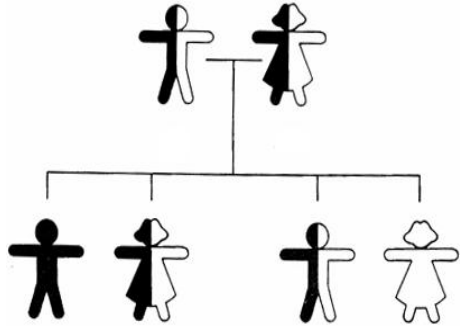




10. prednáška (23.11.2011)



Dedičnosť a polymorfizmus



alebo

rodinkárstvo v OOP





Krátke zopakovanie

- Mali sme program na pohodlnú správu 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 všetkých filmov, kde hral daný herec
 - vypísať filmy, ktoré sú podľa vášho hodnotenia na stupnici od 7 do 10.





Krátke zopakovanie

- O každom DVD sme si uchovávali 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

- Pochopili sme, že v každom rozumnom zadaní sa špecifikujú dve kľúčové (základné) množiny požiadaviek:
 - **S akými dátami** bude program pracovať
 - Z nich sa stanú inštančné premenné
 - **Aké služby** má poskytovať resp. **akú funkcionality** má program mať
 - Z nich sa stanú metódy



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 jedno DVD-čko je objekt triedy `Dvd`
- Správcom pre pole DVD-čiek je objekt triedy `ZoznamDvd`



Zapúzdrenosť

- Ak zistíme, že chceme robiť prácu s viacerými súkromnými premennými cudzích objektov alebo s komplikovanejšími súkromnými premennými
 - vytvoríme si radšej novú metódu pre tieto objekty v ich triede, ktorá túto prácu spraví za nás
 - zoznam DVD-čiek nebudeme zaťažovať výpisom viacerých inštančných premenných DVD-čiek, poprosíme príslušné DVD-čka, nech nám vygenerujú sformátovaný výstup
- **S objektmi chceme komunikovať** a nie hrabať sa v ich inštančných premenných!!!



Napĺňanie inštančných premenných

- Najhorší a neodporúčaný prístup je priame napĺňanie inštančných premenných zvonku cez bodkovú notáciu
- Použijeme **settery** a **gettery**, a/alebo
- Použijeme **konštruktory**
- Bez priameho prístupu vieme ochrániť hodnoty inštančných premenných a ostať konzistentnom stave



Rozširujeme zadanie

- Stalo sa, čo ste nepredpokladali. Vaša rodina bola nadšená vašim zoznamom DVD-čiek a chce od vás, aby ste evidovali aj všetky filmy na videopáskach a v počítači.





Rozširujeme zadanie

- Už im nestačia informácie o filme, chcú vedieť aj to, kde film hľadať a v prípade súborov aj veľkosť súboru:
- Na DVD
 - Očíslované (evidencia je zatiaľ na ošumtelom papieri)
- Na páske
 - Očíslované (papier v šalátovom vydaní)
 - Chceme vedieť aj začiatočnú minútu (kópie z TV)
- V súbore v počítači
 - Názov počítača
 - Cesta k súboru
 - Veľkosť súboru



Veľa rozdielneho aj spoločného...

- Filmy na ľubovoľnom médiu majú niektoré **rozdielne dáta**:
 - Identifikácia média
 - Očíslovanie, meno počítača a cesta k súboru
 - Doplnujúce údaje
 - začiatočná minúta, veľkosť súboru



Veľa rozdielneho aj spoločného...

- Filmy na ľubovoľnom médiu majú niektoré **rozdielne dáta**:
 - Identifikácia média
 - Očíslovanie, meno počítača a cesta k súboru
 - Doplnujúce 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, 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
 - hodnotenie kvality filmu na stupnici od nula do desať.



Veľa rozdielneho aj spoločného...

- Filmy na ľubovoľnom médiu majú ale aj **spoločné dáta**:
 - 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
 - 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



Zásada dobrého programátora

- Dobrý programátor chce mať každú funkcionálnu napísanú **iba v jedinej metóde z celého programu**
- Každá metóda má veľkú šancu na svoju zmenu z rôznych dôvodov
 - prídu dáta, s ktorými sa pred tým nerátalo
 - zákazník chce zmenu chovania metódy
 - Y2K
 - metóda je pomalá a treba ju zmeniť
 - ...



Každá metóda len v jednej kópii !!!

- Nevýhody viac ako 1 kópie tej istej metódy pri zmene:
 - treba veľa písať
 - zabudneme na nejakú kópiu v inej triede
 - ak by sme zabudli nasleduje nezriedka SODOMA-GOMORA, trhanie vlasov, peňažné postihy, ...
- Ak by sme vytvorili 3 triedy pre filmy s kópiami premenných a metód, len s doplnením nových vecí, porušili by sme zásadu jednej metódy s rovnakým telom
 - Čo ak bude mať herec okrem mena aj svoje ocenenia, rasu, vek, fotku, ...



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`

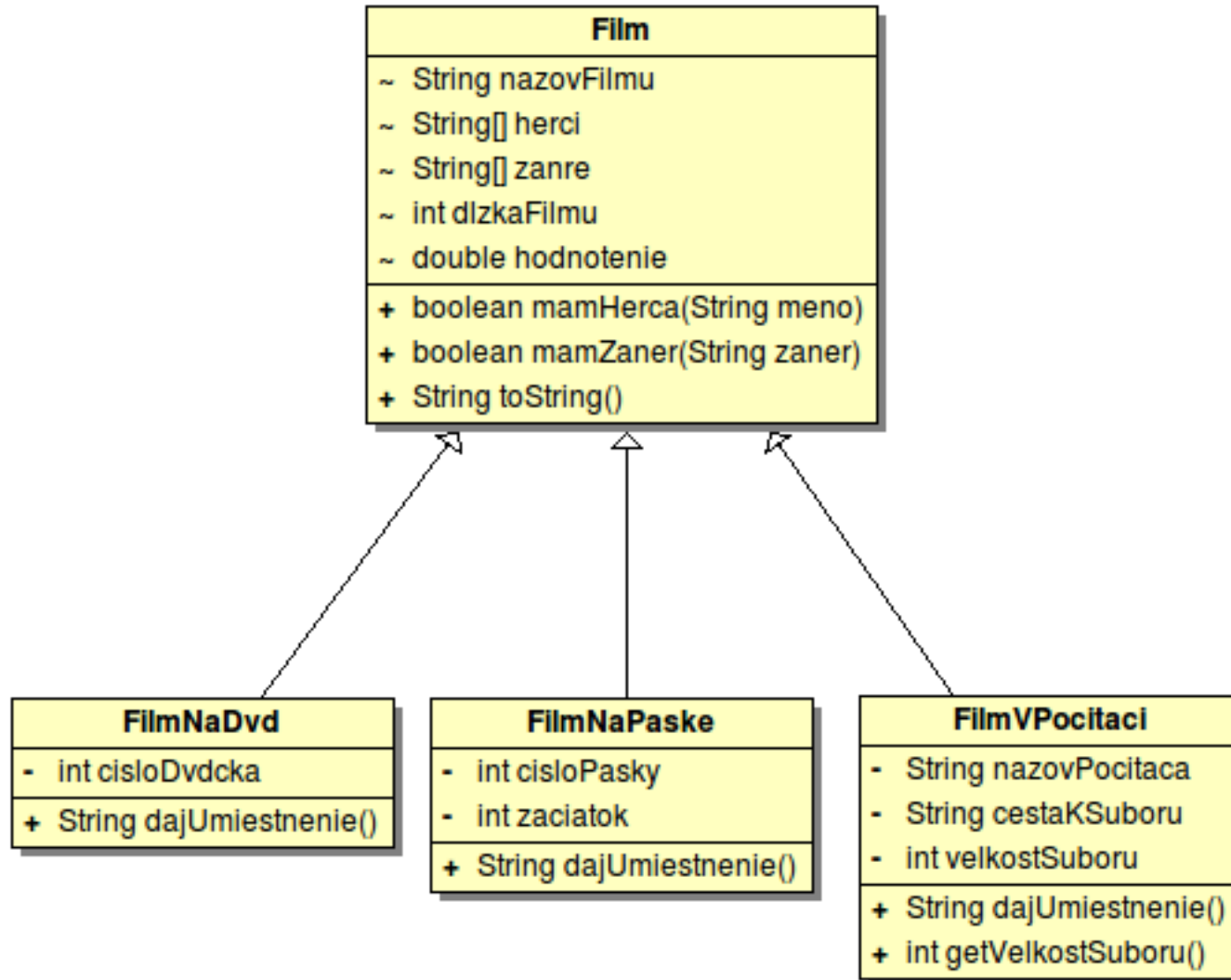


Dedičnosť = rozširovanie

- Každá oddedená trieda vie povedať, kde sa jej film nachádza
 - `String dajUmiestnenie()`
- Každá 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





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

- Lepšie riešenie:
 - `FilmNaDvd` **je** `Film`
 - `FilmNaPaske` **je** `Film`
 - `FilmVPocitaci` **je** `Film`
- Do premennej predka môžeme strkať potomkov
 - `Film f = new FilmNaDvd();`
 - Od premennej `f` môžeme požadovať všetko to, čo je definované v triede `Film` (a jej predkoch)
 - Ale aj napr. `FilmNaDvd` vie všetko to, čo vie `Film`



Renovujeme zoznam filmov

- Opačné pravidlo neplatí !!!
- Nevieme napísať
 - `FilmNaDvd f = new Film();`
 - Lebo od premennej `f` môžeme požadovať to, čo je definované v triede `FilmNaDvd` a predkoch
 - Ale `Film` nevie `dajUmiestnenie()`



Renovujeme zoznam filmov

- Riešenie:
 - Máme síce 3 triedy, ale stačí nám 1 pole

```
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:

```
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`



Drevorubačské riešenie

- 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());
            }
            ...
        }
    }
}
```



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



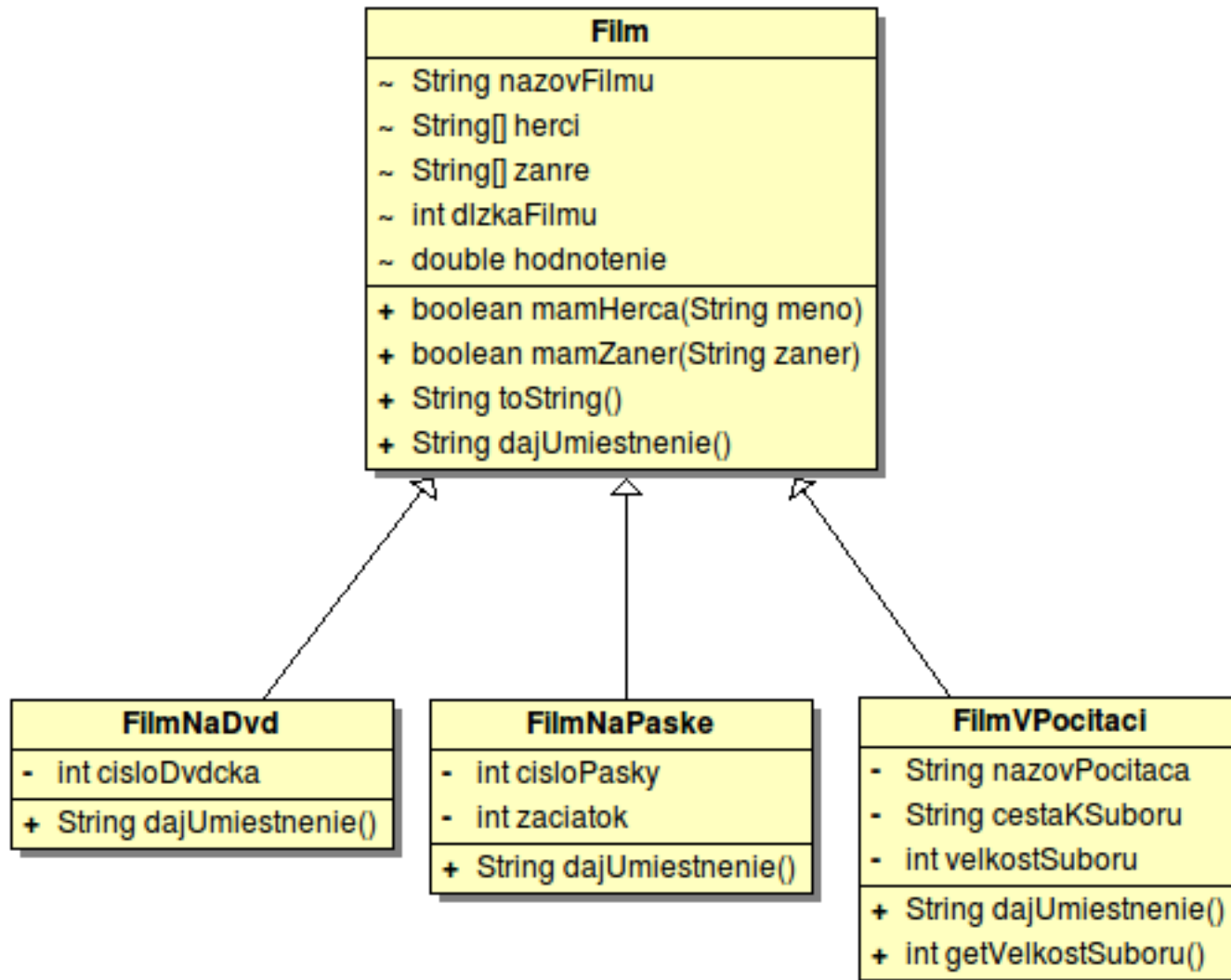
Polymorfizmus

- 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].retazecPreVypis());  
            System.out.println(filmy[i].dajUmiestnenie());  
        }  
    ...  
}
```



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

objekt triedy Film:

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



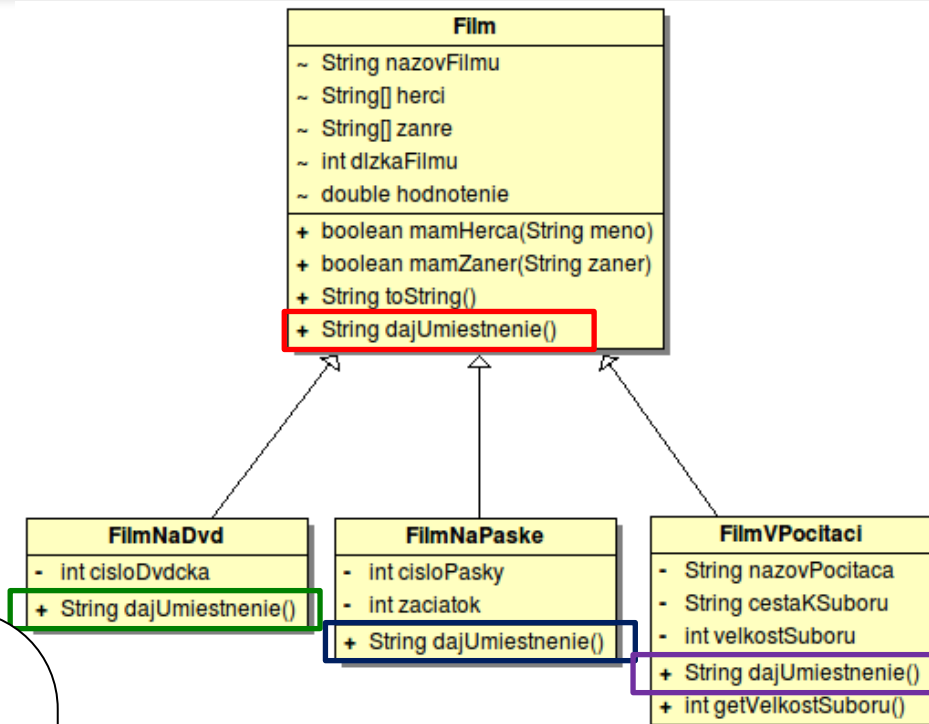
volá

objekt triedy FilmNaDvd:

```
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





Ukladáme zoznam filmov

- Chceme realizovať ukladanie a nahrávanie zo súboru zavolaním metódy v triede `ZoznamFilmov`
- Každé médium sa uloží inak
- Každé médium sa nahrá inak
- **Úloha pre polymorfizmus!**



Ďakujem za pozornosť !

