Soubory

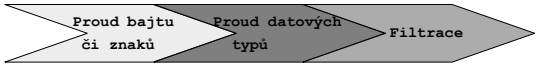
A0B36PR1-Programování 1
Fakulta elektrotechnická
České vysoké učení technické

Soubory

- Soubor je množina údajů uložená ve vnější paměti počítače, obvykle na disku
- Pro soubor jsou typické tyto operace:
  - otevření souboru, čtení údaje, zápis údaje, uzavření souboru
- Přístup k údajům (čtení nebo zápis) může být:
  - sekvenční nebo libovolný (adresovatelný)
- Soubory se sekvenčním přístupem umožňují pouze postupné (sekvenční) čtení nebo zápis údajů
- Soubory s libovolným přístupem umožňují adresovatelné čtení nebo zápis údaje (podobně jako pole)
- Způsob přístupu k údajům v souboru není zakódován v souboru, ale je dán programem
- Zde se budeme zabývat sekvenčními soubory, pro zájemce i adresovatelnými

Soubory a proudy

- Java rozlišuje soubory (file) a proudy (stream)
  - soubor je množina údajů uložená ve vnější paměti počítače
  - proudy jsou nástroje k přenosu informací např. z/do souboru, ale také do/ze sítě, paměti, jiného programu, atd.
- informace může mít tvar znaků, bajtů, skupin bajtů (obrázky…), objektů,..
- přenos informace se děje dvoj/třívrstevně v proudech (streams):
  1. otevření přenosového proudu pro bajty či znaky
  2. otevření přenosového proudu pro datové typy Javy
  3. filtrace dat podle požadavků – bufferování, řádkování, …



Proudy

- Bajtové FileInputStream
  - přenos primitivních datových typů `DataOutputStream`
  - přenos objektů `ObjectOutputStream`
  - bufferování `BufferedOutputStream`
- Znakové `FileReader/FileWriter`
  - bufferování `BufferedReader`
  - tokenizace `StreamTokenizer`
- Libovolný přístup `RandomAccessFile`
- Zpracování souborů/adresářů `File`

Pozn:
- věnujeme se převážně vstupu/výstupu do a ze souborů
- zajišťuje `java.io`, cca 40 tříd

1. Soubor jako **posloupnost bytů**

- Soubory v Javě = třídy definované v balíku *java.io (+newio)*
- Příklad: kopie souboru

```
import java.io.*;
public class Kopie1 {
  public static void main(Strin[] args) throws IOException {
    FileInputStream in = new FileInputStream("vstup.txt");
    FileOutputStream out = new FileOutputStream("vystup.txt");
    int b = in.read();
    while (b!=-1) {
      out.write(b);
      b = in.read();
    }
    out.close();
    in.close();
  }
}        // * throws IOException – výjimka, vysvětlíme později
```

Vstupní soubor

Výstupní soubor, i zde, pokud není uvedena celá cesta, hledá se adresář podle hodnoty systémové proměnné user.dir, její výpis: System.out.println(System.getProperty("user.dir"));

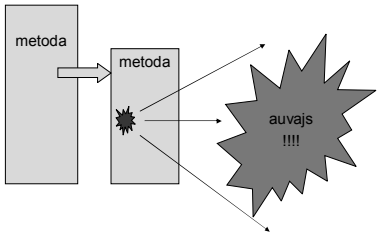
Proud bajtů či znaků

Soubor jako posloupnost bytů

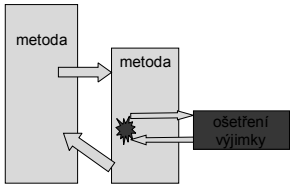
- FileInputStream in = new FileInputStream("vstup.txt");
  - vytvořený objekt *in* reprezentuje vstupní soubor uložený na disku v aktuálním adresáři pod názvem *vstup.txt*; pokud takový soubor neexistuje, nastane výjimka, chyba
- `FileOutputStream out=new FileOutputStream("vystup.txt");`
  - vytvořený objekt *out* reprezentuje výstupní soubor, který bude uložen do aktuálního adresáře pod názvem *vystup.txt*; pokud by soubor nebylo možné vytvořit, nastane chyba
  - ► `b = in.read();`
    - ze souboru *in* se přečte jeden byte (číslo v rozsahu 0..255) a uloží do *b*; není-li v souboru žádný nepřečtený byte, výsledkem metody je –1 typu int
  - ► `out.write(b);`
    - do souboru *out* se zapíše jeden byte s hodnotou *b*
  - ► `out.close();`
  - ► `in.close();`
    - uzavření souborů *in* a *out*

Informace

Výjimky – vznik nestandardní situace



Výjimky – ideální ošetření



- Při operacích se soubory (nejen se soubory) mohou nastat chyby
- Chyba (resp. „výjimečný stav“) při provádění programu v jazyku Java nemusí znamenat ukončení programu – chybu lze ošetřit a pokračovat dál

Výjimky

- K ošetření chyb slouží mechanismus výjimek (`Exceptions`)
  - libovolná metoda (konstruktor, funkce) může skončit standardně (proběhla-li operace bez chyby) nebo **nestandardně „vyhozením“ (vyvoláním) výjimky určitého typu** (v případě, že nastala chyba či „chyba“)
  - pokud příkaz skončil „vyhozením“ výjimky, další příkazy se neprovedou a **řízení se předá konstrukci ošetřující výjimku** daného typu (s touto konstrukcí se seznámíme později)
  - pokud taková konstrukce v těle funkce (metody, konstruktoru) není, skončí funkce nestandardně a **výjimka se šíří** na dynamicky nadřazenou úroveň
  - není-li výjimka ošetřena ani ve funkci *main*, **vypíše se a program** skončí
  - **pro rozlišení různých typů výjimek je v jazyku Java zavedena řada knihovních tříd, výjimky jsou instancemi těchto tříd!**

Ošetření výjimek – try - catch

Příklad: funkce, která si vyžádá zadání jména vstupního souboru z klávesnice a pokud soubor s daným jménem neexistuje, proces se opakuje (při zadání prázdného jména program skončí)

```
static FileInputStream vstup() {// vyjimka.java
  for (;;) {
    System.out.print("zadejte vstupní soubor: ");
    String jmeno = sc.nextLine();
    if (jmeno.equals("")) System.exit(0);
    try {
      FileInputStream in = new FileInputStream(jmeno);
      return in;
    } catch (FileNotFoundException e) {
      System.out.print("soubor neexistuje");
    }
  }
}
```

Podezřelé místo

Ošetření

Ošetření výjimek

Pokračování příkladu: napíšeme podobnou funkci pro zadání výstupního souboru a obě funkce použijeme ve druhé verzi programu pro kopii souboru (třída Kopie2)

```
static FileOutputStream vystup() {
  for (;;) {System.out.print("zadejte výstupní soubor: ");
    String jmeno = sc.nextLine();
    if (jmeno.equals("")) System.exit(0);
    try {
      FileOutputStream out = new FileOutputStream(jmeno);
      return out;
    } catch (FileNotFoundException e) {
      System.out.print("soubor nelze vytvořit");
    }
  }
}
```

Podezřelé místo

Ošetření

Ošetření výjimek

- V hlavní funkci *main* ošetříme výjimku typu *IOException*, která může vzniknout při provádění metod *read, write* a *close*

```
public static void main(String[] args) {
FileInputStream in = vstup();
 FileOutputStream out = vystup();
   try {
     int b = in.read();
     while (b!=-1) {
       out.write(b);
       b = in.read();
     }
     in.close(); out.close();
   } catch (IOException e) {
     e.printStackTrace();
   }
 }
```

Podezřelé místo

Ošetření

Ošetření výjimek - příklad

Dialog:

```
zadejte vstupní soubor: ??
soubor neexistuje
zadejte vstupní soubor: vstup.txt
zadejte výstupní soubor: ...
soubor nelze vytvořit
zadejte výstupní soubor: vystup.txt
```


2. Soubor jako **posloupnost primitivních typů**

Příklad: program, který vytvoří soubor obsahující 100 náhodných čísel typu *double* a pak soubor přečte a vypíše součet čísel

```
public class Cisla {
public static void main(String[] args)throws Exception {
DataOutputStream out = new DataOutputStream(
                              new FileOutputStream("tmp.bin"));
for (int i=0; i<100; i++)
 out.writeDouble(Math.random());
out.close();

DataInputStream in = new DataInputStream(
                            new FileInputStream("tmp.bin"));
double soucet = 0;
while (in.available()>0)
   soucet = soucet + in.readDouble();
System.out.print(soucet);
}}
```

Zavření souboru

Výstupní soubor

Vstupní soubor

Proud bajtu či znaků

Proud datových typů

3. Soubor primitivních typů a objektů

Soubor obsahující jak primitivní typy tak objekty reprezentují třídy ObjectOutputStream a ObjectInputStream

Příklad: program, který vytvoří soubor obsahující dvě hodnoty typu *int* a dva řetězce, a pak soubor přečte a vypíše

```
public class CislaRetezce {
  public static void main(String[] args) throws Except
   ObjectOutputStream out = new ObjectOutputStream(
                              new FileOutputStream("temp.bin"));
  out.writeInt(1); out.writeInt(2);
  out.writeObject("prvni retez"); out.writeObject("druhy retez");
  out.close();
  ObjectInputStream in = new ObjectInputStream(
                            new FileInputStream("temp.bin"));
    System.out.print(in.readInt()+" "+in.readInt());
    String s1 = (String)in.readObject();
    String s2 = (String)in.readObject();
    System.out.print (s1+" "+s2);
  }}
```

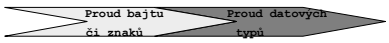
Výstupní soubor

Zavření souboru

Vstupní soubor

Operace se souborem primitivních typů a objektů

Uvedenými metodami lze zapisovat a číst pouze tzv. serializovatelné objekty (patří mezi ně implicitně např. řetězce a pole primitivních typů), jinak je třeba „serializovat“ (implementovat rozhraní **`Serializable !`**)



4. Ukládání objektů do souboru

```
class ProObjekt implements Serializable {
  int i;   String jmeno;   int telefon;
  boolean pohlavi;  double vaha;
  public ProObjekt(int j) {
    i = j;
    jmeno = "JMENO-" + j;
    telefon = 111 + j;
    pohlavi = i%2==0;
    vaha = 25 - i;
  }
public String toString() {
return (i+"\t"+ jmeno+"\t"+ telefon+"\t"+ pohlavi+"\t"+
   vaha);}
```

Datové složky

Konstruktor

Jak zobrazit objekt

Ukládání objektů do souboru

```
public static void main(String[] args) throws IOException,
                                    ClassNotFoundException {
FileOutputStream fos = new FileOutputStream("objekty.bin");
ObjectOutputStream oos = new ObjectOutputStream(fos);
ProObjekt[] poleObjektu = new ProObjekt[3];
for (int i = 0; i < poleObjektu.length; i++) {
  poleObjektu[i] = new ProObjekt(i);  }
System.out.println("Ulozeni");
for (int i = 0; i < poleObjektu.length; i++) {
 System.out.println(poleObjektu[i]);  }
for (int i = 0; i < poleObjektu.length; i++) {
 oos.writeObject(poleObjektu[i]);
    }
fos.close();
for (int i = 0; i < poleObjektu.length; i++) {
   poleObjektu[i] = null;}
```

Výstupní soubor

Zavření souboru

Ukládání objektů do souboru

```
FileInputStream fis = new FileInputStream("objekty.bin");
ObjectInputStream ois = new ObjectInputStream(fis);
for (int i = 0; i < poleObjektu.length; i++) {
 poleObjektu[i] = (ProObjekt) ois.readObject();
    }
fis.close();
System.out.println("Cteni");
for (int i = 0; i < poleObjektu.length; i++) {
 System.out.println(poleObjektu[i]);
    }
  }
}
```

Vstupní soubor

```
Ulozeni
0       JMENO-0  111     true    25.0
1       JMENO-1  112     false   24.0
2       JMENO-2  113     true    23.0
Cteni
0       JMENO-0  111     true    25.0
1       JMENO-1  112     false   24.0
2       JMENO-2  113     true    23.0
```


Užitečná informace

Kolekce

- Kolekce je datová struktura obsahující proměnný počet prvků, pro kterou jsou (mimo jiné) definovány operace
  - vytvoření prázdné kolekce
  - přidání, odebrání, vložení, přístup k prvku
  - zjištění rozsahu, výpis a další
- V knihovně java.util je řada tříd definujících různé druhy kolekcí
  - Příkladem je třída *ArrayList*, ve které se vložené prvky rozlišují pomocí indexu počítaných od 0

Kolekce - ArrayList

```
ArrayList ko = new ArrayList();// nová prázdná kolekce

ko.add( "abcd" );             // přidání na konec
ko.add( "xyz" );              // přidání na konec
System.out.println( ko.get(0) );// vypíše se abcd
System.out.println( ko.get(1) );// vypíše se xyz
System.out.println( ko.size() );
                // aktuální počet prvků: 2
```

Poznámka: Od verze Javy 1.5 se používají typované kolekce, které umožňují vkládat objekty určitého typu. Např.:

```
List<String> seznamRet = new ArrayList<String>();
seznamRet.add("Pepa"); // O.K.
seznamRet.add(new java.awt.Point(4,6));
                      // Chyba při překladu
```

Příklad použití třídy ArrayList

- Přečte řádky zakončené prázdným řádkem a vypíše je v opačném pořadí

```
public class KontejnerRadku {
  public static void main(String[] args) {
  ArrayList radky = new ArrayList();
  Scanner scan = new Scanner(...);
  System.out.println( "zadejte řádky zakončené prázdným
  řádkem" );
  String radek = scan.nextLine();
  while (!radek.equals( "" )) {
   radky.add(radek);
   radek = scan.nextLine();
   }
   System.out.println( "výpis řádků v opačném pořadí" );
    for (int i=radky.size()-1; i>=0; i--)
  System.out.println(radky.get(i));
  }
}
```


Příklad použití třídy ArrayList

- Chceme-li z kolekce *radky* získat např. referenci na první řetěz a uložit jí do referenční proměnné:

```
String prvni;
prvni = radky.get(0);//nelze, chyba při překladu, vrací Object
prvni = (String)radky.get(0); // přetypováno, O.K.
```

- Proč? Metody *add* a *get* jsou specifikovány takto:
  `void add(Object obj)`
  `Object get()`
  - hodnotou proměnné, parametru nebo výsledkem metody typu *Object* může být reference na objekt jakéhokoliv typu
  - jestliže *o* je proměnná, parametr nebo výsledek metody typu *Object* a její hodnotou je reference na objekt typu *T*, pak pro přístup k referencovanému objektu jako k objektu typu *T* je třeba použít operaci přetypování *o* na typ *T* ve tvaru (*T*)*o*
  - operace zkontroluje, zda *o* skutečně referencuje objekt typu *T*; není-li tomu tak, nastane chyba při výpočtu, výjimka `DynamicCastException`

Seznam slov v souboru

- Vstup: textový soubor – vstup.txt
- Výstup: seznam všech slov obsažených v souboru, abecedně řazený

```
public class VypisRuznychSlov {

    private static Scanner scan;

    public static void main(String[] args) {
        SortedSet slova = nactiVsechnaSlova();
        vypisJeVPoradiPodleCetnosti(slova);
    }
....

}
```

Seznam slov v souboru

```
private static SortedSet nactiVsechnaSlova() {
SortedSet slova = new TreeSet();
try {
scan = new Scanner(new FileInputStream("vstup.txt"));
while(scan.hasNext()){
slova.add(scan.next());
        //přidá další slovo,pokud se neopakuje
            }
 }
catch (IOException ex) {
System.out.println("Soubor vstup.txt nebyl nalezen.");
        }
        return slova;
    }
```


Seznam slov v souboru

```
private static void vypisJeVPoradiPodleCetnosti(SortedSet
  slova) {
 for (Iterator it = slova.iterator(); it.hasNext();) {
 String slovo = (String)it.next();
 System.out.println(slovo);
        }
    }
```


Seznam slov v souboru

- Vstup: První slovo druhé a třetí. Máma mele maso. Mleté maso. Máma a táta se učí programovat.
- Výstup: *Mleté*
  Máma
  První
  a
  druhé
  maso.
  mele
  programovat.
  se
  slovo
  táta
  třetí.
  učí

Seznam slov v souboru - Java 5

```
public class VypisRuznychSlovNewJava {
    private static Scanner scan;
    public static void main(String[ ] args) {
        SortedSet<String> slova = nactiVsechnaSlova();
        vypisJeVPoradiPodleCetnosti(slova);
    }
. . . .
}
```

SortedSet<String>
možno vkládat pouze řetězce – kontroluje se to při překladu, je to typově bezpečné

Seznam slov v souboru - Java 5

```
private static SortedSet<String> nactiVsechnaSlova() {
SortedSet<String> slova = new TreeSet<String>();
try {
scan = new Scanner(new FileInputStream("vstup.txt"));
while(scan.hasNext()){
 slova.add(scan.next());
//přidá další slovo, pokud se neopakuje, vkládat lze pouze
//řetězce – typ String
            }
} catch (IOException ex) {
System.out.println("Soubor vstup.txt nebyl nalezen.");
          }
return slova;
    }
```


Seznam slov v souboru - Java 5

```
private static void
   vypisJeVPoradiPodleCetnosti(SortedSet<String> slova) {
   for (Iterator<String> it = slova.iterator(); it.hasNext();)
   {
   String slovo = it.next();
   System.out.println(slovo);
        }
    }
```

- Z kolekce vybíráme slova – typ String, není nutné přetypovávat

Seznam slov v souboru - Java 5

```
private static void
  vypisJeVPoradiPodleCetnosti(SortedSet<String> slova)
  {
   for (String slovo : slova) {
            System.out.println(slovo);
        }
    }
```

- Nový cyklus **`foreach`**
- Tento cyklus se čte: Pro každé **slovo** z kolekce **slova** proveď jeho výpis