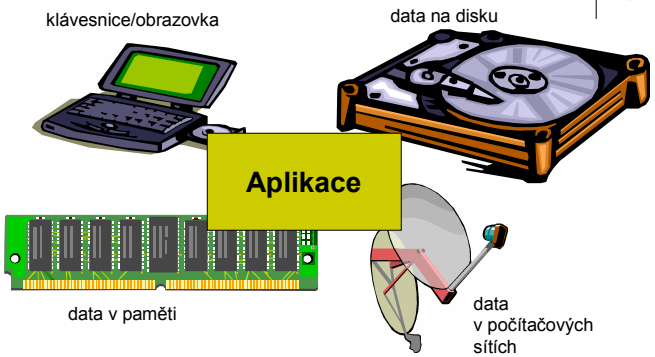
Vstup a výstup
Zpracování výjimek

doc. Ing. Miroslav Beneš, Ph.D.
katedra informatiky FEI VŠB-TUO
A-1007 / 597 324 213
http://www.cs.vsb.cz/benes
Miroslav.Benes@vsb.cz

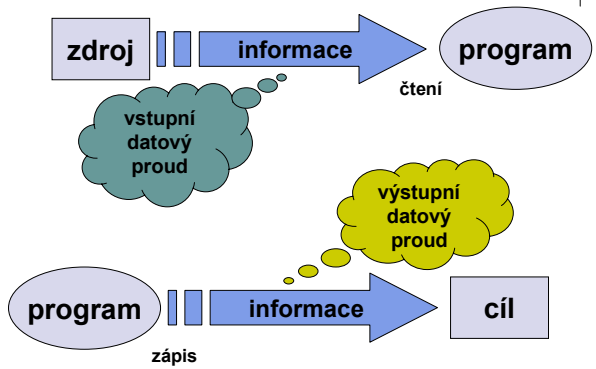
Vstup a výstup



Vstup a výstup

- **Uložení informací**
  - diskové soubory
  - paměť
  - počítačová síť
  - jiný program
- **Typ informací**
  - binární data – obrázky, zvuk, objekty, ...
  - textová data – ASCII, UNICODE مرحبا नमस्ते

Datové proudy – abstrakce V/V



Sekvenční zpracování dat

Čtení

1. otevřít proud
2. dokud jsou data
   čtení dat
3. zavřít proud

Zápis

1. otevřít proud
2. dokud jsou data
   zápis dat
3. zavřít proud

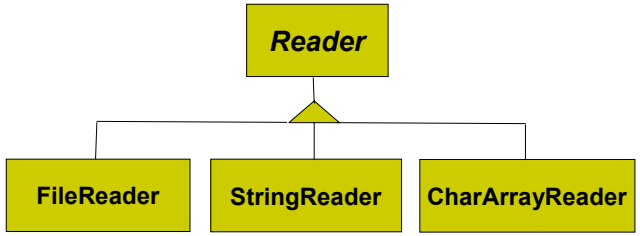
Znakové proudy

- posloupnost 16-bitových znaků UNICODE
- rozhraní *java.io.Reader*
  - `int read()` `-1 => EOF`
  - `int read(char[] buf)`
  - `int read(char[] buf, int offset, int len)`
- rozhraní *java.io.Writer*
  - `void write(int ch)`
  - `void write(char[] buf)`
  - `void write(char[] buf, int offset, int len)`
  - `void write(String str)`

Slabikové proudy

- posloupnost 8-bitových slabik
- rozhraní *java.io.InputStream*
  - `int read()` `-1 => EOF`
  - `int read(byte[] buf)`
  - `int read(byte[] buf, int offset, int len)`
- rozhraní *java.io.OutputStream*
  - `void write(int b)`
  - `void write(byte[] buf)`
  - `void write(byte[] buf, int offset, int len)`

Rozhraní Reader




```
Reader rdr = new FileReader("text.txt");
Reader rdr = new StringReader("2+3*5");
```

Příklad – čtení souboru

```
import java.io.*;
class IOTest {
  public static void main(String[] args)
     throws Exception
  {
     Reader rdr = new FileReader(args[0]);
     int ch;
     while( (ch = rdr.read()) != -1 ) {
        System.out.print((char)ch);
     }
     rdr.close();
  }
}
```

pozor!

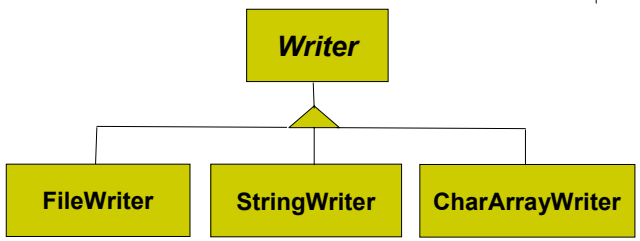
Filtry

- umožňují rozšíření funkčnosti proudů se zachováním jejich rozhraní
  - ***BufferedReader* / *BufferedWriter***
    čtení/zápis s využitím vyrovnávací paměti
  - ***PushbackReader***
    možnost „vrácení“ znaku zpět
    `void unread(int ch)`
  - ***DataInputStream* / *DataOutputStream***
    čtení/zápis hodnot primitivních datových typů
    `void writeDouble(double val)`

Příklad – čtení System.in

```
public static void main(String[] args)
  throws Exception
{
  BufferedReader rdr = new BufferedReader(
     new InputStreamReader(System.in));
  String radek;
  while( (radek = rdr.readLine()) != null )
  {
     System.out.println(radek);
  }
  rdr.close();
}
```


Rozhraní Writer




```
Writer wrt = new FileWriter("text.txt");
Writer wrt = new StringWriter();
```

Příklad – spojení souborů

```
// java Spojit vystup.txt vstup1.txt vstup2.txt ...
public static void main(String[] args)
  throws Exception
{
  Writer wrt = new BufferedWriter(
               new FileWriter(args[0]));
  for(int i = 1; i < args.length; i++ ) {
    Reader rdr = new BufferedReader(
                 new FileReader(args[i]));
    int ch;
    while( (ch = rdr.read()) != -1 )
      wrt.write(ch);
    rdr.close();
  }
  wrt.close();
}
```


Příklad – formátování pole

```
//  {"a","b","c"} => "a,b,c"
public static String arr2str(String[] args)
{
  StringWriter wrt = new StringWriter();
  for(int i = 0; i < args.length; i++ ) {
     if( i > 0 ) wrt.write(',');
     wrt.write(args[i]);
  }
  return wrt.toString();
}
```


Formátovaný výstup

- **Filtry *PrintStream* a *PrintWriter***

```
PrintWriter out = new PrintWriter(
                  new FileWriter("x.txt"));
out.println(10);
out.println(12.5);
out.print("aaaa");
out.println();
```

- **System.out, System.err**
  - objekty typu `PrintStream`
- **System.in**
  - objekt typu `InputStream`

Ošetření chybových stavů

- **Použití návratových kódů**
  - `boolean ok = f();`
    `if( !ok ) { ` *ošetření chyby* ` }`
  - co když zapomeneme otestovat výsledek?
- **Výjimky**
  - při chybě se provede určený úsek programu
  - nestrukturovaná obsluha: Basic, C
    - ON ERROR GOTO 9999
  - strukturovaná obsluha: C++, Java, C#
    - try - catch

Princip ošetření výjimek

```
int deleni(int x, int y) throws DeleniNulou
{
  if( y == 0 ) throw new DeleniNulou();
  return x / y;
}
```

```
int procento(int cast, int celek) {
  try {
     return deleni(100 * cast, celek);
  } catch( DeleniNulou e ) {
     return 100;
  }
}
```


Princip ošetření výjimek

1. Příkazem **try** se označí blok programu, ve kterém se mají ošetřovat výjimky
2. Doplní se varianty **catch**, které popisují zpracování jednotlivých typů výjimek
3. Při výskytu výjimky (příkaz **throw** nebo systémová chyba) se hledá aktivní blok **try**, ve kterém je výjimka ošetřena; provede se příslušná varianta **catch**
4. Není-li výjimka ošetřena, program se ukončí

Příklad – čtení souboru

```
public static void main(String[] args) {
  try {
     Reader rdr = new FileReader(args[0]);
     int ch;
     while( (ch = rdr.read()) != -1 )
        System.out.print((char)ch);
     rdr.close();
  } catch( FileNotFoundException e ) {
     System.err.println("Chyba: Soubor nenalezen");
  } catch( IOException e ) {
     System.err.println("Chyba: " + e);
  }
}
```


Deklarace výjimek

- Pokud funkce může způsobit výjimku, musí to deklarovat
  - `int fn() throws Excp1,Excp2, … { … }`
- Funkce může způsobit výjimku, pokud
  - sama výjimku generuje příkazem **throw**,
  - nebo není ošetřena výjimka generovaná jinou funkcí volanou z jejího těla
- Výjimka: následník třídy java.lang.Throwable
  - Exception, RuntimeException, Error

Příklad – vlastní typ výjimky

```
public class DeleniNulou extends Exception
{
   public DeleniNulou() {}
   public DeleniNulou(String msg) {
     super(msg);
   }
}

…
if( jmenovatel == 0 )
  throw new DeleniNulou("Jmenovatel == 0");
```


Úkoly

1. Vytvořte program, který opíše na standardní výstup zadaný soubor s očíslovanými řádky.
2. Vytvořte program, který přečte posloupnost celých čísel ze vstupu a vypíše nejmenší a největší hodnotu. Předpokládejte, že na každém řádku vstupu je uvedeno pouze jedno číslo.