Informatika 2

Základy dedičnosti

doc. Ing. Ján Janech, PhD.
11. 3. 2024

ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Pojmy zavedené v 3. prednáške (1)

- interface
  - implementácia viacerých interface v triedach
    - implements interface1, interface2, ...

Pojmy zavedené v 3. prednáške (2)

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

Cieľ prednášky

- základy dedičnosti
- príklad: KCalB

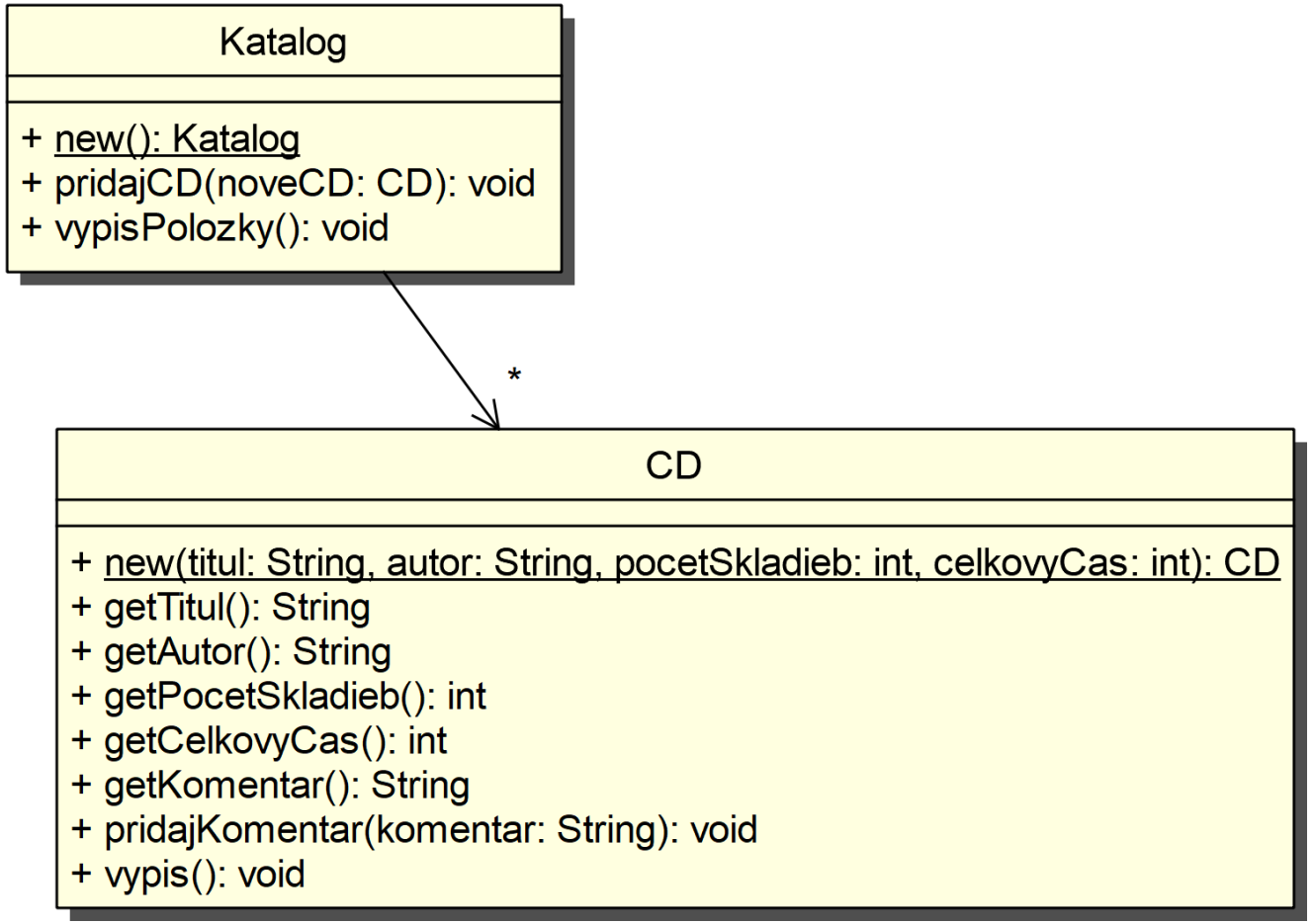
„KCalB“ – úloha 1

- 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

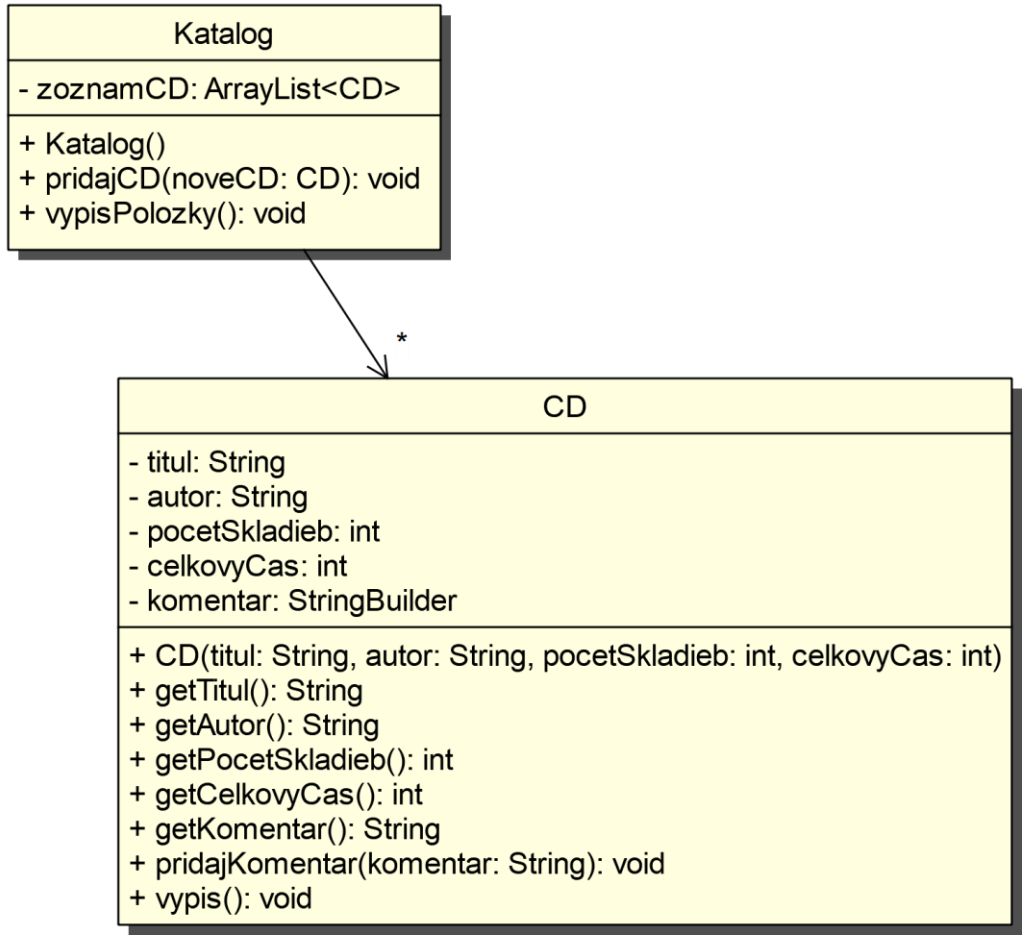
Riešenie

- 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 KCaIB – vonkajší pohľad



Prvá verzia KCaIB – vnútorný pohľad



„KCaIB“ – úloha 2 (1)

- jednoduchý katalóg audiovizuálnych diel
  - pridávanie diela
- výpis zoznamu diela
- užívateľské komentáre

- informácie o CD
- informácie o DVD

„KCaIB“ – 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

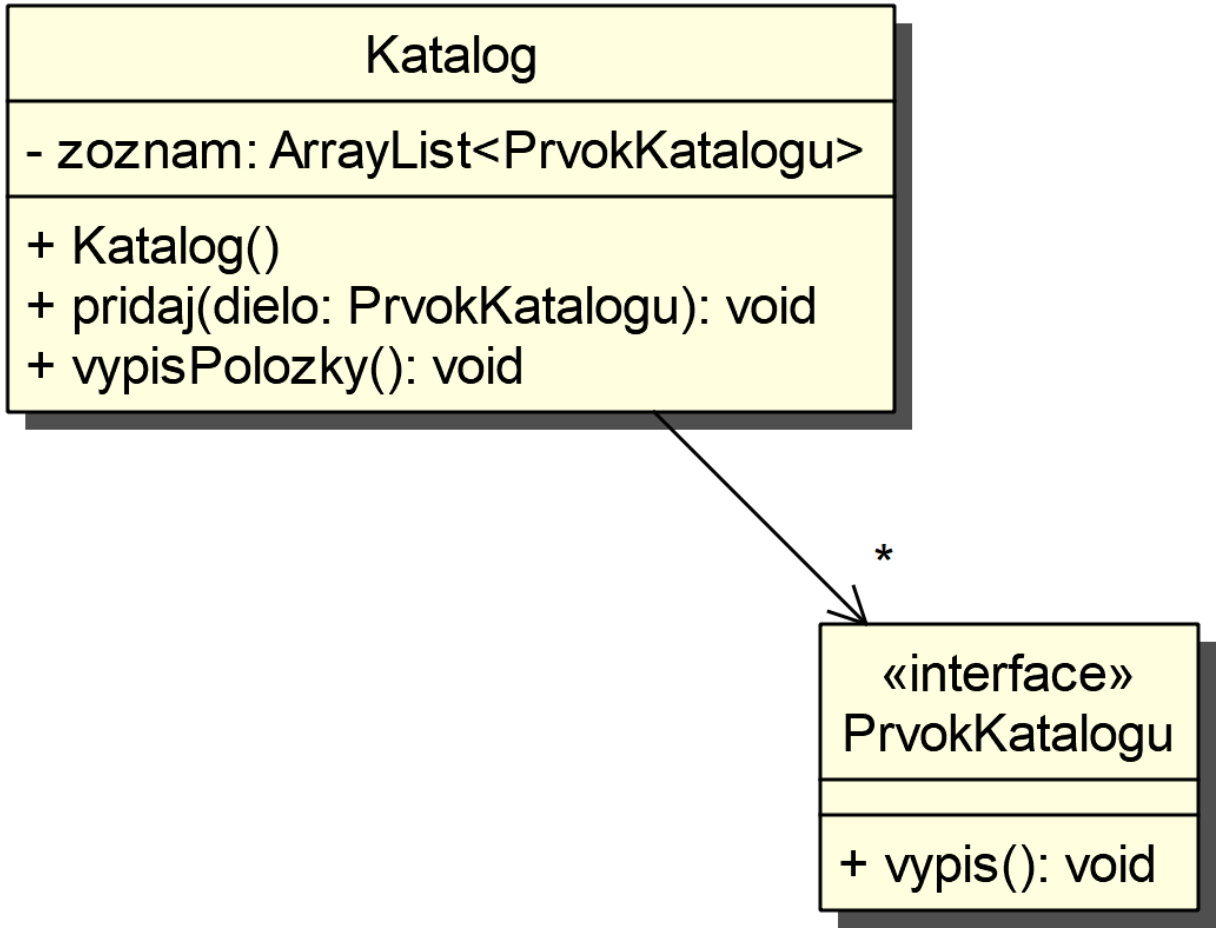
- trieda pre DVD
- trieda pre CD

- možnosti riešenia
  - dva kontajnery ArrayList
  - polymorfizmus

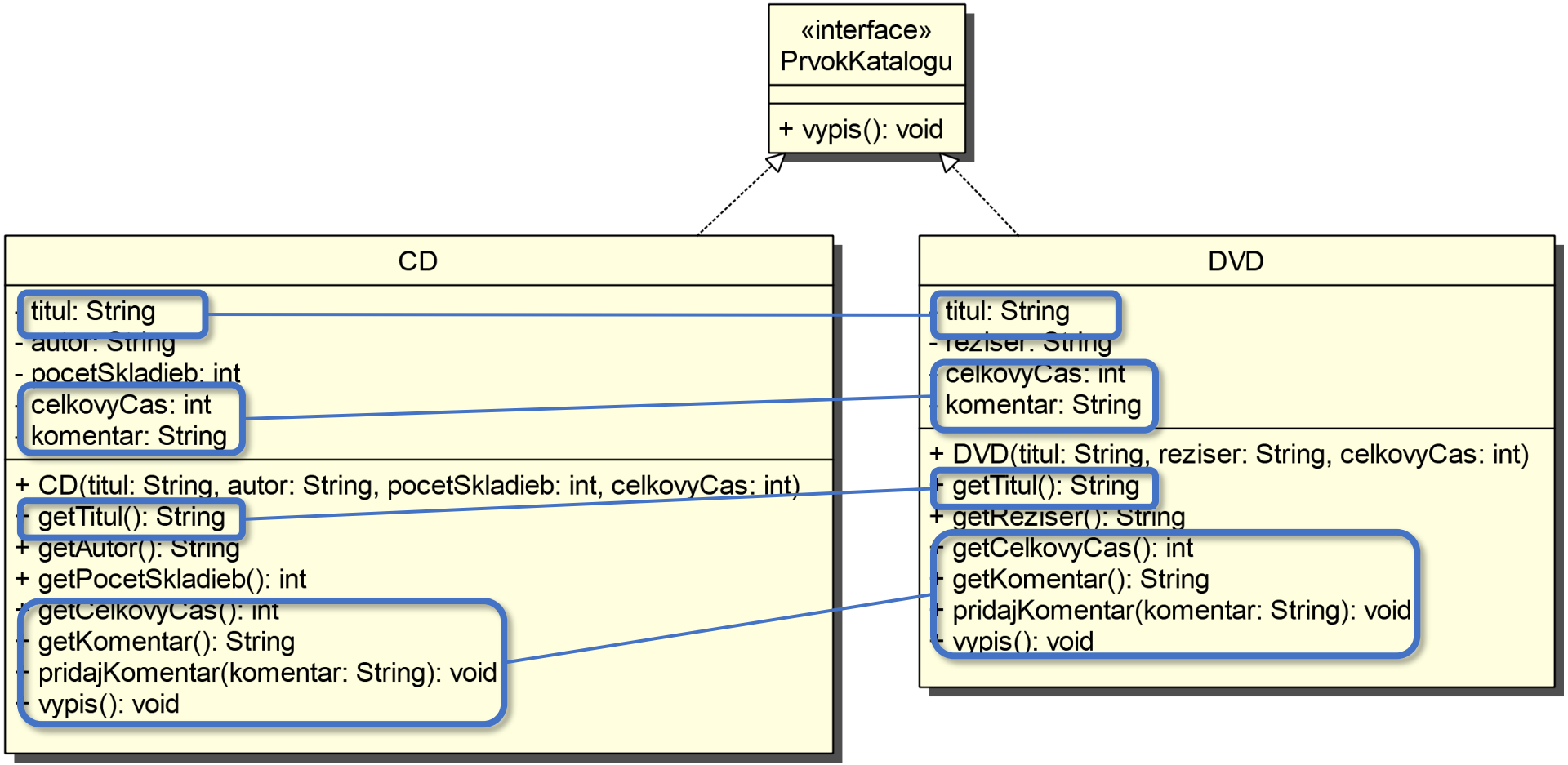
Riešenie pomocou polymorfizmu

- interface so nutnými spoločnými správami
  - vypis

Zavedenie interface (1)



Zavedenie interface (2)



Trieda CD – definícia

```
public class CD implements PrvokKatalogu {
    private String titul;
    private String autor;
    private int pocetSkladieb;
    private int celkovyCas;
    private StringBuilder komentar;
    ...
}
```

Trieda DVD – definícia

```
public class DVD implements PrvokKatalogu {
    private String titul;
    private String reziser;
    private int celkovyCas;
    private StringBuilder komentar;
    ...
}
```

Trieda CD – konštruktor

```
public CD(String titul, String autor, int pocetSkladieb, int celkovyCas) {
    this.titul = titul;
    this.autor = autor;
    this.pocetSkladieb = pocetSkladieb;
    this.celkovyCas = celkovyCas;
    this.komentar = new StringBuilder();
}
```

Trieda DVD – konštruktor

```
public DVD(String titul, String reziser, int celkovyCas){
    this.titul = titul;
    this.reziser = reziser;
    this.celkovyCas = celkovyCas;
    this.komentar = new StringBuilder();
}
```

Trieda CD – prístupové metódy (1)

```
public String getTitul() {
    return this.titul;
}

public String getAutor() {
    return this.autor;
}
```

Trieda CD – prístupové metódy (2)

```
public int getPocetSkladieb() {
    return this.pocetSkladieb;
}

public int getCelkovyCas() {
    return this.celkovyCas;
}
```

Trieda DVD – prístupové metódy

```
public String getTitul() {
    return this.titul;
}

public String getReziser() {
    return this.reziser;
}

public int getCelkovyCas() {
    return this.celkovyCas;
}
```

Trieda CD – práca s komentármi

```
public String getKomentar() {
    return this.komentar;
}

public void pridajKomentar(String riadok) {
    if (this.komentar.length() > 0) {
        this.komentar.append('\n');
    } else {
        this.komentar.append(riadok);
    }
}
```

Trieda DVD – práca s komentármi

```
public String getKomentar() {
    return this.komentar;
}

public void pridajKomentar(String riadok) {
    if (this.komentar.length() > 0) {
        this.komentar.append('\n');
    } else {
        this.komentar.append(riadok);
    }
}
```

Trieda CD – výpis

```
public void vypis() {
    System.out.println("CD:");
    System.out.println("- Skladieb: " + this.pocetSkladieb);
    System.out.println("- Autor: " + this.autor);
    System.out.println("- Titul: " + this.titul);
    System.out.println("- Cas: " + this.celkovyCas);
    if (this.komentar.length() > 0) {
        System.out.println("Komentar ku CD:");
        System.out.println(this.komentar);
    }
}
```

Trieda DVD – výpis

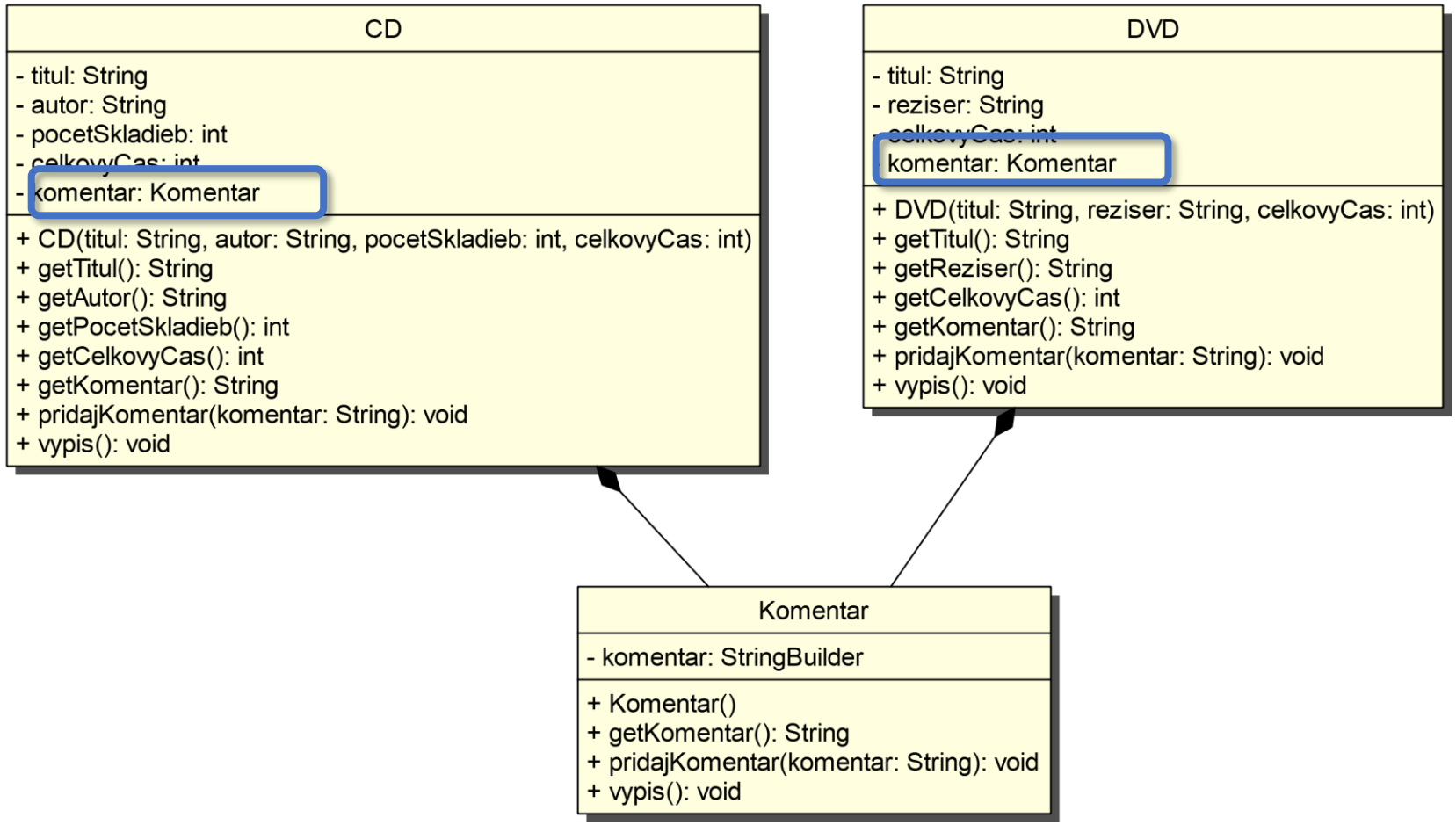
```
public void vypis() {
    System.out.println("DVD:");
    System.out.println("- Reziser: " + this.reziser);
    System.out.println("- Titul: " + this.titul);
    System.out.println("- Cas " + this.celkovyCas);
    if (this.komentar.length() > 0) {
        System.out.println("Komentar ku DVD:");
        System.out.println(this.komentar);
    }
}
```

Duplicita kódu

- triedy CD a DVD zdieľajú veľkú časť kódu
- duplicita kódu
  - sťažuje údržbu kódu
  - potencionálne miesto pre vznik chýb

- riešenie
  - kompozícia
  - trieda so spoločnými časťami
    - vyčleniť sa dá trieda Komentare

Riešenie pomocou kompozície (1)



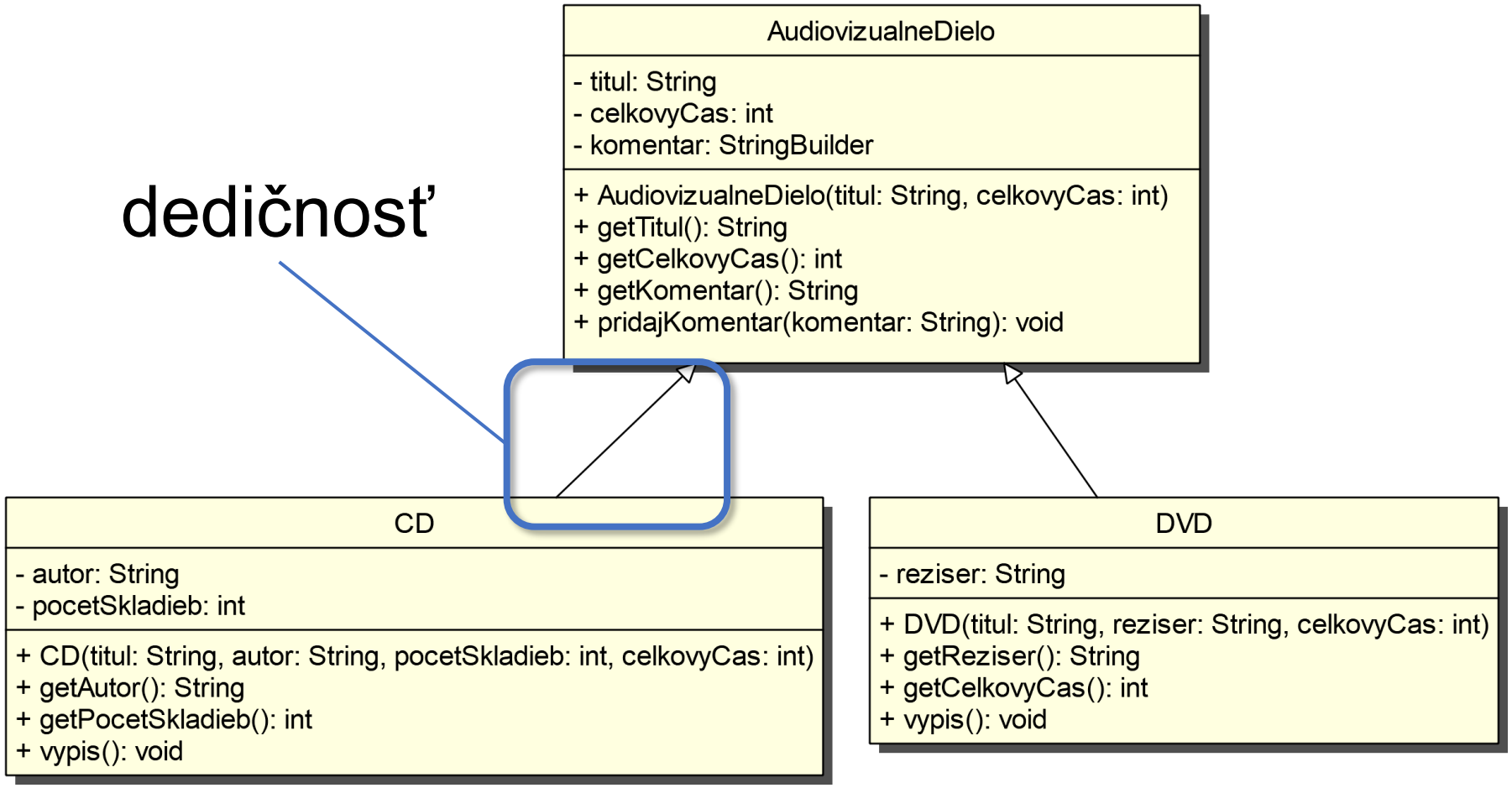
Riešenie pomocou kompozície (2)

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

```
public void pridajKomentar(String riadok) {
    this.komentar.pridajKomentar(riadok);
}
```

- iné riešenie: dedičnosť

Riešenie pomocou dedičnosti



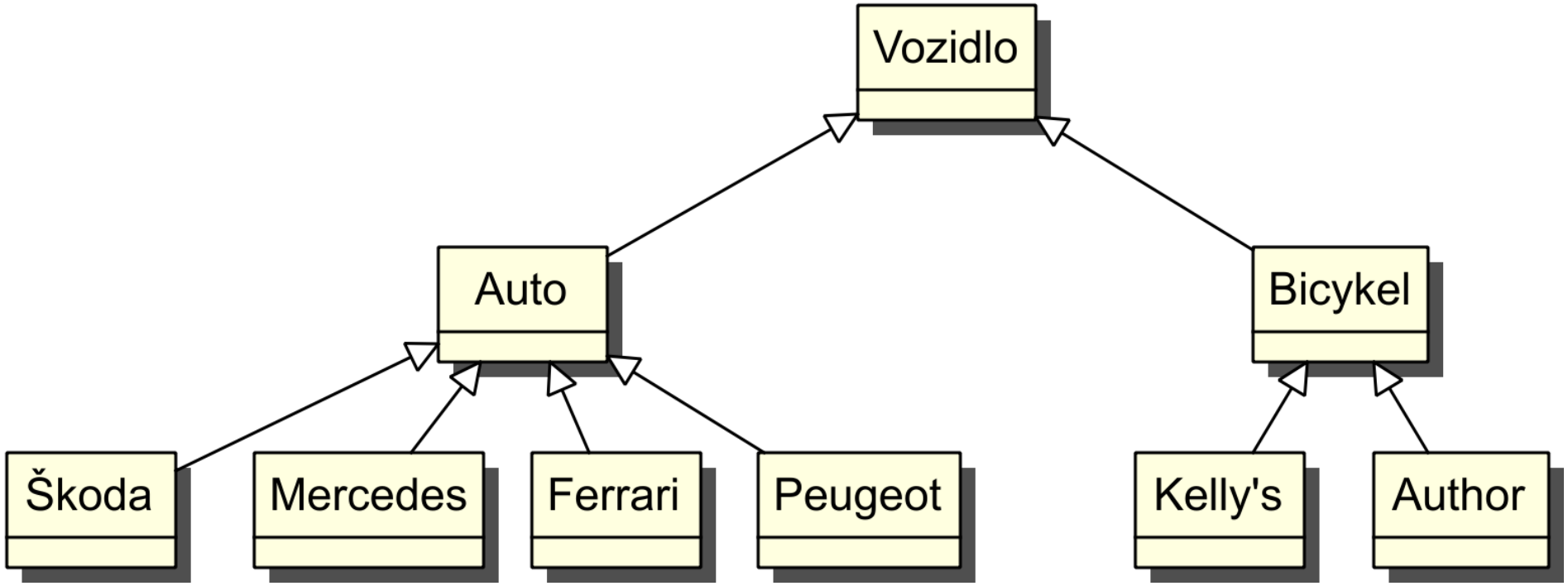
Dedičnosť

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

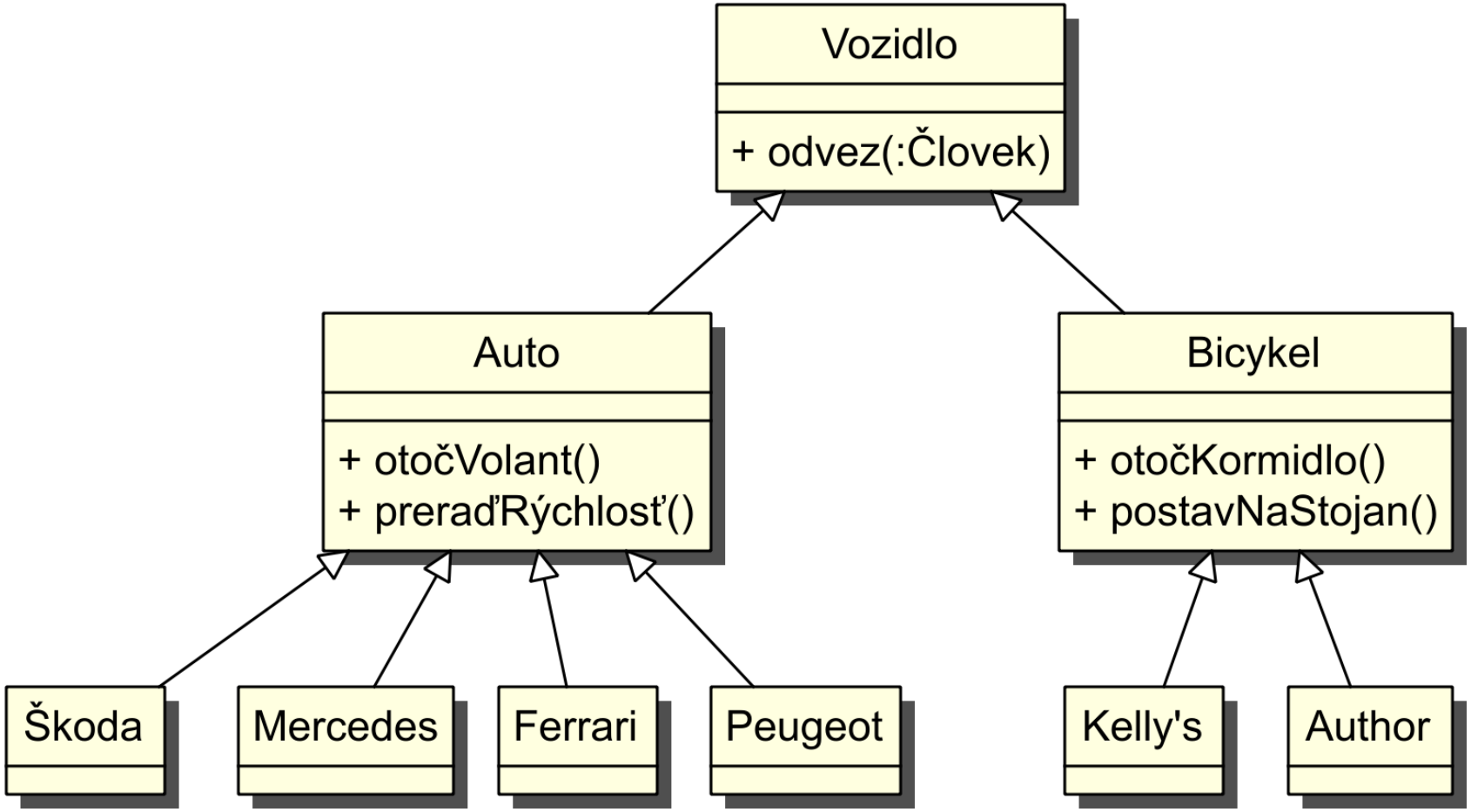
Dedičnosť v reálnom svete (1)

- 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

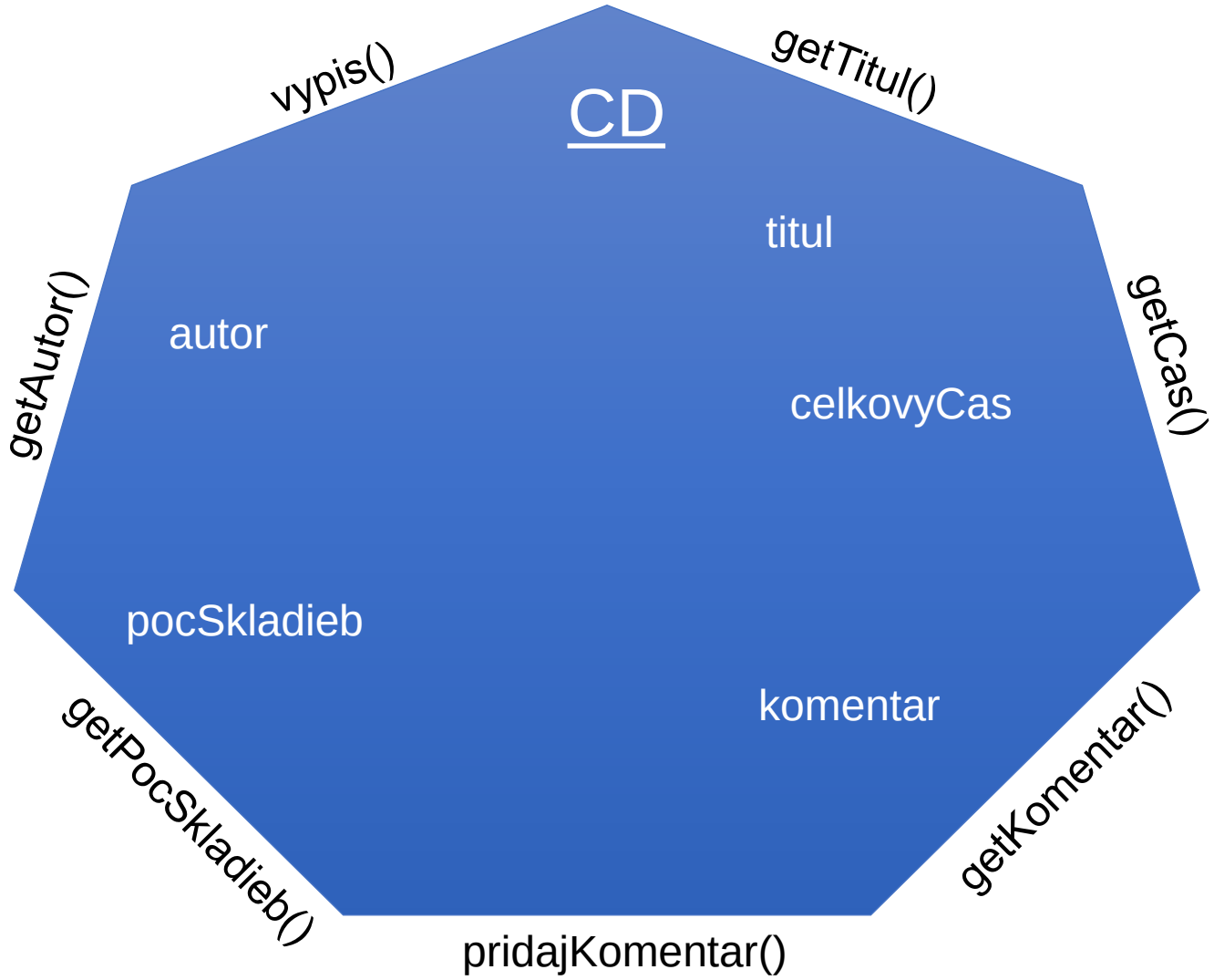
- absolútny predok – „koreň” stromu hierarchie
  - trieda Vozidlo

Potomok dedí od predka

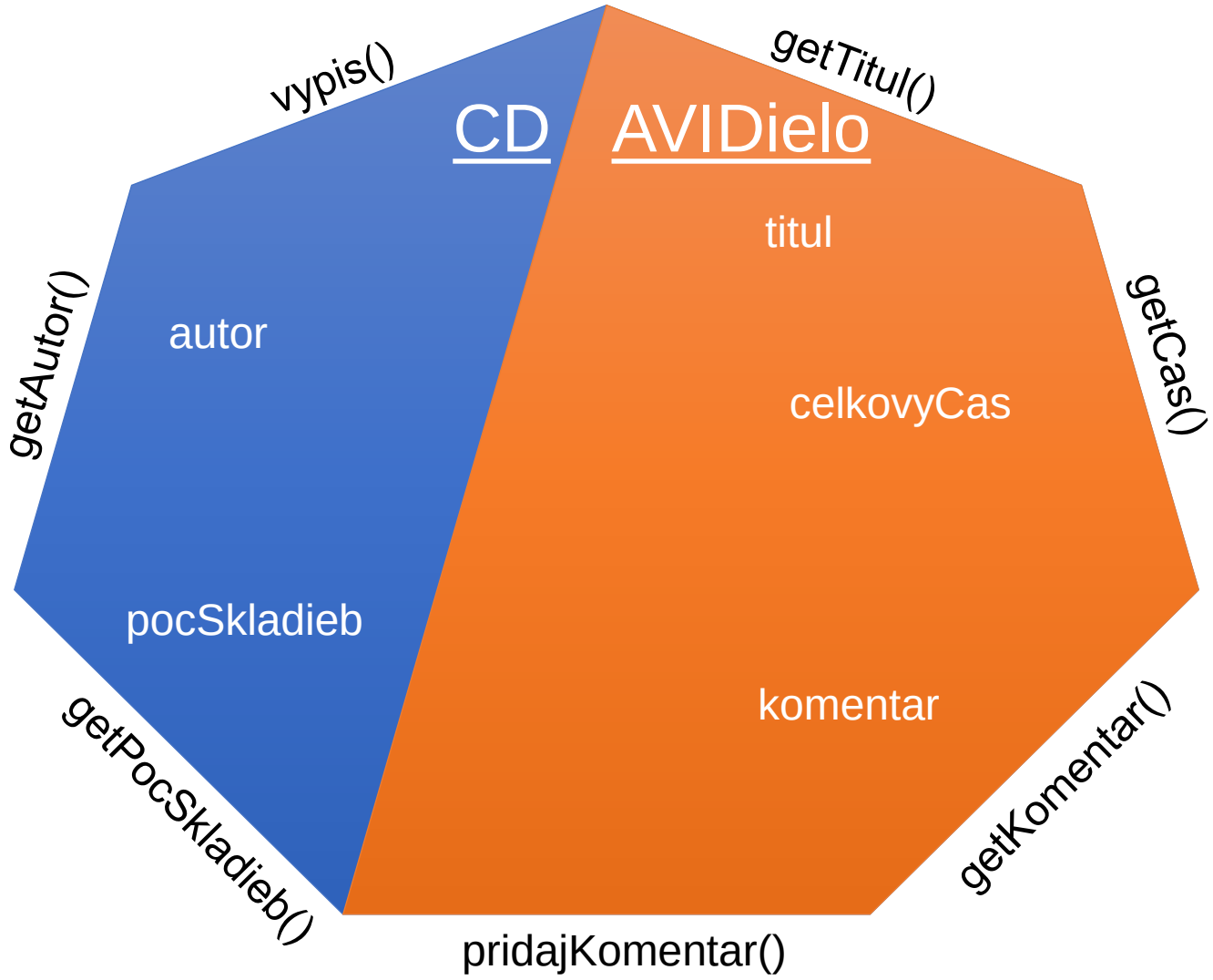
- vonkajší pohľad
  - verejné rozhranie
- vnútorný pohľad
  - atribúty
  - metódy

- 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 PrvokKatalogu
```

- v odvodenej triede
- len jeden predok

- kľúčové slovo extends
  - nemýliť s rozširovaním interface

Dedičnosť v jazyku Java (2)

- ! Potomok nededí konštruktory predka

- konštruktory treba znovu 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

KCaIB – Trieda AudiovizualneDielo

- spoločný predok CD, DVD

- atribúty
  - titul
  - celkovyCas
  - komentar
- metódy
  - getTitul
  - getCelkovyCas
  - getKomentar
  - pridajKomentar

KCaIB – Trieda CD

- potomok triedy AudiovizualneDielo

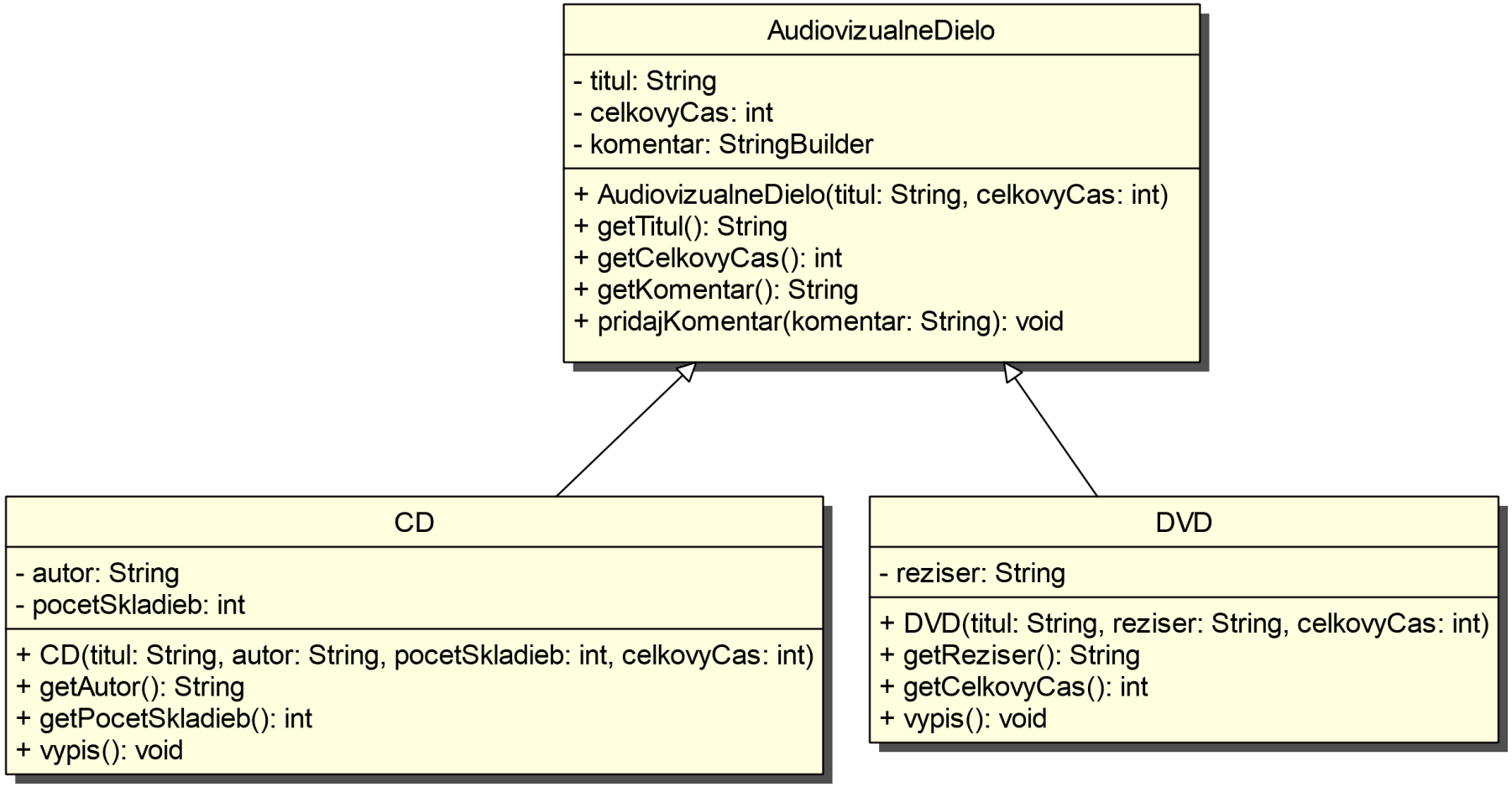
- atribúty
  - autor
  - pocetSkladieb
- metódy
  - getAutor
  - getPocetSkladieb
  - vypis

KCaIB – Trieda DVD

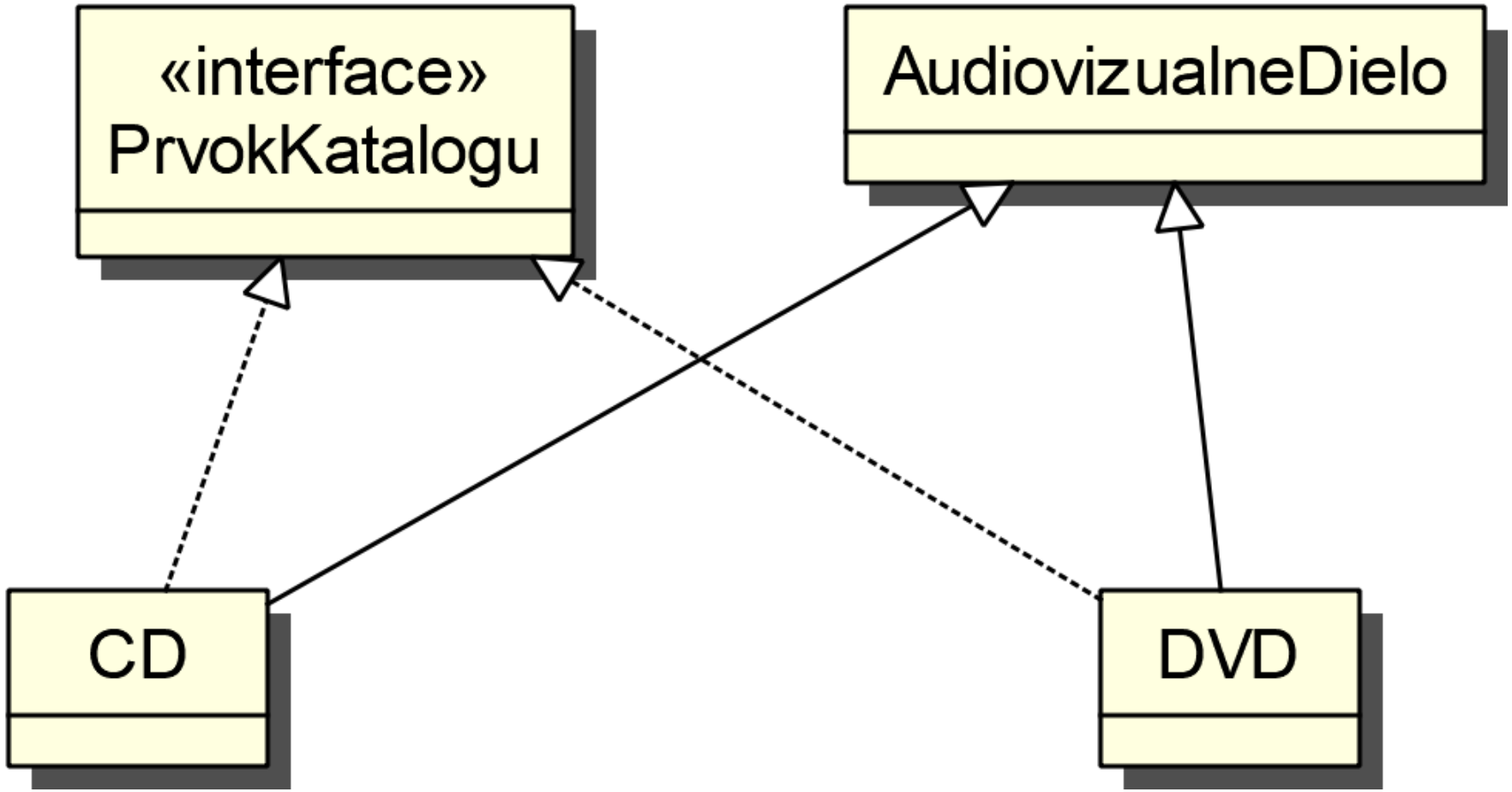
- potomok triedy AudiovizualneDielo

- atribúty
  - reziser
- metódy
  - getReziser
  - vypis

KCaIB v UML (1)


AudiovizualneDielo
- titul: String
- celkovyCas: int
- komentar: StringBuilder
+ AudiovizualneDielo(titul: String, celkovyCas: int)
+ getTitul(): String
+ getCelkovyCas(): int
+ getKomentar(): String
+ pridajKomentar(komentar: String): void
CD
- autor: String
- pocetSkladieb: int
+ CD(titul: String, autor: String, pocetSkladieb: int, celkovyCas: int)
+ getAutor(): String
+ getPocetSkladieb(): int
+ vypis(): void
DVD
- reziser: String
+ DVD(titul: String, reziser: String, celkovyCas: int)
+ getReziser(): String
+ getCelkovyCas(): int
+ vypis(): void

KCaIB v UML (2)



Trieda AudiovizualneDielo

```
public class AudiovizualneDielo {
    private String titul;
    private int celkovyCas; // v minutach
    private StringBuilder komentar;
    ...
}
```

Trieda AudiovizualneDielo – konštruktor

```
public AudiovizualneDielo(String titul, int celkovyCas) {
    this.titul = titul;
    this.celkovyCas = celkovyCas;
    this.komentar = new StringBuilder();
}
```

Trieda DVD

```
public class DVD extends AudiovizualneDielo implements PrvokKatalogu {
    private String reziser;
    ...
}
```

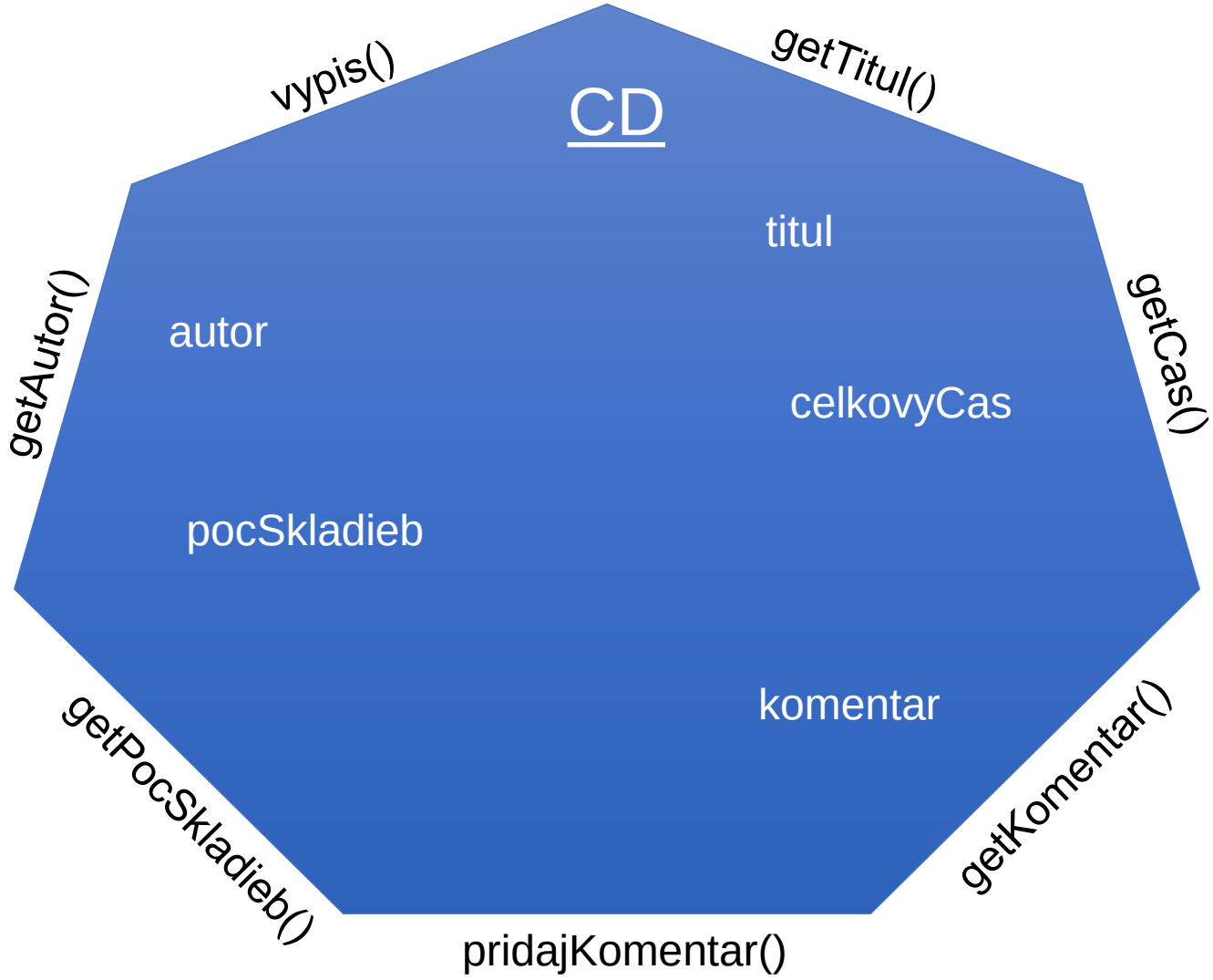
Trieda DVD – konštruktor

```
public DVD(String titul, String reziser, int celkovyCas){
    super(titul, celkovyCas);

    this.reziser = reziser;
}
```

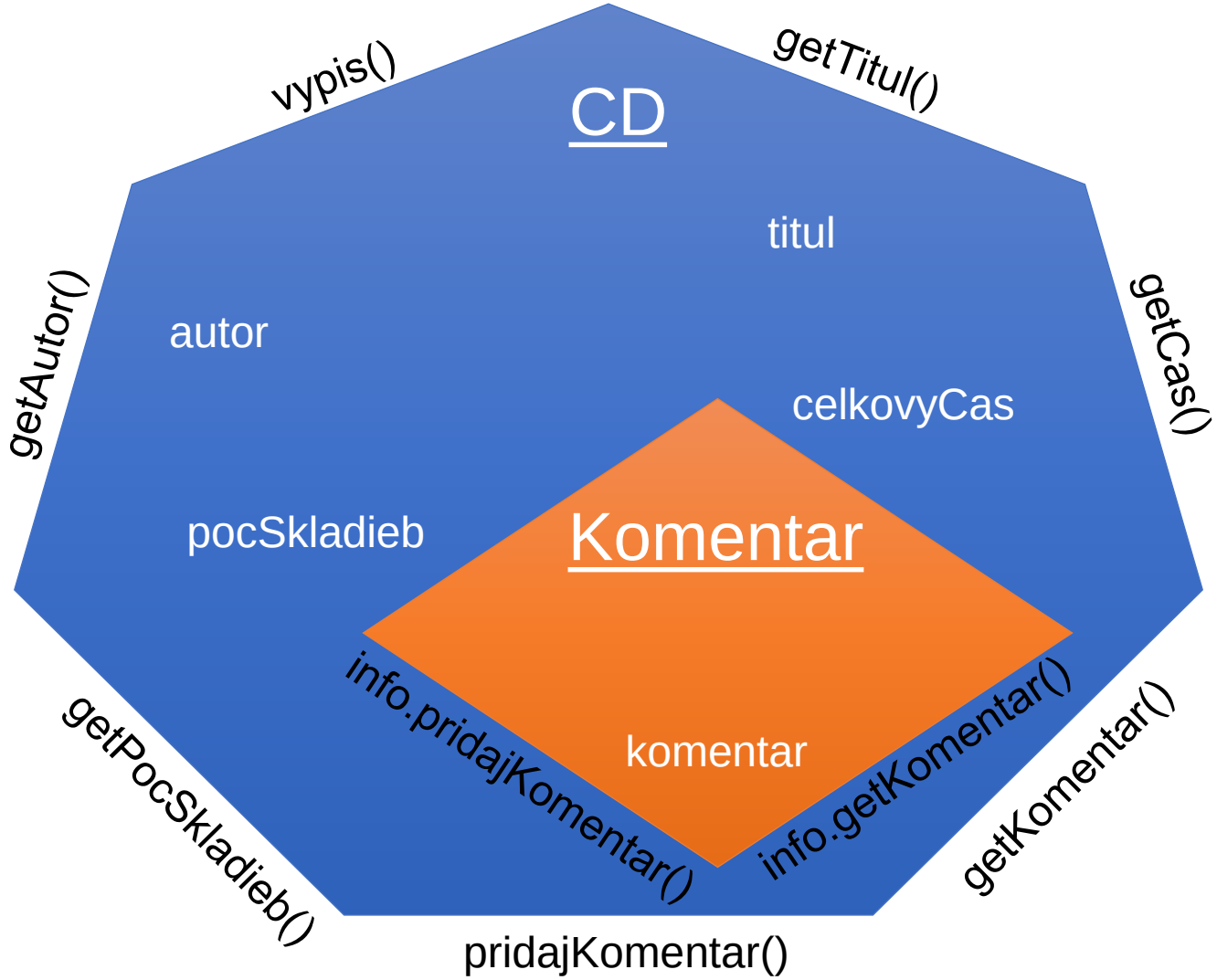
KCaIB – tri rôzne spôsoby riešenia

- nezávislé triedy
- skladanie – časť a celok
- dedičnosť

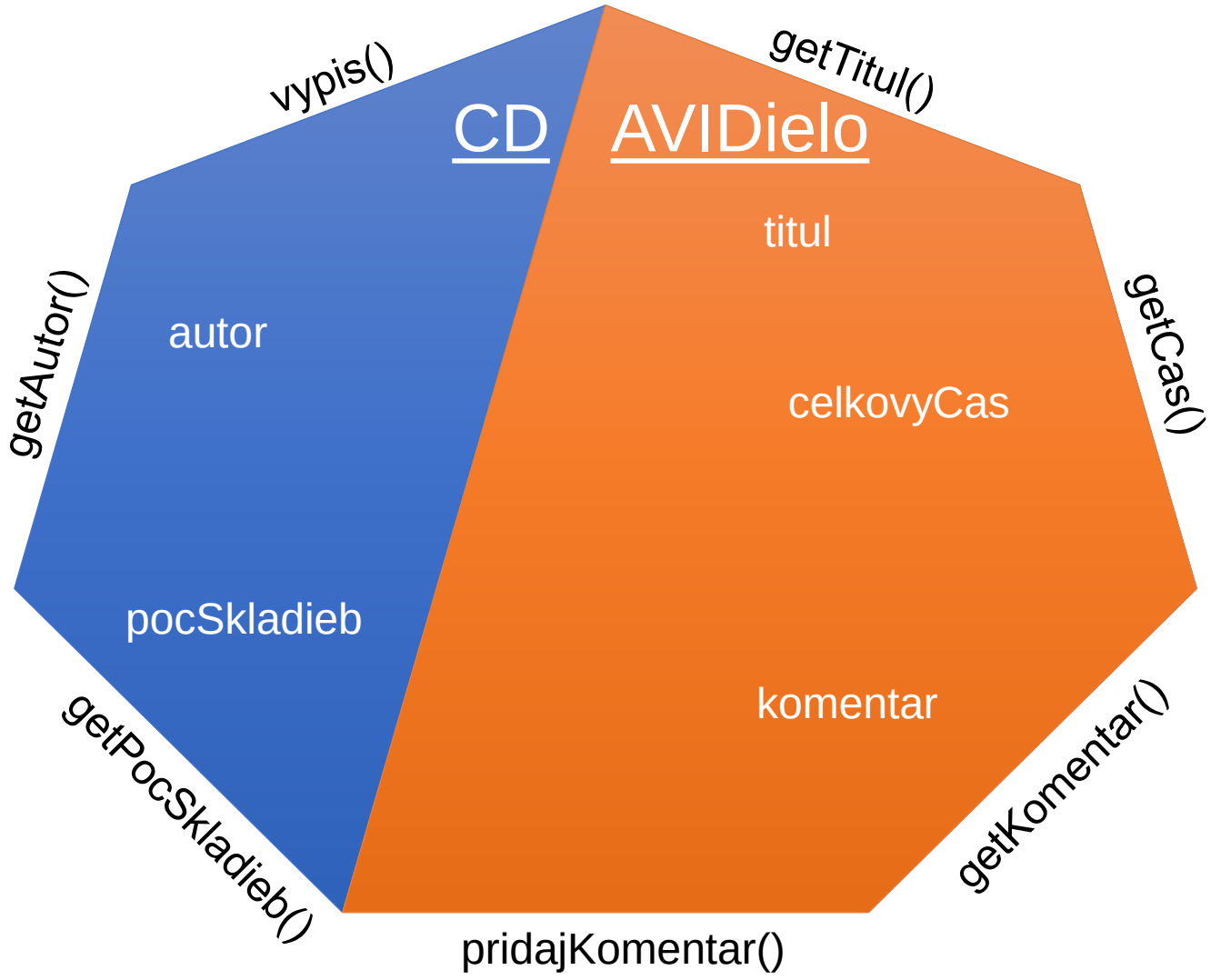
CD: nezávislá trieda – monolit



CD: závislé triedy – skladanie



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

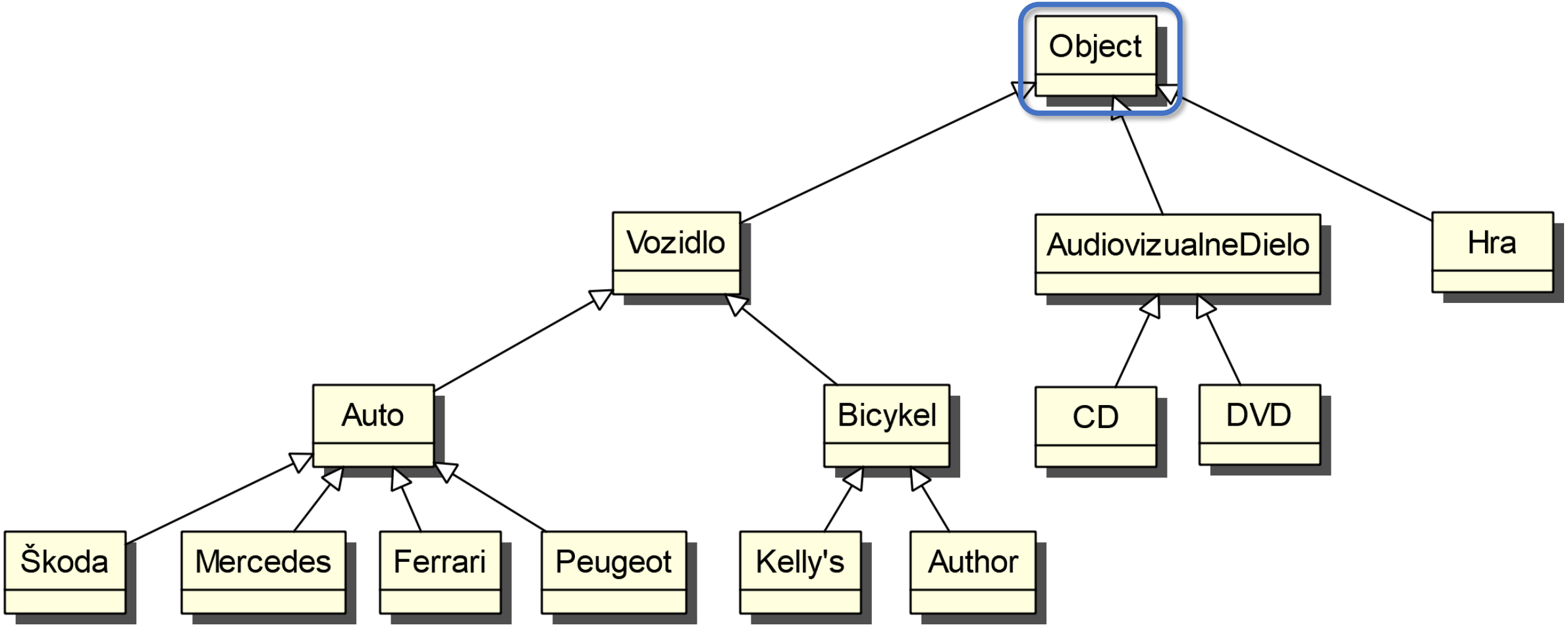
Implementačná závislosť

- 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

Jednoduchá dedičnosť

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

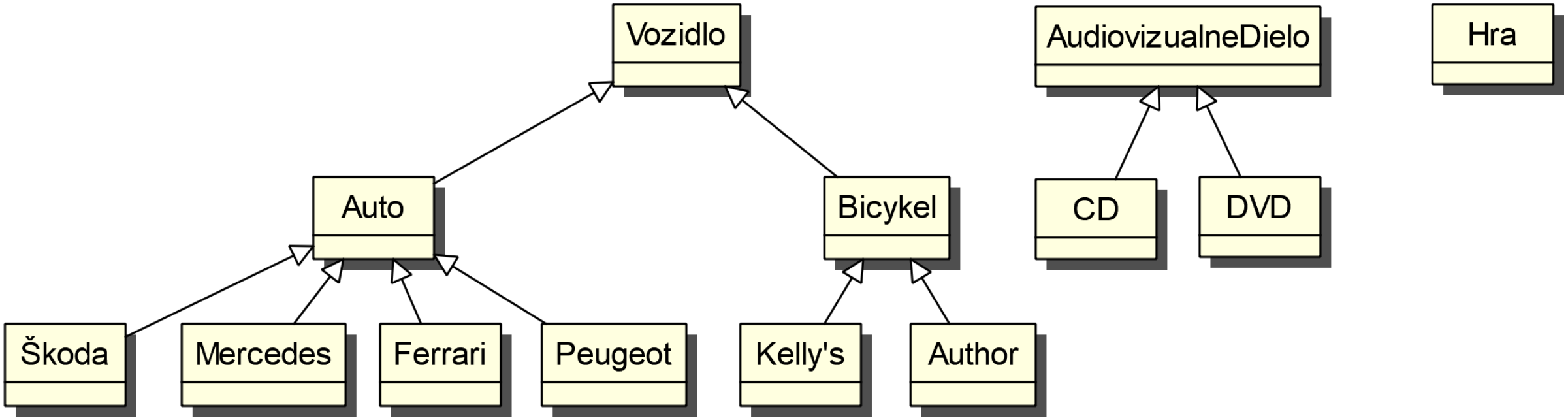
Jednoduchá dedičnosť – Java



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

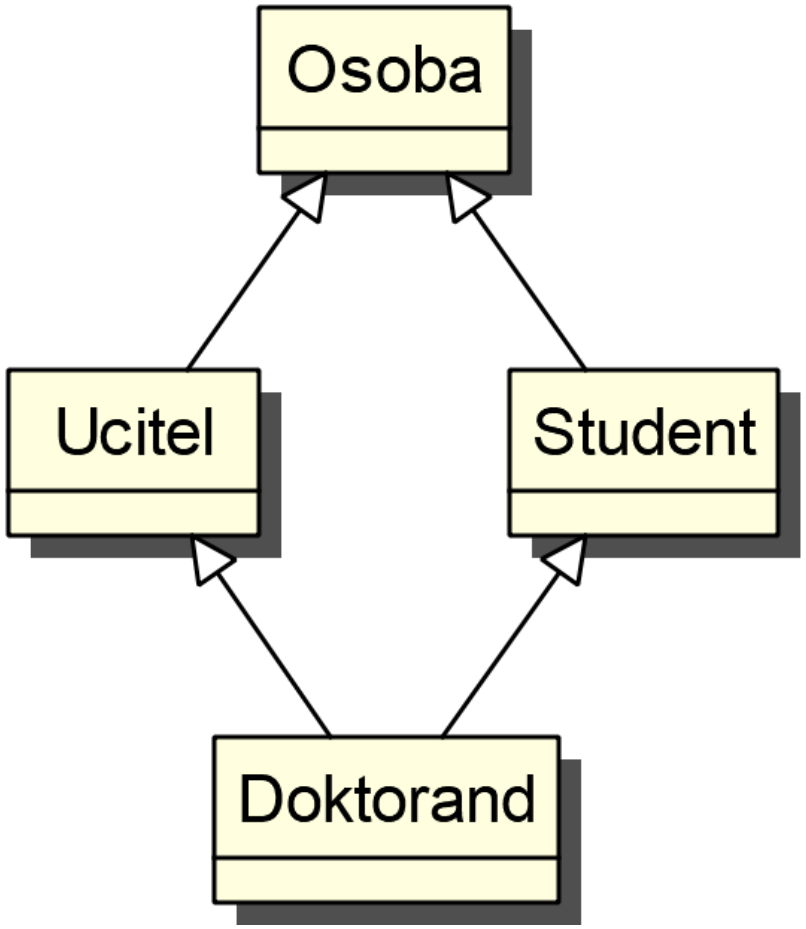


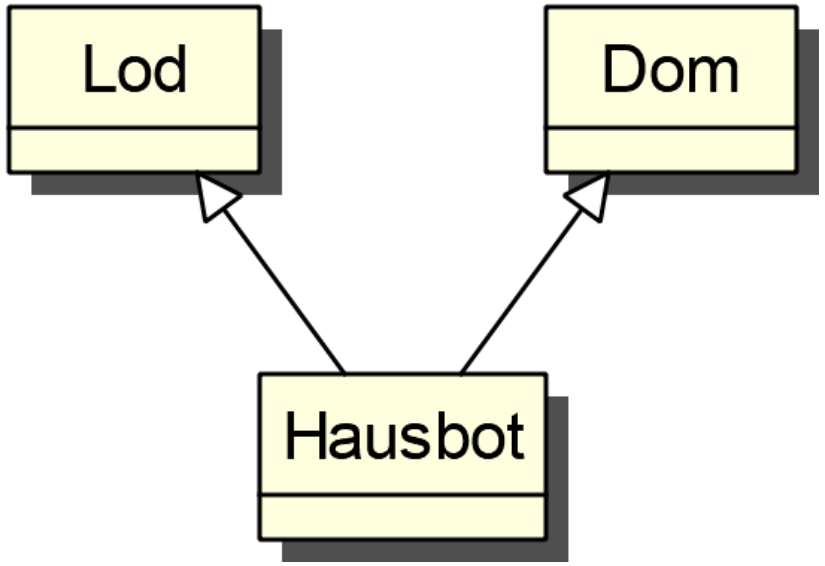
Viacnásobná dedičnosť

- trieda môže mať ľubovoľný počet priamych predkov

- C++, Python, ...

Viacnásobná dedičnosť – C++





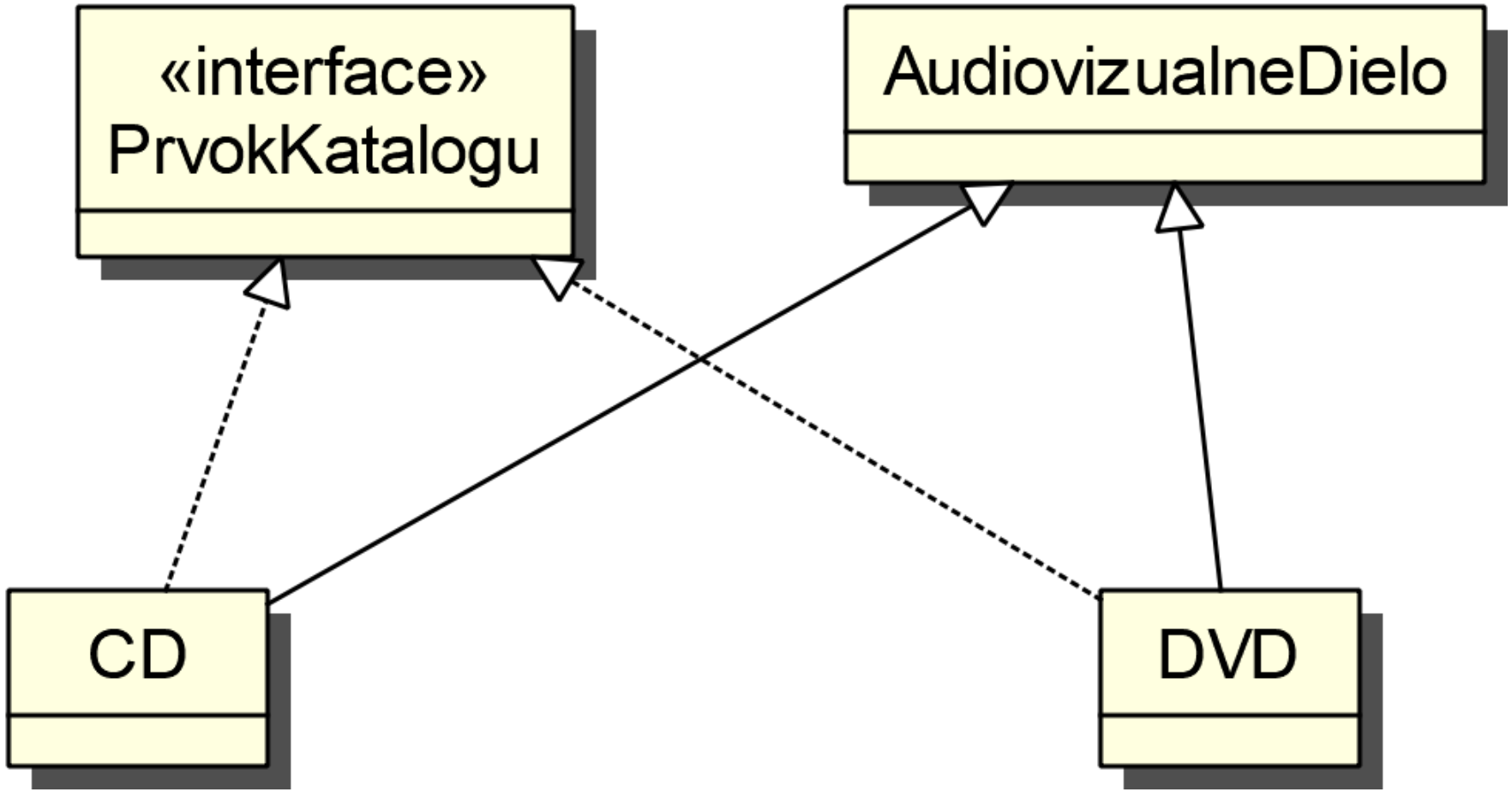
Dedenie implementácie interface

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

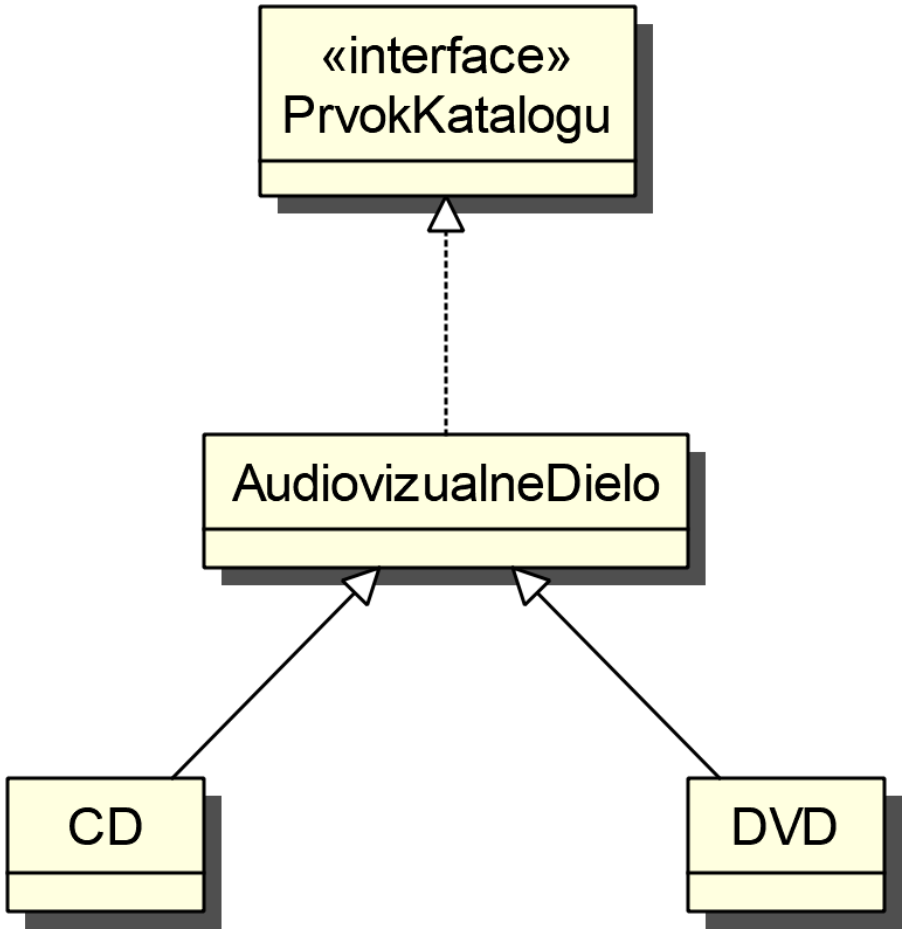
=>

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

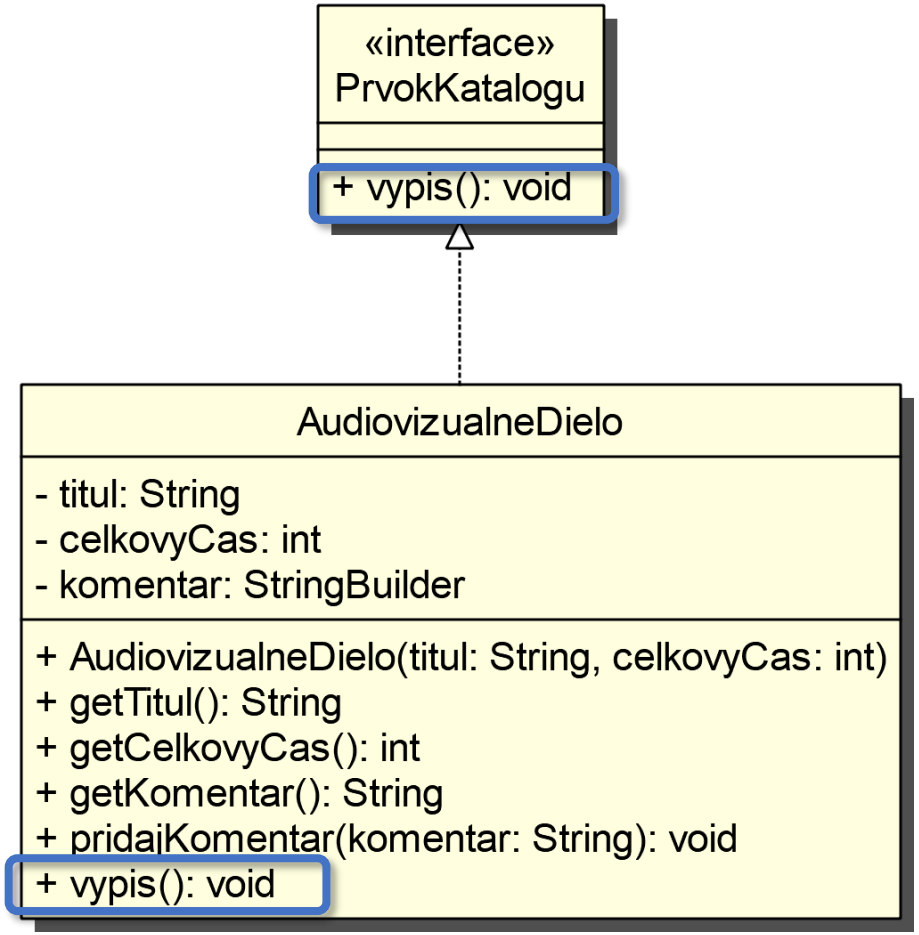
KCaIB – pôvodná verzia



KCaIB – nová verzia (1)



KCaIB – nová verzia(2)



Problém nového riešenia

- AudiovizualneDielo implementuje kompletný interface PrvokKatalogu
  - metóda vypis

- AudiovizualneDielo nepozná režiséra DVD, autora CD a počet skladieb CD
  - nedokáže vypísať – nekompletný výpis

Dedičnosť a typová kompatibilita

- vzťah 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)

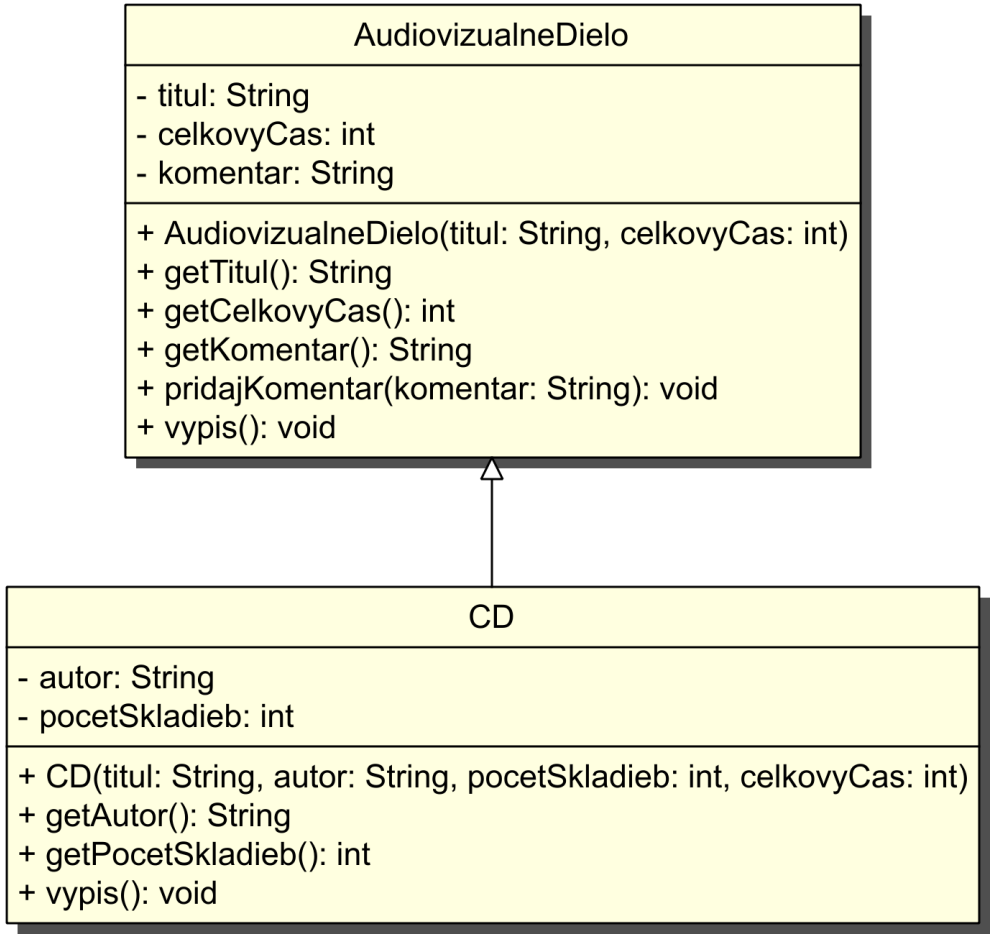
- metódy potomka:
  - zdedené – potomok využíva bez zmeny
  - vlastné – nové metódy, ktoré predok nemá
  - predefinované (prekryté) – predok aj potomok má svoju vlastnú implementáciu – realizujú polymorfizmus
  - POZOR – preťažené metódy potomka = vlastné

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
  - spomeňte na @Override

Polymorfizmus pomocou dedičnosti (1)



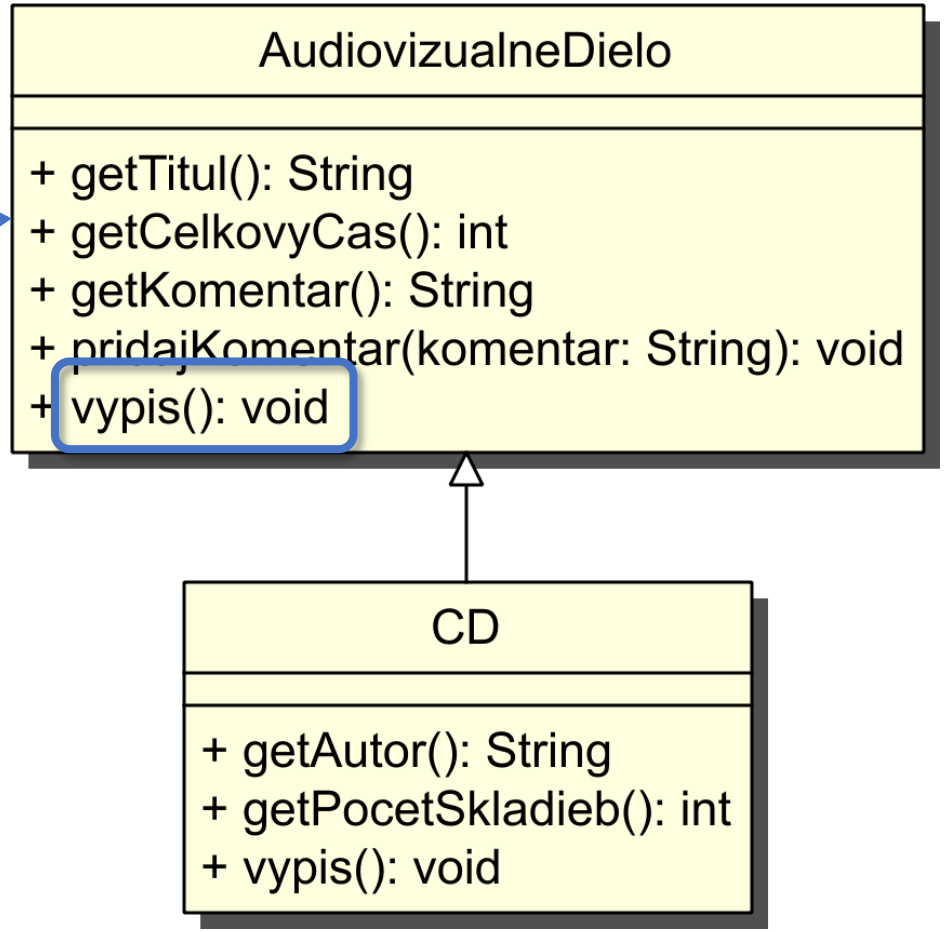
Polymorfizmus pomocou dedičnosti (2)

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


// co vypise?
d.vypis();
```

Poslanie správy, prekladač

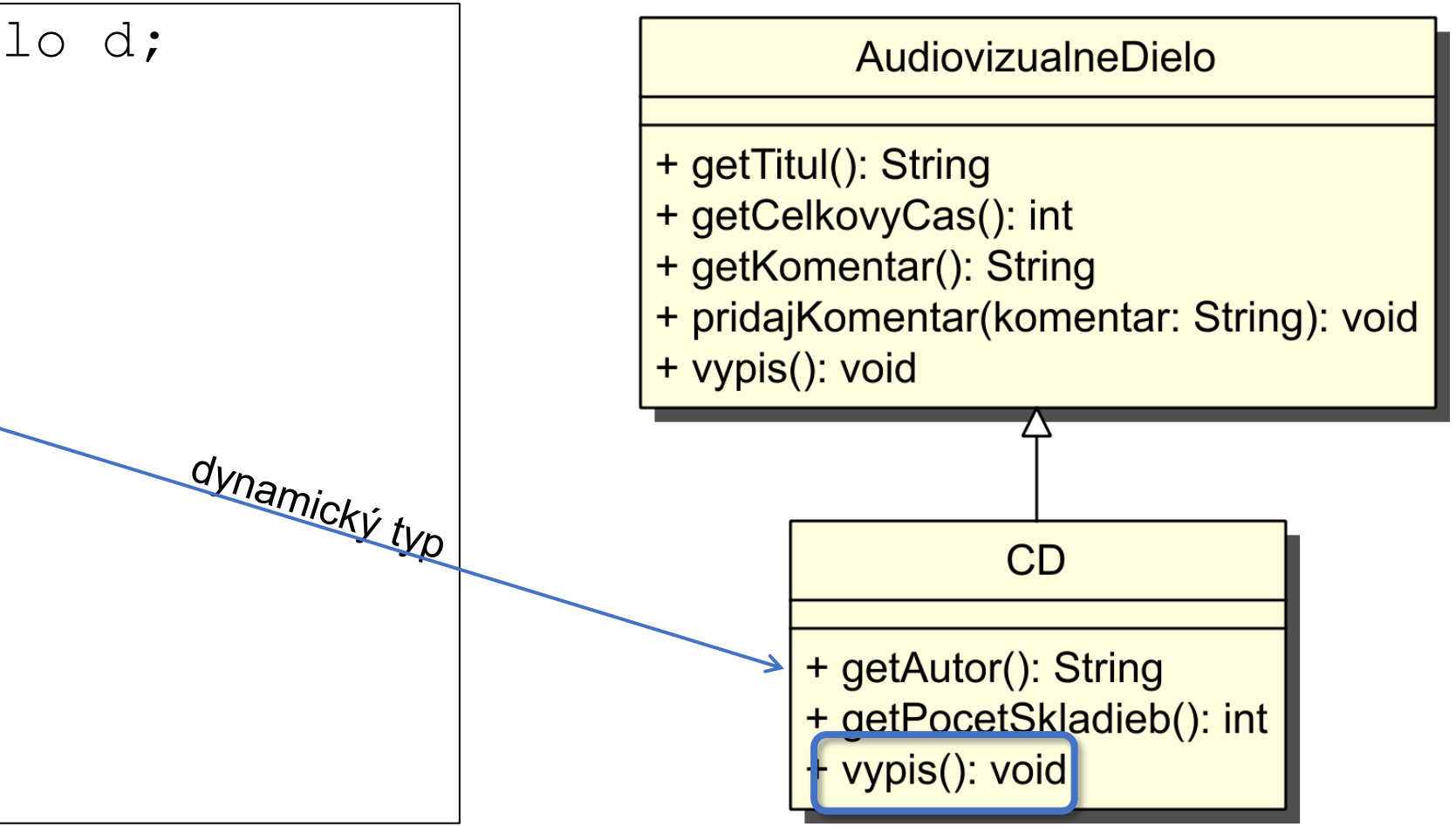
```
AudiovizualneDielo d;
d = new CD(...);

d.vypis();
```

statický typ



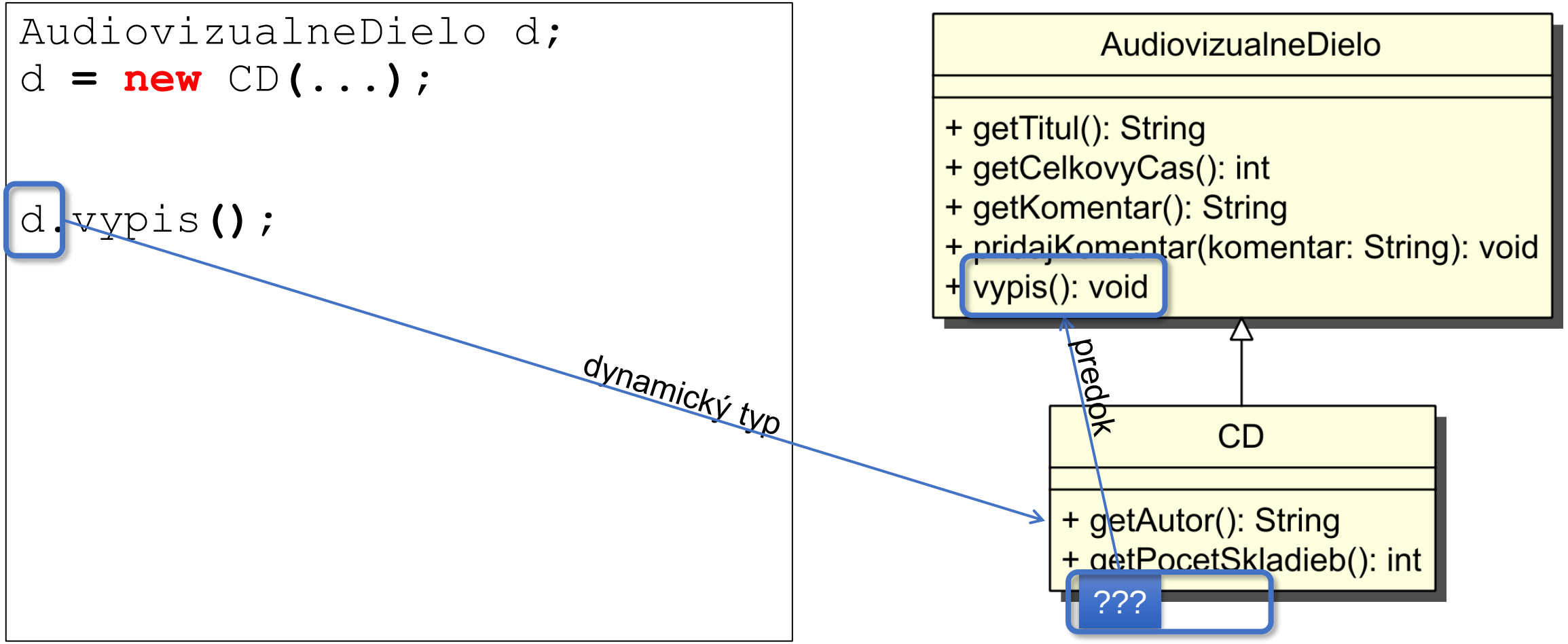
Poslanie správy, vykonanie (1)

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


d.vypis();
```

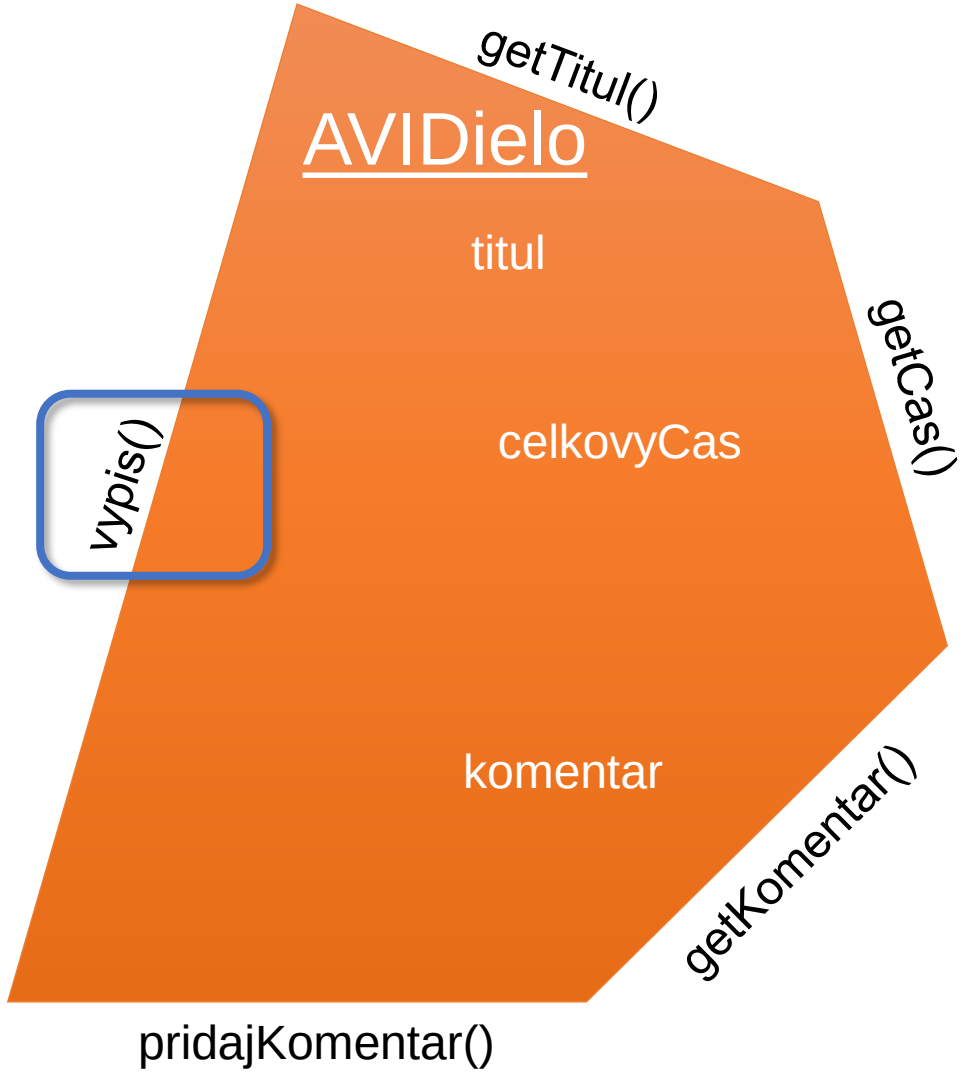


Poslanie správy, vykonanie (2)

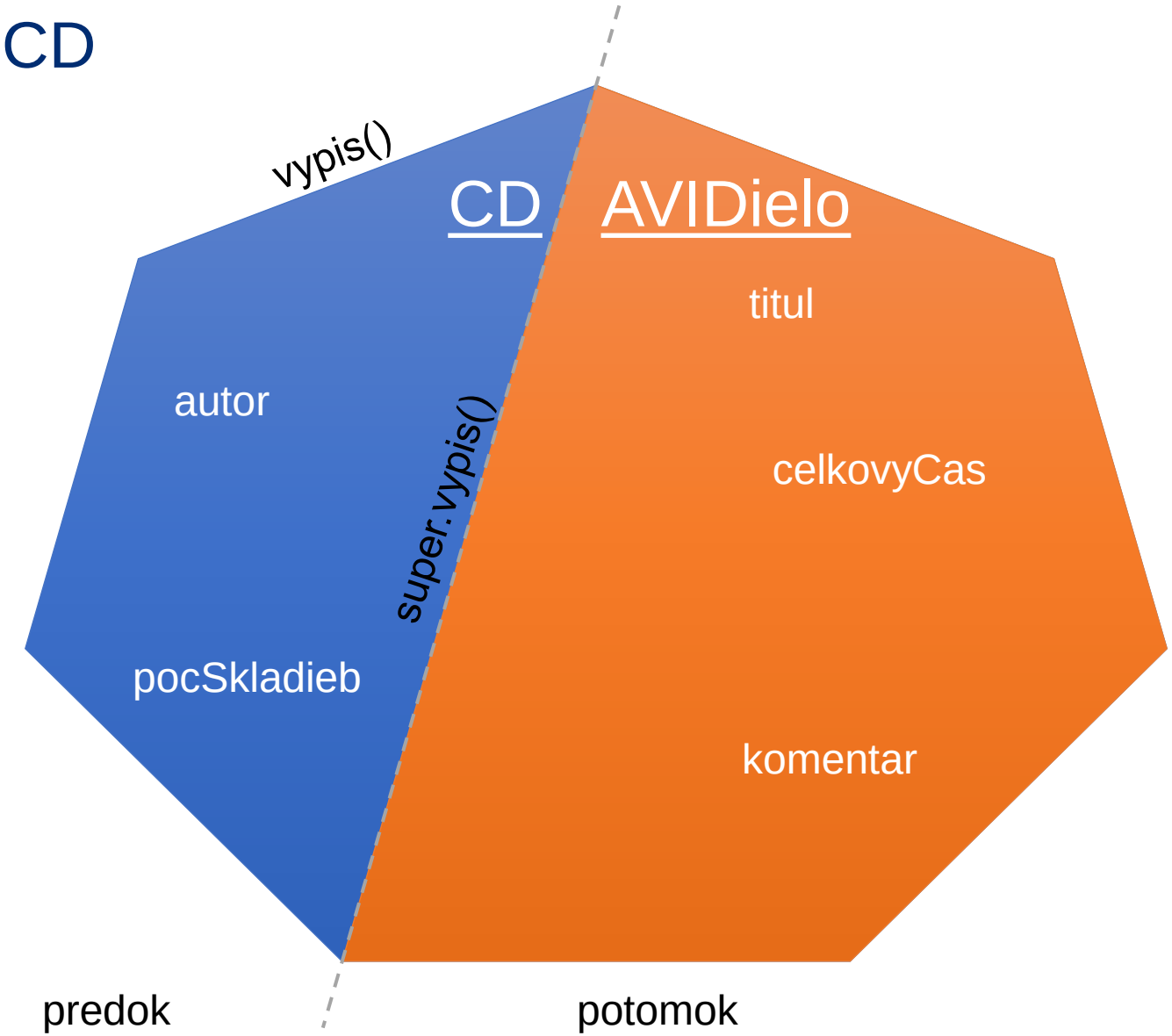
```
AudiovizualneDielo d;
d = new CD(...);


d.vypis();
```



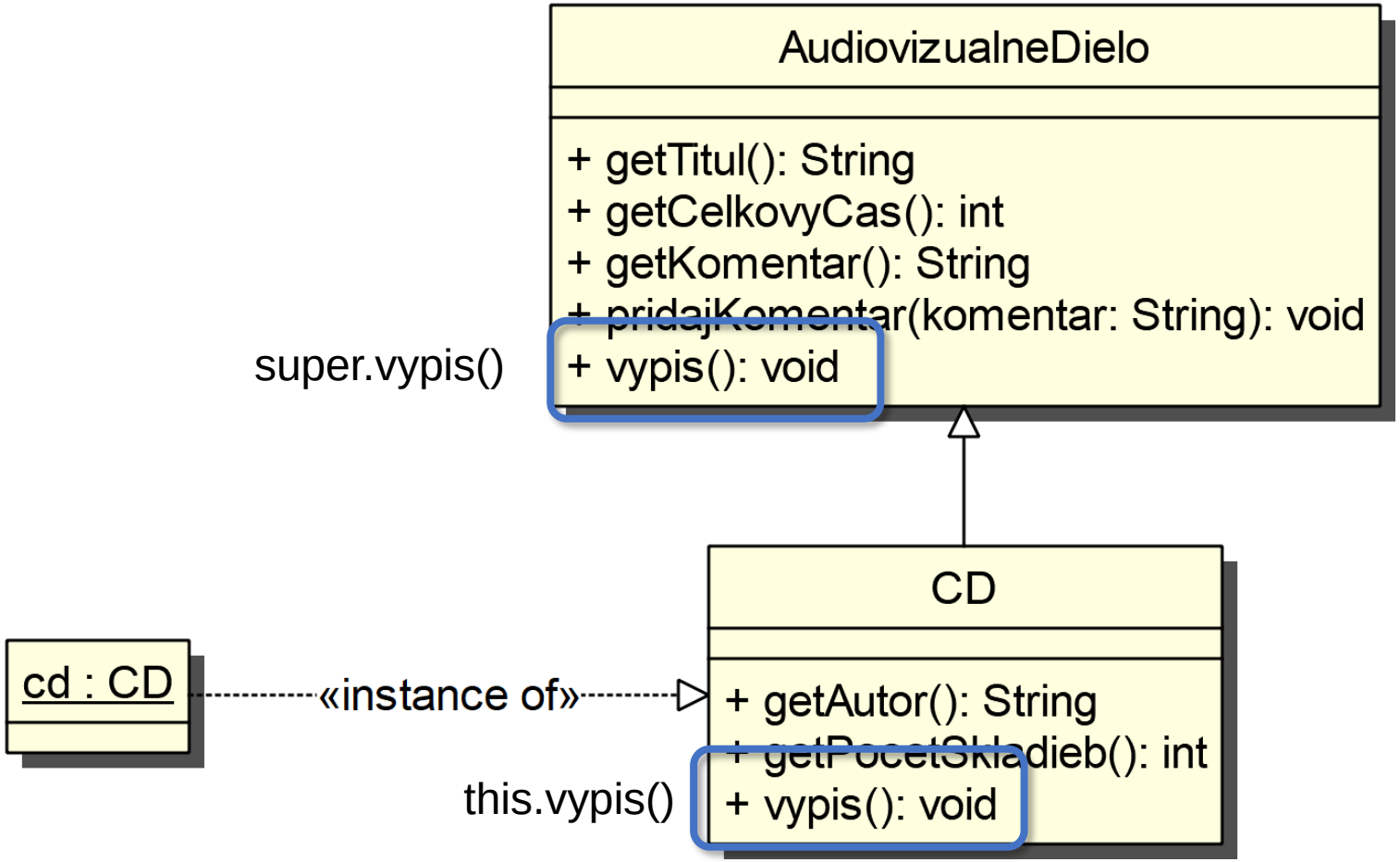
Metóda vypis - AudiovizualneDielo



Metódy vypis - CD



KCaIB: 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(this.titul);
    System.out.println(" (" + this.cas + " min.)");
    if (this.komentar.length() > 0) {
        System.out.println(this.komentar);
    }
}
```

Metóda vypis() – DVD

```
public void vypis() {
    System.out.println("DVD");
    System.out.printl("réžia: " + this.reziser);
    super.vypis();
}
```

Polymorfizmus a jazyky

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