



# Vstup a výstup

Peter Borovanský  
KAI, I-18

borovan 'at' ii.fmph.uniba.sk  
<http://dai.fmph.uniba.sk/courses/JAVA/>

dnes bude:

- " úvod do package java.io
- " **stream** (ako jednosmerný sekvencný tok dát) uvidíme v dvoch podobách
  - . byte (8-bit) stream (triedy InputStream/OutputStream),
  - . 16-bit (unicode/character) stream (triedy Reader/Writer),
- " výnimky (znova a podrobnejšie),
- " obáľkovanie (wrapovanie),
- " formátovaný vstup a výstup,
- " regulárne výrazy,
- " serializácia,
- " práca so súbormi a adresármi

Cvienie:

- " čítanie/zápis zo/do súboru, streamu
- " formátovaný vstup a výstup

Literatúra:

- " Thinking in Java, 3rd Ed. (<http://www.ibiblio.org/pub/docs/books/eckel/TIJ-3rd-edition4.0.zip>) . 12: The Java I/O System,
- " <http://interval.cz/clanky/naucte-se-javu-prace-se-vstupy-a-vystupy-1/>,
- " <http://interval.cz/clanky/naucte-se-javu-prace-se-vstupy-a-vystupy-2/>,

# Buffre

Vstup:

- “ ke sme sa u ili íta , najprv sme zvládli jednotlivé písmenká, M, A, M, A
- “ potom nás nau ili íta oddelené slová, vety, odstavce, celú rozprávku na pochopenie (spracovanie):
- “ rozprávky, musíme vedie spracova text odstavca, ktorý sa oby ajne vojde na stranu/obrazovku/do ohrani eného buffra
- “ vety, musíme na konci vety si pamäta oi. podmet, o ktorom je veta . opä iný buffer . lokálna krátkodobá pamä .

aj súbory (alias rozprávky) preto vieme íta po

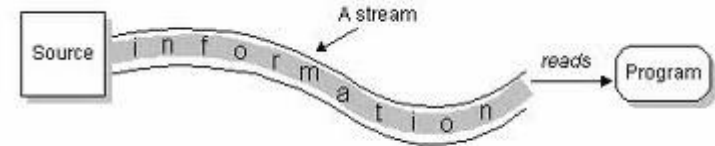
- “ bajtoch (8 bitov),
- “ znakoch (16-bit), ktoré majú rôznu interpretáciu v rôznych kódovaniach
- “ riadkoch (textové), i blokoch (binárne)
- “ oddelených slovách (formátovaný vstup)
- “ vetách i paragrafoch (regulárne výrazy)

Princíp buffrovania nám oddeluje

- “ proces pomalého nízko-úrov ového vstupu zo súboru
- “ proces rýchleho vnútorného spracovania údajov pomocou vyrovnávacej pamäte.

# InputStream/OutputStream

(byte stream)

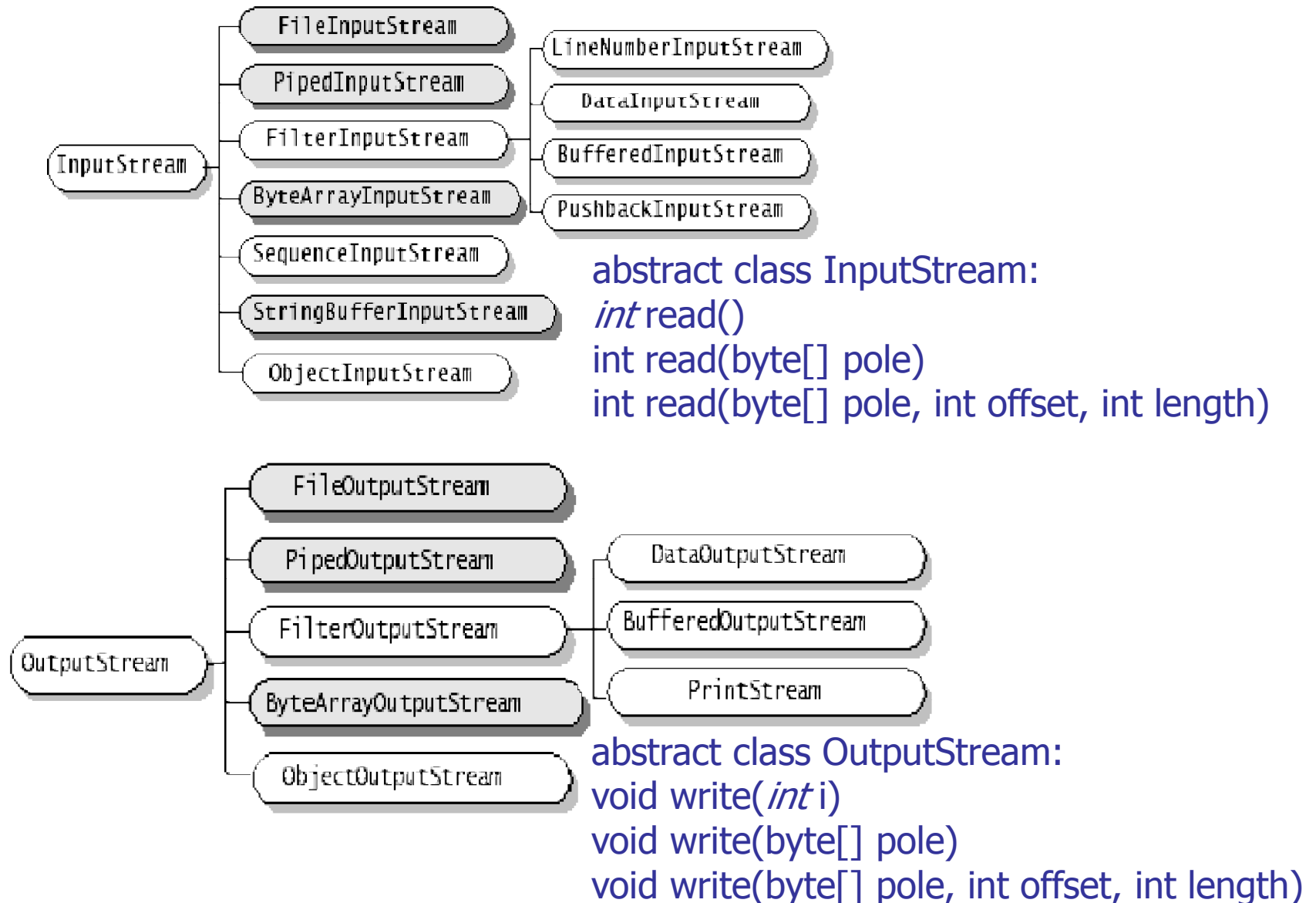


**InputStream/OutputStream** je jednosmerný sekvenčný tok bajtov (8bits)  
zdrojom/cieľom môže byť

- pole bajtov → [ByteArrayInputStream](#)(byte[] buf)\_
- String → [StringBufferInputStream](#)([String](#) s)\_
- súbor → [FileInputStream](#)([File](#) file)\_  
[FileInputStream](#)([String](#) name)
- pipe → [PipedInputStream](#)([PipedOutputStream](#) src)\_
- sekvencia iných  
zreťazených streamov  
(len pre InputStream) → [SequenceInputStream](#)([InputStream](#) s1, [InputStream](#) s2)\_  
[SequenceInputStream](#)([Enumeration](#) e)

Pomocou týchto podtried tried InputStream/OutputStream uniformným spôsobom čítame/píšeme z/do súboru, konzoly, byte[], pipe, ...

# InputStream/OutputStream (byte)



# Byte-Stream Input/Output

íťanie EXIF formátu z JPEG obrázku

íťanie .exe súboru

“ <http://www.delorie.com/djgpp/doc/exe/>

íťanie .class súboru

“ [http://en.wikipedia.org/wiki/Java\\_class\\_file](http://en.wikipedia.org/wiki/Java_class_file)

íťanie a zápis komprimovaného súboru (váz .zyp)

“ [http://en.wikipedia.org/wiki/ZIP %28file format%29](http://en.wikipedia.org/wiki/ZIP_%28file_format%29)

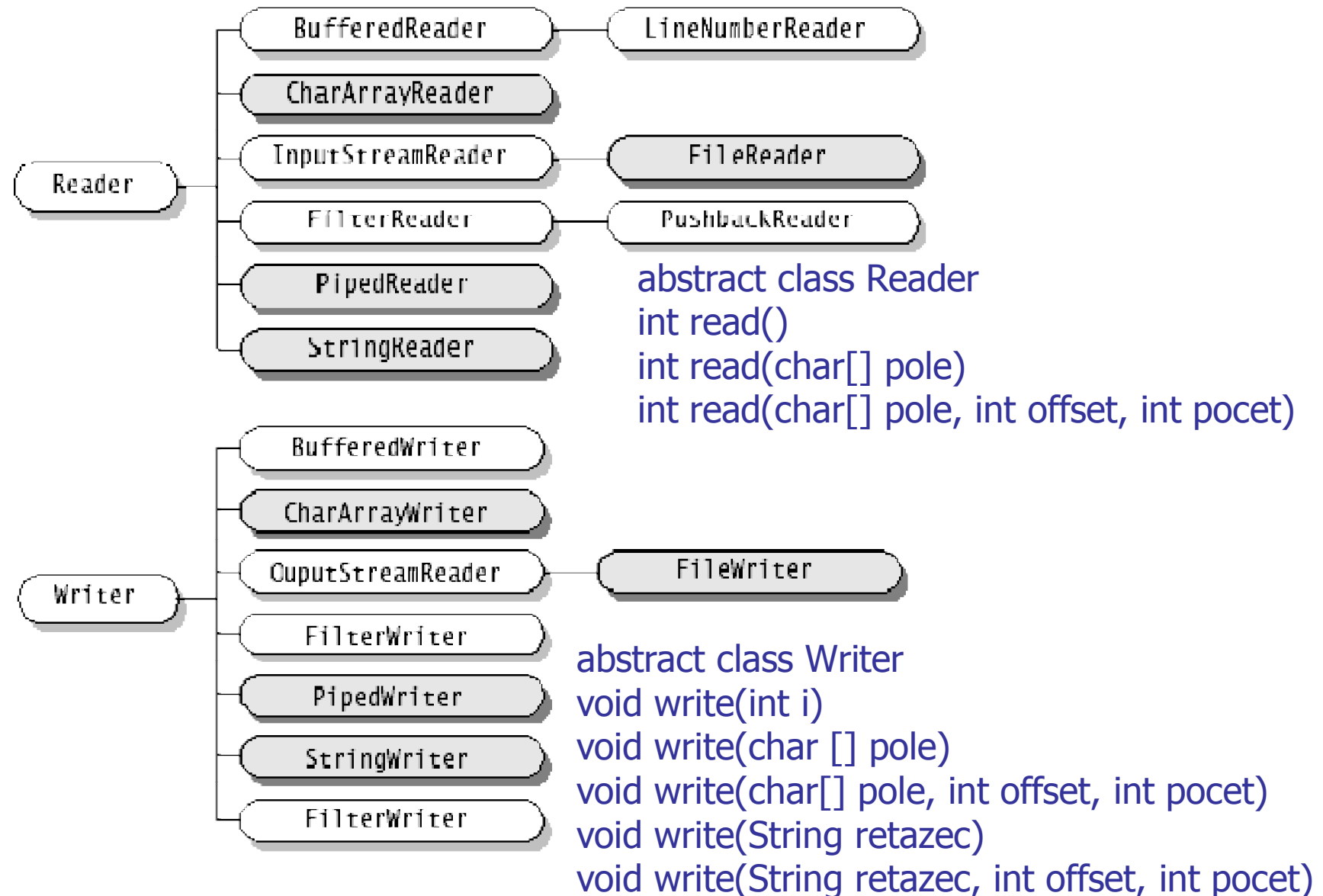
# Writer/Reader

(char stream)

**Reader/Writer** je jednosmerný sekvenčný tok znakov/charakterov/ (16bits)  
zdrojom/cieľom môže byť

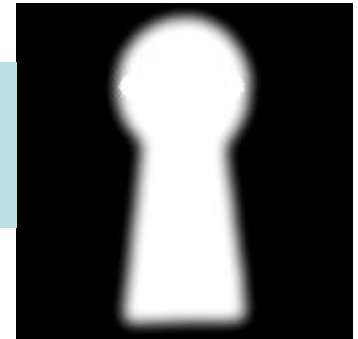
- pole znakov → [CharArrayWriter](#)(int initialSize)\_
- String → [StringWriter](#)(int size)\_
- Input/OutputStream (8bit) → [OutputStreamWriter](#)([OutputStream](#) out)  
→ [OutputStreamWriter](#)([OutputStream](#) out, [String](#) charsetName)
- pipe → [PipedWriter](#)([PipedReader](#) snk)\_
- súbor → [FileWriter](#)([String](#) fileName)  
[FileWriter](#)([File](#) file)

# Character (unicode) Streams (16bit)





# Konzola



**System.in** : InputStream – vstup

**System.out** : PrintStream – výstup

**System.err** : PrintStream – chybové hlásenia

// čítanie znaku s echom

**System.out.println**("Klepní znak"); 😊

char ch = ' ';

try {

    ch = (char)**System.in.read**(); 😞

} catch (IOException e) {

**System.err.println**("chyba");

    e.printStackTrace();

}

**System.out.println** ("klepol si: " + ch);

import java.io.\*;

import java.io.InputStream;

import java.io.PrintStream;

// čítanie riadku s echom

**System.out.println**("Napis riadok");

String inputLine = "";

while (true) {

    try {

        int tmp = **System.in.read**();

        if (tmp == -1) break;

        if (tmp == '\n') break;

        char c = (char) tmp;

        inputLine = inputLine + c;

    } catch (IOException e) {

**System.err.println**("chyba");

    }

}

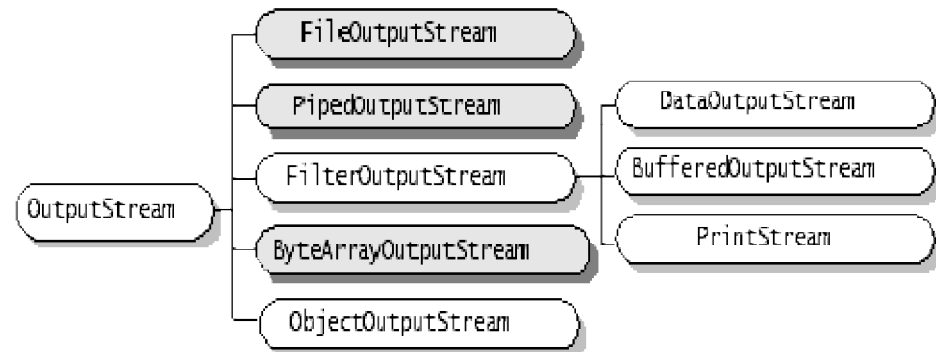
**System.out.println**("echo: "+inputLine);

Súbory: InputChar.java, InputLine.java

# OutputStream vs. PrintStream

kým OutputStream poskytuje:

- void write(*int* i)
- void write(byte[] pole)
- void write(byte[] pole, int offset, int pocet)
- close()
- flush()



podtrieda PrintStream rozširuje o:

- void print(int i), print(double d),..., print(Object o)
- void println(int i), println(double d),..., println(Object o)

# Ošetrenie výnimky

Ošetrenie výnimky v mieste, kde vznikla

```
public static void citaj() {
```

```
    . . . .
```

```
    try {
```

```
        int tmp = System.in.read();
```

```
        char c = (char)tmp;
```

```
    } catch (IOException e) {
```

```
        System.err.println("chyba ");
```

```
        e.printStackTrace();
```

```
    }
```

```
    . . . .  
}
```

profil read v java.io je  
public int **read**(byte[] b)  
**throws IOException**

// odchytenie výnimky

// vypíš zásobník volaní pri výnimke

Neošetrenie (propagovanie) výnimky:

výnimka IO vzniknúcšia v metóde citaj() nebude tam spracovaná, preto citaj() môže skončiť výnimkou

```
public static void citaj() throws IOException {
```

```
    int tmp = System.in.read();
```

```
    char c = (char)tmp;
```

```
}
```

# Propagovanie výnimky

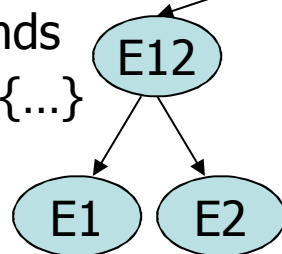
```
public static void citaj() throws IOException {  
    . . . .  
    try {  
        char c = (char)System.in.read ();  
    } catch (IOException e) { // odchytenie výnimky  
        System.err.println ("chyba pri citani");  
        throw e;           // propagovanie (vyvolanie) výnimky  
    }  
    . . . .  
}  
v mieste odkiaľ voláme metódu citaj():  
try {  
    citaj();  
} catch (IOException e) { // opätovné odchytenie výnimky  
    System.err.println("rachlo to");  
}
```

# Hierarchia výnimiek

class **Exception12** extends  
    **RuntimeException** {...}

class **Exception1** extends  
    **Exception12** {...}

class **Exception2** extends  
    **Exception12** {...}

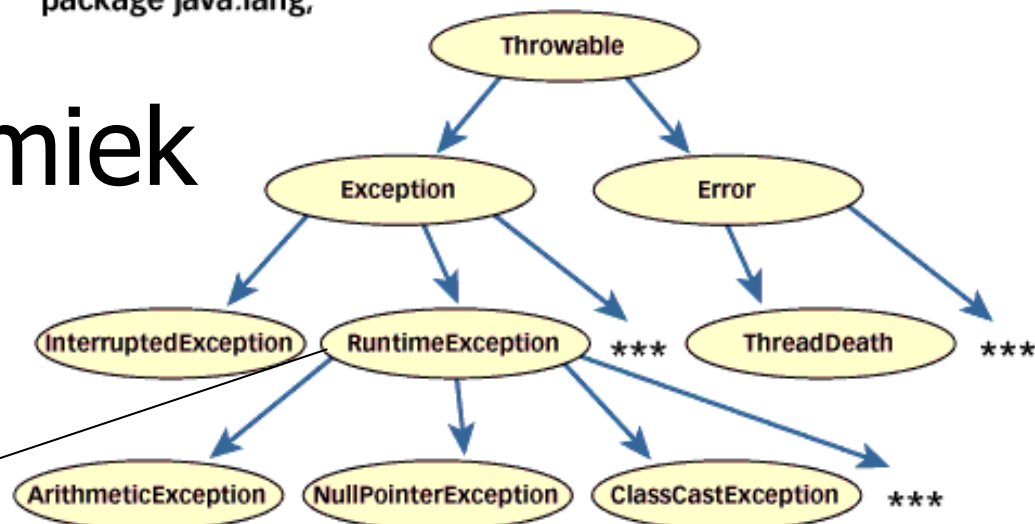


**Exception 2**  
**finally**

**Exception 12**  
**finally**

Súbor: DoException.java

package java.lang;

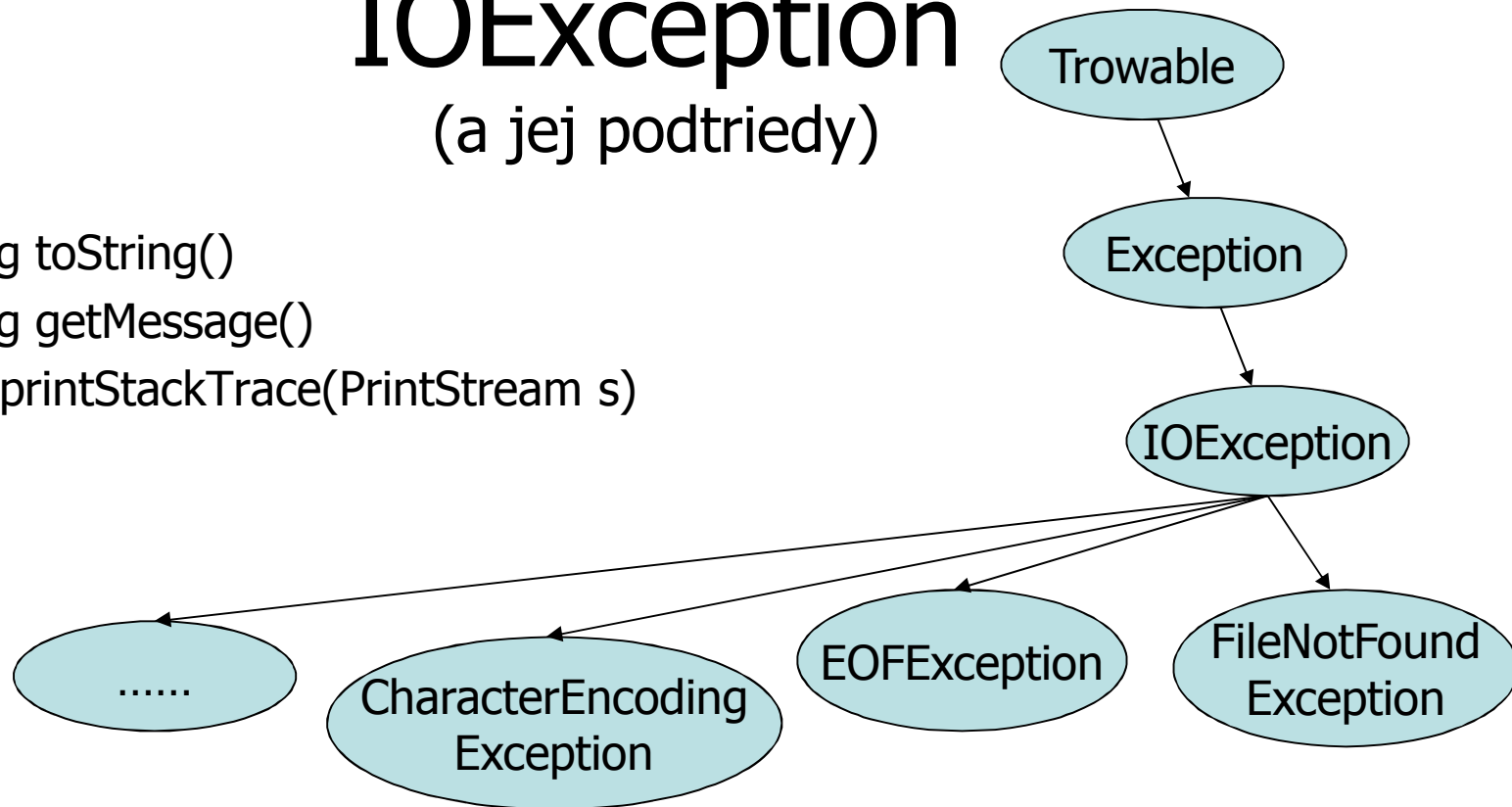


```
try { // kód produkujúci Ex1, Ex2 resp. Ex12
    switch((int)(3*Math.random())) {
        case 0: throw new Exception12();
        case 1: throw new Exception1();
        case 2: throw new Exception2(); }
    } catch (Exception1 e1) {
        System.out.println("Exception 1");
    } catch (Exception2 e2) {
        System.out.println("Exception 2");
    } catch (Exception12 e12) {
        System.out.println("Exception 12");
    } finally {
        System.out.println("finally");
    }
}
```

# IOException

(a jej podtriedy)

- String toString()
- String getMessage()
- void printStackTrace(PrintStream s)



zlozvyk:

```
public static void citaj() {  
    try {  
        int tmp = System.in.read();  
        char c = (char)tmp;  
    } catch (Exception e) {}  
}
```

# Kopírovanie súboru

```
import java.io.FileInputStream;  
import java.io.FileOutputStream
```

```
import java.io.*;
```

```
public class Kopiruj {  
    public static void main(String[] args) throws IOException {
```

```
        File frName = new File("a.txt");           // file descriptor  
  
                                                    // vytvor InputStream (bajt)  
        FileInputStream fr = new FileInputStream(frName);  
                                                    // vytvor OutputStream (bajt)  
        FileOutputStream fw = new FileOutputStream(new File("b.txt"));  
  
        long dlzka = frName.length();           // zisti dĺžku súboru  
        for (long i = 0; i < dlzka; i++)  
            fw.write(fr.read());  
  
        fr.close();  
        fw.close();  
    }  
}
```

```
        int c;  
        while ((c = fr.read()) != -1)  
            fw.write(c);           // zatvor oba streamy
```

Súbor: Kopiruj.java

# Obáľkovanie

(wrapovanie)

```
import java.io.InputStreamReader;  
import java.io.BufferedReader;
```

```
InputStreamReader in = new InputStreamReader(System.in);  
                                     // byte stream -> char stream  
BufferedReader bufIn = new BufferedReader(in); // buffrovaný vstup  
try {  
    System.out.println ("zadaj cislo = ");  
    String inputLine = bufIn.readLine(); // čítanie riadku  
    int tmp=Integer.parseInt(inputLine.trim()); // konverzia riadku na číslo  
    System.out.println ("echo = " + tmp); // echo prečítaného čísla  
    int sum = 0;  
    for(;;) {  
        inputLine = bufIn.readLine(); // čítanie riadkov  
        if (inputLine == null) break; // až po koniec vstupu (Ctrl-Z)  
        for(int j=0; j<inputLine.length(); j++) // spočítanie '*'  
            sum += (inputLine.charAt(j)=='*')?1:0;  
    }  
    System.out.println("pocet *="+sum);  
} catch (IOException e) {  
    System.err.println ("IO exception = " + e);  
}
```

Súbor: InputLines.java



# InputStream

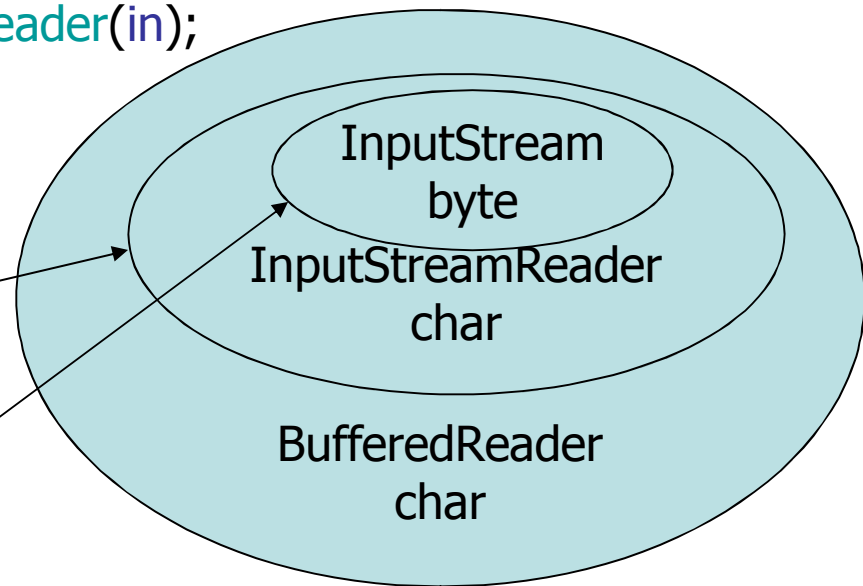
# BufferedReader

```
InputStreamReader in = new InputStreamReader(System.in);
```

```
BufferedReader bufIn = new BufferedReader(in);
```

InputStreamReader konvertuje  
ByteStream na CharStream  
(US-ASCII, ISO-8859-1, ...)

```
int read()  
int read(byte[] pole)  
int read(byte[] pole, int offset, int pocet)  
  
skip(long n)  
void mark(int lookAhead)  
void reset()
```



```
String readLine()  
for(;;) {  
    inputLine = bufIn.readLine();  
    if (inputLine == null) break;  
    . . . .  
}
```

# Konvertovanie súboru (utf/1250)

```
File file1 = new File("a_utf8.txt");
File file2 = new File("a_cp1250.txt");

InputStreamReader isr = new InputStreamReader( // kódovanie vstupného
                                             new FileInputStream(file1), "UTF-8"); // súboru
BufferedReader br = new BufferedReader(isr);
                                             // kódovanie vystupného
BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(
                                         new FileOutputStream(file2), "cp1250")); // súboru

int c;
System.out.println(isr.getEncoding()); // kódovanie vstupného súboru
for(;;) { // while (true) {...}
    String line = br.readLine(); // čítaj riadok
    if (line == null) break; // eof ak už niet čo čítať
    bw.write(line); // zapíš riadok
    bw.newLine(); // zapíš nový riadok
}
br.close();
bw.close();
```

Súbor: Konvertuj.java

# Formátovaný vstup

```
import java.util.Scanner;

Scanner scanner = new Scanner(System.in);
try {
    for(;;)
        if (scanner.hasNextInt())                // nasleduje int ?
            System.out.println ("Ivstup " + scanner.nextInt() + "\n");
        else if (scanner.hasNextFloat())          // nasleduje float ?
            System.out.println ("Fvstup " + scanner.nextFloat() + "\n");
        else if (scanner.hasNextBoolean())        // nasleduje boolean ?
            System.out.println ("Bvstup " + scanner.nextBoolean() + "\n");
        else
            break;
} catch (InputMismatchException e) {
    System.err.println("Mismatch exception:" + e );
}
```

```
3,14 15 true false 13 5,8 11
Fvstup 3,14
Ivstup 15
Bvstup true
Bvstup false
Ivstup 13
Fvstup 5,8
Ivstup 11
```

# Príklad Histogram

```
Scanner scan = new Scanner(  
    new FileInputStream(  
        new File("lenBody.txt")));  
  
TreeMap<Double,Integer> tmap =  
    new TreeMap<Double,Integer>();  
  
while(scan.hasNextDouble()) {  
    double val = Math.floor(  
        scan.nextDouble());  
    if (tmap.get(val)==null) tmap.put(val,1);  
    else tmap.put(val,1+tmap.get(val));  
}  
  
for(double i=0; i<50; i++)  
    System.out.println(i+"\t"+  
        "*****".  
        substring(0,(tmap.get(i)==null)?0:tmap.get(i)));
```

// scanner na vstupnom súbore  
// interná tabuľka  
// výskytov/početností  
// kým sú ...  
// čítanie realov  
// update tmap  
// tlač  
// histogramu

4.0\*  
5.0\*\*  
6.0  
7.0  
8.0\*  
9.0\*\*  
10.0\*  
11.0\*  
12.0\*\*  
13.0\*\*\*\*\*  
14.0\*  
15.0\*\*\*\*\*  
16.0\*\*  
17.0\*\*\*\*\*  
18.0\*\*\*  
19.0\*\*\*\*\*  
20.0\*\*\*\*  
21.0\*\*\*\*\*  
22.0\*\*  
23.0\*\*  
24.0\*  
25.0\*\*  
26.0\*\*\*  
27.0\*  
28.0  
29.0\*  
30.0\*

Súbor: Histogram.java

# Príklad Oddelovač

```
Scanner scan_csv = new Scanner(  
    new FileInputStream(  
        new File("body.csv")));  
  
double sucty[] = {0,0,0,0,0,0};  
int pocet = 0;  
  
scan_csv.useDelimiter("[;\r\n]");  
while(scan_csv.hasNextLine()) {  
    int i = 0;  
    /*String meno = */ scan_csv.next();  
    /*String priezvisko = */ scan_csv.next();  
    while(scan_csv.hasNextDouble())  
        sucty[i++] += scan_csv.nextDouble();  
    scan_csv.nextLine(); pocet++;  
}  
for(int i=0; i<sucty.length; i++)  
    System.out.println((i+1)+".priklad ma priemer: "+sucty[i]/pocet);
```

AlPsbeta;BachronpkovB;19.6;7;0.9;3.5;4;4.2  
Patrik;BaraniŮin;19.1;7;0.9;4.5;2.5;4.2  
Kšbroly;Belokostolsk<sup>2</sup>;13.9;2.5;4.9;2.5;4;0

// súčty bodov za jednotl.příklady  
// počet študentov  
  
// oddeľovač .csv  
// kým nie je posledný riadok  
// nastav index cvičenia  
// preskoč meno  
// preskoč priezvisko  
// kým sú body  
// čítaj body  
// na nový riadok  
  
// tlač priemerov

Súbor: Oddelovac.java

1.priklad ma priemer: 17.888524590163936  
2.priklad ma priemer: 5.049180327868853  
3.priklad ma priemer: 3.106557377049181  
4.priklad ma priemer: 3.30327868852459  
5.priklad ma priemer: 2.8524590163934427  
6.priklad ma priemer: 3.577049180327869

# Regulárne výrazy

[abc]	a, b, c	.	ľub.znak
[^abc]	okrem a, b, c	\d	[0-9]
[a-zA-Z]	a..z,A..Z (interval)	\D	[^0-9]
[a-d[m-p]]	[a-dm-p] (zjednotenie)	\s	[ \t\n\x0B\f\r]
[a-z&&[def]]	d, e, f (prienič)	\S	[^\s]
[a-z&&[^bc]]	[ad-z] (rozdiel')	\w	[a-zA-Z_0-9]
[a-z&&[^m-p]]	[a-lq-z] (rozdiel')	\W	[^\w]

X?	raz či vöbec X	^	zač.riadku
X*	viackrát X	\$	koniec riadku
X+	aspoň raz X	\b	hranica slova
X{n}	n krát X	\A	zač.vstupu
X{n,}	aspoň n krát X	\Z	koniec vstupu
X{n,m}	n až m krát X		

`[--]?([0-9]*\.[0-9]+|[0-9]+)`

`(19|20)\d\d[- ./](0[1-9]|1[012])[- ./](0[1-9]|12[0-9]|3[01])`

`^\.*\b(one|two|three)\b.*$`

```
Scanner fr = new Scanner(new FileReader(new File("d.txt")));
sc.useDelimiter("[\\r\\n]");
for(;sc.hasNextLine();sc.nextLine()) {
    String pat;
```

# Príklad Regular

```

if (sc.hasNext(pat="[A-Z][a-zA-Z]*"))
    System.out.println("meno: "+sc.next(pat));
else if (sc.hasNext(pat="\\d{3}\\s\\d{2}"))
    System.out.println("psc: "+sc.next(pat));
else if (sc.hasNext(pat="09\\d{2}\\s\\d{3}\\s\\d{3}"))
    System.out.println("mobil: "+sc.next(pat));
else if (sc.hasNext(pat="[0]\\d+[/-]\\d+"))
    System.out.println("pevna: "+sc.next(pat));
else if (sc.hasNext(pat=
    "(19|20)\\d\\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|12)[0-9]|3[01]"))
    System.out.println("dat.nar.: "+sc.next(pat));
else if (sc.hasNext(pat="(muz|zena)"))
    System.out.println("sex: "+sc.next(pat));
else if (sc.hasNext(pat="[a-zA-Z0-9_.]+[@][a-zA-Z0-9_.]+"))
    System.out.println("mail: "+sc.next(pat));
else
    break;
}
```

meno: Peter  
 psc: 821 06  
 dat.nar.: 1982-12-26  
 mobil: 0905 819 123  
 pevna: 02/2517293  
 sex: muz  
 mobil: peter@gmail.com  
 meno: Jano  
 psc: 034 21  
 pevna: 033/232329  
 mobil: jano@maznet11.com

Súbor: Regular.java

# Formátovaný výstup

```
import java.util.Formatter;  
import java.io.PrintStream;
```

```
Formatter formatter = new Formatter((OutputStream)System.out);
```

```
formatter.format ("Text%n");
```

```
boolean a_boolean = false;  
int    an_int    = 1234567;  
float  a_float   = 983.6f;
```

```
formatter.format ("boolean = %9b %n", a_boolean);  
System.out.printf("boolean = %9b %n", a_boolean);
```

```
formatter.format ("int    = %9d %n", an_int);  
System.out.printf("int    = %9d %n", an_int);
```

```
formatter.format ("float  = %9.3f %n", a_float);  
System.out.printf("float  = %9.3f %n", a_float);
```

```
formatter.flush ();  
formatter.close ();
```

```
Text  
boolean =      false  
int      =    1234567  
float    =    983.600
```

Súbor: Formater.java



# String.format

```
import java.util.*;
```

%[argument\_index\$][width][.precision]conversion

'b', 'B' boolean

's', 'S' String

'c', 'C' char

'd' decimal

'o' oktal

'x', 'X' hexa

'e', 'E' scientific notation

'f' float

'g', 'G' float or scientific

't', 'T' date/time

'n' new line

```
formatter.format("%4$s %3$s %2$s",  
"a", "b", "c", "d")
```

d c b

```
Calendar rightNow = Calendar.getInstance();
```

```
... = String.format("%1$tm %1$te,%1$tY", rightNow );
```

```
... = String.format("%1$tB %1$te,%1$tY",rightNow );
```

**10 29,2007**

**október 29,2007**

<http://java.sun.com/j2se/1.5.0/docs/api/java/util/Formatter.html>

# Pokračovanie

- ilustrovali sme základné možnosti tried **InputStream/OutputStream**,
- princíp obáľkovania (napr. Buffered..., Scanner s reg.výrazmi),

Pokračovanie:

- serializácia – spôsob ako ľubovoľný objekt pretransformovať na postupnosť bajtov tak, aby sme ho vedeli spätne reštaurovať. Využitie: zápis a čítanie objektu do/z súboru, pajpy, ...
- práca s adresármi
- priamy prístup k dátam v súbore – najčastejšie v binárnom súbore, keď pristupujeme k dátam v súbore cez ich adresu – pozíciu v súbore, trieda `RandomAccessFile`,
- sekvencovaný vstup – zreťazenie viacerých vstupov do jedného streamu – trieda `SequenceInputStream`,
- komprimované súbory – triedy `ZipInputStream`, `ZipOutputStream`,

# Serializácia

ObjectOutputStream  
ObjectInputStream

## ObjectOutputStream

```
FileOutputStream out = new FileOutputStream("AVL.obj");  
ObjectOutputStream fs = new ObjectOutputStream(out);  
fs.writeObject("avl"); // zapíš String  
fs.writeObject(s);      // AVLTree  
fs.flush();              // fs.close();
```

Obsah súboru AVL.obj:  
~í | t ^ avlsr AVLTree;6û ^ | Gš >\_ L ^ roott  
LAVLNode;xpsr  
AVLNode◀Û=%ö½â~\_ ^ | xL ^ leftq ~\_ L  
| rightq ~\_ xp ssq ~ ^ /sq ~ ^  
sq ~ ^ ppsq ~ ^ ppsq ~ ^ Qsq ~ ^  
@ppsqs ~ ^ bppsqs ~ ^ "sq ~ ^ spsq ~ ^  
!psq ~ ^ .pp

## ObjectInputStream

```
FileInputStream in = new FileInputStream("AVL.obj");  
ObjectInputStream is = new ObjectInputStream(in);  
String str = (String)is.readObject();  
AVLTree ss = (AVLTree)is.readObject();
```

Objekt musí byť serializovateľný: Serializable Interface

```
class AVLNode implements Serializable  
class AVLTree implements Serializable
```

Súbor: AVLTree.java

# Ulož a prečítaj konfiguráciu

Ak máme uložiť a opätovne vedieť načítať konfiguráciu, napr. hry, a nechceme vytvárať vlastný formát, napr. v textovom súbore, použijeme serializáciu.

```
public class PiskyStav implements Serializable {  
    char piskvorky[][] = new char [10][10]; // reprezentácia plochy  
    boolean XNaTahu = true; // stav hry  
    int pocetXPiskvoriek = 0; // ďalšie informácie o hre  
    int pocetOPiskvoriek = 0;  
  
    public static PiskyStav load(String fileName) throws Exception {  
        ObjectInputStream is = new ObjectInputStream(new FileInputStream(fileName));  
        return (PiskyStav)is.readObject();  
    }  
    public void save(String fileName) throws Exception {  
        ObjectOutputStream fs = new ObjectOutputStream(new FileOutputStream(fileName));  
        fs.writeObject(this);  
        fs.close();  
    }  
}
```

Súbory: PiskyStav.java a Piskvorky.java

```
import java.io.File;
```

# Súbory a adresáre

```
String aktDir = System.getProperty("user.dir"); // adresár v ktorom beží aplik.
```

Relatívna cesta:

```
File f = new File("TMP" + File.separator + "a.txt"); // TMP\a.txt  
f.getAbsolutePath() // C:\borovan\java\eclipse\Subory\TMP\a.txt  
f.getName() // a.txt  
f.getParent() // TMP  
f.exists() // true  
f.createNewFile()  
f.isFile() // true
```

Absolútna cesta k súboru:

```
File d = new File(aktDir, "a.txt"); // C:\borovan\java\eclipse\Subory\a.txt  
d.getAbsolutePath() //C:\borovan\java\eclipse\Subory\a.txt  
d.getName() // a.txt  
d.getParent() // C:\borovan\java\eclipse\Subory  
d.exists() // false  
d.mkdir  
d.isDirectory() // true
```

```
import java.io.File;
```

# Vlastnosti súborov

```
File subor = new File("b.txt");  
File adr = new File("TMP");
```

```
subor.lastModified()  
adr.lastModified()
```

```
// Sun Oct 24 18:32:12  
// Sun Oct 24 18:32:12
```

```
subor.length()  
adr.length()
```

```
File iny = new File("c.txt");  
subor.renameTo(iny);  
adr.renameTo(new File("TMP-OLD"));
```

```
// premenuje súbor b.txt na c.txt  
// premenuje TMP na TMP-OLD
```

```
subor.delete();  
adr.delete();  
iny.delete();
```

```
// nevymaže c.txt ale b.txt  
// nevymaže TMP-OLD ale TMP  
// vymazanie c.txt
```

# System properties

**System.getProperty("user.dir");**

"java.class.path"	Java classpath
"java.class.version"	Java class version number
"java.home"	Java installation directory
"java.vendor"	Java vendor-specific string
"java.vendor.url"	Java vendor URL
"java.version"	Java version number
"line.separator"	Line separator
"os.arch"	Operating system architecture
"os.name"	Operating system name
"os.version"	Operating system version
"path.separator"	Path separator (for example, ":")
"user.dir"	User's current working directory
"user.home"	User home directory
"user.name"	User account name

```
java.vm.version=1.5.0-b64
java.vm.vendor=Sun Microsystems
path.separator=;
user.dir=C:\Documents and
Settings\peterb\GetP...
os.arch=x86
os.name=Windows 2000
sun.jnu.encoding=Cp1252
myProperty=myValue
. . . . .
```

# Výpis adresára

```
String menaAktDir = System.getProperty("user.dir");  
File aktDir = new File(menaAktDir);
```

```
String[] mena = aktDir.list();    // výpis adresára  
if (mena != null)  
    for (int i = 0; i < mena.length; i++)  
        if (mena[i].indexOf(".java")>0) // ak prípona .java  
            System.out.println(mena[i]);
```

```
File[] subory = aktDir.listFiles();  
if (subory != null)  
    for (int i = 0; i < subory.length; i++)  
        if (subory[i].length() > 1024)  
            System.out.println(subory[i].getName()+  
                                "\t"+subory[i].length());
```

Súbor: DirList.java

DirList.java  
DoException.java  
Exception1.java  
Exception12.java  
Exception2.java  
Formater.java  
InputChar.java  
InputLine.java  
InputLines.java  
Patern.java  
Scanner.java

DirList.class	1419
Formater.class	1736
InputLine.class	1139
InputLines.class	1689
Patern.class	1293
Scanner.class	2051
Scanner.java	1154



# Filtrovanie adresára podľa prípony

```
class FilterPripony implements FilenameFilter {  
    String maska;  
    FilterPripony(String maska) { // zapamätaj si %ile mask+  
        this.maska = maska;  
    }  
    public boolean accept(File dir, String name) { // padnú do výberu, ak true  
        if (name.lastIndexOf(maska) > 0)  
            return true;  
        else // t.j. return name.lastIndexOf(maska)>0;  
            return false;  
    }  
}
```

**FilterPripony FilterPr = new FilterPripony(".java");**  
**String[] mena = aktDir.list(FilterPr);**

DirList.java  
DoException.java  
Exception1.java  
Exception12.java  
Exception2.java  
FilterPripony.java

FilterVelkosti.java  
Formater.java  
InputChar.java  
InputLine.java  
InputLines.java  
Patern.java  
Scanner.java

# Filtrovanie adresára podľa veľkosti

```
class FilterVelkosti implements FilenameFilter {  
    int velkost;  
    FilterVelkosti(int velkost) {                // zapamätaj si ve kos súborov  
        this.velkost = velkost;  
    }  
    public boolean accept(File dir, String name) {  
        File f = new File(dir, name);            // ako sa dosta k ve kosti súboru  
        if (f.length() > velkost)  
            return true;  
        else  
            return false;  
    }  
}  
  
FilterVelkosti FilterVel = new FilterVelkosti(2048);  
File subory = aktDir.listFiles(FilterVel);
```

# Adresáre rekurzívne

```
class FilterAdresara implements FilenameFilter {
    public boolean accept(File dir, String name) {
        File f = new File(dir, name);
        return f.isDirectory();
    }
}

static void rekVypis(String aktualnyAdr) {
    String[] mena;
    File aktDir = new File(aktualnyAdr);
    FilterAdresara FilterAdr = new FilterAdresara();
    mena = aktDir.list(FilterAdr);
    if (mena != null) {
        for (int i = 0; i < mena.length; i++) {
            String podadr = new String (aktualnyAdr + File.separator + mena[i]);
            System.out.println(podadr);
            rekVypis(podadr);
        }
    }
}

// zaujímajú ma len adresáre
rekVypis("c:\\Program Files");

c:\\Program Files\\Accessories
c:\\Program Files\\Accessories\\Imagevu
c:\\Program Files\\ACD Systems
c:\\Program Files\\ACD Systems\\ACDSee
c:\\Program Files\\ACD Systems\\ACDSee

// chod' do podadresára
```

Súbor: RecDirList.java

# Priamy prístup

priamy prístup k dátam v súbore – najčastejšie v binárnom súbore, keď prístupujeme k dátam v súbore cez ich adresu – pozíciu v súbore

Trieda: RandomAccessFile

- `new RandomAccessFile("a.txt", "r");` // len čítanie zo súboru
- `new RandomAccessFile("b.txt", "rw");` // čítanie a zápis do súboru

Pohyb s súbore:

- `int skipBytes(int n)` // preskoč n bytov dopredu
- `void seek(long pos)` // skoč na pozíciu pos
- `long getFilePointer()` // vráť pozíciu, kde sa nachádzaš

```
File f = new File("filename");
RandomAccessFile raf = new RandomAccessFile(f, "rw");
char ch = raf.readChar();
raf.seek(f.length()); // seek to end
raf.writeChars("<" + ch + ">"); // append
raf.close();
```

java.io.SequenceInputStream;

# Sekvencovaný vstup

Trieda `SequenceInputStream` slúži na zret'azenie viacerých `InputStream`ov do jedného tak, že pri čítaní z neho nezbadáme, kedy dochádza k prechodu medzi posebeidúcimi vstupmi

```
SequenceInputStream s =  
    new SequenceInputStream(  
        new FileInputStream("a.txt"),    // prvý vstup  
        new FileInputStream("b.txt"));    // druhý vstup  
int c;  
try {  
    while ((c = s.read()) != -1)  
        System.out.write(c);  
    s.close();  
} catch (Exception e) { }  
}
```

# Zip file1+file2->file.zip

```
String[] filenames = {"file1", "file2"};
String outFilename = "file.zip";
ZipOutputStream out = new ZipOutputStream(
    new FileOutputStream(outFilename));
for (int i=0; i<filenames.length; i++) {
    FileInputStream in = new FileInputStream(filenames[i]);
    out.putNextEntry(new ZipEntry(filenames[i]));
    int len;
    byte[] buf = new byte[1024];
    while ((len = in.read(buf)) > 0)
        out.write(buf, 0, len);
    out.closeEntry();
    in.close();
}
out.close();
```