

Dedičnosť

Pojmy zavedené v 5. prednáške₍₁₎

- interface
 - implementácia viacerých interface v triedach
 - implements interface1, interface2, ...
 - rozšírenie jedného interface v inom interface
 - extends interface1, interface2, ...
- interface v štandardnej knižnici Java

Pojmy zavedené v 5. prednáške₍₂₎

- operátor instanceof
- pretypovanie
 - implicitné
 - explicitné
 - bezpečné explicitné pretypovanie

Pojmy zavedené v 5. prednáške₍₃₎

- menný priestor
- balíčky
 - nepomenovaný balíček
 - príkaz package
 - konvencie Java
 - javadoc – package-info.java
 - UML

Ciel' prednášky

- základy dedičnosti
- príklad: KCalB 

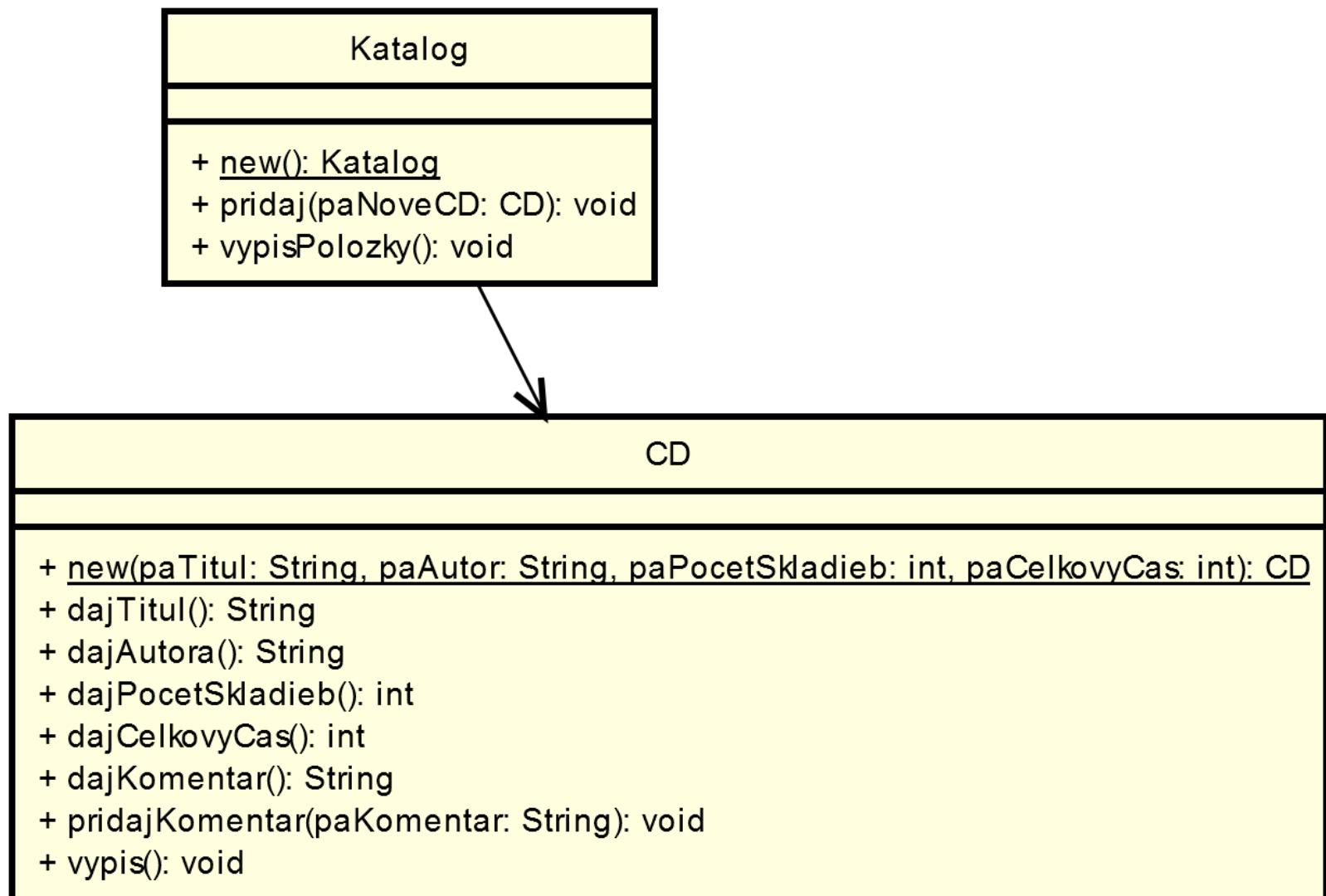
„KCalB“ – úloha 1

- jednoduchý katalóg CD
 - pridávanie CD
 - výpis zoznamu CD
 - používateľské komentáre
- informácie o CD
 - titul CD (názov albumu)
 - autor (spevák, skupina)
 - počet skladieb na CD
 - celkový čas v minútach
 - používateľov komentár

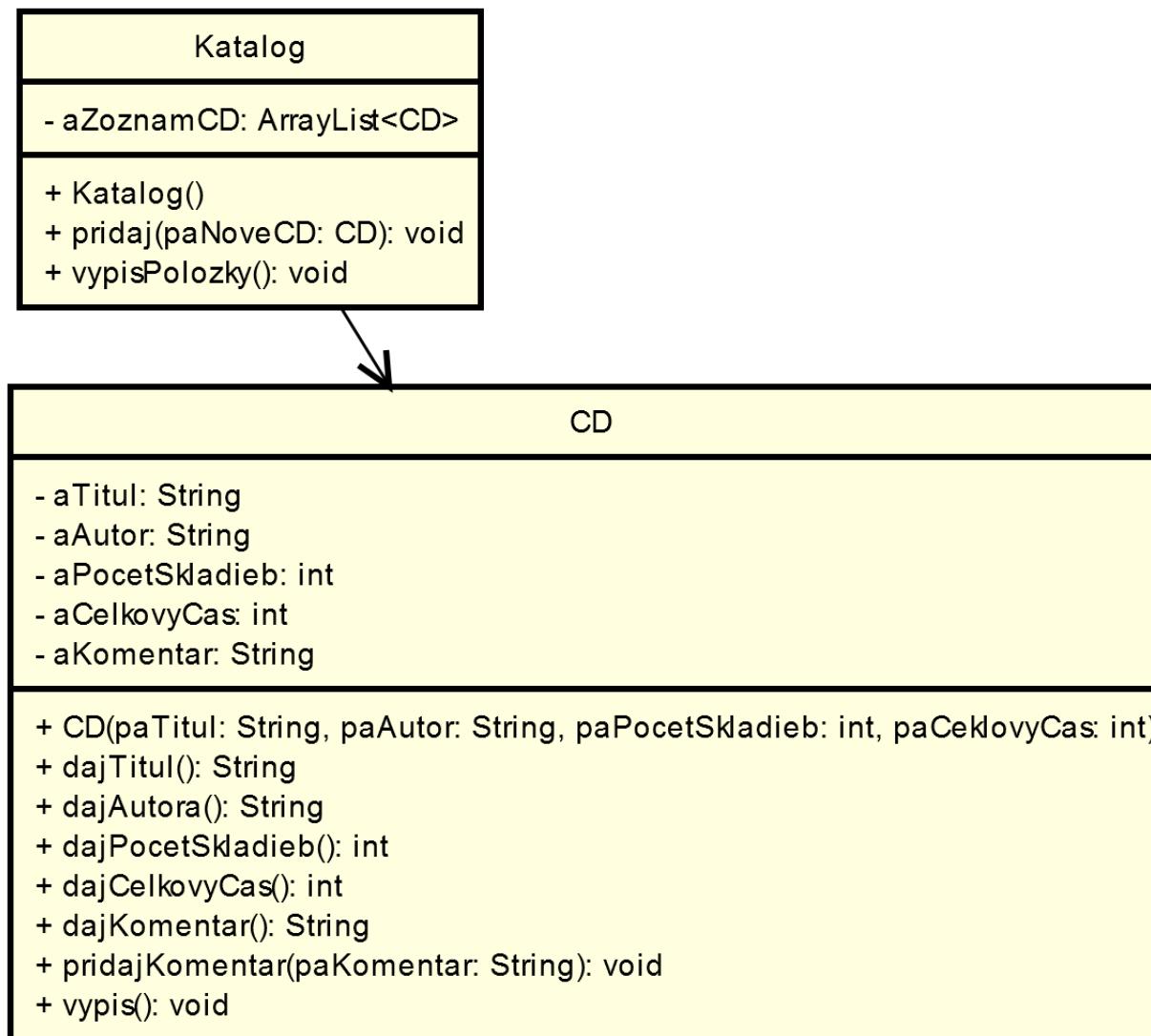
Riešenie

- dve triedy
 - Katalog
 - CD
- ukladanie zoznamu CD do kontajnera ArrayList
- CD
 - vie vrátiť informácie o sebe
 - vie pridať riadok poznámky
 - vie sa vypísať na terminál

Prvá verzia KCalB – vonkajší pohľad



Prvá verzia KCalB – vnútorný pohľad



„KCaIB“ – úloha 2₍₁₎

- jednoduchý katalóg audiovizuálnych diel
 - pridávanie diela
- výpis zoznamu diela
- užívateľské komentáre
- informácie o CD
- informácie o DVD

„KCalB“ – úloha 2₍₂₎

- informácie o CD
 - titul CD (názov albumu)
 - autor (spevák, skupina)
 - počet skladieb na CD
 - celkový čas v minútach
 - používateľov komentár

„KCalB“ – úloha 2₍₃₎

- informácie o DVD
 - titul DVD (názov filmu)
 - režisér
 - celkový čas v minutách
 - používateľov komentár

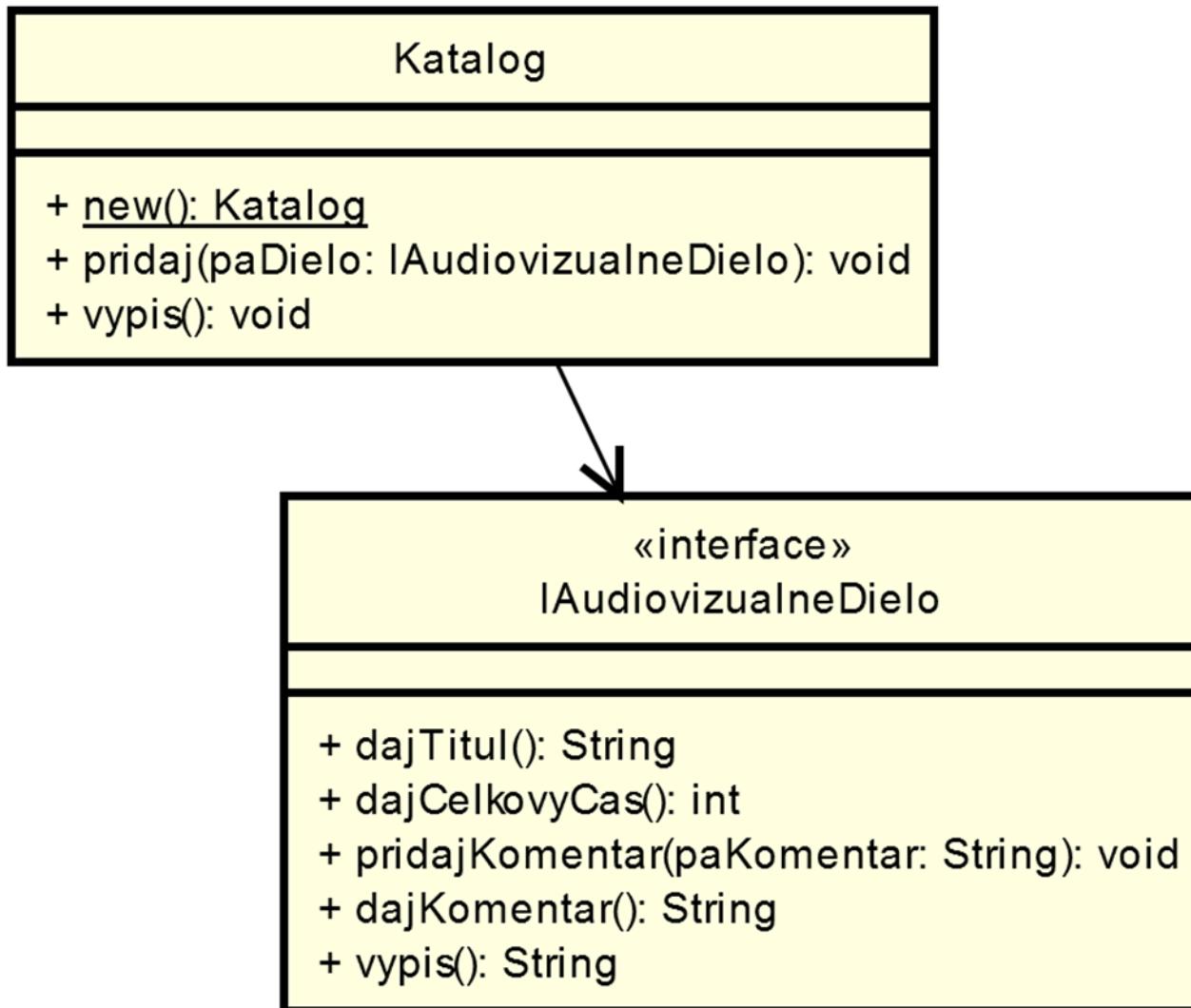
Ako na to

- trieda pre DVD
- trieda pre CD
- možnosti riešenia
 - dva kontajnery ArrayList
 - polymorfizmus

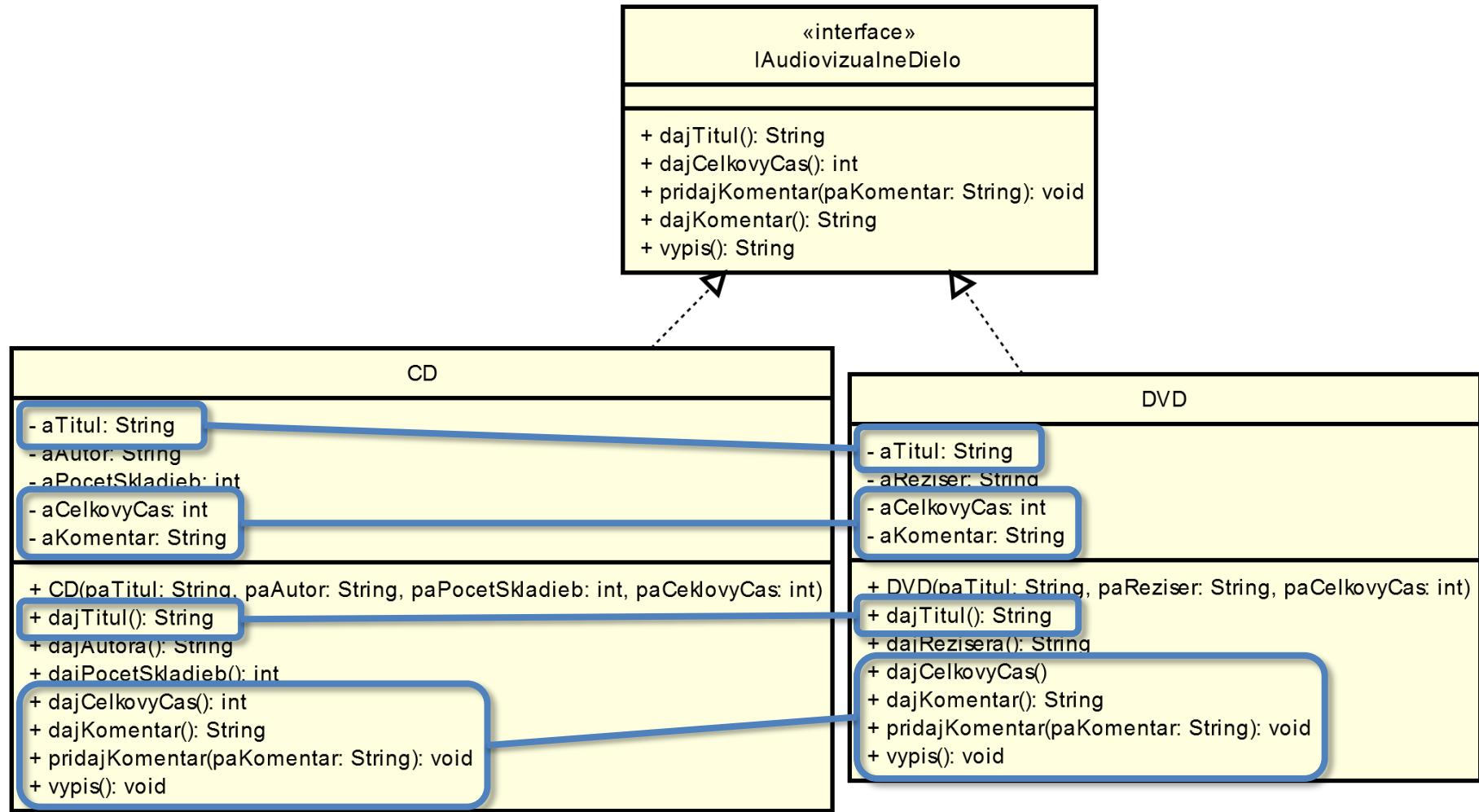
Riešenie pomocou polymorfizmu

- interface so spoločnými správami
 - dajTitul
 - dajCelkovyCas
 - pridajKomentar
 - dajKomentar
 - vypis

Zavedenie interface₍₁₎



Zavedenie interface₍₂₎



Trieda CD – definícia

```
public class CD implements IAudiovizualneDielo
{
    private String aTitul;
    private String aAutor;
    private int aPocetSkladieb;
    private int aCelkovyCas;
    private String aKomentar;

    ...
}
```

Trieda DVD – definícia

```
public class DVD implements IAudiovizualneDielo  
{  
    private String aTitul;  
    private String aReziser;  
    private int aCelkovyCas;  
    private String aKomentar;  
    ...  
}
```

Trieda CD – konštruktor

```
public CD(String paTitul, String paAutor,  
          int paPocetSkladieb, int paCelkovyCas)  
{  
    aTitul = paTitul;  
    aAutor = paAutor;  
    aPocetSkladieb = paPocetSkladieb;  
    aCelkovyCas = paCelkovyCas;  
    aKomentar = null;  
}
```

Trieda DVD – konštruktor

```
public DVD(String paTitul, String paReziser,  
          int paCelkovyCas)  
{  
    aTitul = paTitul;  
    aReziser = paReziser;  
    aCelkovyCas = paCelkovyCas;  
    aKomentar = null;  
}
```

Trieda CD – prístupové metódy₍₁₎

```
public String dajTitul()
```

```
{
```

```
    return aTitul;
```

```
}
```

```
public String dajAutora()
```

```
{
```

```
    return aAutor;
```

```
}
```

Trieda CD – prístupové metódy₍₂₎

```
public int dajPocetSkladieb()
```

```
{
```

```
    return aPocetSkladieb;
```

```
}
```

```
public int dajCelkovyCas()
```

```
{
```

```
    return aCelkovyCas;
```

```
}
```

Trieda DVD – prístupové metódy

```
public String dajTitul()
```

```
{
```

```
    return aTitul;
```

```
}
```

```
public String dajRezisera()
```

```
{
```

```
    return aReziser;
```

```
}
```

```
public int dajCelkovyCas()
```

```
{
```

```
    return aCelkovyCas;
```

```
}
```

Trieda CD – práca s komentármí

```
public String dajKomentar()
{
    return aKomentar;
}
public void pridajKomentar(String paKomentar)
{
    if (aKomentar == null) {
        aKomentar = paKomentar;
    } else {
        aKomentar = aKomentar + "\n" + paKomentar;
    }
}
```

Trieda DVD – práca s komentármami

```
public String dajKomentar()
{
    return aKomentar;
}
public void pridajKomentar(String paKomentar)
{
    if (aKomentar == null) {
        aKomentar = paKomentar;
    } else {
        aKomentar = aKomentar + "\n" + paKomentar;
    }
}
```

Trieda CD – výpis

```
public void vypis()
{
    System.out.println("CD:");
    System.out.println("- Skladieb: " + aPocetSkladieb);
    System.out.println("- Autor: " + aAutor);
    System.out.println("- Titul: " + aTitul);
    System.out.println("- Cas: " + aCelkovyCas);
    if (aKomentar != null) {
        System.out.println("Komentar ku CD:");
        System.out.println(aKomentar);
    }
}
```

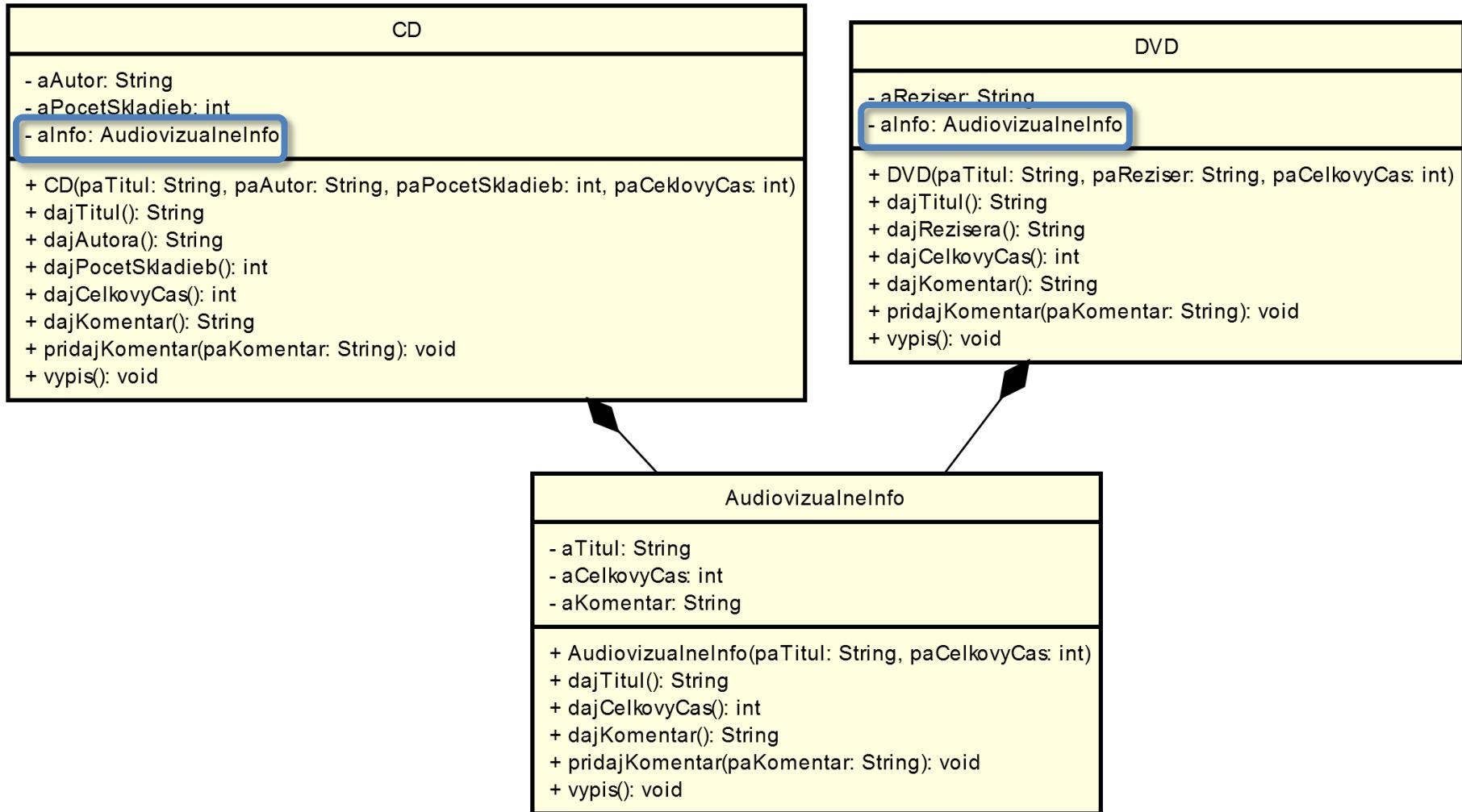
Trieda DVD – výpis

```
public void vypis()
{
    System.out.println("DVD:");
    System.out.println("- Reziser: " + aReziser);
    System.out.println("- Titul: " + aTitul);
    System.out.println("- Cas " + aCelkovyCas);
    if (aKomentar != null) {
        System.out.println("Komentar ku DVD:");
        System.out.println(aKomentar);
    }
}
```

Duplicita kódu

- triedy CD a DVD zdieľajú veľkú časť kódu
- duplicita kódu
 - sťažuje údržbu kódu
 - potencionálne miesto pre vznik chýb
- riešenie
 - samostatná trieda so spoločnými časťami, kompozícia

Riešenie pomocou kompozície₍₁₎



Riešenie pomocou kompozície₍₂₎

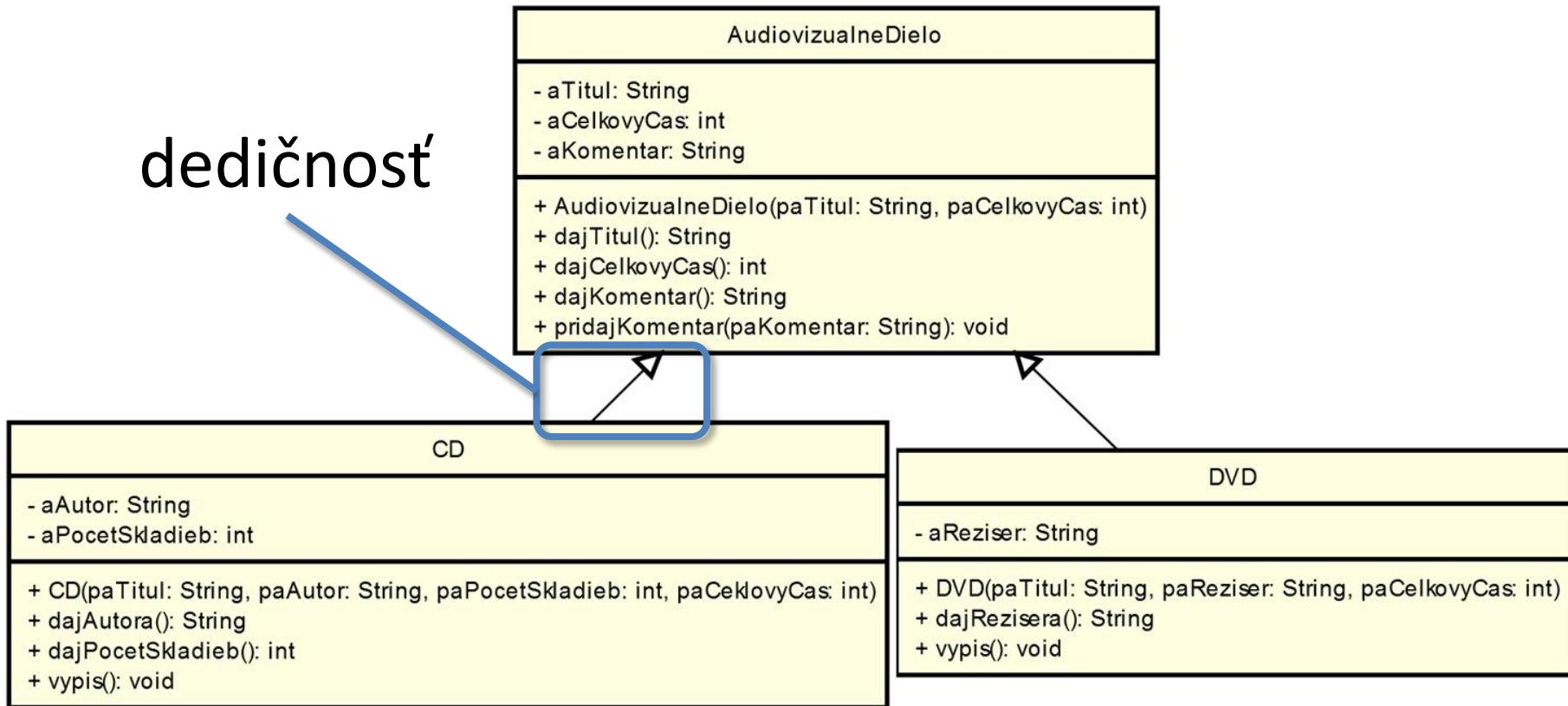
- CD aj DVD si ako atribút v konštruktore vytvorí inštanciu AudiovizualneInfo
- vykonávaním spoločnej funkcionality poveria túto inštanciu

```
public void pridajKomentar(String paKomentar)  
{  
    alno.pridajKomentar(paKomentar);  
}
```

- iné riešenie: dedičnosť

Riešenie pomocou dedičnosti

dedičnosť



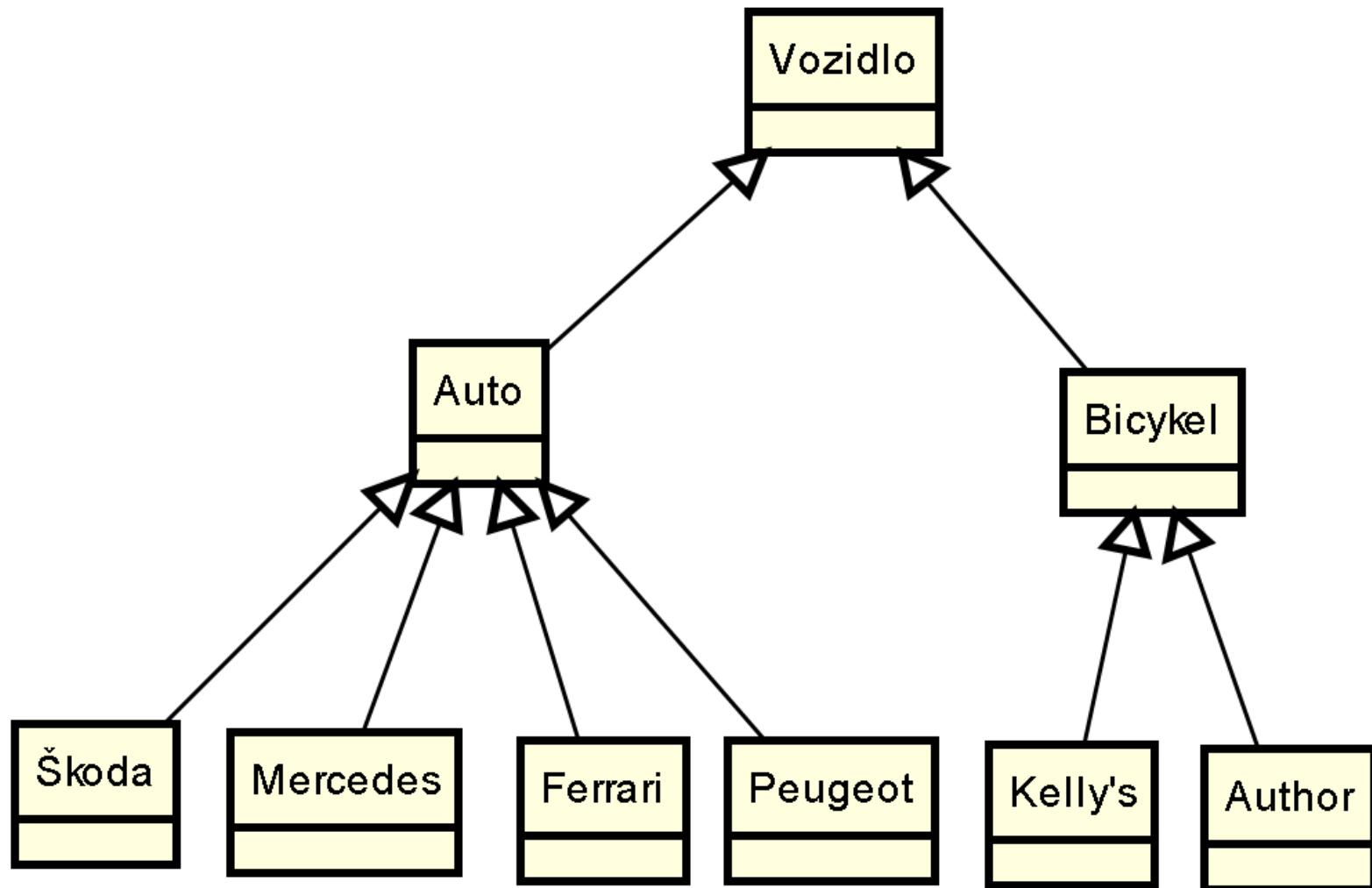
Dedičnosť

- vzťah medzi triedami
- trieda ako typ objektu
- hierarchia dedičnosti – vyjadruje typológiu objektov
 - delenie objektov na typy a podtypy
 - ľubovoľná úroveň

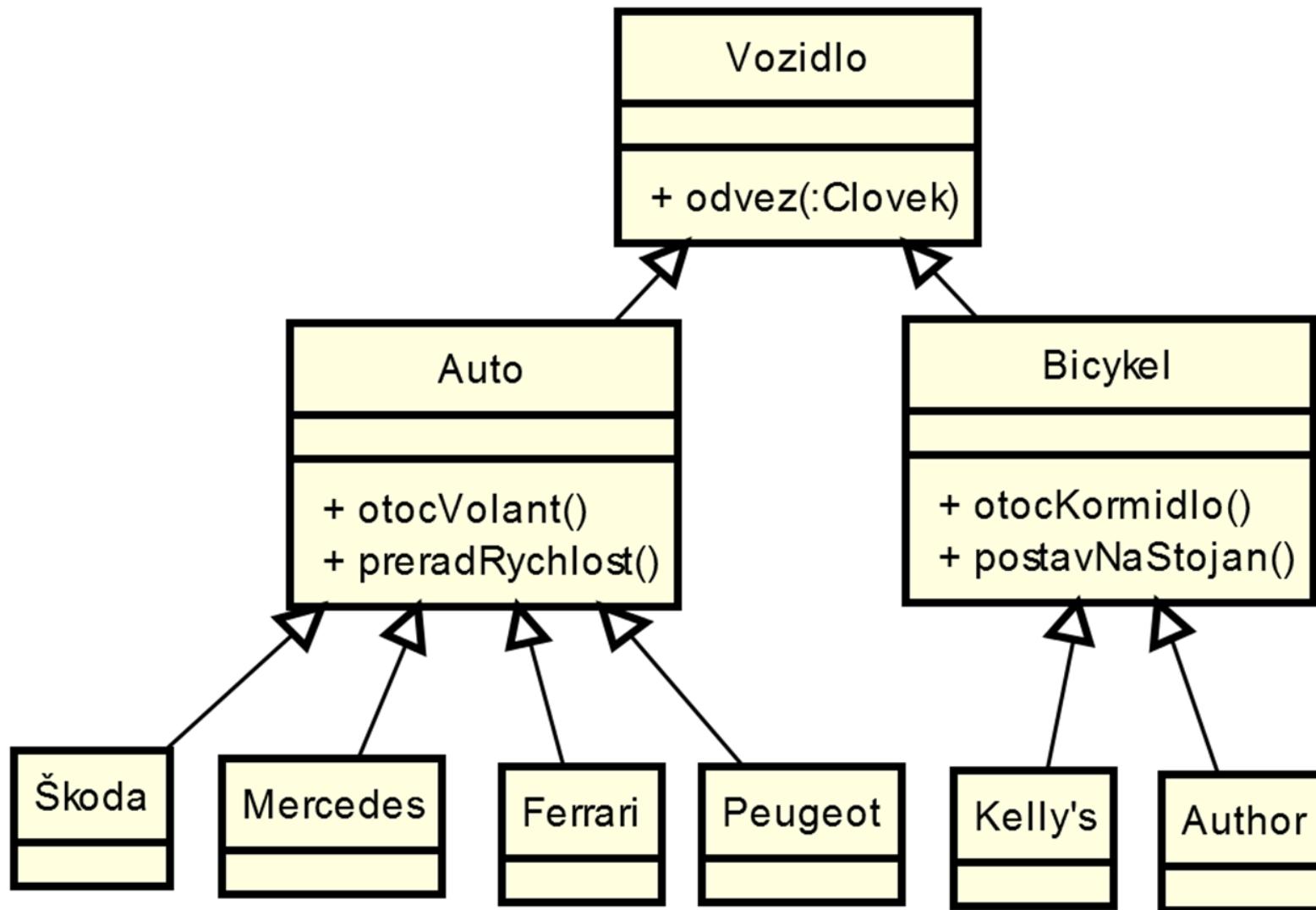
Dedičnosť v reálnom svete₍₁₎

- vozidlo
 - bicykel
 - Kelly's
 - Author
 - auto
 - Škoda
 - Mercedes
 - Ferrari
 - Peugeot

Dedičnosť v reálnom svete₍₂₎



Dedičnosť v reálnom svete₍₃₎



Základné pojmy₍₁₎

- potomok – podtyp – odvodená trieda
 - trieda Auto je potomok triedy Vozidlo
 - trieda Author je potomok triedy Vozidlo
- predok – nadtyp – základná trieda
 - trieda Bicykel je predok triedy Author
 - trieda Vozidlo je predok triedy Author

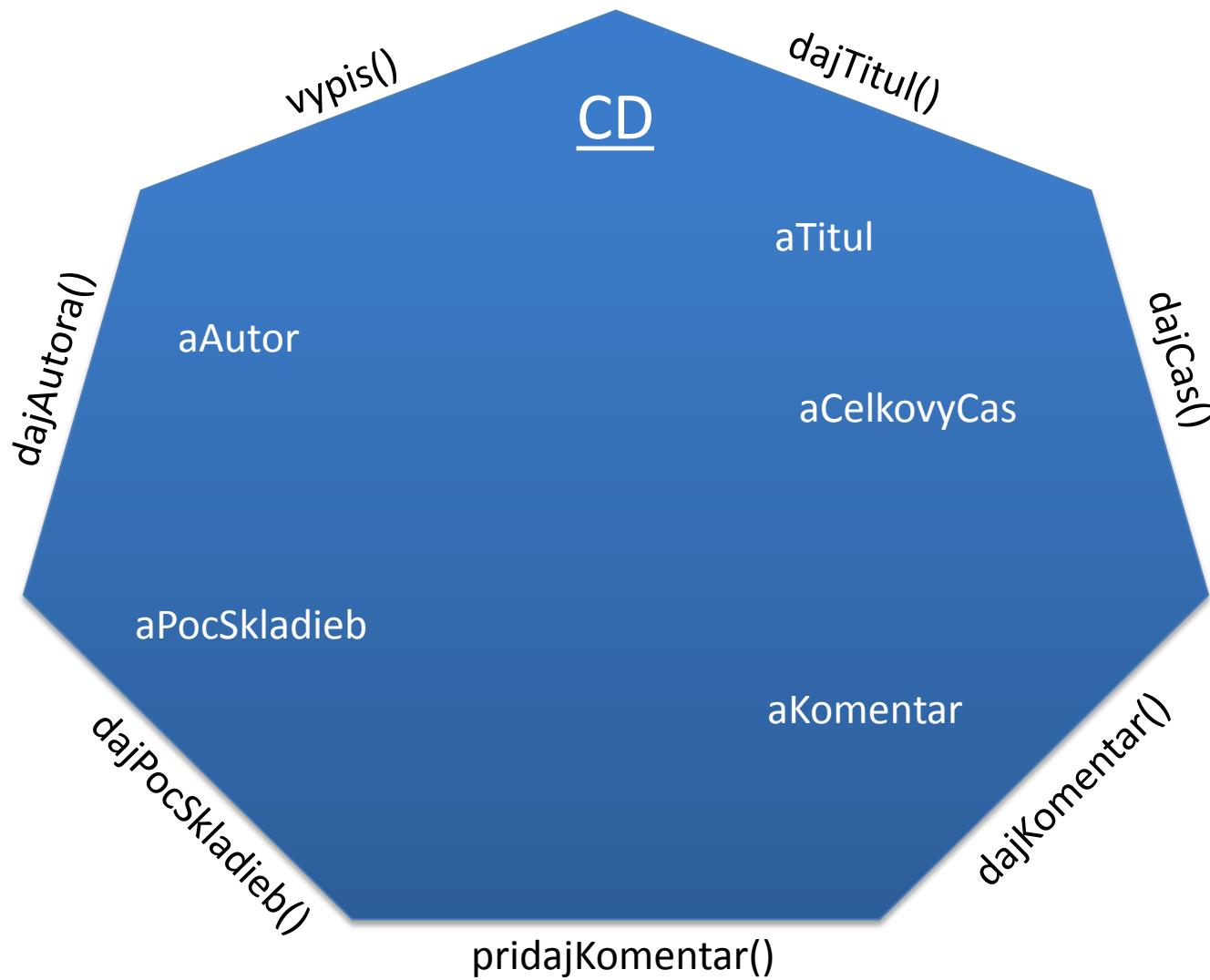
Základné pojmy₍₂₎

- priamy potomok
 - trieda Auto je priamy potomok triedy Vozidlo
- priamy predok
 - trieda Bicykel je priamy predok triedy Author
- absolútny predok – „koreň“ stromu hierarchie
 - trieda Vozidlo

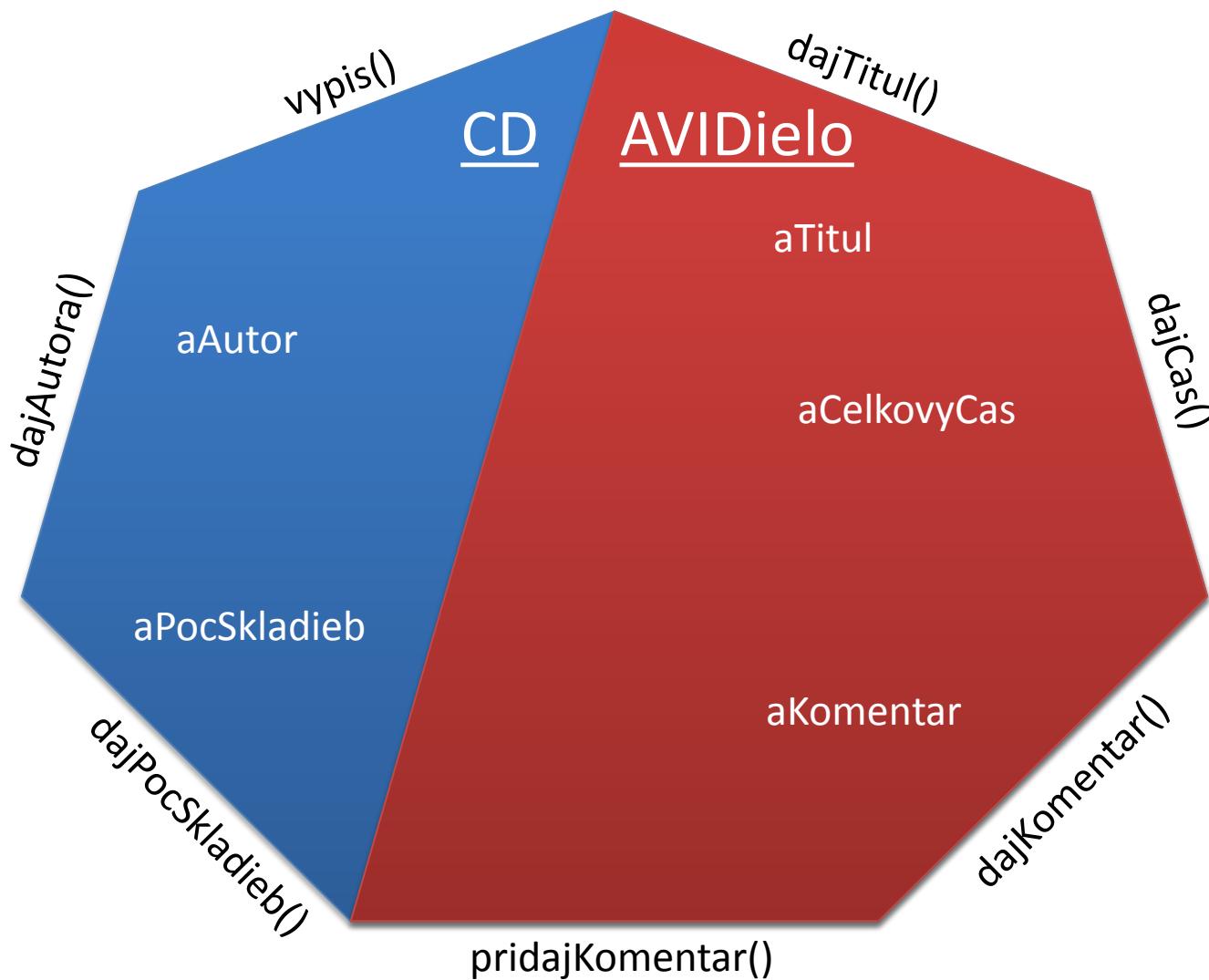
Potomok dedí od predka

- vonkajší pohľad
 - verejné rozhranie
- vnútorný pohľad
 - atribúty
 - metódy
- potomok nemá priamy prístup k neverejným zložkám predka – interné ukrývanie informácií

Vonkajší a vnútorný pohľad



Vonkajší a vnútorný pohľad



Dedičnosť v jazyku Java₍₁₎

public class CD extends AudiovizualneDielo

- alebo

**public class CD extends AudiovizualneDielo
implements IAudiovizualneDielo**

- v odvodenej triede
- len jeden predok

Dedičnosť v jazyku Java₍₂₎

- ! Potomok nededí konštruktory predka
- konštruktory treba znova definovať
 - prvý príkaz v tele konštruktora – kľúčové slovo super

Dedičnosť v jazyku Java₍₃₎

- kľúčové slovo super
 - použije konštruktor predka
 - inicializácia začiatočného stavu predka

```
super(zoznamParametrov);
```

- zoznamParametrov – zoznam skutočných parametrov, ktoré dostane konštruktor predka

KCalB – Trieda AudiovizualneDielo

- spoločný predok CD, DVD
- atribúty
 - aTitul
 - aCelkovyCas
 - aKomentar
- metódy
 - dajTitul
 - dajCelkovyCas
 - dajKomentar
 - pridajKomentar

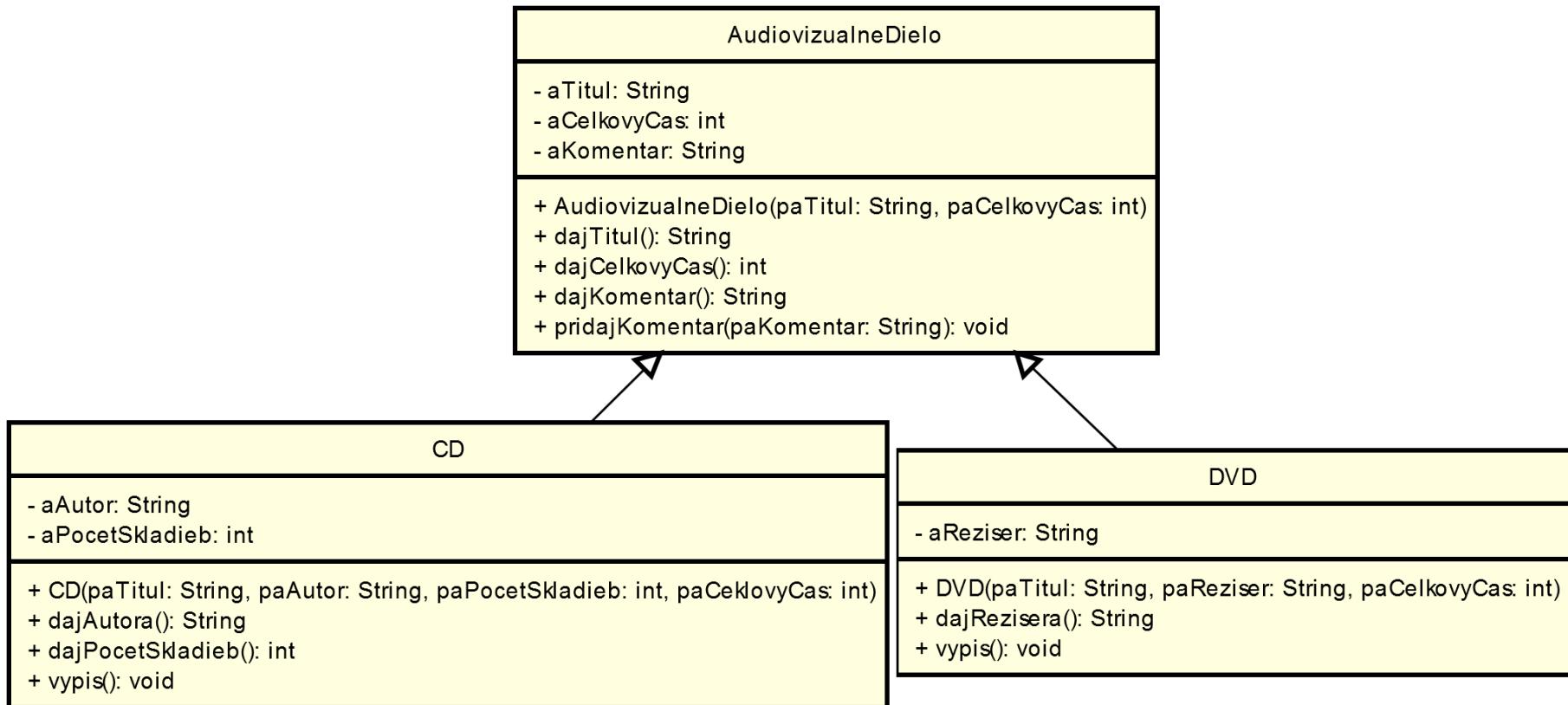
KCalB – Trieda CD

- potomok triedy `AudiovizualneDielo`
- atribúty
 - `aAutor`
 - `aPocetSkladieb`
- metódy
 - `dajAutora`
 - `dajPocetSkladieb`
 - `vypis`

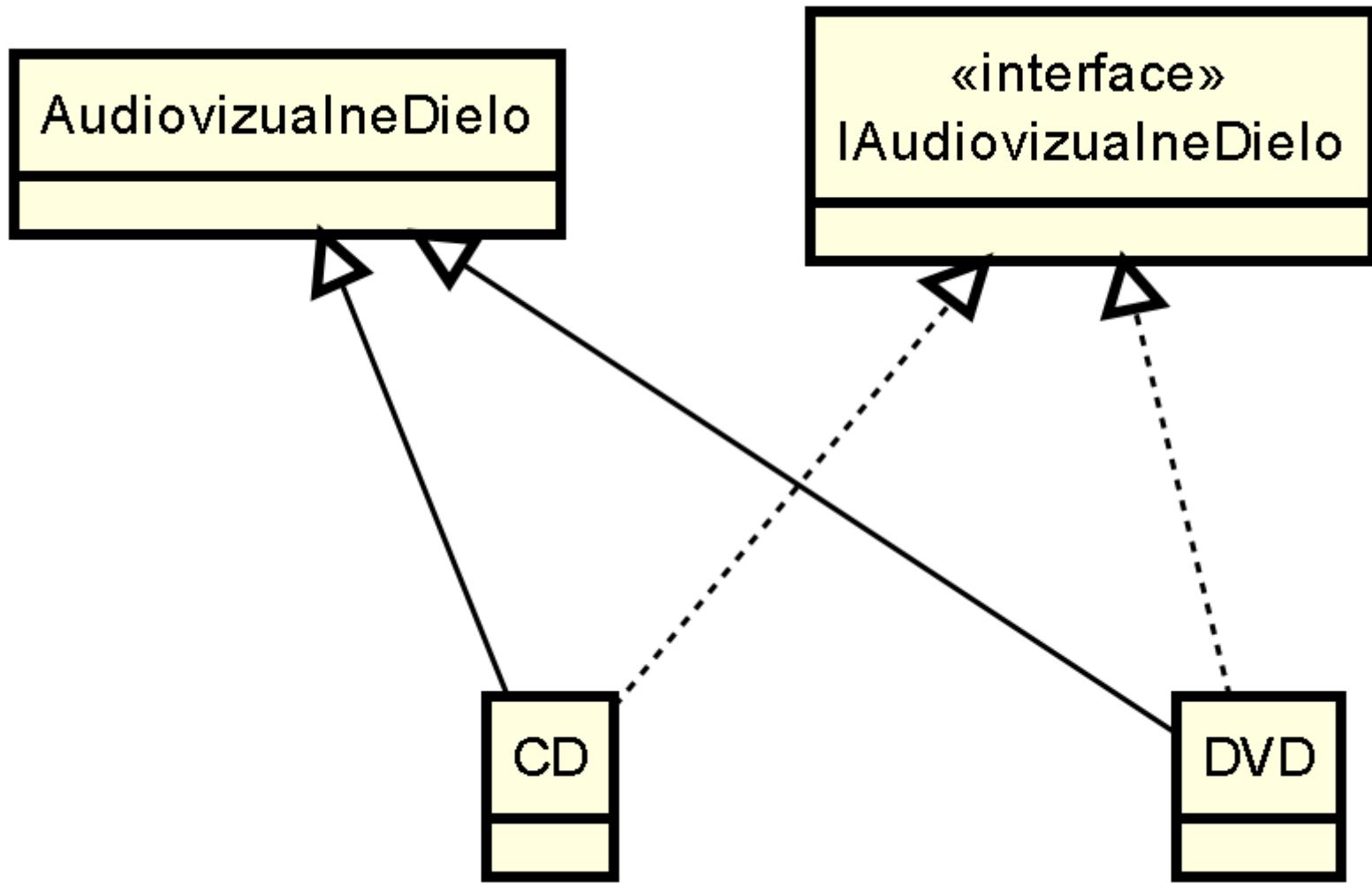
KCalB – Trieda DVD

- potomok triedy `AudiovizualneDielo`
- atribúty
 - `aReziser`
- metódy
 - `dajRezisera`
 - `vypis`

KCalB v UML₍₁₎



KCalB v UML₍₂₎



Trieda AudiovizualneDielo

```
public class AudiovizualneDielo  
{  
    private String aTitul;  
    private int aCelkovyCas; // v minutach  
    private String aKomentar;  
    ...  
}
```

Trieda AudiovizualneDielo – konštruktor

```
public AudiovizualneDielo(String paTitul,  
                           int paCelkovyCas)  
{  
    aTitul = paTitul;  
    aCelkovyCas = paCelkovyCas;  
    aKomentar = null;  
}
```

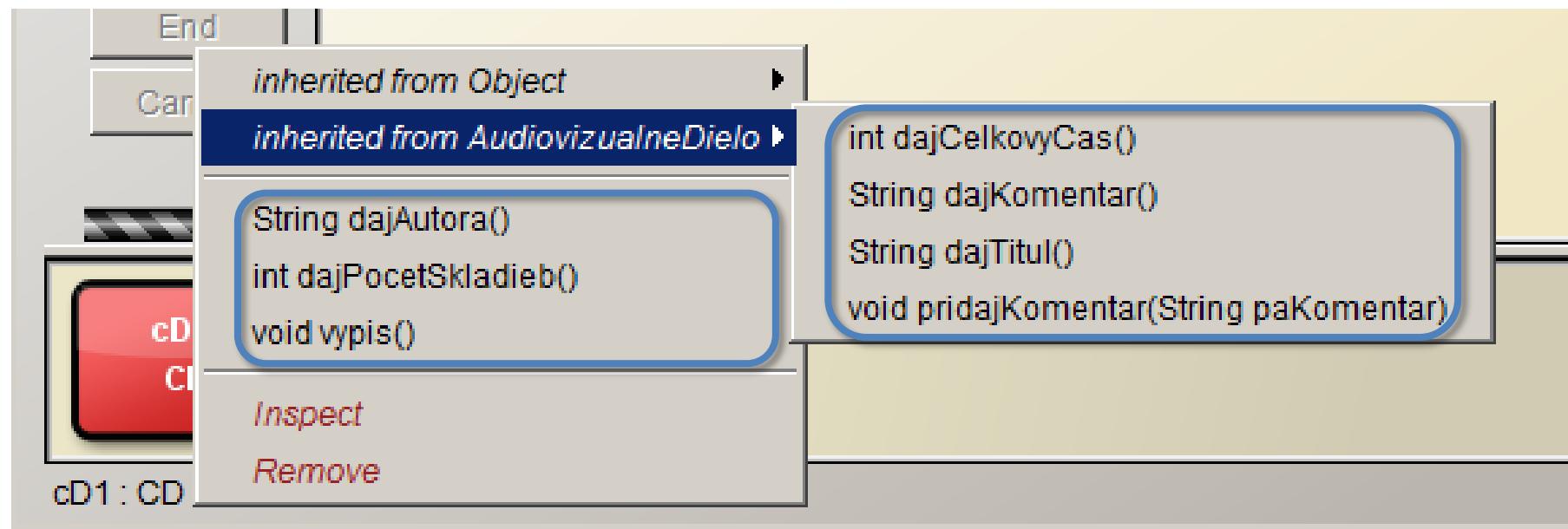
Trieda DVD

```
public class DVD extends AudiovizualneDielo  
    implements IAudiovizualneDielo  
{  
    private String aReziser;  
    ...  
}
```

Trieda DVD – konštruktor

```
public DVD(String paTitul, String paReziser,  
          int paCelkovyCas)  
{  
    super(paTitul, paCelkovyCas);  
  
    aReziser = paReziser;  
}
```

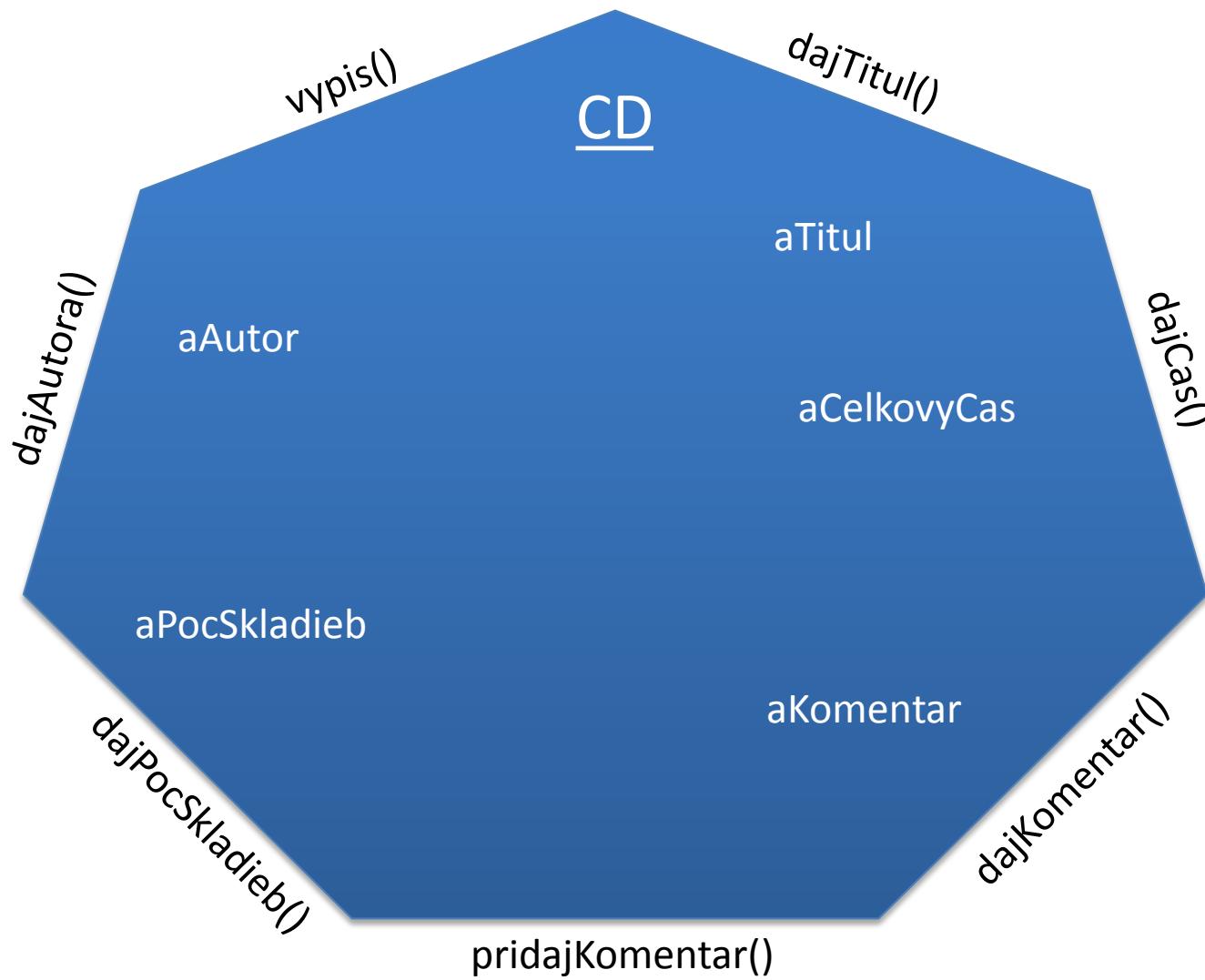
Rozhranie v nástroji BlueJ



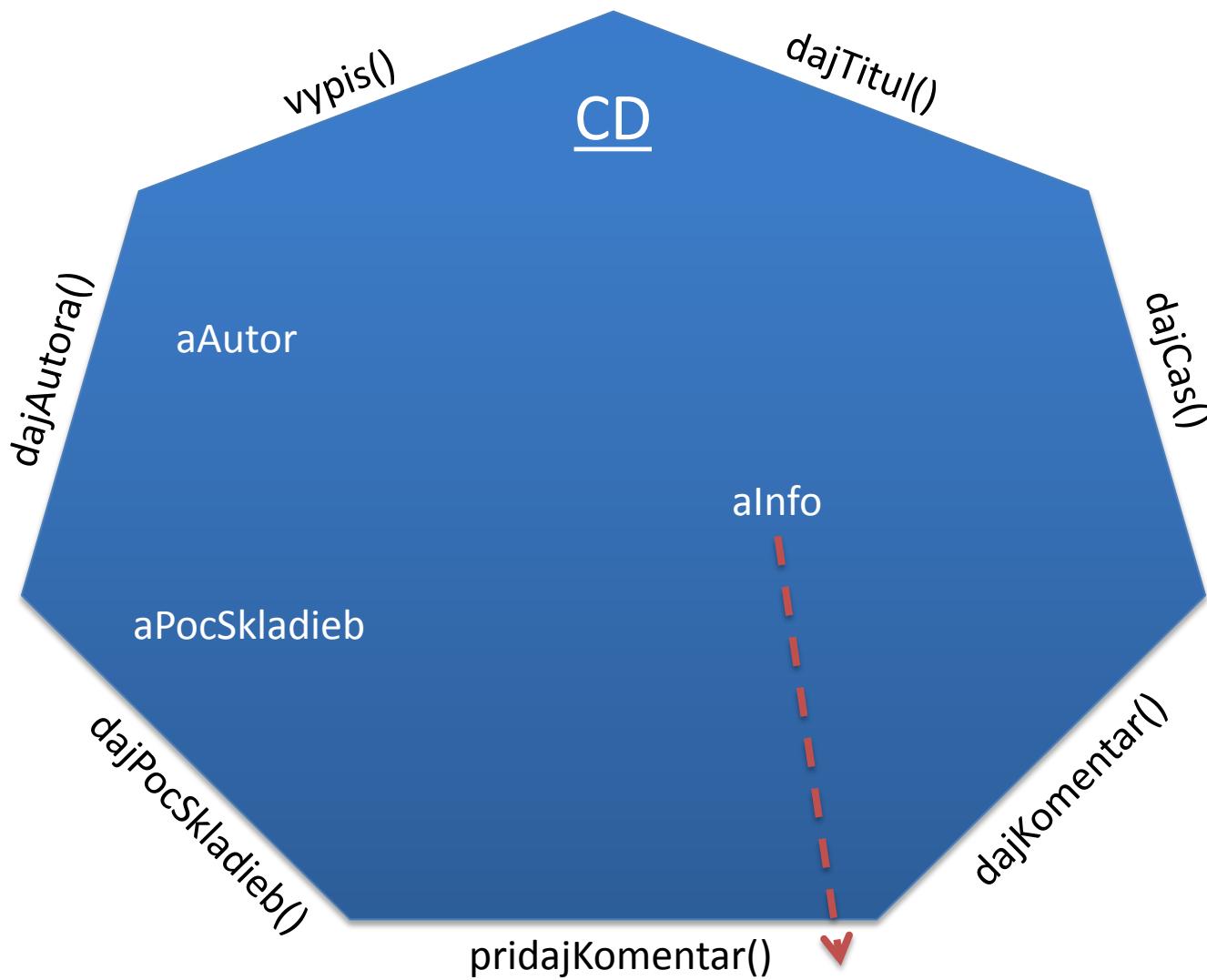
KCalB – tri rôzne spôsoby riešenia

- nezávislé triedy
- skladanie – časť a celok
- dedičnosť

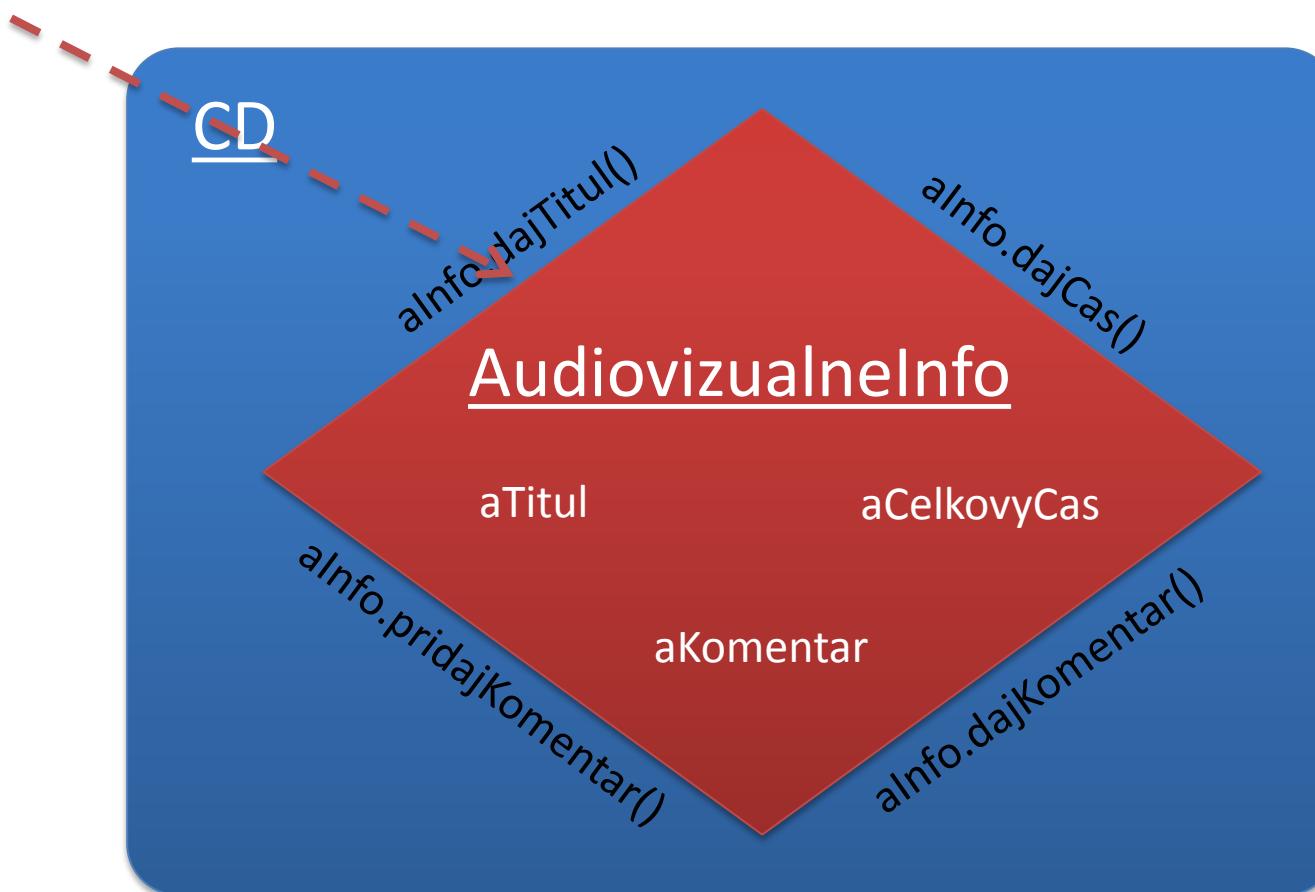
CD: nezávislá trieda – monolit



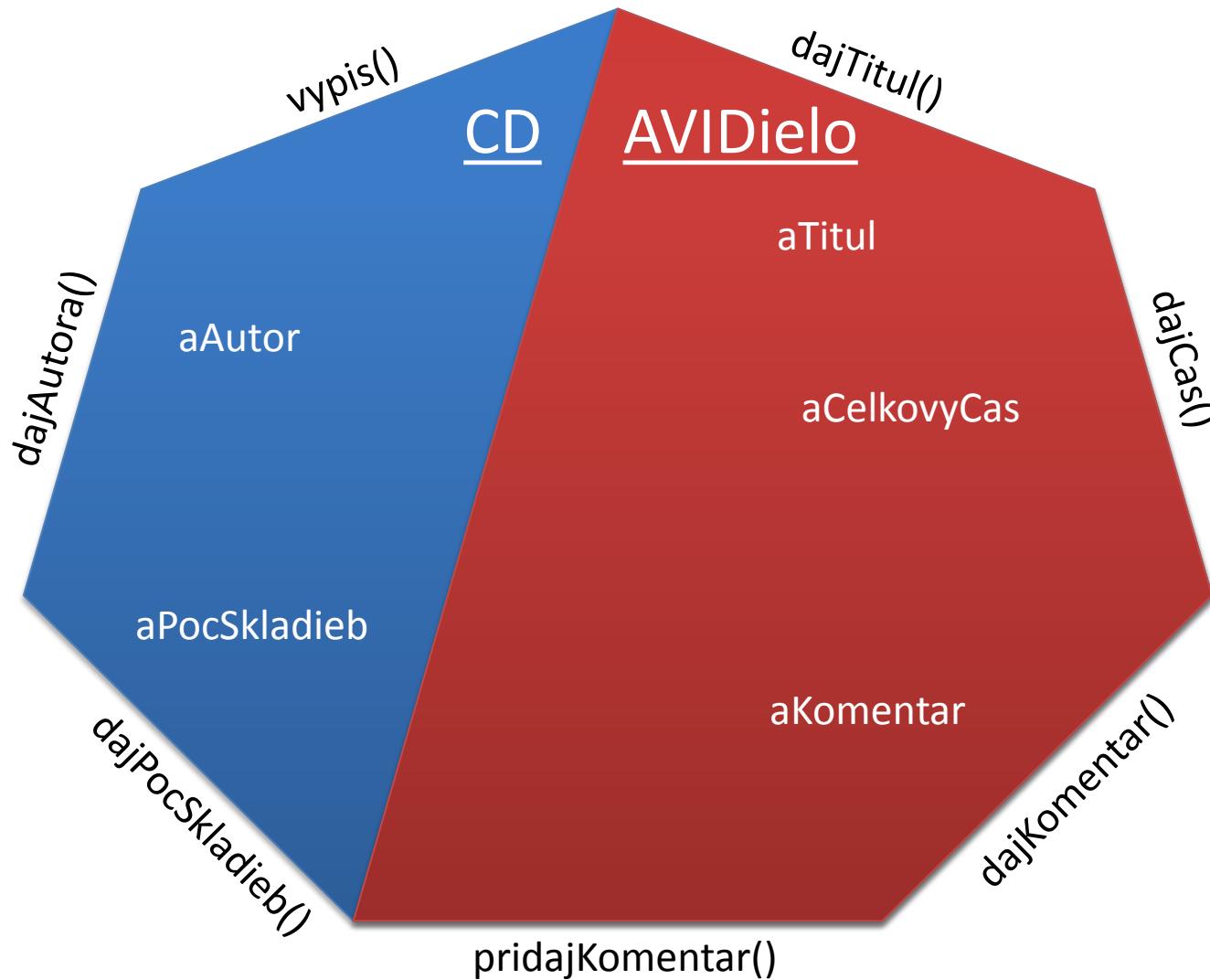
CD: závislé triedy – skladanie₍₁₎



CD: závislé triedy – skladanie₍₂₎



CD: závislé triedy – dedičnosť



Znovapoužiteľnosť (reuse)

- nezávislé triedy
 - znovapoužiteľnosť na úrovni inštancií tried
 - každá trieda implementuje všetky svoje metódy
- skladanie
 - znovapoužiteľnosť na úrovni implementácie častí
 - objektové atribúty nevyžadujú implementáciu svojich metód
 - rozhranie definujem kompletne celé
- dedičnosť
 - znovapoužiteľnosť na úrovni rozhrania aj implementácie
 - zdedené je aj rozhranie

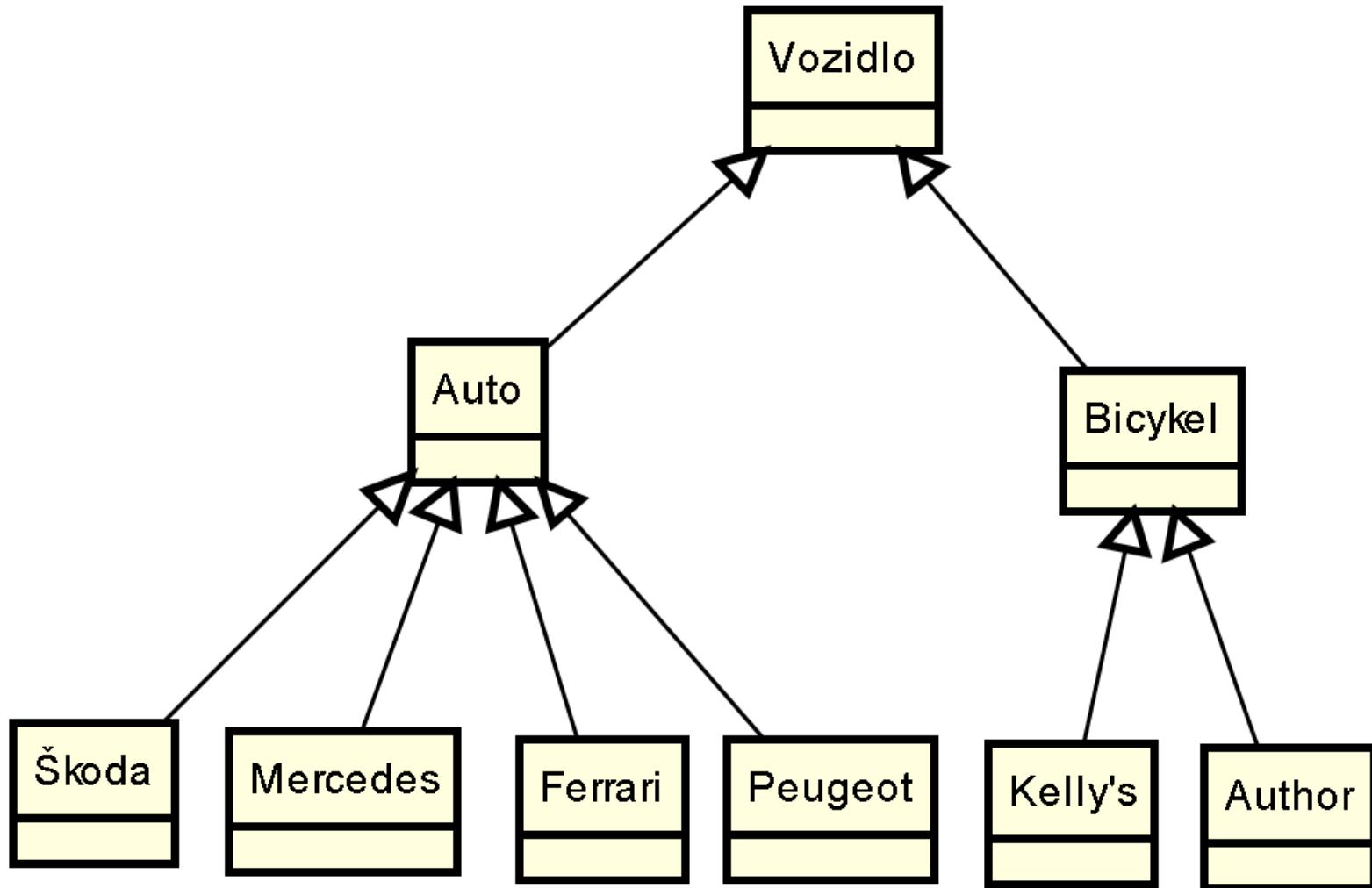
Implementačná závislosť

- nezávislé triedy – žiadna závislosť
 - ale nevedia spolupracovať
- skladanie – celok závisí od častí
 - závislosť je neverejná
 - implementácia je skrytá – dá sa meniť
- dedenie – potomok závisí od predkov
 - informácia o predkovi je verejná

Graf hierarchie dedičnosti

- digraf: orientovaný, acyklický, len jednoduché hrany
- vrcholy: triedy
- hrany: vztah dedičnosti
- orientácia hrán:
 - priamy potomok → priamy predok

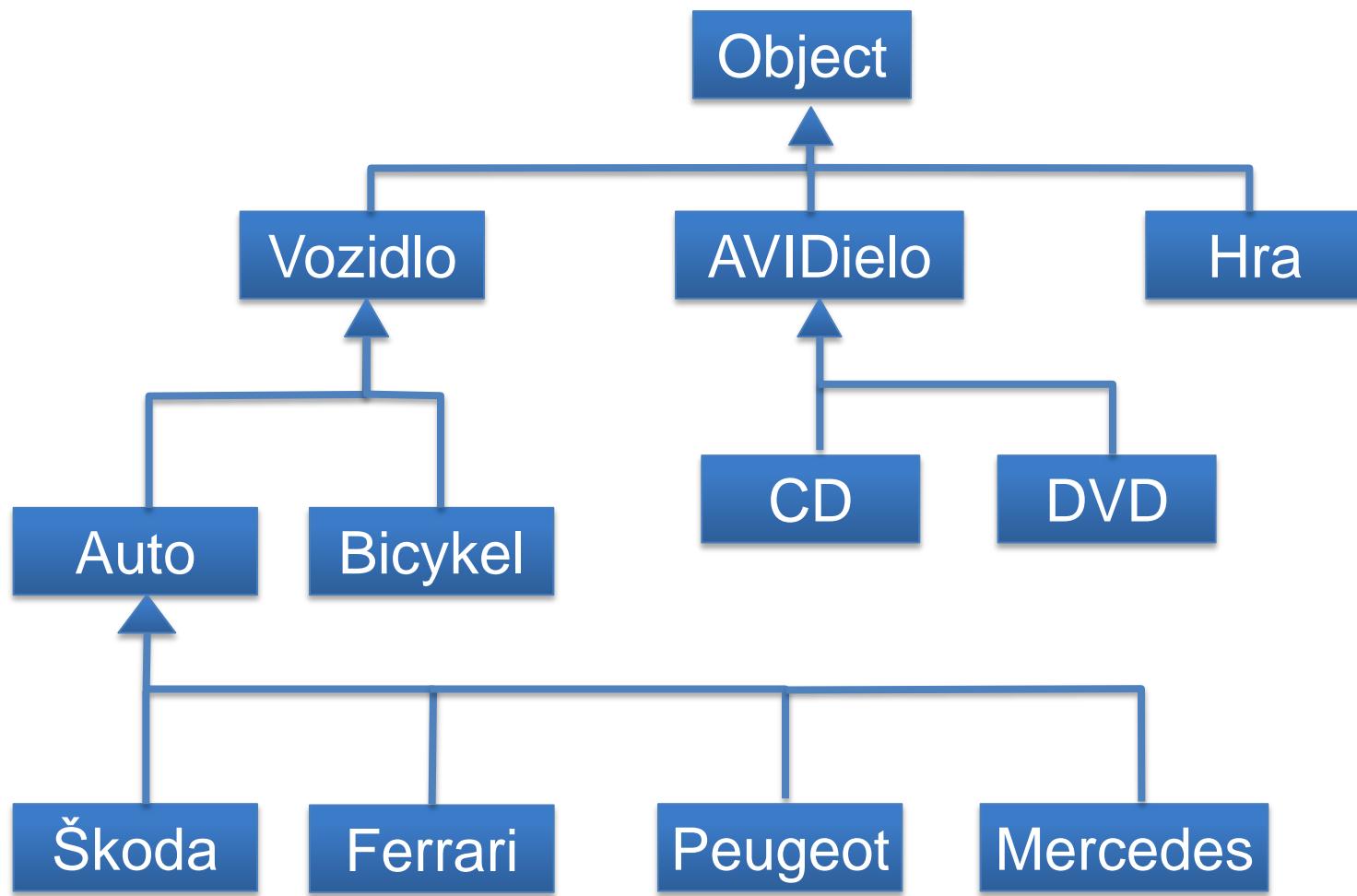
Príklad grafu hierarchie dedičnosti



Jednoduchá dedičnosť

- práve jeden predok
 - jediná hierarchia – strom
 - spoločný preddefinovaný koreň – absolútny predok
 - koreň – implicitný predok
 - Java – trieda Object
 - Delphi Pascal – trieda TObject

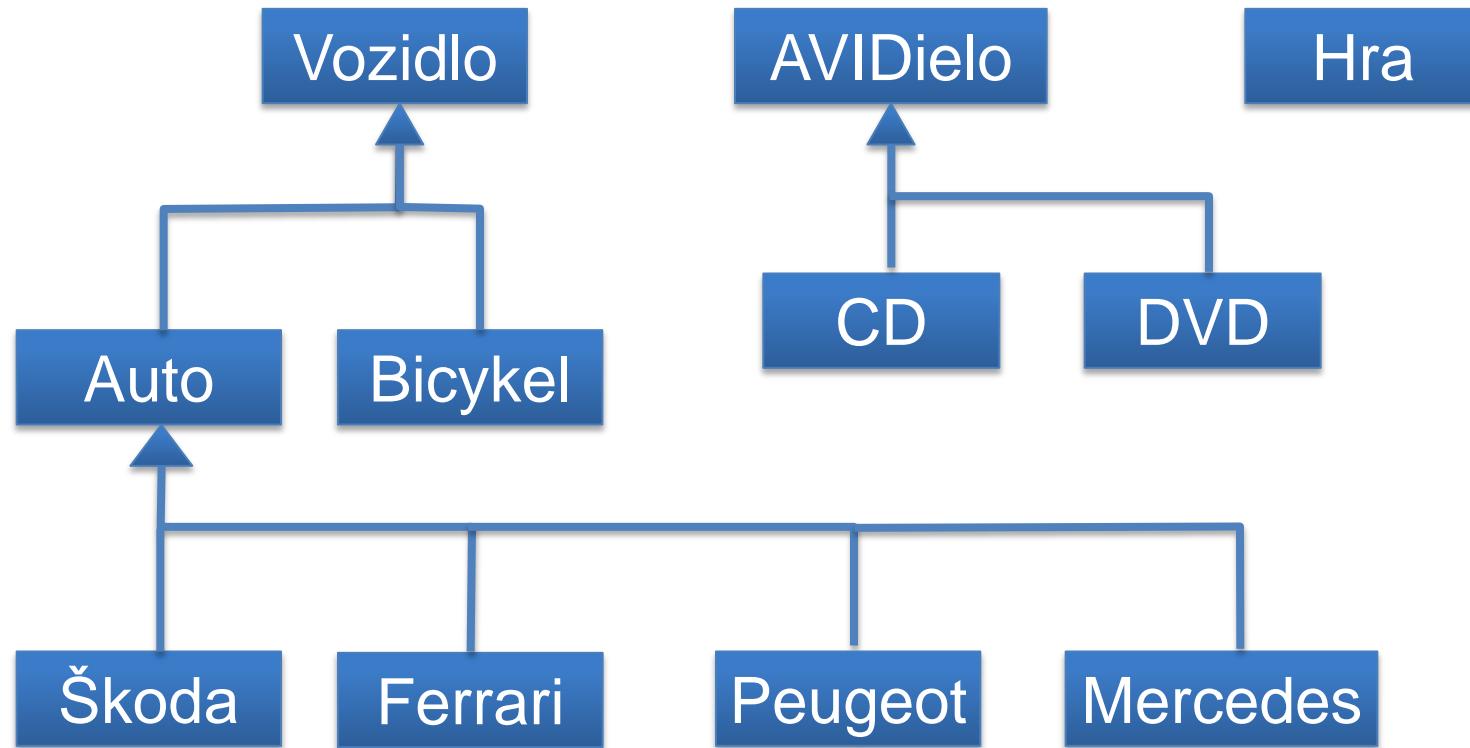
Strom dedičnosti – Java



Jednoduchá dedičnosť

- najviac jeden predok
 - ľubovoľný počet nezávislých hierarchií
 - každá hierarchia – jeden strom
 - spoločne vytvárajú les
 - koreň hierarchie
 - trieda bez predka, od nikoho nededí
 - absolútny predok
 - Turbo Pascal

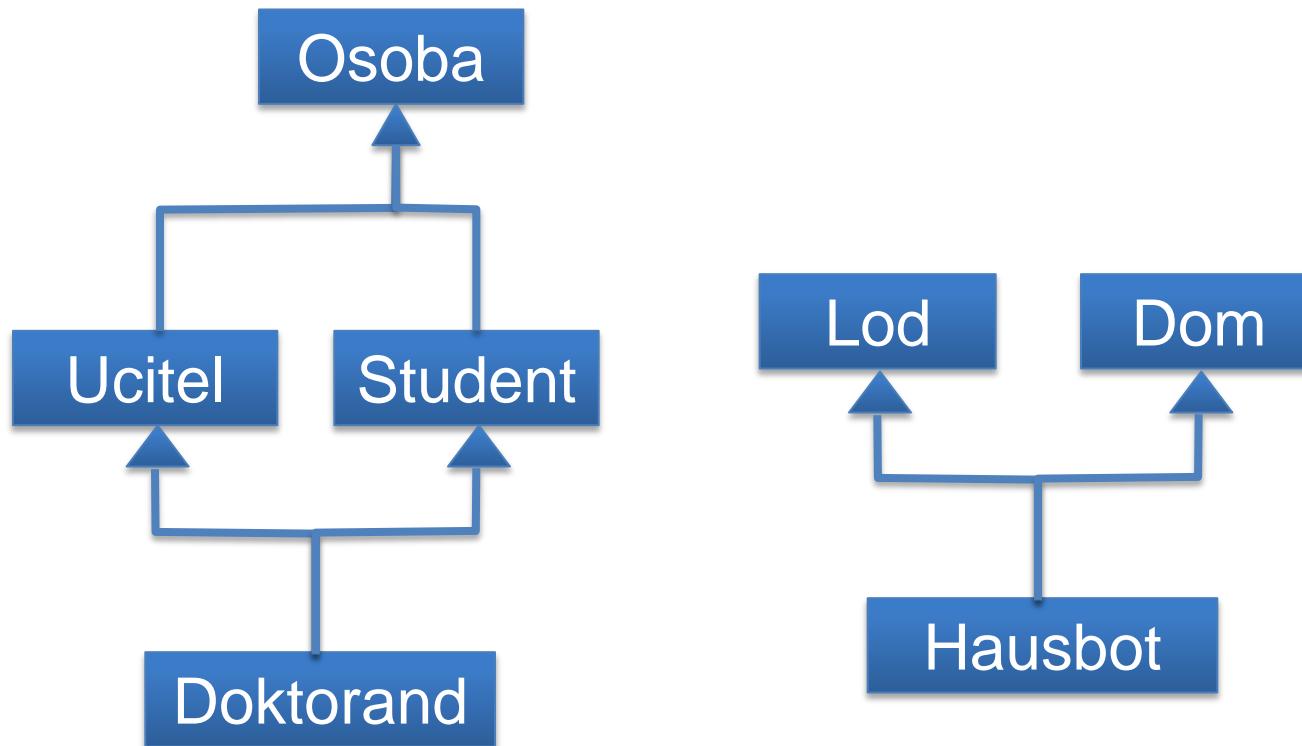
Les dedičnosti – Turbo Pascal



Viacnásobná dedičnosť

- trieda môže mať ľubovoľný počet priamych predkov
- hierarchia závislých tried – mriežka
 - orientované hrany – vzťah dedičnosti
 - neorientované hrany – vzťah závislosti
 - graf dedičnosti → graf závislosti
 - graf závislosti – súvislý graf
- množina mriežok
- C++, Python, ...

Mriežky dedičnosti – C++



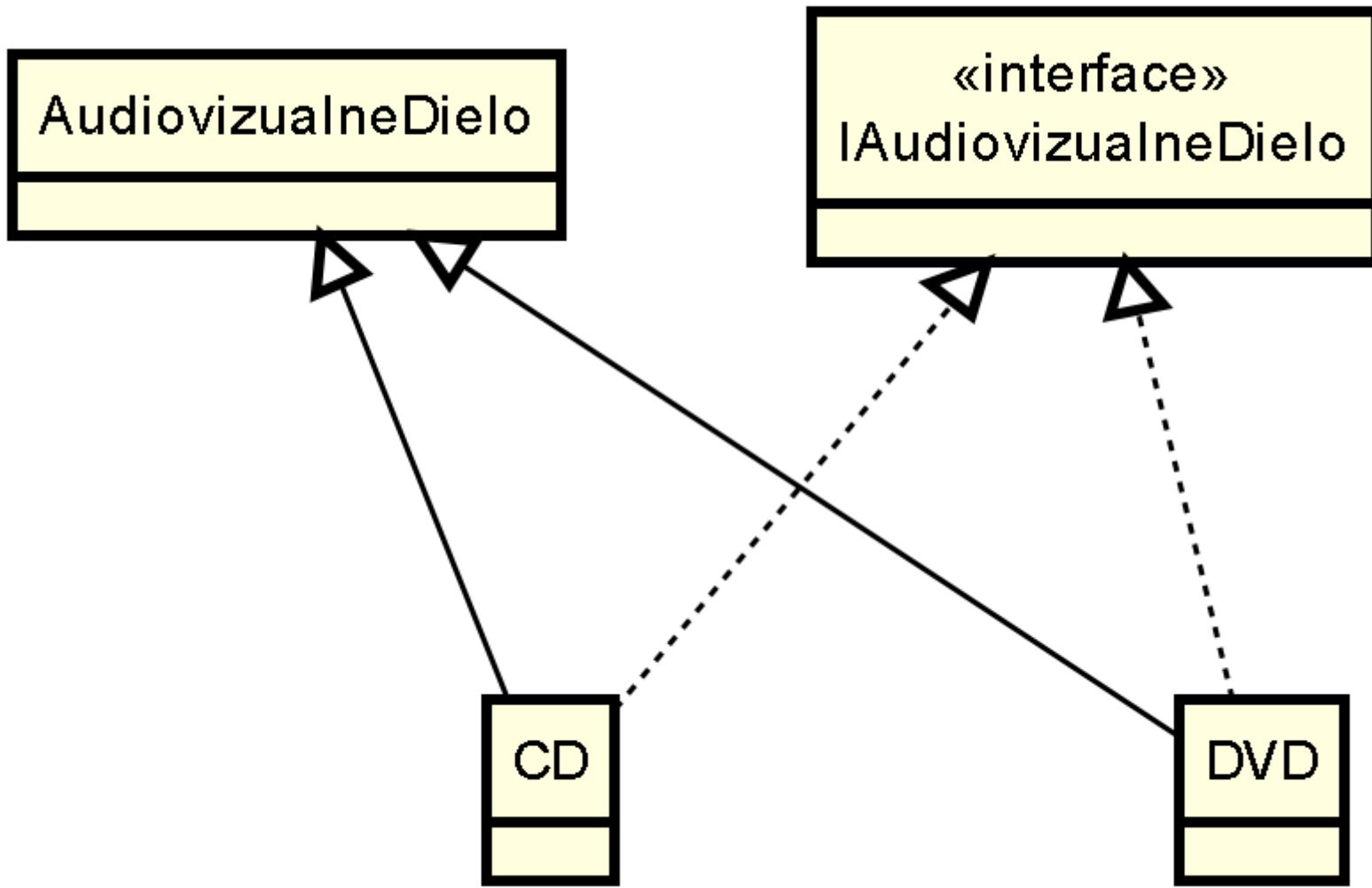
Dedenie implementácie interface

- trieda AudiovizualneDielo implementuje interface IAudiovizualneDielo
- triedy CD a DVD dedia po triede AudiovizualneDielo

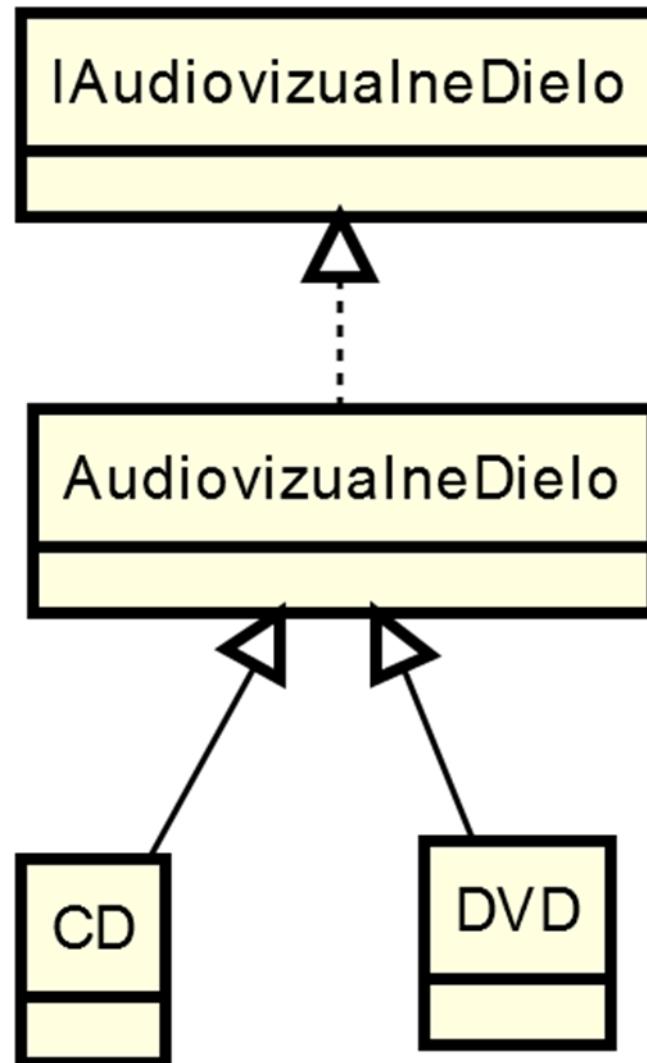
=>

- triedy CD a DVD implementujú interface IAudiovizualneDielo
 - je zachovaná typová kompatibilita CD a DVD s IAudiovizualneDielo

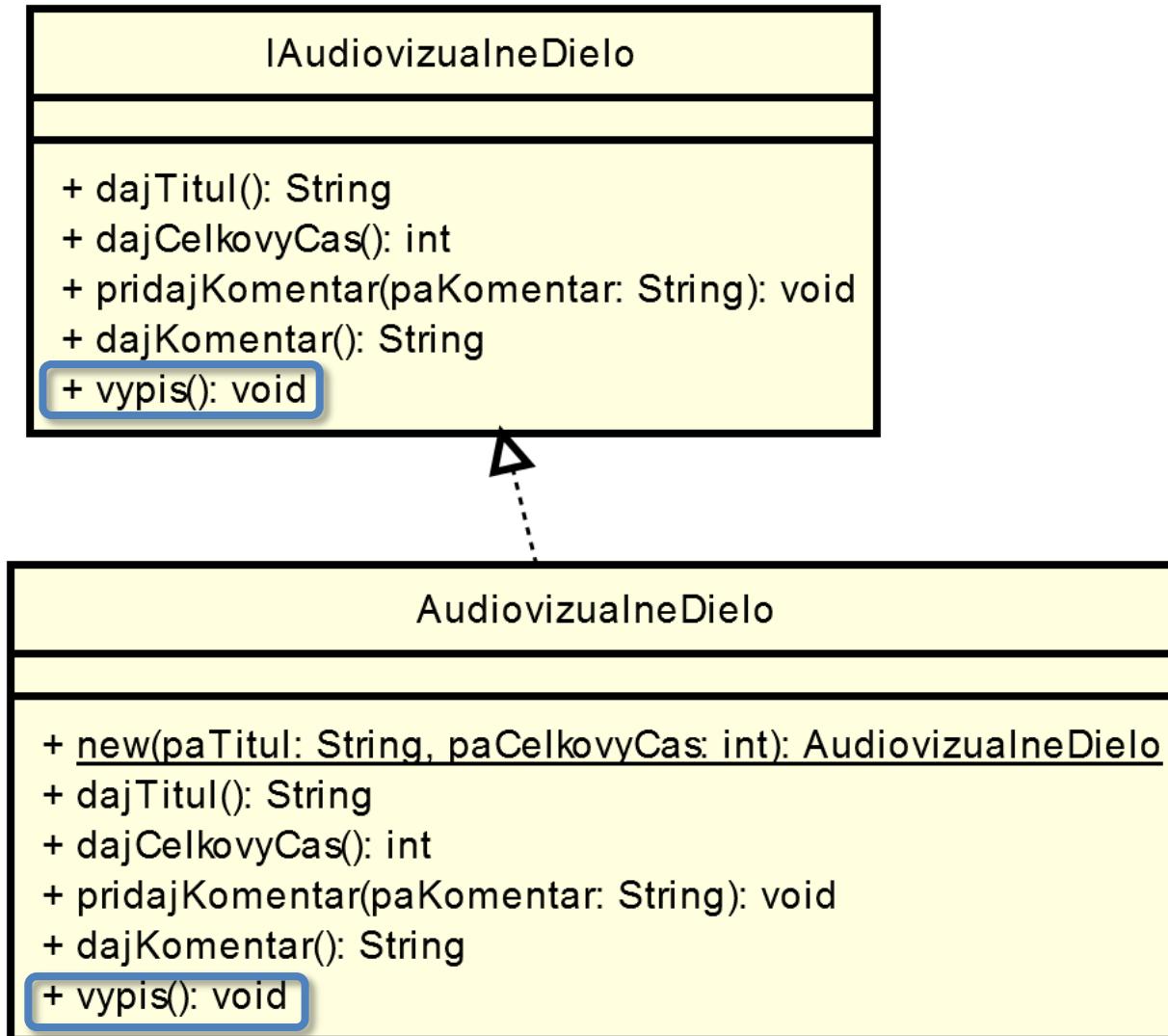
KCalB – pôvodná verzia



KCalB – nová verzia₍₁₎



KCalB – nová verzia₍₂₎



Problém nového riešenia

- `AudiovizualneDielo` implementuje kompletný interface `IAudiovizualneDielo`
 - metóda `vypis`
- `AudiovizualneDielo` nepozná režiséra DVD, autora CD a počet skladieb CD
 - nedokáže vypísať – nekompletný výpis

Dedičnosť a typová kompatibilita

- vztah predok – potomok:
 - každý potomok (nielen priamy) je typovo kompatibilný s každým svojim predkom
- typová kompatibilita => implicitné pretypovanie
- žiadny predok nie je kompatibilný so žiadnym svojim potomkom
 - explicitné pretypovanie
 - instanceof – bezpečné explicitné pretypovanie

Dedičnosť a polymorfizmus

- polymorfizmus – rôzne chovanie
 - určuje rozhranie
 - zhoda v rozhraní – rozdiel v implementácii
- dedičnosť – hierarchia a štruktúra tried
 - určuje vnútorná štruktúra tried
- typová kompatibilita => polymorfizmus
 - typ určitého predka – statický (definovaný) typ
 - typ nejakého potomka – dynamický (skutočný) typ
 - predok aj potomok majú v rozhraní rovnakú správu
 - predok aj potomok definuje vlastnú metódu

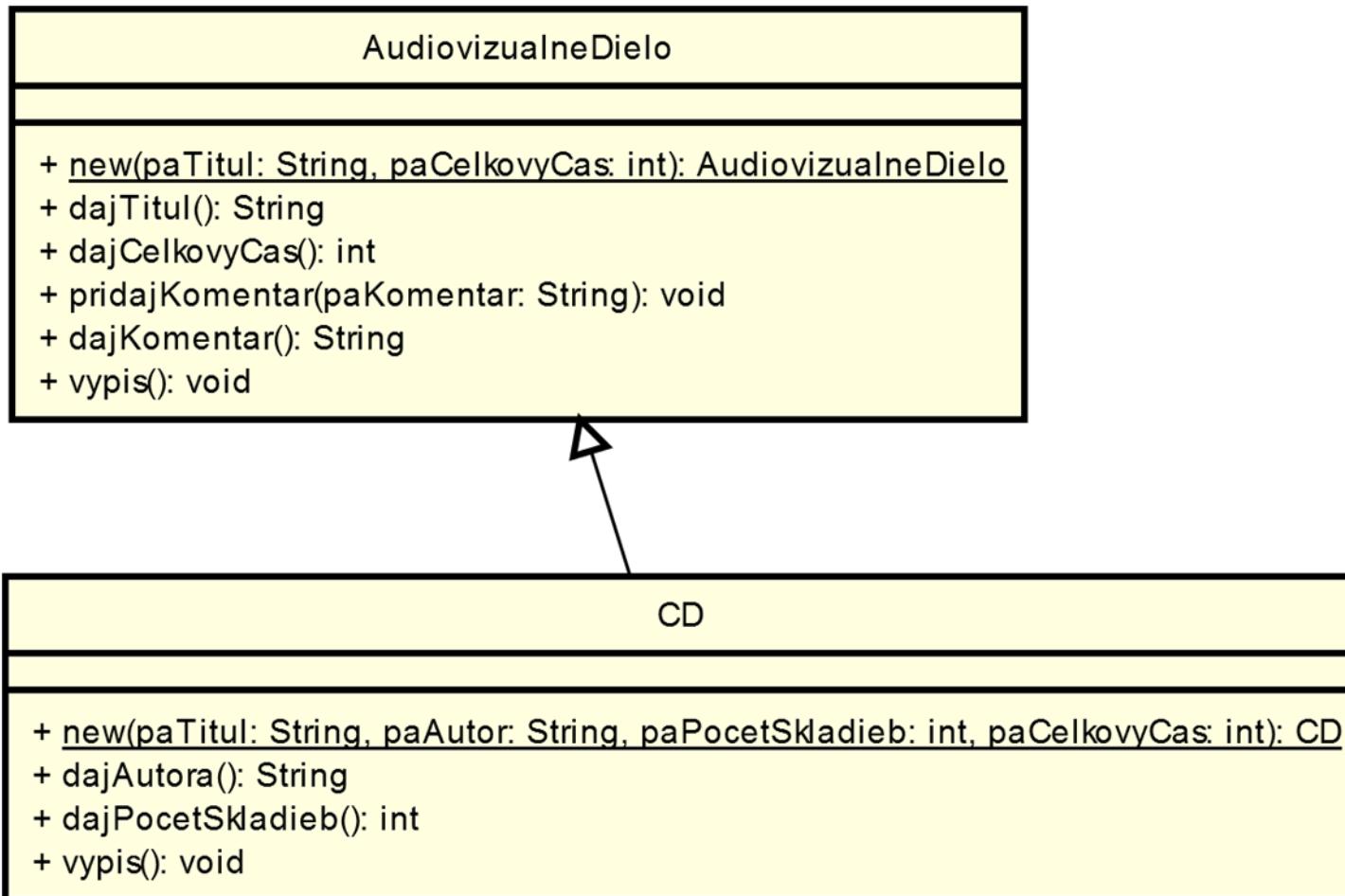
Metódy a polymorfizmus₍₁₎

- metódy potomka:
 - zdedené – potomok využíva bez zmeny
 - vlastné – nové metódy, ktoré predok nemá
 - predefinované (prekryté) – predok aj potomok má svoju vlastnú implementáciu – realizujú polymorfizmus
 - POZOR – preťažené metódy potomka = vlastné

Metódy a polymorfizmus₍₂₎

- preťaženie – dve rôzne správy aj metódy
 - aj v jednej triede, aj v rôznych triedach
- prekrytie – jedna správa a dve rôzne metódy
 - vždy v rôznych triedach

Polymorfizmus pomocou dedičnosti₍₁₎



Polymorfizmus pomocou dedičnosti₍₂₎

AudiovizualneDielo d;

d = **new** CD("The Best Of", "Beatles", 12, 75);

// co vypise?

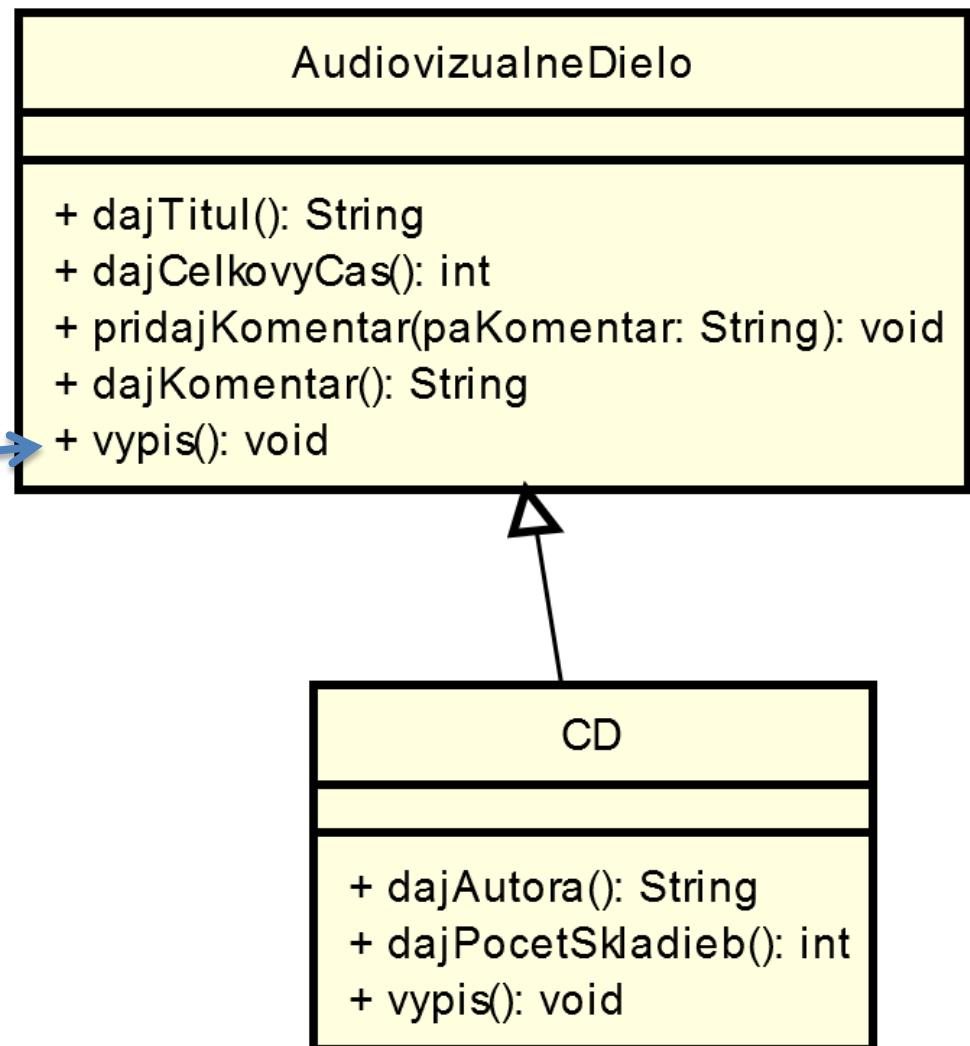
d.vypis();

Poslanie správy, prekladač

```
AudiovizualneDielo d;  
d = new CD(...);
```

statický typ

```
d.vypis();
```



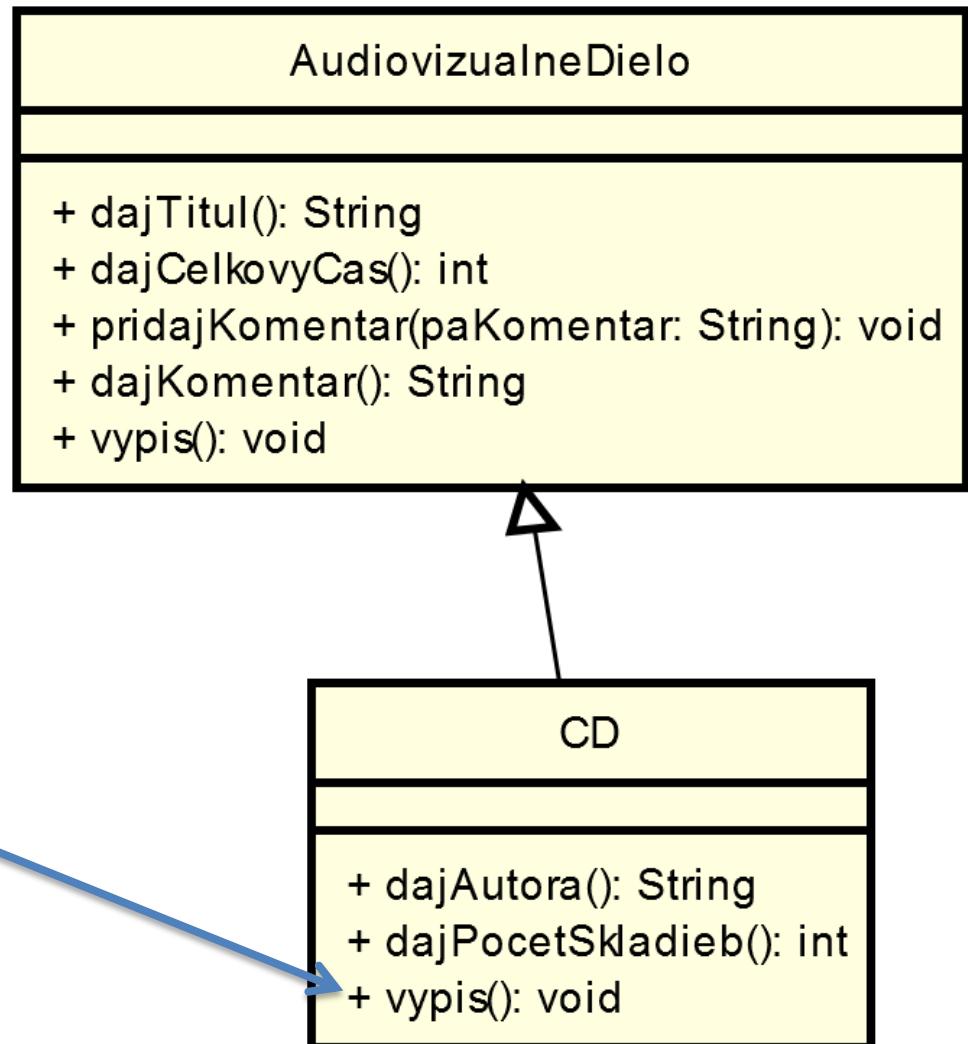
Poslanie správy, vykonanie₍₁₎

AudiovizualneDielo d;

d = new CD(...);

d.vypis();

dynamický typ

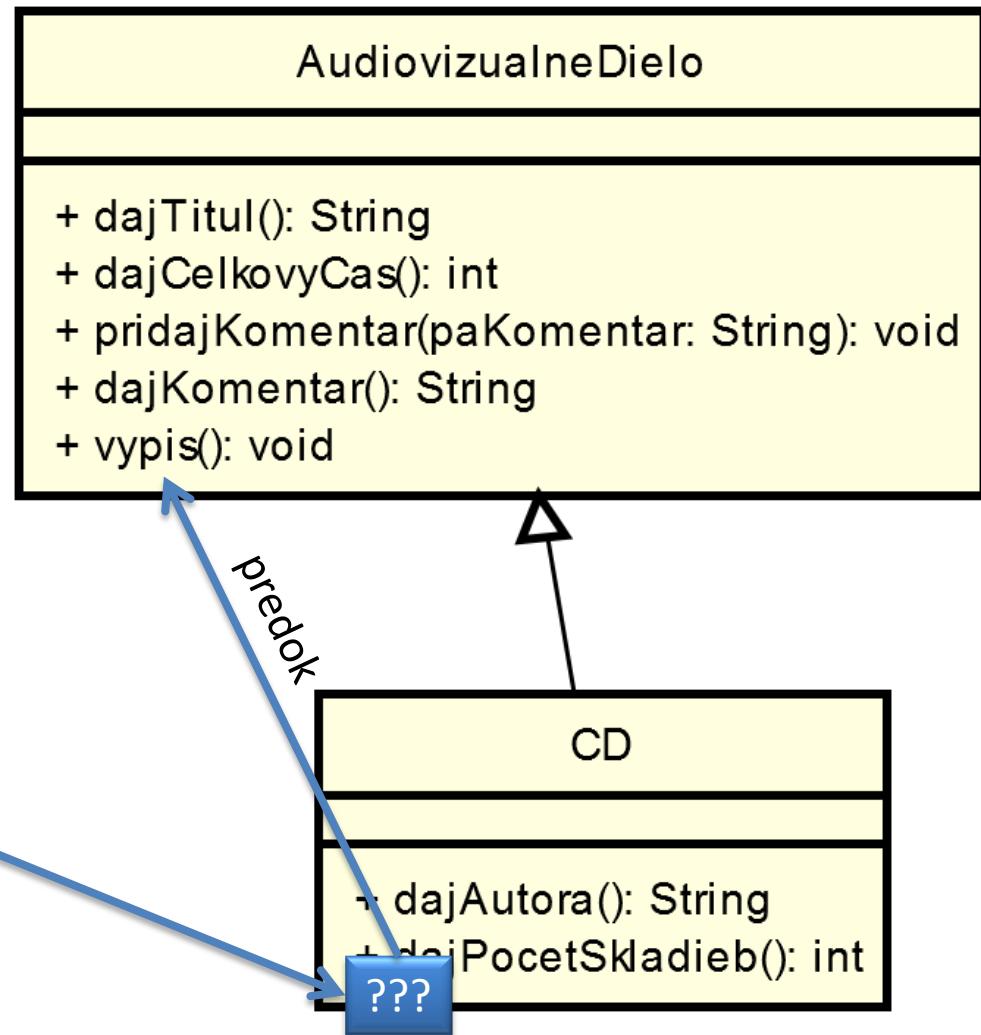


Poslanie správy, vykonanie₍₂₎

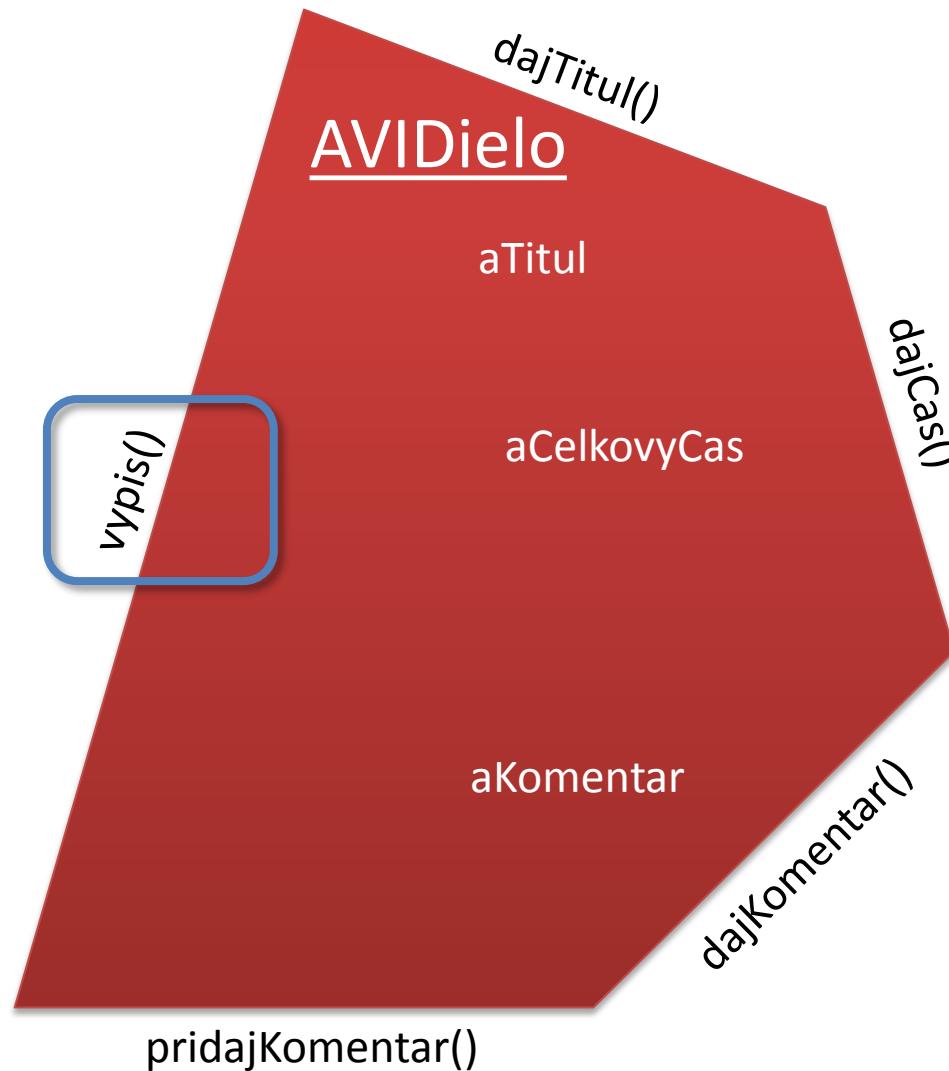
```
AudiovizualneDielo d;  
d = new CD(...);
```

```
d.vypis();
```

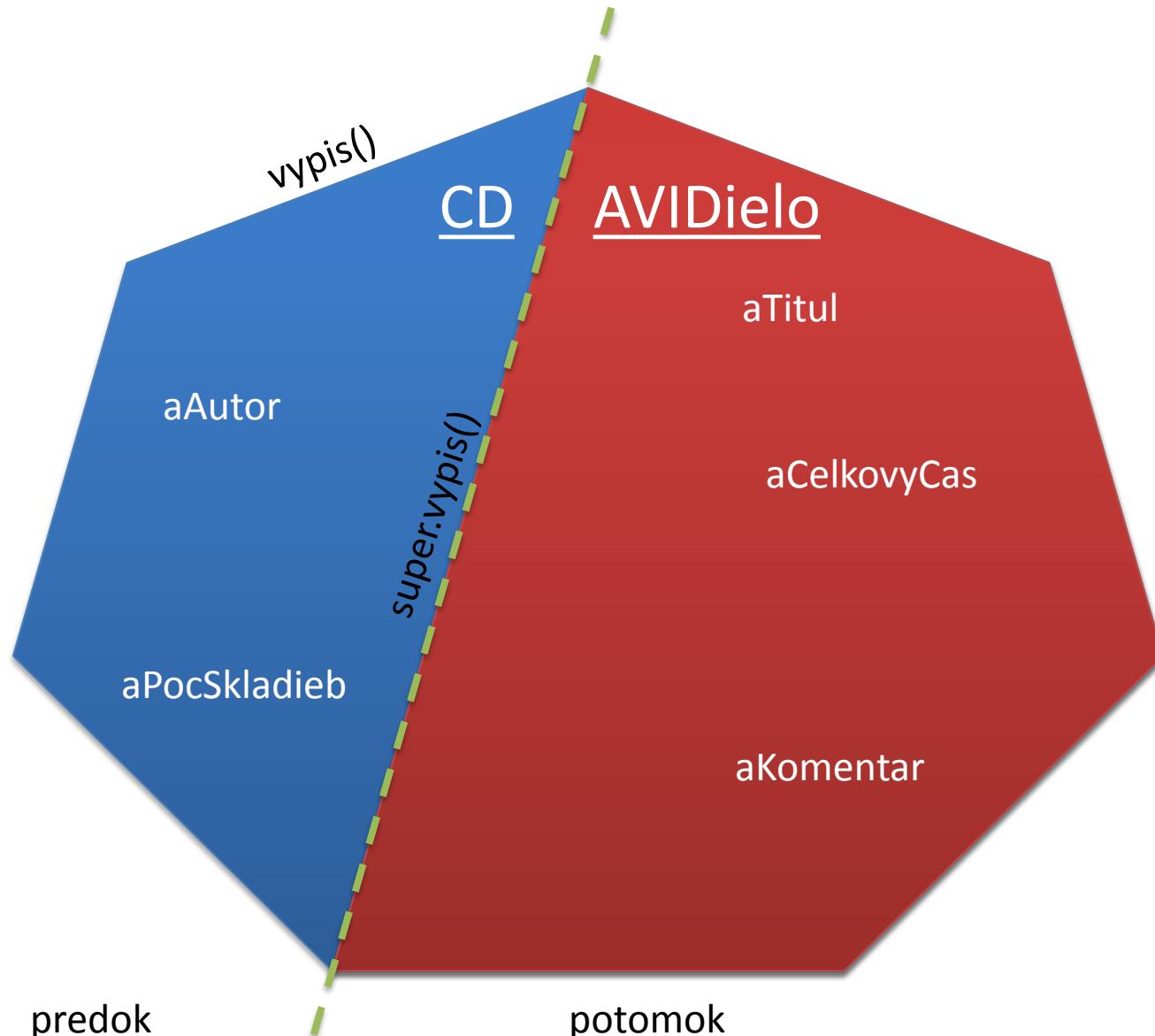
dynamický typ



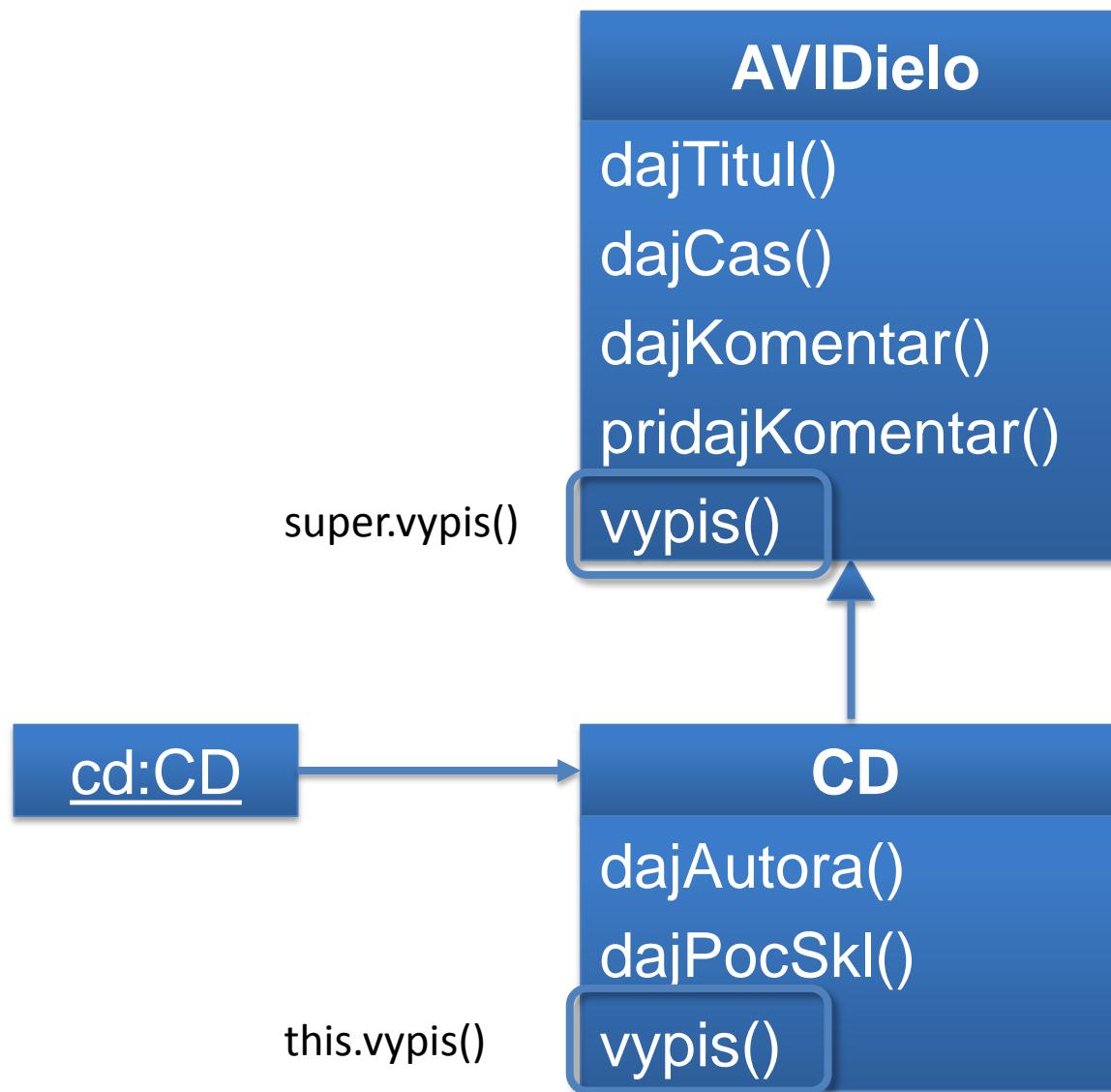
Metóda vypis - AudiovizualneDielo



Metódy vypis - CD



KCalB: polymorfizmus



Java – prekrytie metód

- super – explicitné použitie metód predka
- nutnosť pre prekryté metódy

```
super.vypis();
```

- super potláča dynamický typ objektu
- super vnúti objektu typ predka

Metóda vypis() – AudiovizualneDielo

```
public void vypis()
```

```
{
```

```
    System.out.print(aTitul);
```

```
    System.out.println(" (" + aCas + " min.)");
```

```
    if (aKomentar != null) {
```

```
        System.out.println(aKomentar);
```

```
}
```

```
}
```

Metóda vypis() – DVD

```
public void vypis()
```

```
{
```

```
    System.out.println("DVD");
```

```
    System.out.printl("réžia: " + aReziser);
```

```
    super.vypis();
```

```
}
```

Polymorfizmus a jazyky

- všetky metódy môžu realizovať polymorfizmus
 - Java, Python
- len explicitne označené metódy realizujú polymorfizmus
 - Delphi, C++
 - označenie – virtual

Vďaka za pozornosť