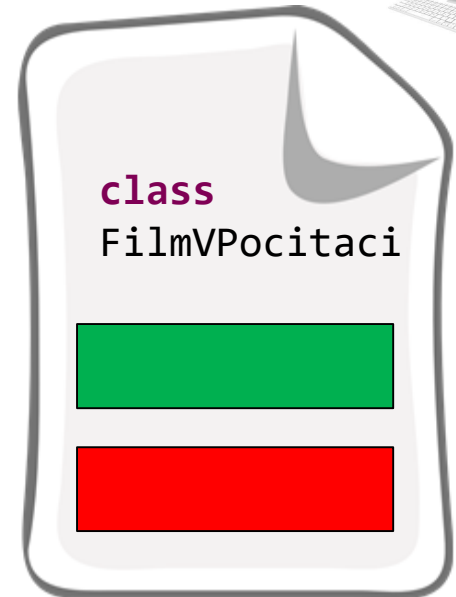
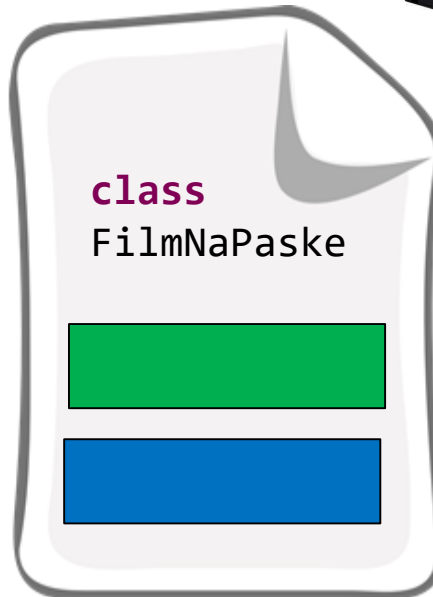
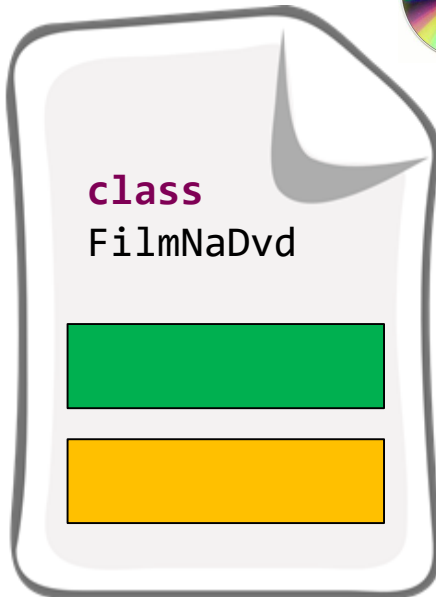
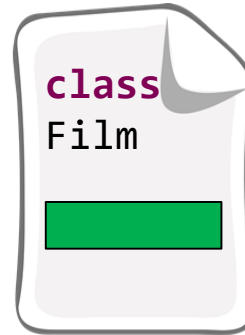


- Riešenie:
  - metódy
  - triedy
  - „knižnice“



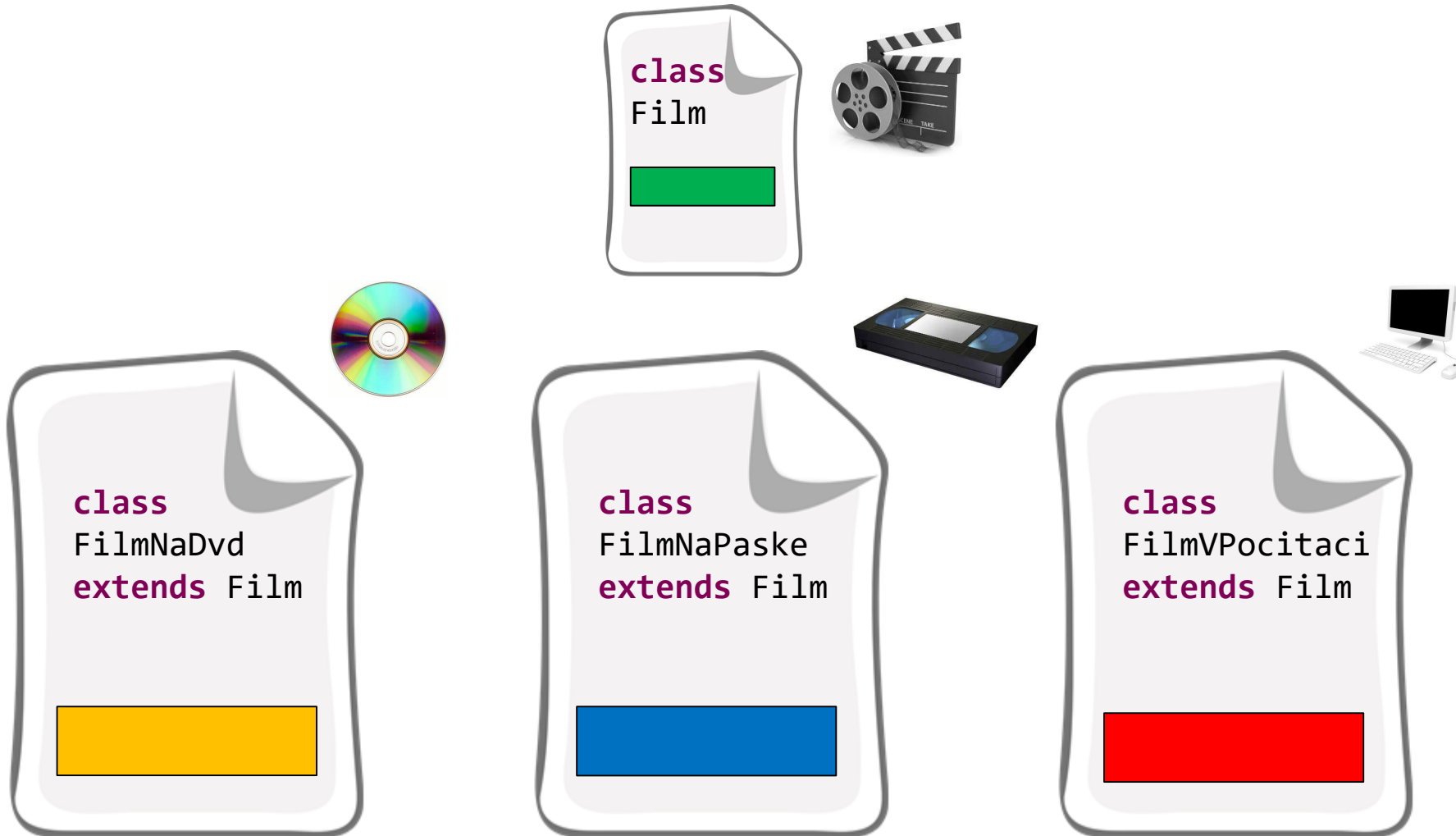


# Aké triedy?





# Eliminácia duplicity rozšírením





# Dedičnosť = rozširovanie

- Vytvoríme si triedu `Film`, ktorá
  - obsahuje **spoločné dáta a metódy** pre všetky filmy bez ohľadu na médiá, na ktorých sú uložené
- Od nej oddedené triedy, t.j. triedy, ktoré rozširujú vlastnosti triedy `Film` o:
  - `FilmNaDvd`
    - `cisloDvdcka`
  - `FilmNaPaske`
    - `cisloPasky`, `zaciatok`
  - `FilmVPocitaci`
    - `nazovPocitaca`, `cestaKSsuboru`, `velkostSuboru`

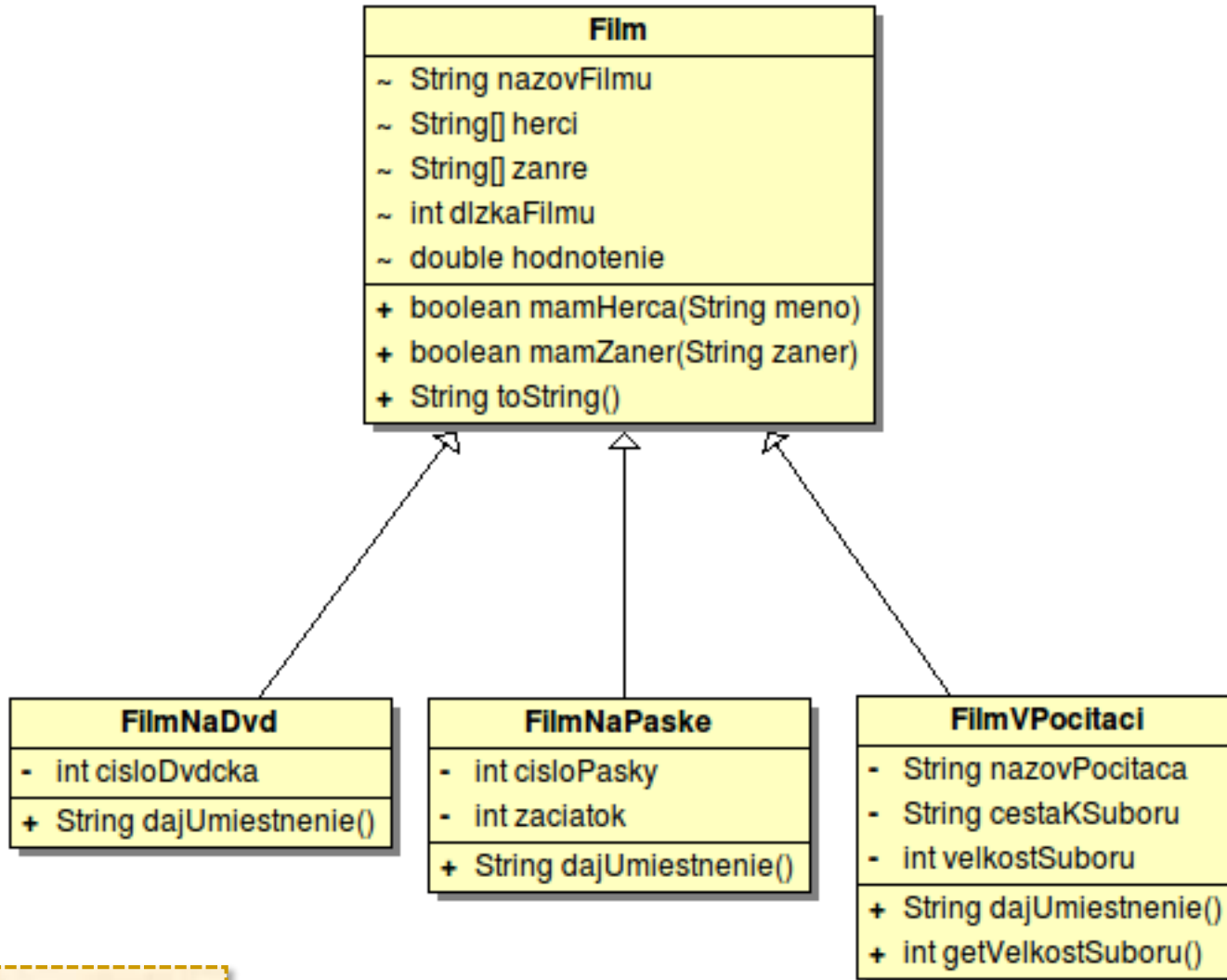


# dajUmiestnenie

- Každá „filmový“ objekt vie povedať, kde sa nachádza
  - `String dajUmiestnenie()`
- Každý to povie po svojom (nejde o kópiu)
  - `FilmNaDvd`
    - DVD číslo 34
  - `FilmNaPaske`
    - Páska číslo 22 od 97. minúty
  - `FilmVPocitaci`
    - Počítač Žofka v súbore `C:\filmy\janosik.avi`



# Triedový diagram



Programujeme...



# Konštruktory a dedičnosť

- Prvý príkaz konštruktora **musí byť vždy** volanie **konštruktora rodičovskej** (rozširovanej) triedy alebo iného konštruktora vytváranej triedy.
- Konštruktor rodičovskej triedy voláme cez  
`super(...parametre...);`
- Java pre „lenivých“:
  - ak programátor pravidlo nedodrží, Java doplňuje do prvého riadku konštruktora: `super();`
  - pozor: rodičovská trieda nemusí mať bezparametrový konštruktor → problém (chyba) už pri vytvorení triedy



# Implicitný konštruktor

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

```
public class Film {
```

```
    public Film() {  
        super();  
    }
```

Takto by vyzeral  
implicitný konštruktor  
keby ho bolo vidieť

```
    ...  
}
```

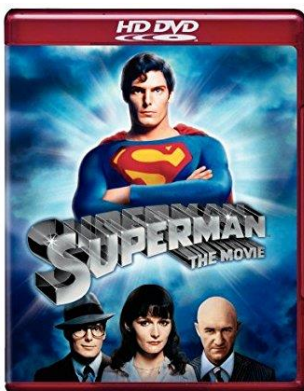




# Minule...



Zmysel života?



Zoznam/Správca DVDčiek

```
public class ZoznamDvd
```

ZoznamDvd závisí od Dvd

```
public class Dvd
```

DVD



# Dnes...



Zoznam/Správca filmov

**public class** ZoznamFilmov  
čo a ako má uchovať?



**public class** Film



**public class** FilmNaPaske



**public class** FilmNaDvd



**public class** FilmVPocitaci



# Renovujeme zoznam filmov

- Prvý nápad:
  - Máme 3 triedy, dáme 3 polia

```
public class ZoznamFilmov {  
    private FilmNaDvd[]    filmyDvd;  
    private FilmNaPaske[]  filmyPasky;  
    private FilmVPocitaci[] filmyPocitac;  
    ...  
}
```



# Renovujeme zoznam filmov

- Prvý nápad:
  - Ale potom máme všade 3 cykly

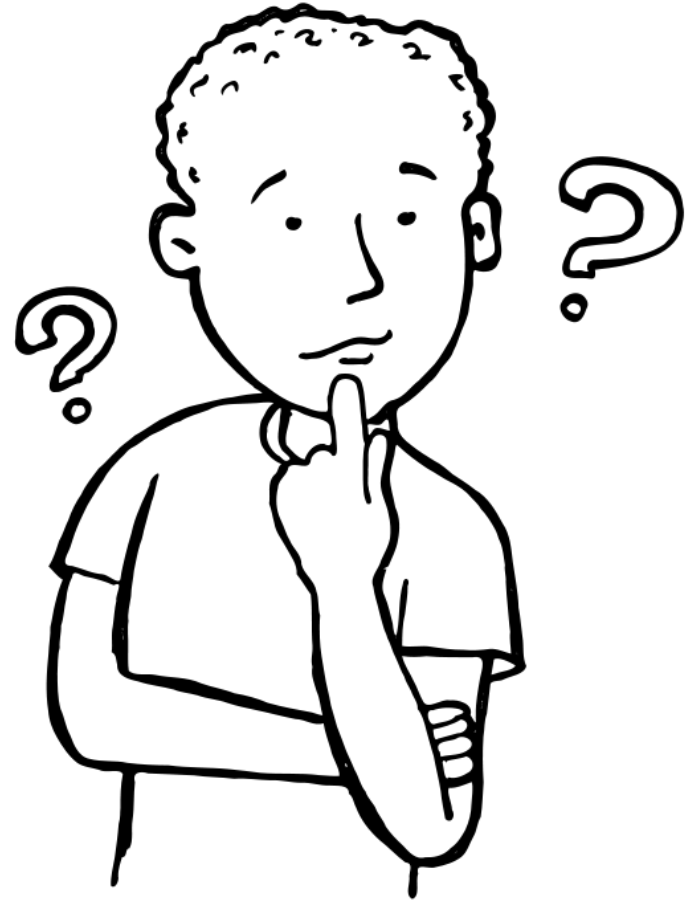


```
public class ZoznamFilmov {  
    ...  
    public void vypisVsetko() {  
        for (int i = 0; i < filmyDvd.length; i++)  
            System.out.println(filmyDvd[i].toString());  
        for (int i = 0; i < filmyPasky.length; i++)  
            System.out.println(filmyPasky[i].toString()); 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



# Renovujeme zoznam filmov

- Filozofická úvaha:
  - FilmNaDvd je Film
  - FilmNaPaske je Film
  - FilmVPocitaci je Film



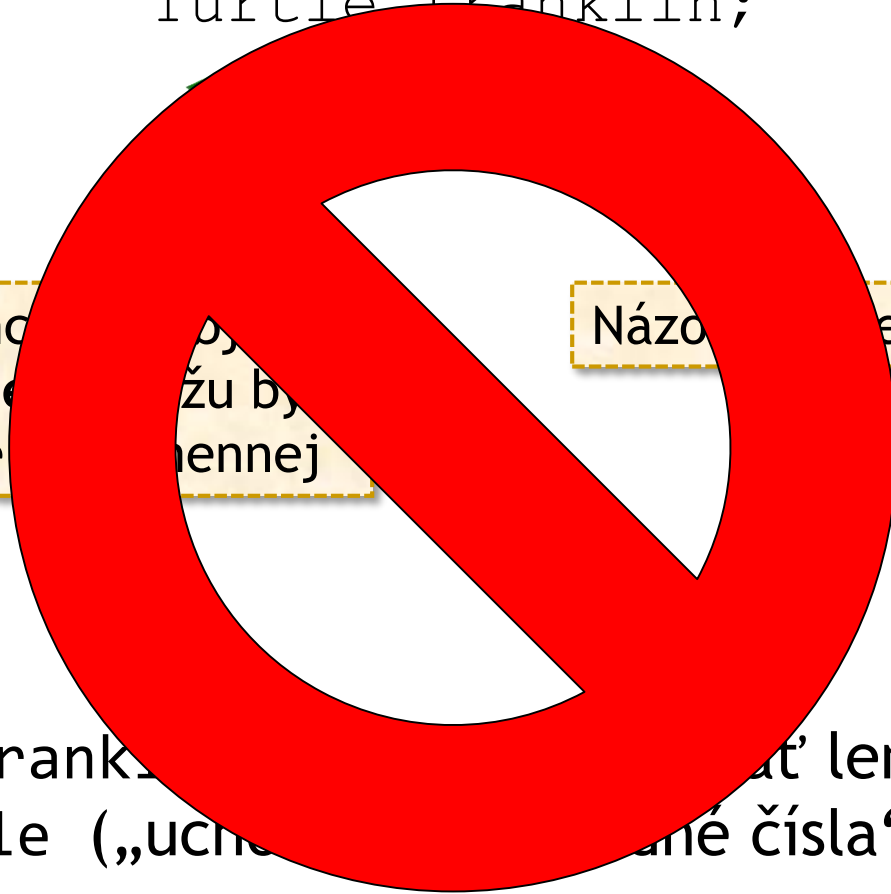


# Premenné referenčného typu

```
Turtle franklin;
```

Referencia na objekt  
akej triedy  
uložené v pamäti

Názov premennej



- Premenná franklin môže obsahovať len objekty triedy Turtle („učenie o číslach“)
- Špeciálna hodnota **null** určujúca, že premenná neuchováva referenciu na objekt.



# Premenné referenčného typu

```
Turtle franklin;
```

Referencie na objekty  
akej triedy môžu byť  
uložené v premennej


Názov premennej

- Premenná franklin môže referencovať len objekty triedy Turtle **a tried, ktoré rozširujú triedu Turtle**



# Premenné referenčného typu

```
Turtle franklin = new SmartTurtle();  
franklin.step(100);  
franklin.smartMethod();
```



**Chyba:** Cez premennú franklin môžem volať len metódy z triedy Turtle. Metódy definované v triede SmartTurtle sú nedostupné.

```
SmartTurtle franklin = new Turtle();
```





# Renovujeme zoznam filmov

- Referencie typu `Film` môžu uchovávať aj referencie na objekty tried `FilmNaDvd`, `FilmNaPaske`, `FilmVPocitac`
  - vieme pristupovať k metódam z triedy `Film`
  - nevieme pristupovať k metóde `dajUmiestnenie`

```
public class ZoznamFilmov {  
    private Film[] filmy;  
    ...  
}
```



# Renovujeme zoznam filmov

- Riešenie:
  - Máme síce 3 triedy, ale stačí nám 1 pole
  - Stačí nám všade iba 1 cyklus

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





# Ale máme problém

- Problém:
  - Keď všetko je film, ako zistíme umiestnenie?
  - Nevieme predsa zavolať `dajUmiestnenie...`

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

The method `dajUmiestnenie()` is undefined for the type `Film`



# Identifikujeme a pretypujeme

- Operátor **instanceof**:

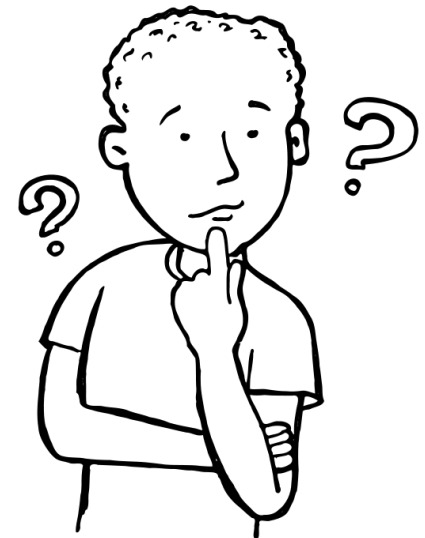
- Vieme porovnať triedu objektu referencovaného z premennej

```
public class ZoznamFilmov {
...
    public void vypisVsetko() {
        for (int i = 0; i < filmy.length; i++)
            System.out.println(filmy[i].toString());
            if (filmy[i] instanceof FilmNaDvd) {
                FilmNaDvd film = (FilmNaDvd) filmy[i];
                System.out.println(film.dajUmiestnenie());
            }
            if (filmy[i] instanceof FilmNaPaske) {
                FilmNaPaske film = (FilmNaPaske) filmy[i];
                System.out.println(film.dajUmiestnenie());
            }
            ...
        }
    }
}
```



# Renovujeme zoznam filmov

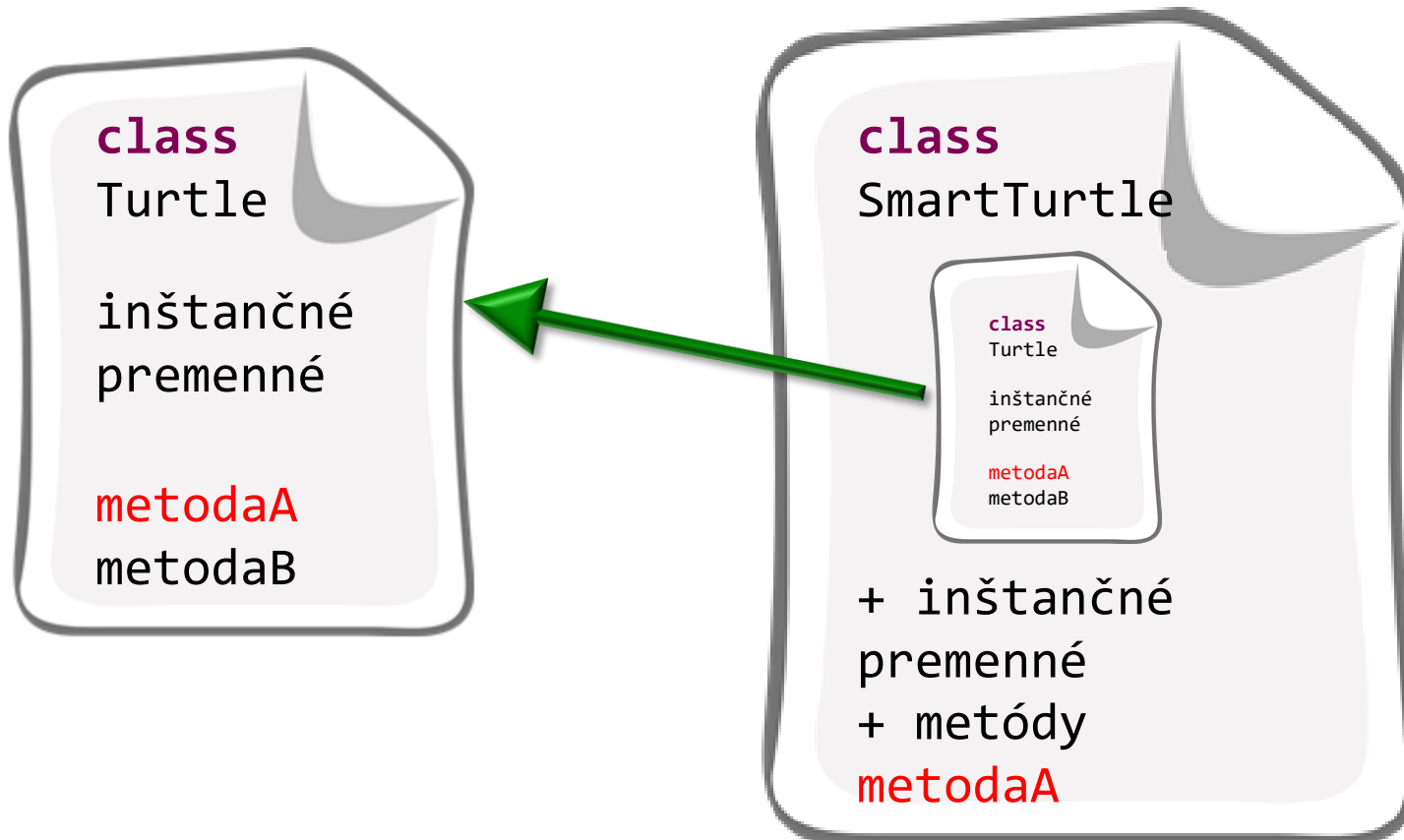
- Každý film má nejaké umiestnenie
  - pridáme metódu `dajUmiestnenie` do triedy `Film`?
- `FilmNaDvd`, `FilmNaPaske`, `FilmVPocitaci` v rámci metódy `dajUmiestnenie` spraví úplne inú vec a s inými dátami
  - ...o ktorých tvorca triedy `Film` nemá poňatia...





# Prekrývavanie metód

```
public class SmartTurtle extends Turtle
```





# Prekrývanie metód

- Tvorca triedy môže **prekryt'** implementáciu **metódy zdedenej** z rodičovskej triedy
  - čo dedím, musím mať
  - aj keď dedím metódu, môžem ju „preprogramovať“
  - prekrytie = ak ktokoľvek zavolá metódu (cez akúkoľvek premennú referenčného typu) vykoná sa preprogramovaná implementácia metódy
  
- Prekrytie = override



# Polymorfizmus

- Polymorfizmus (viactvarovosť)
  - Na objekte vieme volať metódy definované v jeho triede, alebo v jej predkoch
  - Ak trieda-potomok definuje **rovnakú metódu** ako trieda-predok nastáva **prekrytie metódy**
    - Metódu predka z objektu „nevidno“ - použije sa „nová“ metóda potomka
    - Rovnaká metóda = rovnaký názov a rovnaký počet, poradie a typy parametrov
- Tento mechanizmus je nezávislý od typu referencujúcej premennej
  - Objekt vie to, akej triedy je, a nie to, akého typu je premenná, ktorá ma uloženú referenciu naňho
  - Typ premennej určuje, čo vieme na referencovaných objektoch volať





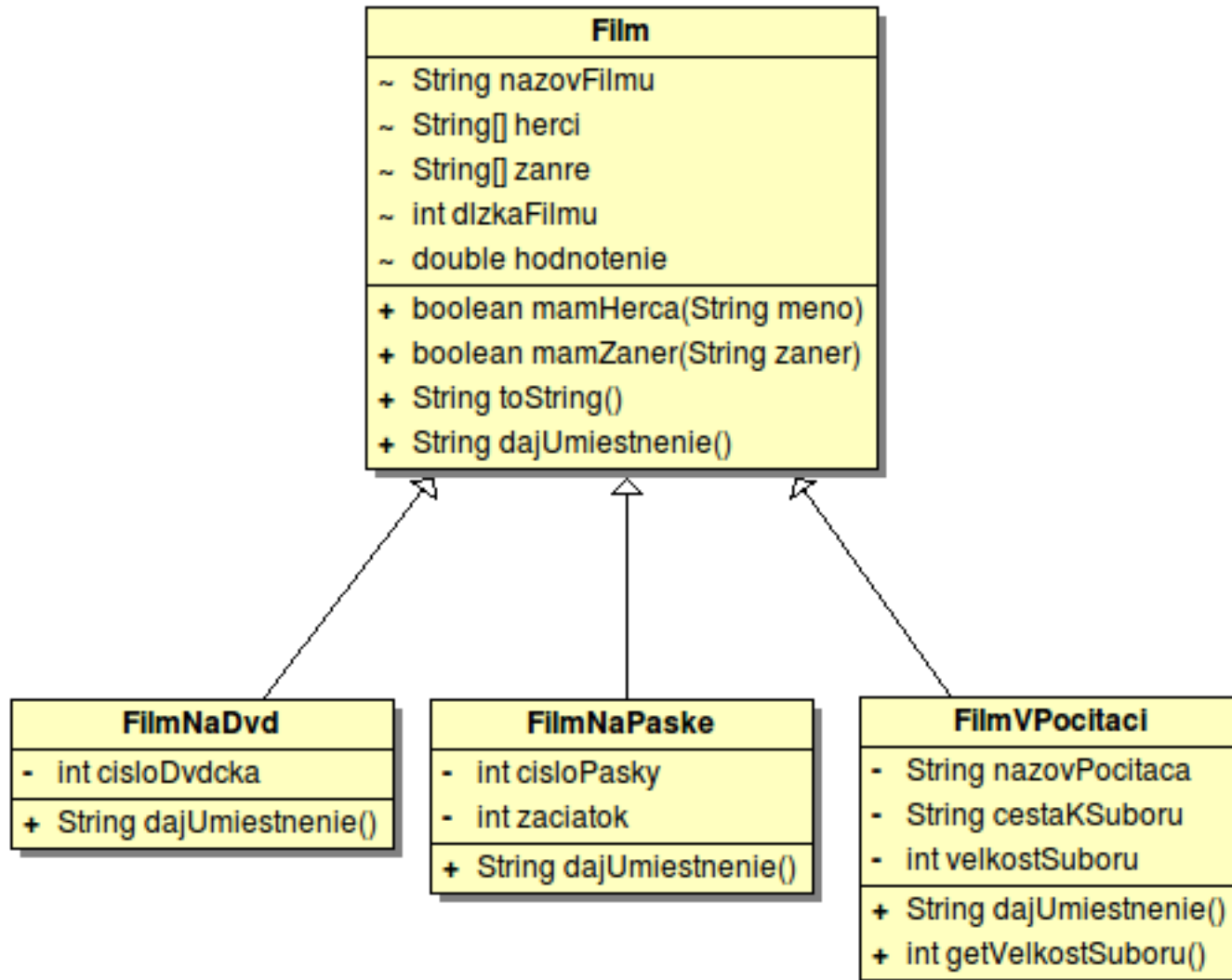
# Polymorfizmus

- Dopíšeme metódu `dajUmiestnenie()` aj do triedy `Film`.
  - Premenné typu `Film` už vedia takúto metódu zavolať
  - Túto metódu však objekty tried `FilmNaDvd`, `FilmNaPaske` a `FilmVPocitaci` nebudú používať, lebo použijú svoje metódy `dajUmiestnenie()`, ktorými túto metódu prekryjú

```
public class Film {  
    ...  
    public String dajUmiestnenie() {  
        return "nemám umiestnenie";  
    }  
    ...  
}
```



# Triedový diagram






# Polymorfizmus filmov

- Každý objekt si zavolá `dajUmiestnenie()` zo svojej triedy

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



**Polymorfizmus:** Vykoná sa implementácia zodpovedajúca triede, ktorej ten objekt je inštanciou.



# Problém vyriešený

- Ak by sme sa chceli predsa len dostať k pôvodnej metóde rodiča použijeme v metóde dieťaťa volanie cez **super**

```
public class FilmNaDvd {  
    ...  
    public String dajPovodneUmiestnenie() {  
        return "povodne:" + super.dajUmiestnenie();  
    }  
    ...  
}
```



# Polymorfizmus najhrubšieho zrna :)

- Čo keby sme chceli, aby `toString()` vrátil aj umiestnenie?
  - Ale ved' `toString()` je v triede `Film` a nevidí na inštančné premenné tried `FilmNaDvd`, `FilmNaPaske` a `FilmVPocitaci`
  - Zavoláme `dajUmiestnenie()` v metóde `toString()` v triede `Film`
    - voláme `toString()` **na objekte** triedy potomka
    - objekt si teda zavolá svoju prekrytú metódu `dajUmiestnenie()`



# Triedový diagram

```

nazovFilmu: "Pacho, Hybský zbojník"
herci: ["Jozef Kroner", ...]
zanre: ["komédia"]
dlzkaFilmu: 91
hodnotenie: 8,5
boolean mamHerca(String)
boolean mamZaner(String)
String toString()
String dajUmiestnenie()
    
```



volá

objekt triedy Film:

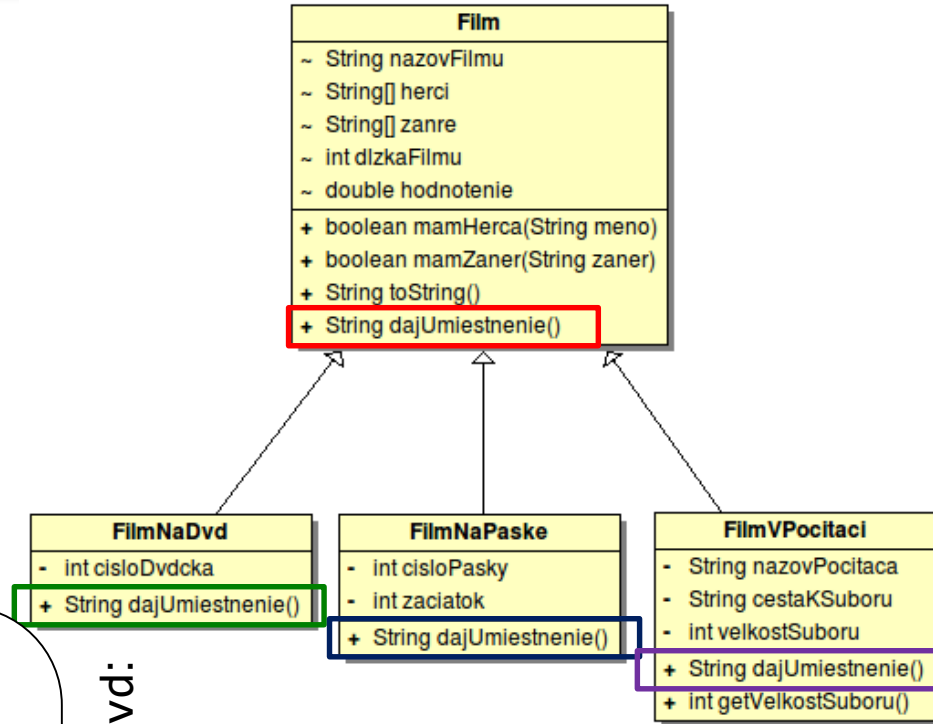
```

nazovFilmu: "The Matrix"
herci: ["Keanu Reeves", ...]
zanre: ["akčný", "Sci-fi"]
dlzkaFilmu: 136
hodnotenie: 8,7
cisloDvdcka: 21
boolean mamHerca(String)
boolean mamZaner(String)
String toString()
String dajUmiestnenie()
    
```



volá

objekt triedy FilmNaDvd:





# Polymorfizmus – novinka?

- Myšacie udalosti v JPAZe:
  - onMousePressed - len obyčajné **prekrytie** metódy z triedy WinPane, kde sme dali vlastnú implementáciu a vďaka polymorfizmu sa zavolať náš obslužný kód
- Záhadný toString a System.out.println:
  - metóda toString pochádza z triedy Object
    - má ju každý objekt
    - **jej prekrytím** implementujeme ako má vyzerat' reťazcová (textová) reprezentácia objektov danej triedy
  - metódu toString využíva System.out.println, ale aj Java, keď pri zlepovaní reťazcov potrebuje referenciu prerobiť na reťazec



**Ďakujem za pozornosť !**

