8. přednáška: Soubory a proudy

- Soubor jako posloupnost bytů
- Ukládání/čtení primitivních typů
- Ukládání/čtení primitivních typů a objektů (řetězců)
- Ukládání/čtení objektů do souboru - serializace

Soubory a proudy

- Java rozlišuje **soubory** (file) a **proudy** (stream)
  - **soubor** je množina údajů uložená ve vnější paměti počítače
  - **proudy** (streams) 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:
  1. otevření přenosového proudu pro bajty či znaky
  2. otevření přenosového proudu pro datové typy Javy
  3. filtrace dat podle požadavků – bufferování, řádkování, …

Princip

Proudy

- **Bajtové proudy** — `FileInputStream`
  - přenos primitivních datových typů — `DataOutputStream`
  - přenos objektů — `ObjectOutputStream`
  - bufferování — `BufferedOutputStream`
- **Znakové proudy** — `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

Soubory

- **Soubor** je množina údajů uložená ve vnější paměti počítače, obvykle na disku.
- **Typické operace** pro pro práci se souborem jsou:
  - otevření souboru
  - čtení údaje
  - zápis údaje
  - uzavření souboru
- **Přístup** k údajům (čtení nebo zápis) může být:
  - **sekvenční** - soubory se sekvenčním přístupem umožňují pouze postupné (sekvenční) čtení nebo zápis údajů.
  - **libovolný (adresovatelný)** - 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

1. Soubor jako posloupnost bytů

- V jazyku Java slouží pro práci se soubory třídy definované v balíku ***java.io***.
- Soubor jako posloupnost bytů reprezentují třídy:
  - ***FileInputStream*** vstupní soubor (bude se z něj číst)
  - ***FileOutputStream*** výstupní soubor (bude se do něj zapisovat)
- Příklad: kopie souboru

```
import java.io.*;
public class Kopie1 {
      public static void main(String[] 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

Proud bajtů či znaků

Princip

Soubor jako posloupnost bytů

- Komentář k příkazům:
  - `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 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.
- Metody:
  - `b = in.read();`

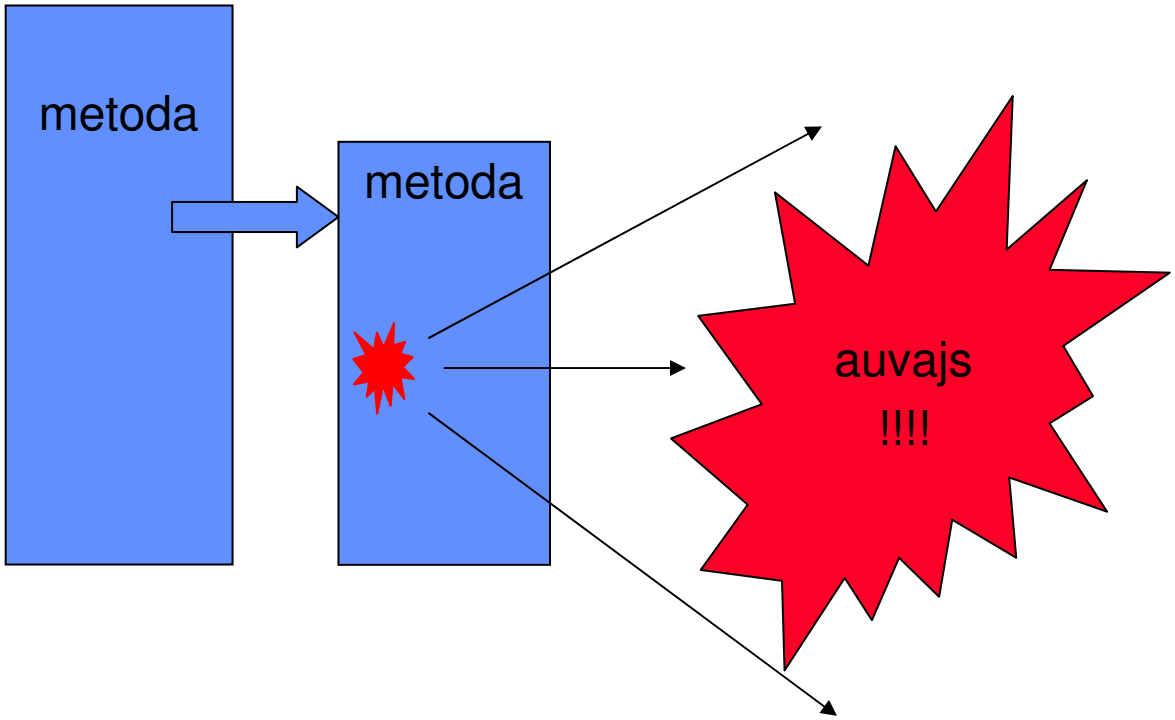
  Ze souboru *in* se přečte jeden byte (číslo v rozsahu 0..255) a uloží do *b*; není-li v souboru žádný nepřečtený byte, výsledkem metody je –1.
  - `out.write(b);`

  Do souboru *out* se zapíše jeden byte s hodnotou *b.*
  - `out.close();`
  - `in.close();`

  Uzavření souborů *in* a *out.*

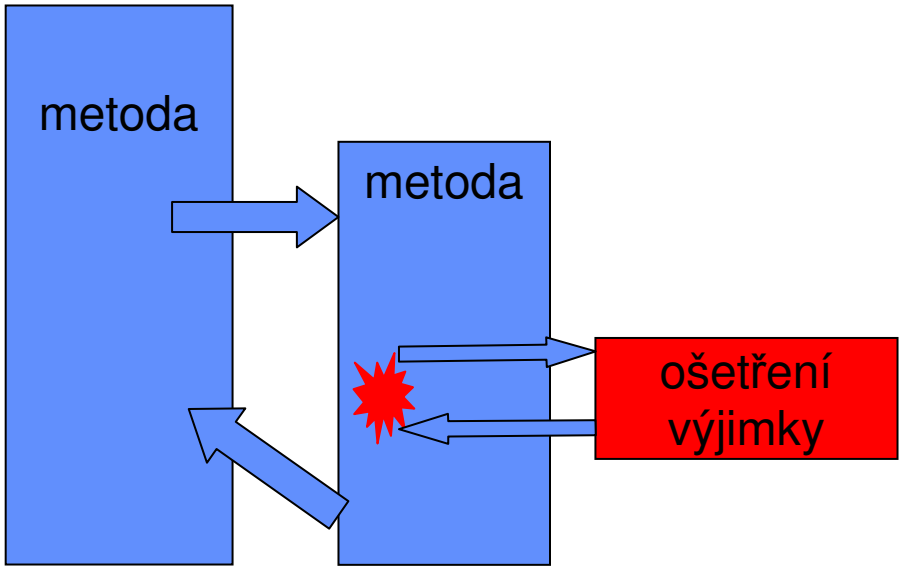
Výjimky – vznik nestandardní situace

- Co by se stalo, kdybychom programově neošetřili možné nestandardní situace.



Výjimky – ideální ošetření

- Ošetření výjimky.



Výjimky

- Při operacích se soubory (ale i při jiných operacích) mohou nastat chyby.
- Chyba (resp. „výjimečný stav“) při provádění programu v jazyku Java nemusí znamenat ukončení programu – chybu lze ošetřit a pokračovat dál.
- K ošetření chyb slouží mechanismus výjimek (**`Exceptions`**).
- **Princip výjimek v jazyku Java**:
  - 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

- Chceme-li ošetřit výjimku, která může vzniknout při provádění určité operace, je třeba operaci vložit do bloku ***try***, ke kterému je připojen blok (klauzule) ***catch*** ošetřující (zachytávající) výjimky daného typu.

  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() {
   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 = Sys.readLine();
      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

Příklad

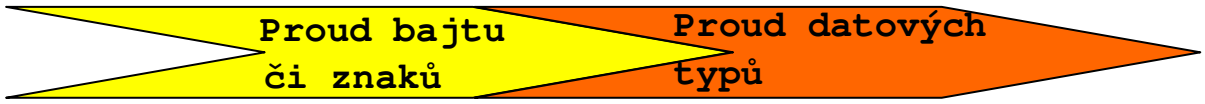
Ošetření výjimek - příklad

```
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("temp.bin"));
        for (int i=0; i<100; i++)
              out.writeDouble(Math.random());
        out.close();
        DataInputStream in = new DataInputStream(
                                 new FileInputStream("temp.bin"));
        double soucet = 0;
        while (in.available()>0)
           soucet = soucet + in.readDouble();
        System.out.print(soucet);
     }
}
```



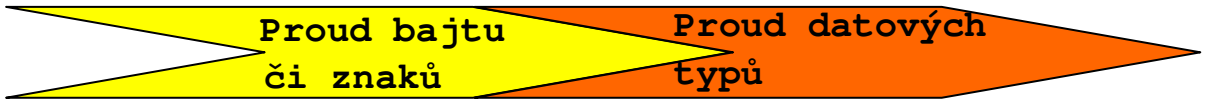
3. **Soubor** primitivních typů a objektů

- 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.

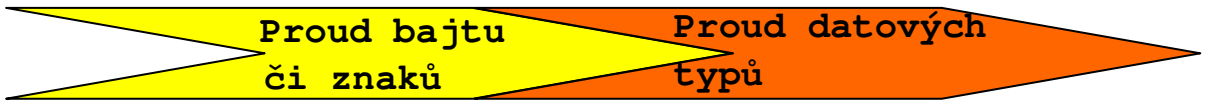
```
public class CislaRetezce {
        public static void main(String[] args) throws Exception {
        ObjectOutputStream out = new ObjectOutputStream(
                                    new FileOutputStream("temp.bin"));
    out.writeInt(1);
           out.writeInt(2);
    out.writeObject("prvni 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);
        }
}
```



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“, tzn. 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

... pokračování

```
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]);
       oos.writeObject(poleObjektu[i]);
    }
    fos.close();
    for (int i = 0; i < poleObjektu.length; i++)
       poleObjektu[i] = null;
}
```

Ukládání objektů do souboru

... pokračování

```
   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]);
      }
   }
}
```

```
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