Fakulta riadenia a informatiky ŽU

Výnimky

Pojmy zavedené v 7. prednáške(1)

- Abstraktná trieda
  - v reálnom svete
  - v jazyku Java
  - v BlueJ/UML
- Konkrétna trieda
- Abstraktná metóda
  - v reálnom svete
  - v jazyku Java

Pojmy zavedené v 7. prednáške(2)

- Vzťah is-a
- Dedičnosť a extenzia triedy
- Triáda triedy
  - meno
  - intenzia
  - extenzia
- instanceof v hierarchii dedičnosti

Pojmy zavedené v 7. prednáške(3)

- viac o LSP
- riešenie problému štvorec-obdĺžnik pomocou abstraktnej triedy
- riziká porušenia LSP
  - znižovanie rizika
- trieda Object a LSP

- polymorfizmus vs dedičnosť

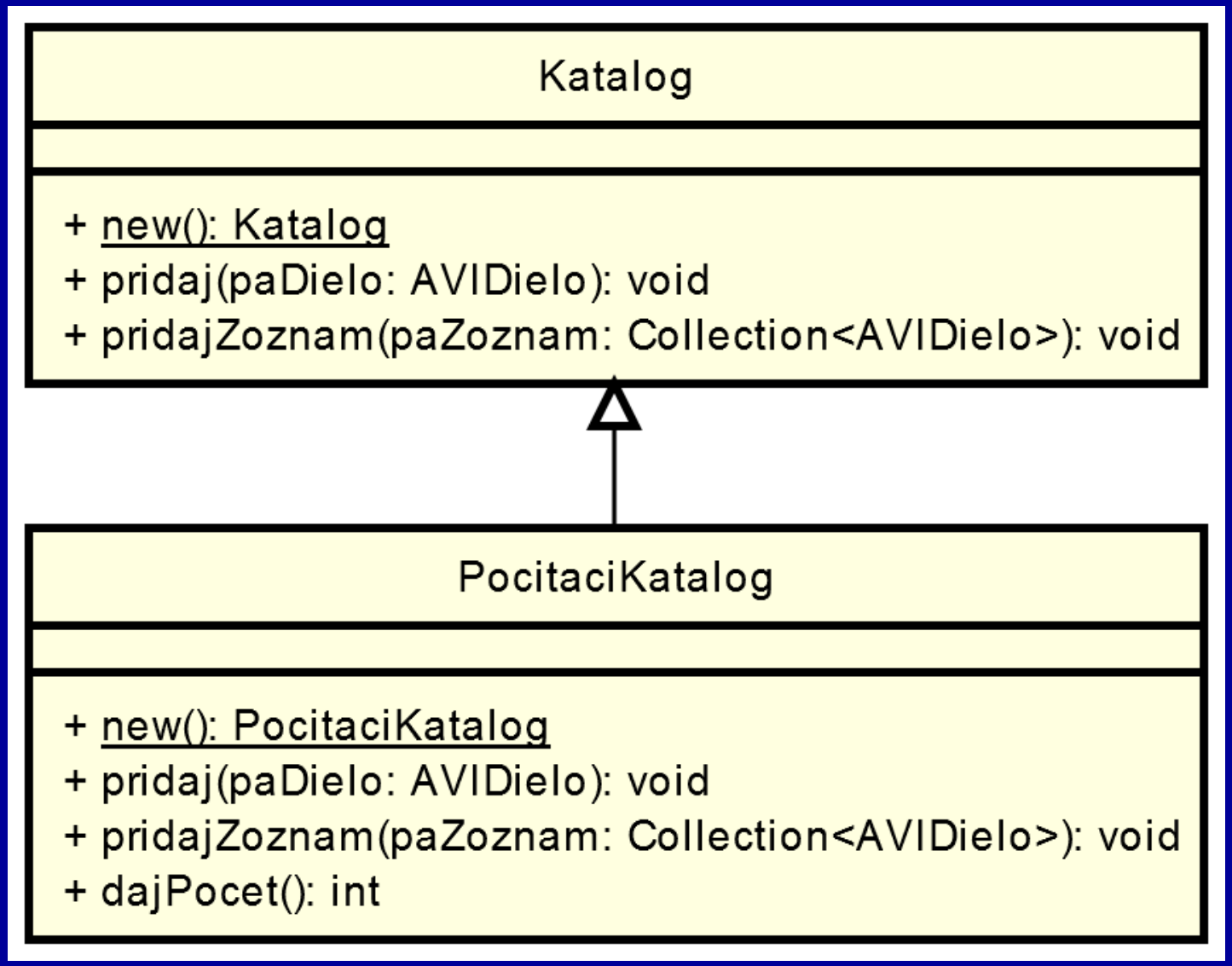
Pojmy zavedené v 7. prednáške(4)

- vytváranie inštancií tried
  - klasická trieda
  - singleton
  - enum
  - abstraktná trieda
- modifikátor final
  - zabránenie prekrytia metódy
  - koncová trieda

Cieľ

- dedičnosť vs. interface
  - nahradenie dedičnosti návrhovým vzorom Dekorátor
  - imitácia interface dedičnosťou
- projekt: KCaIB

Problém(1)



Problém(2)

```
public class PocitaciKatalog extends Katalog
{
    private int aPocet;

    public PocitaciKatalog()
    {
        aPocet = 0;
    }
    ...
}
```

Problém(3)

```
public void pridaj(AVIDielo paDielo) {
    aPocet++;
    super.pridaj(paDielo);
}

public void pridajZoznam
                    (Collection<AVIDielo> paZoznam) {
    aPocet += paZoznam.size();
    super.pridajZoznam(paZoznam);
}
```

PocitaciKatalog – fungovanie

- pridaj – funguje podľa očakávania
- pridajZoznam – funguje nesprávne
  - pridá dvojnásobný počet prvkov z paZoznam

- Kde je chyba?

Problém(4)

```
public void pridaj(AVIDielo paDielo) {
    aDiela.add(paDielo);
}

public void pridajZoznam
                        (Collection<AVIDielo> paZoznam) {
    for (AVIDielo dielo : paZoznam) {
        this.pridaj(dielo);
    }
}
```

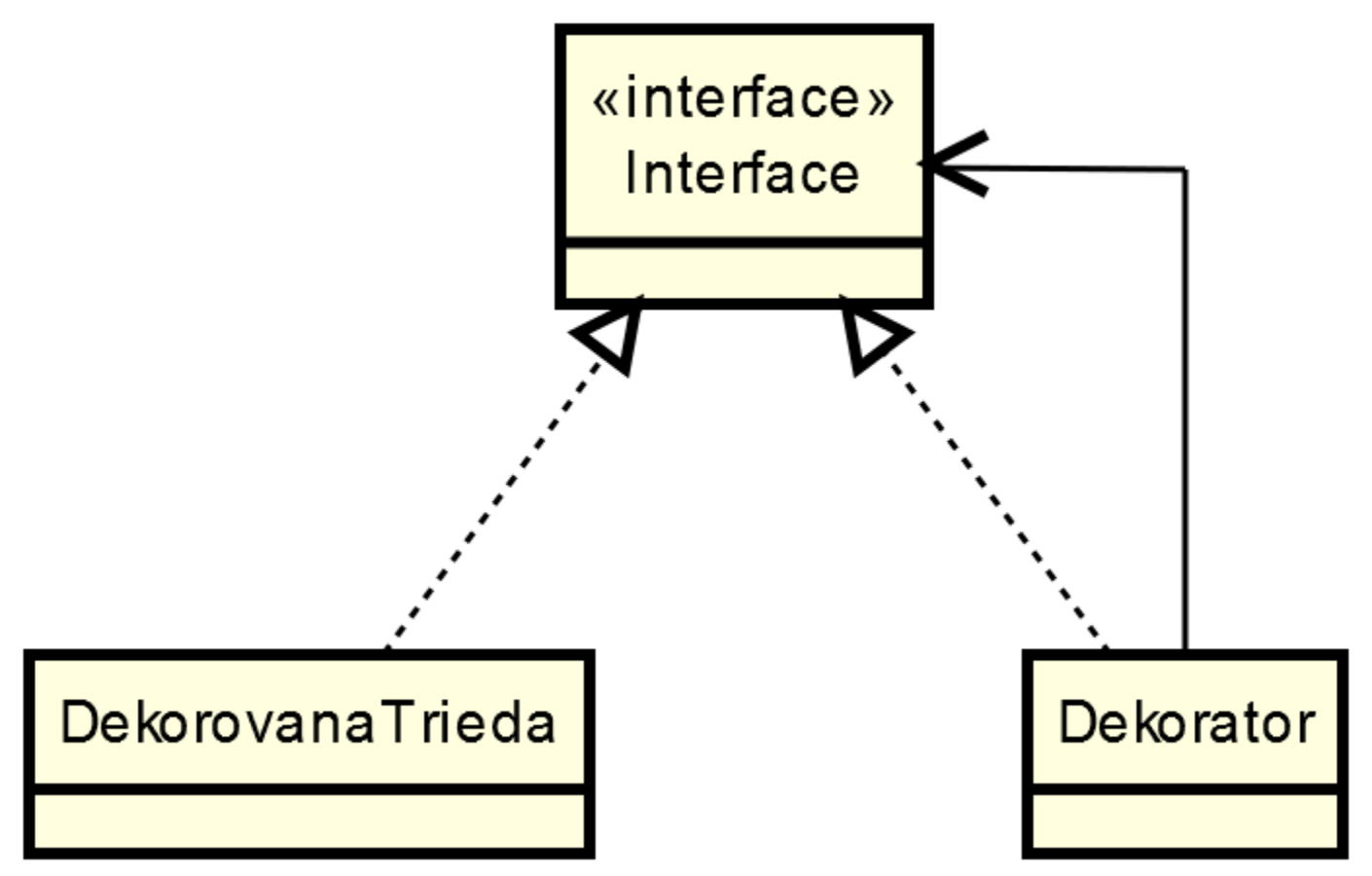
Riešenia

- upraviť predka – riešenie1
  - odstrániť polymorfizmus v pridajZoznam
    - this.pridaj(dielo) - problém
    - aDiela.add(dielo) – riešenie
- upraviť potomka so znalosťou predka
  - neprekryť metódu pridajZoznam – riešenie 2
- odstrániť dedičnosť – riešenie 3

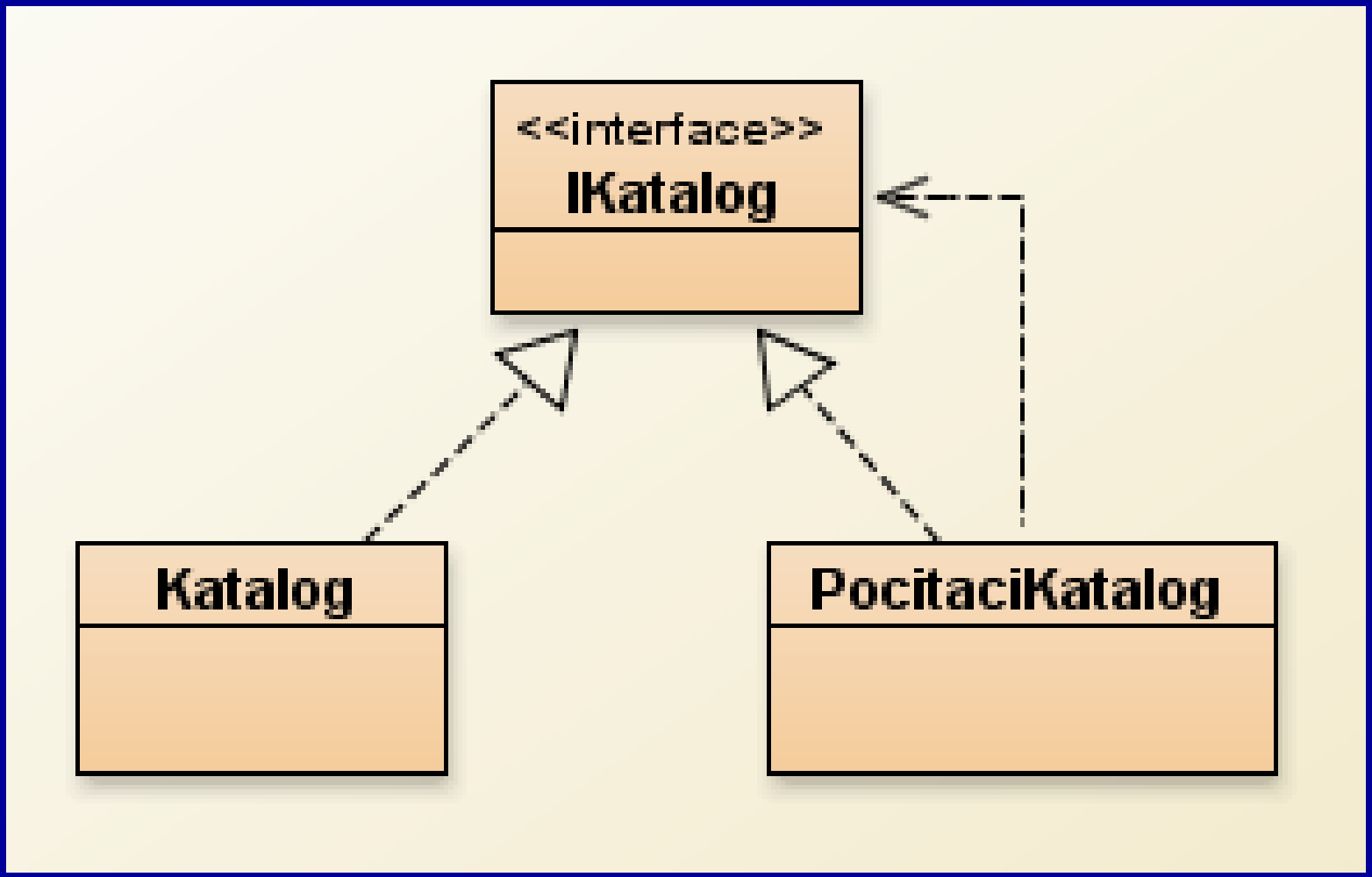
Rozšírenie funkcionality objektu

- dedičnosť
  - vytvorenie podtypu
- návrhový vzor Dekorátor
  - zabalenie objektu – skladanie
  - delegovanie časti služieb na zabalený objekt

Návrhový vzor Dekorátor



Aplikácia na KCalB



Riešenie cez Dekorátor(1)

```
public class PocitaciKatalog implements IKatalog {
    private int aPocet;
    private IKatalog aZabaleny;

    public PocitaciKatalog(IKatalog paZabaleny) {
        aPocet = 0;
        aZabaleny = paZabaleny;
    }
    ...
}
```

Riešenie cez Dekorátor(2)

```
public void pridaj(AVIDielo paDielo) {
    aPocet++;
    aZabaleny.pridaj(paDielo);
}

public void pridajZoznam
                    (Collection<AVIDielo> paZoznam) {
    aPocet += paZoznam.size();
    aZabaleny.pridajZoznam(paZoznam);
}
```

Riešenie cez Dekorátor(3)

```
public void vypisPolozky()
{
    aZabaleny.vypisPolozky();
}
```

Využitie katalógu cez dekorátor

```
IKatalog katalog = new Katalog();
```

```
IKatalog katalog = new PocitaciKatalog(
                                    new Katalog());
```

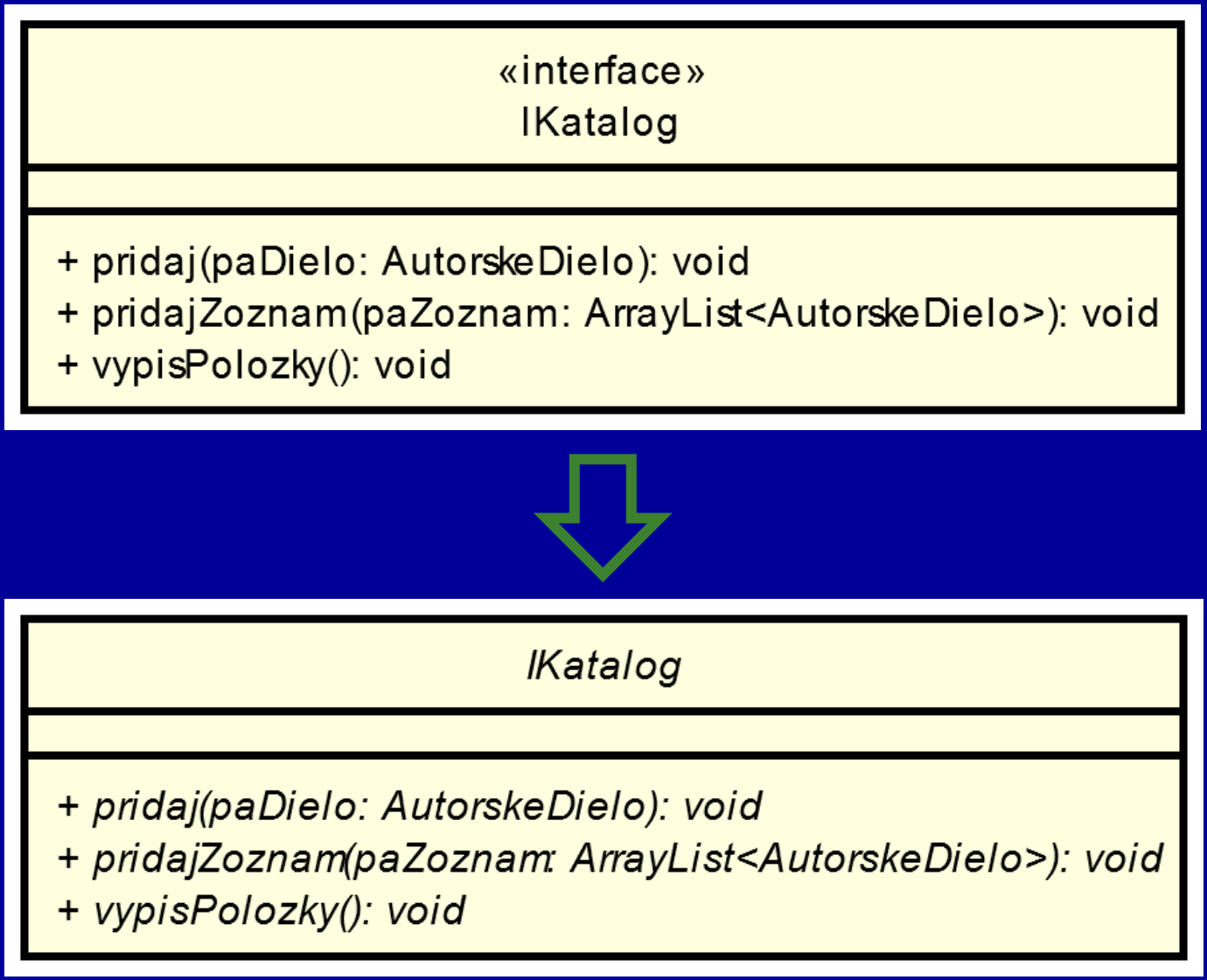
Dekorátor – pre a proti

- Výhody
  - nenastáva polymorfizmus cez this
  - triedy nie sú naviazané – ich vzťah možno kedykoľvek meniť
    - minimálna až žiadna závislosť
- Nevýhody
  - nenastáva polymorfizmus cez this
  - chýba reuse rozhrania
    - kúsok ukecanejšie

Imitácia Interface

- jazyky bez konceptu Interface
  - napr. C++
- polymorfizmus viazaný na dedičnosť
- nahrádzanie abstraktnou triedou
  - čisto abstraktná trieda

Imitácia Interface abstraktnou triedou



Fakulta riadenia a informatiky ŽU

Ošetrovanie chybových stavov

Cieľ

- komunikácia klient-server
- defenzívne programovanie
- ošetrenie chýb
  - na strane servera
  - na strane klienta
- výnimky

- projekt: KCaIB

Behové chyby(1)

- chyby v kóde, ktoré spôsobia pád aplikácie
  - nekorektné ukončenie
  - aplikácia sa vymkne spod kontroly
  - nečakané ukončenie v lepšom prípade

Behové chyby(2)

- Má padnúť internetový prehliadač, ak žiadame otvoriť neexistujúcu web stránku?
  - (Napr. sme pri písaní adresy urobili chybu.)
- Má sa zavrieť (ukončiť prácu) súborový manažér, ak chceme uložiť obrázok na USB kľúč, na ktorom už nie je dostatok miesta?
- Má skončiť hra, ak hráč napíše nesprávny príkaz?

Behové chyby(3)

- Problémy tohto typu chceme riešiť.

Komunikácia klient-server(1)

- komunikácia dvojice objektov – odosielateľ správou žiada adresáta o službu

- odosielateľ – klient žiada o službu
- klient je aktívny objekt

- adresát – server poskytuje službu
- server nevykonáva žiadnu akciu z vlastnej vôle, len reakcia na žiadosť klienta

Komunikácia klient-server(2)

- správa – parametre
- správnosť parametrov – prípustné hodnoty

- kto je zodpovedný?

- klient – definícia hodnôt parametrov
- server – použitie hodnôt parametrov

Klient

- klient – definícia hodnôt parametrov

- doplňujúce informácie správy
- podrobnejšia charakteristika žiadosti
- klient vychádza zo svojho stavu
- možná behová chyba

- negatívne testovanie – zámerná behová chyba

Server

- server – použitie hodnôt parametrov
- parametre sú charakterizované typom
- typ je abstrakcia hodnôt
- nie každá hodnota musí byť prípustná
- určitá hodnota parametra môže vyžadovať určitý stav servera
- parametre môžu vyžadovať zmenu stavu servera

- server nesmie dopustiť, aby sa dostal do nekorektného stavu

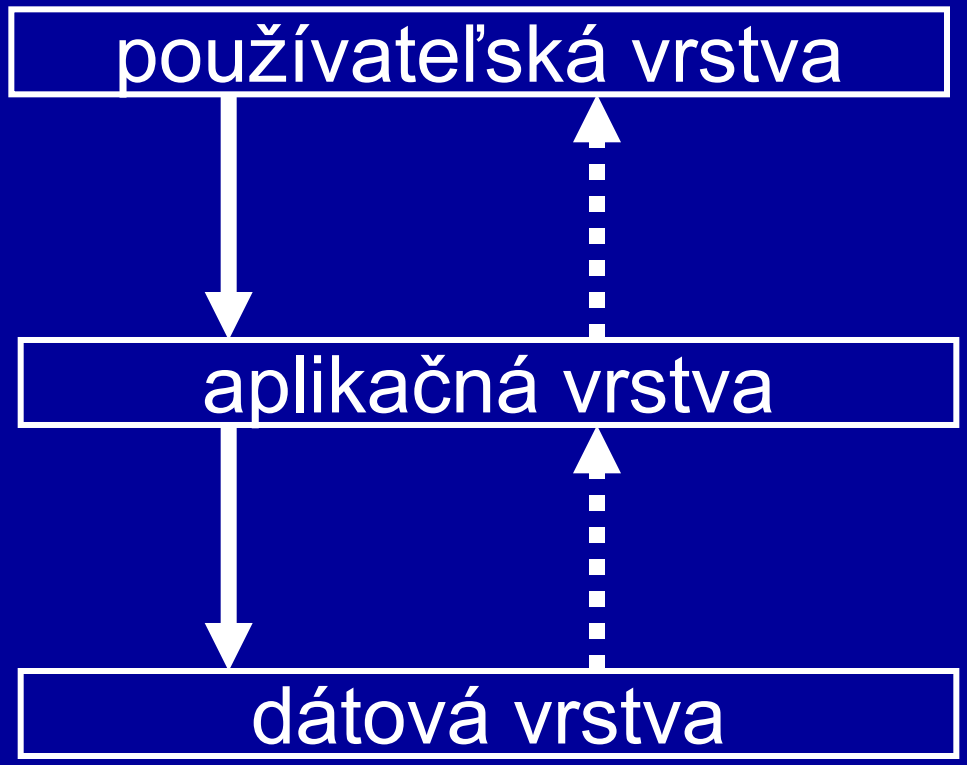
Trojvrstvový model aplikácie(1)

- tri skupiny – vrstvy – objektov aplikácie

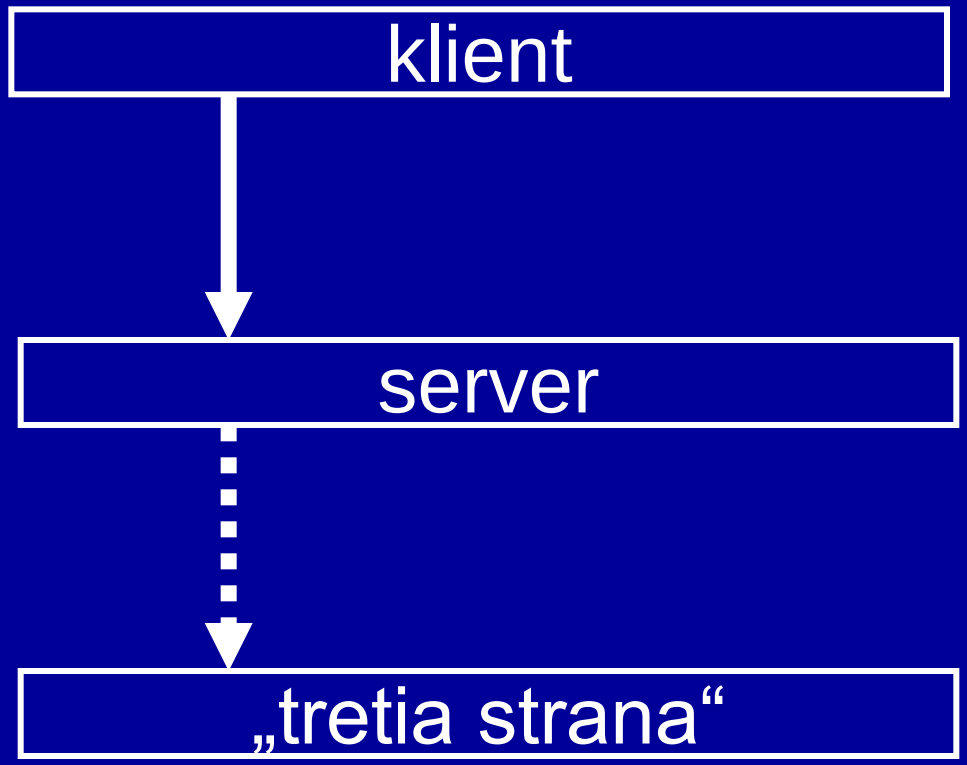
- aplikačná (biznis) vrstva – objekty logiky aplikácie
- používateľská vrstva (GUI, TUI) – komunikácia človek – aplikácia, preberanie vstupných údajov, prezentácia výsledkov
- dátová vrstva – uloženie dát na opätovné použitie

Trojvrstvový model aplikácie(2)

- tri vrstvy sú voči sebe vo vzťahu klient – server



Porušenie vzťahu klient-server



Komunikácia klient-server

- kto je zodpovedný?
- extrémne spôsoby riešenia
- klient vie presne, čo chce a ako to má od servera korektne žiadať
- server predpokladá, že sa môže stať čokoľvek a musí sám zabezpečiť, aby pracoval len korektne

Východiská

- klient nekladie nekorektné požiadavky zámerne
- existujúce nekorektné požiadavky – chyby
- existujú problémy mimo aplikácie – klient nemôže poznať

Riešenie – odpovede na otázky

- Aká veľká kontrola správ od klienta má byť v metódach servera?
- Ako má server oznamovať chyby klientovi?
- Ako má klient predchádzať nekorektným požiadavkám na server?
- Ako má klient reagovať na oznámenie o chybe

Defenzívne programovanie

- kontrola parametrov – server
- kontrola návratovej hodnoty – klient

Riešenie na strane servera

- kontrola parametrov
- reakcia na zistenú chybu
  - nevykoná metódu
  - nezmení svoj stav
- informovanie o chybe
  - žiadne - ??? (urazená osoba)
  - používateľa - výpis
  - klienta

Príklad – Katalog.vymaz(1)

```
public void vymaz(String paTitul)
{
    AVIDielo polozka = this.vyhladaj(paTitul);
    this.aZoznamPoloziek.remove(polozka);
}
```

- `paTitul`
  - nesmie byť `null`
  - nesmie byť prázdny reťazec
  - musí existovať AVIDielo s takým titulom

Príklad – Katalog.vymaz(2)

```
public void vymaz(String paTitul)
{
  if (paTitul == null || paTitul.isEmpty()) {
    // osetrenie chyby 1,2
  } else {
  ...
```

Príklad – Katalog.vymaz(3)

```
  ...
  AVIDielo polozka = this.vyhladaj(paTitul);
  if (polozka == null) {
    // Osetrenie chyby 3
  } else {
    this.aZoznamPoloziek.remove(polozka);
  }
}
```

Informovanie používateľa

- informácia o chybe sa zobrazí používateľovi

- textová forma – terminál, okno
- zvukový výstup – „pípanie“ pri stlačení nesprávneho klávesu
- ...

- doteraz v projektoch – výpis na terminál

Príklad – Katalog.vymaz(4)

```
public void vymaz(String paTitul)
{
  if (paTitul == null || paTitul.isEmpty()) {
    System.out.println("Titul musi byt zadany!");
  } else {
  ...
```

Príklad – Katalog.vymaz(5)

```
  AVIDielo polozka = this.vyhladaj(paTitul);
  if (polozka == null) {
    System.out.println("Titul nie je v katalogu!");
  } else {
    this.aZoznamPoloziek.remove(polozka);
  }
}
```

Informovanie klienta

- dve možnosti
  - návratová hodnota správy
  - výnimky

návratová hodnota

- kód chyby

- `boolean` – `false` (nastala chyba), `true` (nenastala)
- `int` – číselný kód (1 – taká chyba; 2 – iná chyba; 0 – bez chyby)
- objekt – `null` v prípade chyby (napr. pri vyhľadávaní)
- enum – položky – rôzne typy chýb

Návratová hodnota objekt(1)

```
public AudiovizualneDielo vyhladaj(String paTitul)
{
   for (AVIDielo polozka : aZoznamPoloziek) {
      if (polozka.dajTitul().equals(paTitul)) {
         return polozka;
      }
   }
   return null;
}
```

ok

chyba

Návratová hodnota objekt(2)

- štandardné riešenie pri vyhľadávaní
- hľadaný objekt neexistuje => `null`

Návratová hodnota boolean(1)

```
public boolean vymaz(String paTitul)
{
  if (paTitul == null || paTitul.isEmpty()) {
    return false;
  } else {
  ...
```

Návratová hodnota boolean(2)

```
  AVIDielo polozka = this.vyhladaj(paTitul);
  if (polozka == null) {
    return false;
  } else {
    this.aZoznamPoloziek.remove(polozka);
    return true;
  }
}
```

Návratová hodnota boolean(3)

- jediná informácia – nastala chyba
- vhodné pre jedinú možnosť
- nevhodné pre viac rôznych chýb v metóde – ak existuje viac možností ich ošetrenia

Návratová hodnota enum(1)

```
public KodChybyVymazania vymaz(String paTitul)
{
  if (paTitul == null || paTitul.isEmpty()) {
    return KodChybyVymazania.titulNezadany;
  } else {
  ...
```

Návratová hodnota enum(2)

```
  AVIDielo polozka = this.vyhladaj(paTitul);
  if (polozka == null) {
    return KodChybyVymazania.nenasielSa;
  } else {
    this.aZoznamPoloziek.remove(polozka);
    return KodChybyVymazania.ok;
  }
}
```

Výnimky(1)

- kontrola výnimočných stavov
- oznamovanie chyby klientovi
- vo väčšine prípadov preferované riešenie

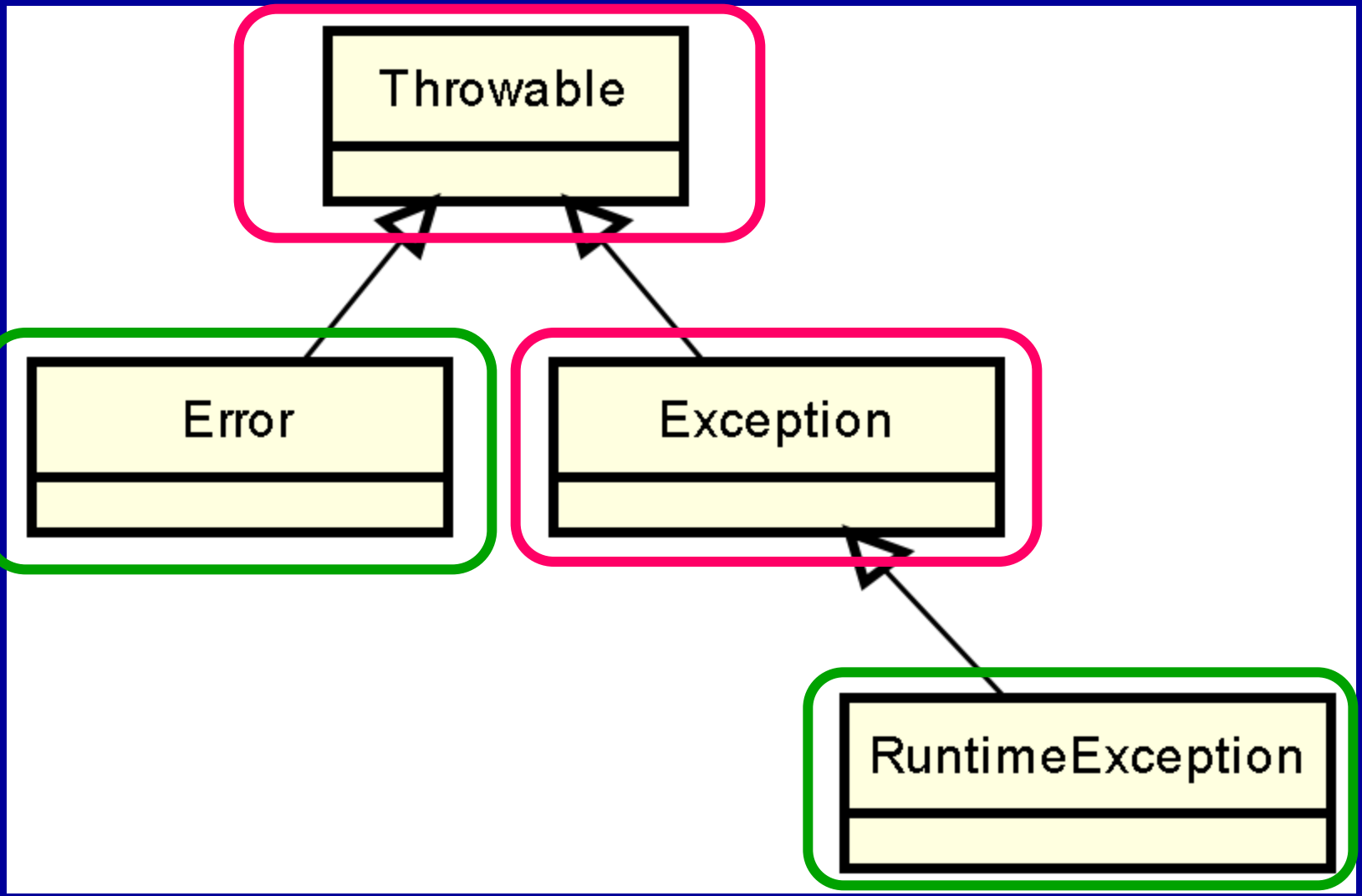
Výnimky(2)

- výnimka – objekt
- informácia – názov triedy výnimky
- informácia – popis situácie

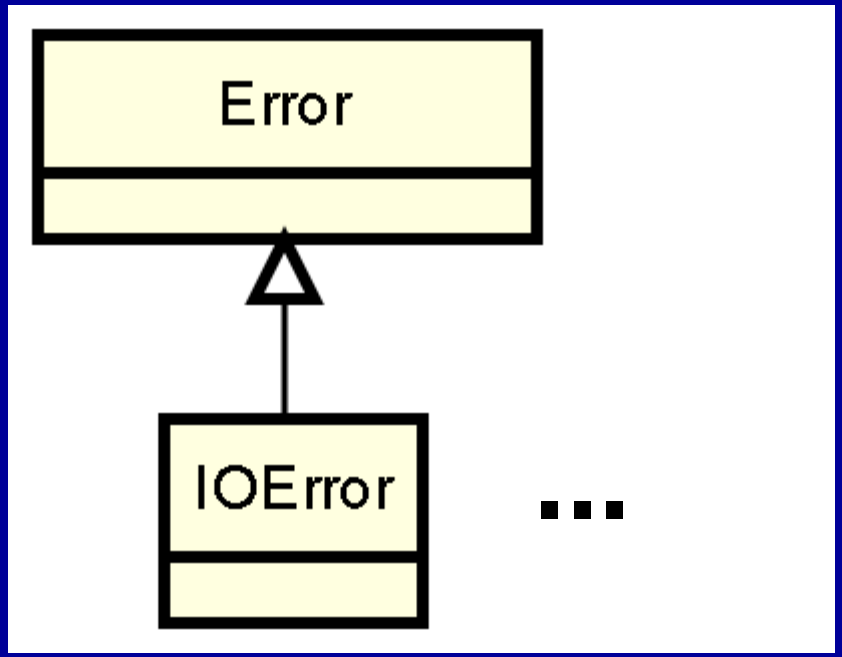
Druhy výnimiek

- kontrolované
  - klient nemôže ignorovať
  - kontroluje prekladač
- nekontrolované
  - klient môže ignorovať
  - spôsobí predčasné ukončenie aplikácie – pád
  - prekladač nerieši

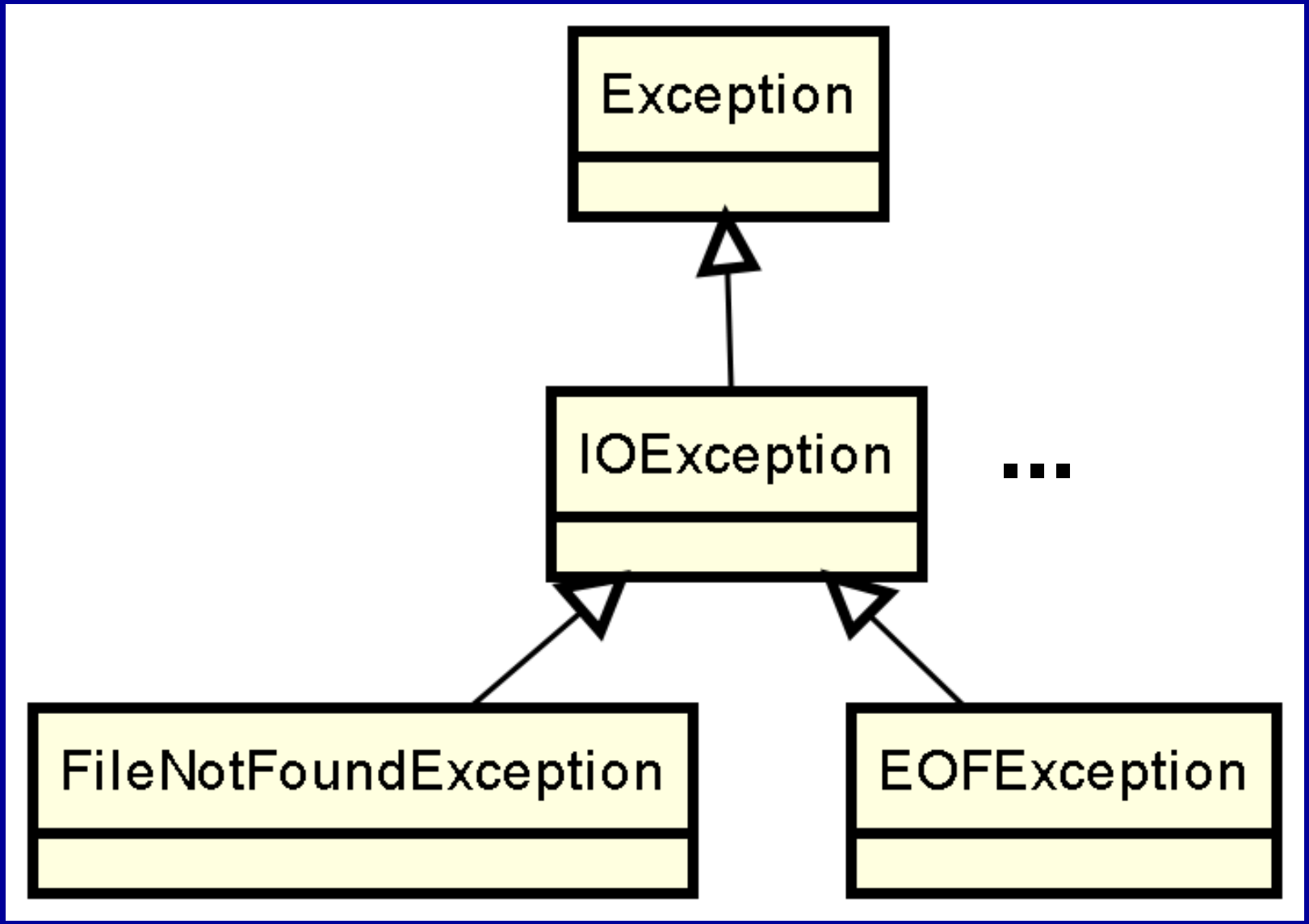
Hierarchia výnimiek – koreň Throwable



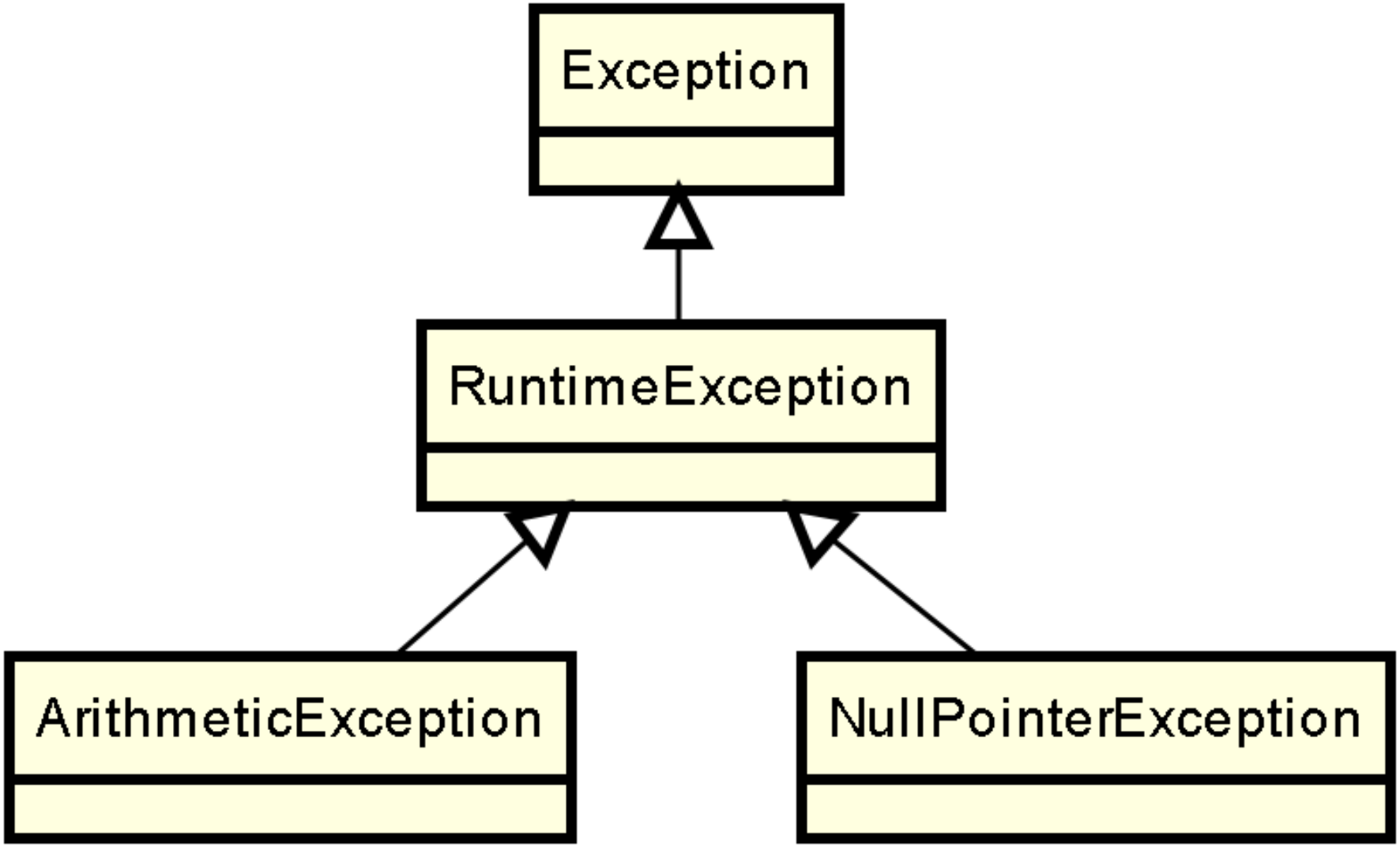
Chyby



Kontrolované výnimky



Nekontrolované výnimky



Výnimky a klient-server

- server výnimky vytvára – „vyhadzuje“
- klient výnimky zachytáva

Vyhodenie výnimky(1)

- príkaz `throw`

```
TypVynimky vynimka = new TypVynimky("popis");
throw vynimka;
```

- skrátený – bežný zápis

```
throw new TypVynimky("popis");
```

Vyhodenie výnimky(2)

- príkaz `throw` ukončuje vykonávanie metódy
- server vracia riadenie klientovi
- informuje klienta o výnimke

Klauzula throws

- `throws` v hlavičke metódy – zoznam výnimiek vyhadzovaných v metóde
  - kontrolované povinne
  - nekontrolované fakultatívne

```
public void uloz(String paNazovSuboru)
                                    throws IOException
```

Priklad – vyhodenie výnimky(1)

```
public void vymaz(String paTitul)
                          throws IllegalParameterException
{
  if (paTitul == null || paTitul.isEmpty()) {
    throw new IllegalParameterException(
                                        "paTitul je prazdny");
  } else {
    ...
```

Priklad – vyhodenie výnimky(2)

```
  AVIDielo polozka = this.vyhladaj(paTitul);
  if (polozka == null) {
    throw new IllegalParameterException(
                                        "Titul sa nenasiel");
  } else {
    this.aZoznamPoloziek.remove(polozka);
  }
}
```

Zachytávanie výnimky

- na strane klienta
- definujeme akcie, ktoré sa vykonajú v prípade výnimky

Príkaz try-catch

- bloky: `try`, `catch`
- samostatne `catch` pre každú triedu výnimiek

```
try {
    // chranene prikazy
} catch (TriedaVynimky vynimka) {
    // prikazy vykonane v pripade vynimky
}
```

Príkaz try-catch

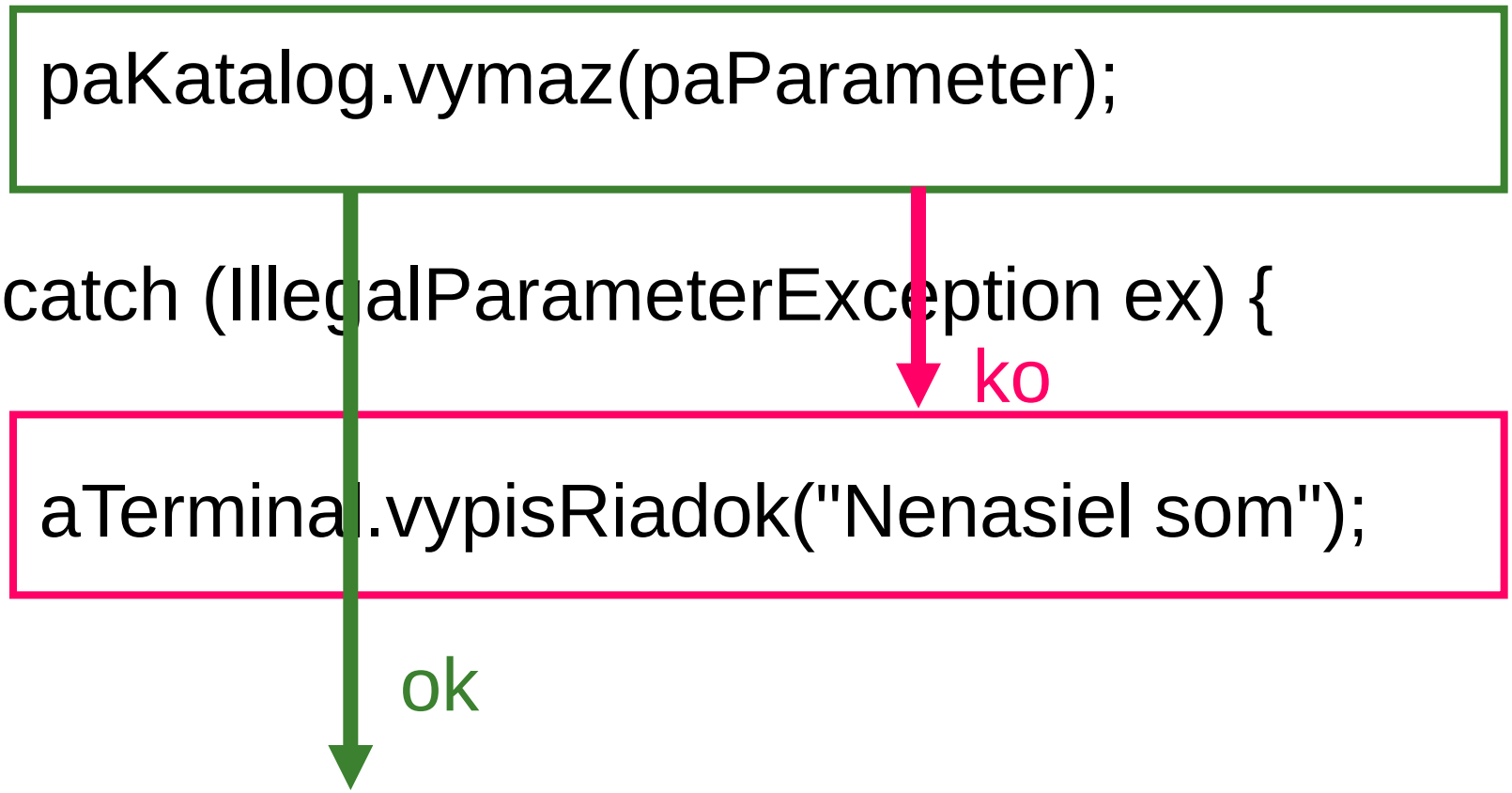
```
try {

    paKatalog.vymaz(paParameter);

} catch (IllegalParameterException ex) {

    aTerminal.vypisRiadok("Nenasiel som");

}
```



Viac blokov catch

- postupné vyhľadávanie typu výnimky
- POZOR! len prvý vhodný blok `catch`

- bloky `catch` musia byť usporiadané
- najskôr potomok – potom predok

Ošetrenie výnimky v bloku catch

- oznámenie používateľovi
- postúpenie výnimky ďalej
  - `throw ex`

Príkaz try-catch-finally

- blok `finally` – príkazy vykonávané vždy
  - ok – všetky príkazy bloku `try` + blok `finally`
  - ko – blok `catch` (ak taký je) + blok `finally`
- uzatvorenie systémových zdrojov – súbory

- verzia `try-finally`
- príkazy `try` môžu byť vnorené – `catch`, `finally`

Ďakujem za pozornosť