Informatika 2

Výnimky – dokončenie

KST Katedra softvérových technológií
Fakulta riadenia a informatiky, Žilinská univerzita v Žiline

Ludum Dare

- vytvorte hru za 48 hodín
- http://www.ludumdare.com/compo/
- 23. konanie súťaže
  - 20.-23. Apríla

Pojmy zavedené v 8. prednáške(1)

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

Pojmy zavedené v 8. prednáške(2)

- behové chyby
- komunikácia klient-server
- trojvrstvový model aplikácie

Pojmy zavedené v 8. prednáške(3)

- defenzívna programovanie
- server - informovanie o chybách
- používateľa
- klienta
  - návratová hodnota
  - výnimka

Pojmy zavedené v 8. prednáške(4)

- výnimky - hierarchia
- druhy
  - Error
  - Exception
  - RuntimeException

- kontrolované výnimky
- nekontrolované výnimky

Pojmy zavedené v 8. prednáške(5)

- vyhadzovanie výnimiek
- príkaz throw
- klauzula throws

Pojmy zavedené v 8. prednáške(6)

- zachytávanie výnimiek
- prikaz try
- try-catch
- try-catch-finaly
- try-finally

Cieľ prednášky

- výnimky
  - posunutie, vlastné
  - príkaz assert
  - zotavenie sa po chybách
  - predchádzanie chybám
- vstupy a výstupy
  - štandardný vstup a výstup
  - textové súbory
  - objektové prúdy

Zachytávanie výnimiek

- metóda nesmie ignorovať kontrolované výnimky
- príkaz try a v bloku catch zachytí príslušnú výnimku
- klauzula throws v hlavičke metódy klienta
  - klazula throws pre triedu výnimky, ktorú nezachytáva žiadny blok catch
  - posunutie výnimky vyššie

Vlastné triedy výnimiek

- nová kontrolovaná výnimka – potomok triedy Exception (alebo od nej odvodenej triedy)
- nová nekontrolovaná výnimka – potomok triedy RuntimeException
- konvencia – názov konci slovom Exception

Dôvody použitia vlastných výnimiek

- neexistuje štandardná výnimka, ktorá dostatočne popisuje chybu
- vyčlenenie samostatnej výnimky
- existuje doplňujúca informácia, ktorá môže mať vplyv na riešenie chyby – kontrolované výnimky

Príklad triedy vlastnej výnimky

```
public class NeznameDieloException
                                        extends Exception
{
    public NeznameDieloException(String paNazov)
    {
        super("Dielo s nazvom " + paNazov +
                                          " nebolo najdene.");
    }
}
```

Vlastná výnimka

- NeznameDieloException
- definícia triedy – odčlenenie výnimky od štandardných
- kontrolovaná výnimka
- zostavenie textu chybového hlásenia v konštruktore
- využitie konštruktora predka
- ostatné metódy získava od predka

Využitie vlastnej výnimky – Katalog

```
public void vymaz(String paTitul)
                                throws NeznameDieloException
{
    ...
    AVIDielo polozka = this.vyhladaj(paTitul);
    if (polozka == null) {
        throw new NeznameDieloException(paTitul);
    } else {
        aZoznamPoloziek.remove(polozka);
    }
}
```

Metóda vymaz triedy Katalog

- klauzula throws – vyhadzuje kontrolovanú výnimku
- test výsledku vyhľadania
- vyhodenie výnimky

Zachytenie vlastnej výnimky(1)

```
public class VykonavacVymaz implements IVykonavac
{
    private ITerminal aTerminal;
    ...
    public boolean vykonaj(Katalog paKatalog,
                           String paParameter)
    {
        ...
    }
}
```

Zachytenie vlastnej výnimky(1)

```
try {
    paKatalog.vymaz(paParameter);
} catch (NeznameDieloException ex) {
    aTerminal.vypisRiadok(ex.getMessage());
    ex.printStackTrace();
}
```

Zachytenie vlastnej výnimky(3)

- príkaz try v metóde vykonaj vykonávača
- chránený príkaz vymazania z katalógu
- zachytenie výnimky = zotavenie sa z chyby
  - informácia pre používateľa
- využitie zdedených metód
  - getMessage() – text správy o chybe
  - printStackTrace() – výpis textu na štandardný chybový výstup – objekt System.err

Kontrola stavu v konštruktore

- nesprávne parametre
- dve možnosti reakcie:
  - zmena počiatočného stavu na správny
  - vyhodenie výnimky

Zmena počiatočného stavu na správny

- riešenie používané doteraz
- napr. AutomatMHD
  - nekladná hodnota ceny lístka => použitie prednastavenej hodnoty
- klient nevie o probléme
  - možnosť informovať používateľa cez terminál
  - možnosť informovať klienta cez špeciálnu správu dajChybu()

Vyhodenie výnimky v konštruktore

- oznámenie chyby klientovi
- inštancia sa vôbec nevytvorí

Príklad – server

```
public CD(String paTitul, String paAutor, int paPocetSkladieb,
                                                    int paCelkovyCas)
{
   super(paTitul, paCelkovyCas);

   if (paAutor == null || paAutor.isEmpty()) {
      throw new IllegalArgumentException("paAutor musi byt
                                                    nastaveny");
   }
   aAutor = paAutor;

   if (paPocetSkladieb <= 0 ) {
      throw new IllegalArgumentException("paPocetSkladieb musi
                                                    byt vacsi ako nula");
   }
   aPocetSkladieb = paPocetSkladieb;
}
```

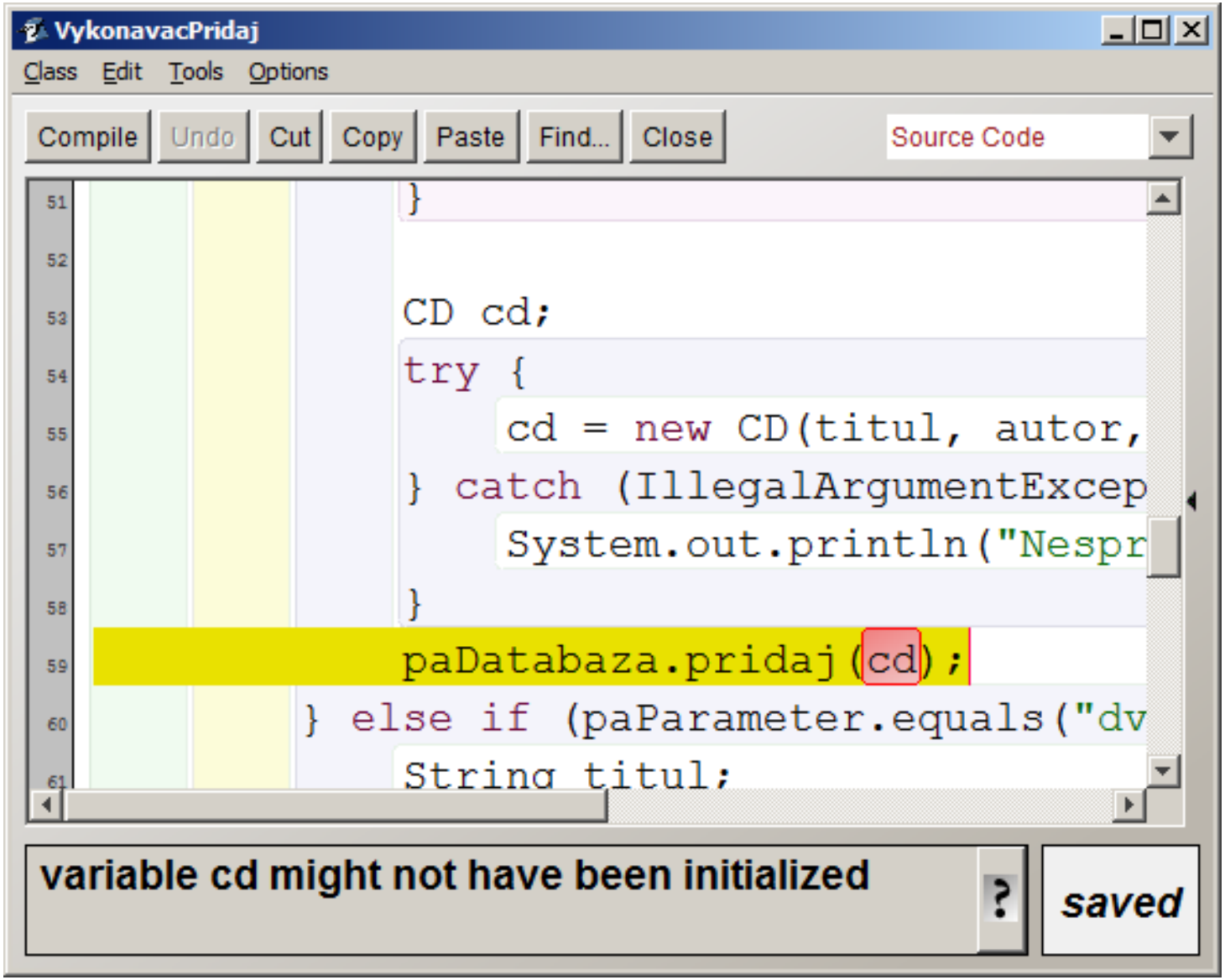
Príklad – klient

```
try {
    CD cd = new CD(titul, autor, pocetSkladieb, cas);
    paDatabaza.pridaj(cd);
} catch (IllegalArgumentException ex) {
    System.out.println("Nespravne parametre!");
}
```

Príklad – klient – nesprávne

```
CD cd;
try {
    cd = new CD(titul, autor, pocetSkladieb, cas);
} catch (IllegalArgumentException ex) {
    System.out.println("Nespravne parametre!");
}
paDatabaza.pridaj(cd);
```

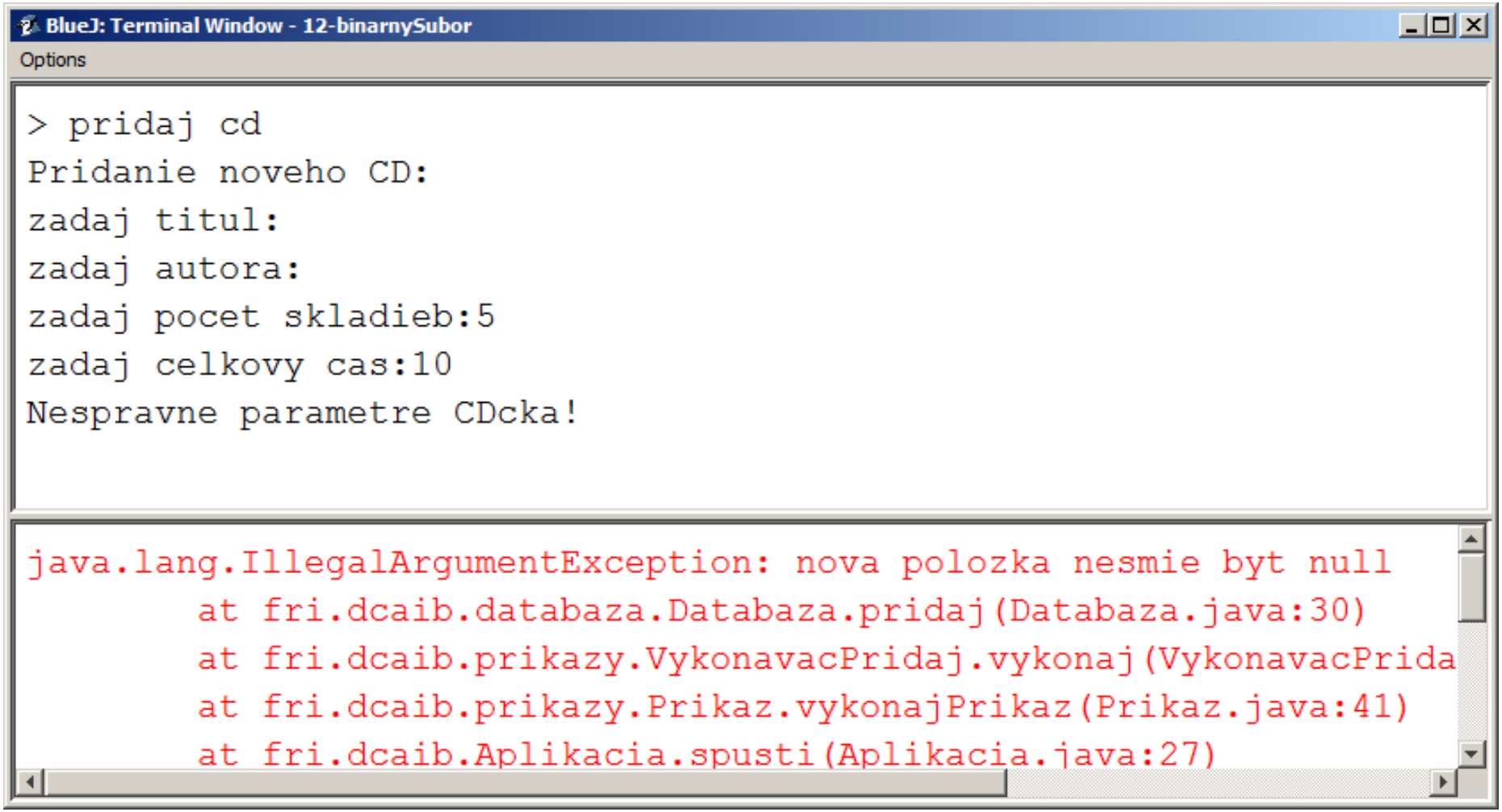
Chyba pri preklade



Príklad – klient – tiež nesprávne

```
CD cd = null;
try {
    cd = new CD(titul, autor, pocetSkladieb, cas);
} catch (IllegalArgumentException ex) {
    System.out.println("Nespravne parametre!");
}
paDatabaza.pridaj(cd);
```

Chyba za behu



Možnosti práce s výnimkami v klientovi

- zotavenie sa z chyby
- predchádzanie chybám

Zotavenie sa z chyby

- príkazy v bloku catch
- niekedy nový pokus s celým príkazom try
  - napr. vstupy
- nemusí sa podariť
- pozor na nekonečný cyklus

Príklad – zápis do súboru(1)

```
…
boolean uspech = false;
int pocetPokusov = 0;
while (uspech == false) {
    // pokus o zapis do suboru – prikaz try
}
…
```

Príklad – zápis do súboru(2)

```
try {
    // zapis do suboru
    uspech = true; // posledny prikaz bloku
} catch (IOException e) {
    if (pocetPokusov < MAX_POCET) {
        pocetPokusov++;
        // vyber inu moznost, kam zapisat subor
    } else {
        throw e; // vynimka sa posuva vyssie
    }
}
```

Predchádzanie chybám(1)

- vyžaduje spoluprácu klienta so serverom
- príklad – duplicita v katalógu CD a DVD
  - vyhladaj – vráti null, ak titul v katalógu nie je
  - ak sa kontroluje, netreba vyhadzovať výnimku
- server musí poskytnúť možnosť
- klient nemusí riešiť príslušnú výnimku

Predchádzanie chybám(2)

- nevýhody
  - zvyšuje sa implementačná závislosť
  - server sa poisťuje – dvojité testovanie

Výnimky a testovanie

- JUnit využíva výnimky na oznamovanie chýb
- assertEquals, assertNull, ...
  - AssertionError (potomok Error)
- test končí chybou po ľubovoľnej výnimke

Negatívne testy

- neprípustné parametre spôsobia výnimku?
- ! výnimka ukončí test chybou
  - nie je to, čo chceme

- dve možnosti riešenia
  - try-fail-catch
  - expected (JUnit 4.x)

try-fail-catch

```
@Test
public void newCDZlyTitul()
{
    try {
        new CD(null, "Niekto", 5, 30);
        fail("Neošetrená hodnota null ako titul");
    } catch (IllegalArgumentException ex) {
        // vynimka nastala = ok
    }
}
```

expected (JUnit 4.x)

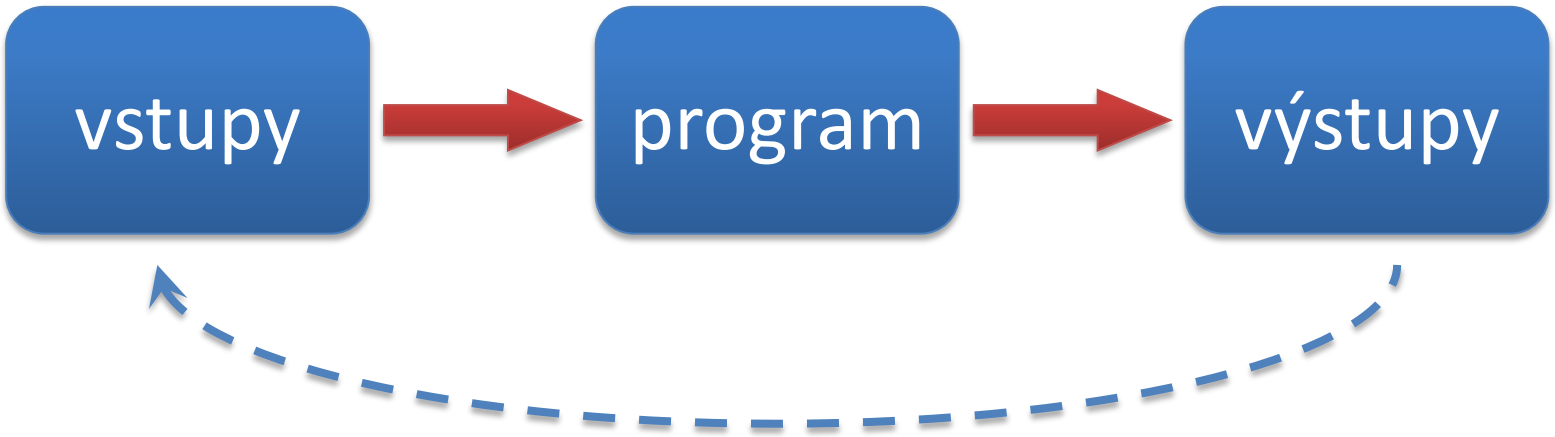
```
@Test(expected = IllegalArgumentException.class)
public void newCDZlyAutor()
{
    new CD("Nieco", null, 5, 30);
}
```

Vstupy a výstupy

Vstupy a výstupy programov

- programy – algoritmy
  - hromadnosť => nie na jediné použitie
  - jeden druh úlohy
  - rôzne začiatočné hodnoty
  - rôzne výsledky

Vstupy a výstupy programov



Štandardný vstup a výstup

Štandardný vstup a výstup

- Trieda System
- atribúty objektového typu:
  - in – vstup
  - out, err – výstup

Štandardný výstup

- výpis textu na obrazovku
- priame použitie v programe

- out – statický typ PrintStream
  - najčastejšie správy
    - print(String paText)
    - println() – nový riadok
    - println(String paText)

Štandardný vstup

- čítanie textu (postupnosti znakov) z klávesnice
- čaká na ukončenie písania na klávesnici – enter
- kontrolný opis na obrazovku – JVM (+OS)

- in – statický typ InputStream
  - System.in sa väčšinou nevyužíva v programe priamo
  - využitie v programe pomocou triedy Scanner
  - Scanner – obaľuje System.in

Scanner next a hasNext

- oddeľovače - skupina znakov (Whitespace): medzera, tabulátor, novy riadok
- rozdelenie vstupného riadku pomocou oddeľovačov na slová (tokeny)
- boolean hasNext()
  - existuje ešte slovo?
- String next()
  - prečíta nasledujúce slovo, všetky oddelovače pred slovom vynechá

Scanner

- ďalšie správy inštancii triedy Scanner

- boolean hasNext*PrimitivnyTyp*()
  - nasledujúce slovo
- PrimitivnyTyp next*PrimitivnyTyp*()
  - InputMismatchException

- boolean hasNext*Line*() -
- String next*Line*()

Scanner – vytvorenie

- čítanie štandardného vstupu:

```
Scanner klavesnica = new Scanner(System.in);
```

- čítanie ľubovoľného reťazca

```
String riadok = ...;
Scanner parser = new Scanner(riadok);
```

Projekt KCaIB

- katalóg CD a DVD

- objekty treba vždy nanovo vytvárať
- (alebo nechať navždy zapnutý program)

- úloha: ukladanie katalógu do súborov

Súbory

Trieda File

- poskytuje základné informácie o súbore
  - názov súboru
  - čas zmeny
  - práva na súbor
  - existencia súboru na disku
  - absolútna cesta
  - ...
- o spolupráci jazyka Java s OS
- nepracuje s obsahom súboru

Možnosti realizácie

- textové súbory
- binárne súbory
- serializácia objektov

Práca so súbormi

- čítanie
  - otvorenie súboru na čítanie
  - postupné čítanie obsahu
  - zatvorenie súboru

- zápis
  - otvorenie súboru na zápis
  - postupný zápis nového obsahu
  - zatvorenie súboru

Textové súbory

- čitateľné človekom
- najťažšie na strojové spracovanie

- Príklady
  - textové súbory .txt
  - zdrojové kódy .java
  - ...

Textové súbory v jazyku Java

- čítanie
  - trieda Scanner
  - Scanner(File paSubor)
- zápis
  - trieda PrintWriter
  - PrintWriter(File paSubor)
  - rozhranie podobné ako System.out

- nutnosť uzavretia súboru správou close()

Metóda zapis v triede Katalog

```
public void zapis() throws IOException
{
    File subor = new File("Katalog.txt");
    PrintWriter zapisovac = new PrintWriter(subor);
    for (AudiovizualneDielo dielo : aZoznam) {
        dielo.zapis(zapisovac);
    }
    zapisovac.close();
}
```

Metóda zapis v triede AudiovizualneDielo

```
public abstract void zapis(PrintWriter paZapisovac);
```

Metóda zapis v triede CD

```
public void zapis(PrintWriter paZapisovac) {
    paZapisovac.println("CD");
    paZapisovac.println(this.dajTitul());
    paZapisovac.println(this.dajCelkovyCas());
    paZapisovac.println(aAutor);
    …
}
```

Metóda zapis v triede DVD

```
public void zapis(PrintWriter paZapisovac)
{
    paZapisovac.println("DVD");
    paZapisovac.println(this.dajTitul());
    paZapisovac.println(this.dajCelkovyCas());
    paZapisovac.println(aReziser);
    …
}
```

Výsledok v súbore

CD
The Best of
30
The Beatles
12

DVD
Tenkrat na zapade
90
Neznamy autor

Metóda nacitaj v triede Katalog

```
public void nacitaj() throws IOException
{
    aZoznam.clear();
    File subor = new File("Katalog.txt");
    Scanner citac = new Scanner(subor);
    while (citac.hasNextLine()) {
        ...
    }
    citac.close();
}
```

Metóda nacitaj v triede Katalog

```
String typ = citac.nextLine();
if (typ.equals("CD")) {
    aZoznam.add(new CD(citac));
} else {
    aZoznam.add(new DVD(citac));
}
```

Nový konštruktor v triede CD

```
public CD(Scanner paCitac)
{
    super(paCitac.nextLine(), paCitac.nextInt());
    aAutor = paCitac.nextLine();
    …
}
```

Nový konštruktor v triede DVD

```
public DVD(Scanner paCitac)
{
    super(paCitac.nextLine(), paCitac.nextInt());
    aReziser = paCitac.nextLine();
    …
}
```

Problémy s textovými súbormi(1)

- príklad nebude fungovať

- problém s koncom riadku za číslom (30, 12)
- nextLine po nextInt prečíta prázdny reťazec

- možné riešenia
  - zapisovať čísla bez konca riadku – viac problémov ako úžitku
  - prečítať najskôr riadok a ten postupne čítať pomocou triedy Scanner
  - po čísle prečítať zvyšok riadku a zabudnúť

Problémy s textovými súbormi(2)

- ďalší problém – viacriadkové komentáre
- riešenie = DÚ
- problémy s formátovaním
  - zápis – bez problémov ľubovoľný formát
  - čítanie – komplikované

Binárne súbory

- nečitateľné človekom, resp. čitateľné komplikovane
- formát rovnaký ako v pamäti – jednoducho spracovateľný počítačom
- pri dodržaní poradia zápisu a čítania dát (musia byť zhodné) nehrozí problém s formátovaním

Binárne súbory v jazyku Java

- čítanie
  - trieda FileInputStream
  - FileInputStream(File paSubor)

- zápis
  - trieda FileOutputStream
  - FileOutputStream(File paSubor)

- komplikované – možnosť zapisovať iba byte[]
- nutnosť uzavretia súboru správou close()

Binárne súbory v jazyku Java

- zjednodušenie – práca s primitívnymi typmi jazyka Java
- čítanie
  - trieda DataInputStream
  - DataInputStream(FileInputStream paStream)

- zápis
  - trieda DataOutputStream
  - DataOutputStream(FileOutputStream paStream)
- nutnosť uzavretia súboru správou close()

Trieda DataOutputStream

- write*PrimitivnyTyp*(*PrimitivnyTyp* paHodnota)
  - zapíše hodnotu primitívneho typu do súboru

- write*UTF*(String paRetazec)
  - zapíše reťazec do súboru

Trieda DataInputStream

- *PrimitivnyTyp* read*PrimitivnyTyp*()
  - prečíta hodnotu primitívneho typu zo súboru

- String read*UTF*()
  - prečíta reťazec zo súboru

Metóda zapis v triede Katalog

```
public void zapis() throws IOException
{
   File subor = new File("Katalog.txt");
   FileOutputStream stream
                              = new FileOutputStream(subor);
   DataOutputStream zapisovac
                             = new DataOutputStream(stream);
   for (AudiovizualneDielo dielo : aZoznam) {
      dielo.zapis(zapisovac);
   }
}
```

Metóda zapis v triede AudiovizualneDielo

```
public abstract void zapis
                    (DataOutputStream paZapisovac)
                    throws IOException;
```

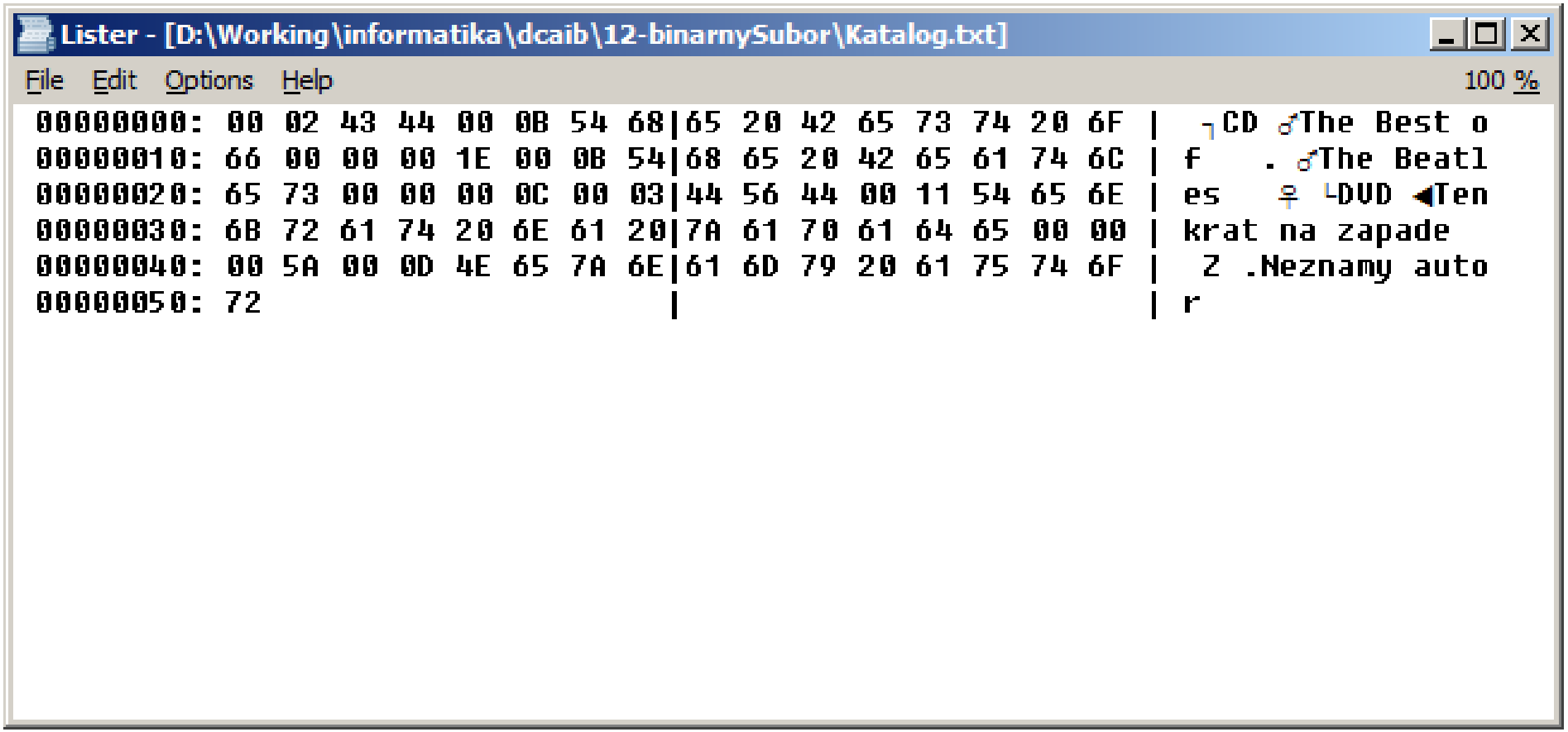
Metóda zapis v triede CD

```
public void zapis(DataOutputStream paZapisovac)
                                        throws IOException
{
    paZapisovac.writeUTF("CD");
    paZapisovac.writeUTF(this.dajTitul());
    paZapisovac.writeInt(this.dajCelkovyCas());
    paZapisovac.writeUTF(aAutor);
    ...
}
```

Metóda zapis v triede DVD

```
public void zapis(DataOutputStream paZapisovac)
                                        throws IOException
{
    paZapisovac.writeUTF("DVD");
    paZapisovac.writeUTF(this.dajTitul());
    paZapisovac.writeInt(this.dajCelkovyCas());
    paZapisovac.writeUTF(aReziser);
    ...
}
```

Výsledok v súbore



Metóda nacitaj v triede Katalog

```
public void nacitaj()
{
    boolean koniecSuboru = false;
    aZoznam.clear();
    File subor = new File("Katalog.txt");
    FileInputStream stream = new FileInputStream(subor);
    DataInputStream citac = new DataInputStream(subor);
    while (!koniecSuboru) {
        ...
    }
}
```

Metóda nacitaj v triede Katalog

```
try {
    String typ = citac.readUTF();
    if (typ.equals("CD")) {
        aZoznam.add(new CD(citac));
    } else {
        aZoznam.add(new DVD(citac));
    }
} catch (EOFException ex) {
    koniecSuboru = true;
}
```

Nový konštruktor v triede CD

```
public CD(DataInputStream paCitac)
{
    super(paCitac.readUTF(), paCitac.readInt());
    aAutor = paCitac.readUTF();
    …
}
```

Nový konštruktor v triede DVD

```
public DVD(DataInputStream paCitac)
{
    super(paCitac.readUTF(), paCitac.readInt());
    aReziser = paCitac.readUTF();
    …
}
```

Vďaka za pozornosť