

# 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`, `java.nio`
- **Stream** (ako jednosmerný sekvenčný tok dát) uvidíme v dvoch podobách
  - **byte** (8-bit) **stream** (podtriedy `InputStream/OutputStream`),
  - **16-bit** (unicode/character) **stream** (podtriedy `Reader/Writer`),
- **try-catch-finally/try-catch-with resources**
- výnimky (ešte znova a podrobnejšie),
- obáľkovanie (wrapovanie),
- formátovaný vstup a výstup (`scan`, `printf`),
- regulárne výrazy (bez nedeterministických automatov :D,
- serializácia,
- práca so súbormi a adresármi

Cvičenie:

- 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) (<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:

- keď sme sa učili čítať, najprv sme zvládli jednotlivé písmenká, M, A, M, A
- potom nás naučili čítať oddelené slová, vety, odstavce, celú rozprávku, román,...

na pochopenie (spracovanie):

- rozprávky, musíme vedieť spracovať text odstavca, ktorý sa obyčajne vojde na stranu/obrazovku/do ohraničeného buffra
- vety, musíme na konci vety si pamätať oi. podmet, o ktorom je veta – opäť iný buffer – lokálna krátkodobá pamäť.

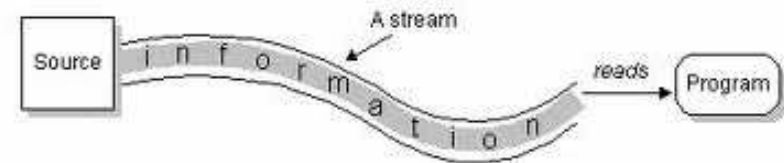
**aj súbory** (alias rozprávky) preto **vieme čítať po:**

- bajtoch (8 bitov),
- znakoch (16-bit), ktoré majú rôznu interpretáciu v rôznych kódovaniach,
- riadkoch (textové), či blokoch (binárne),
- slovách. číslach, reálnych číslach (formátovaný vstup),
- regulárnych výrazoch
- vetách či paragrafoch (regulárne výrazy)

컴퓨터orientedka

# InputStream/OutputStream

(byte stream)

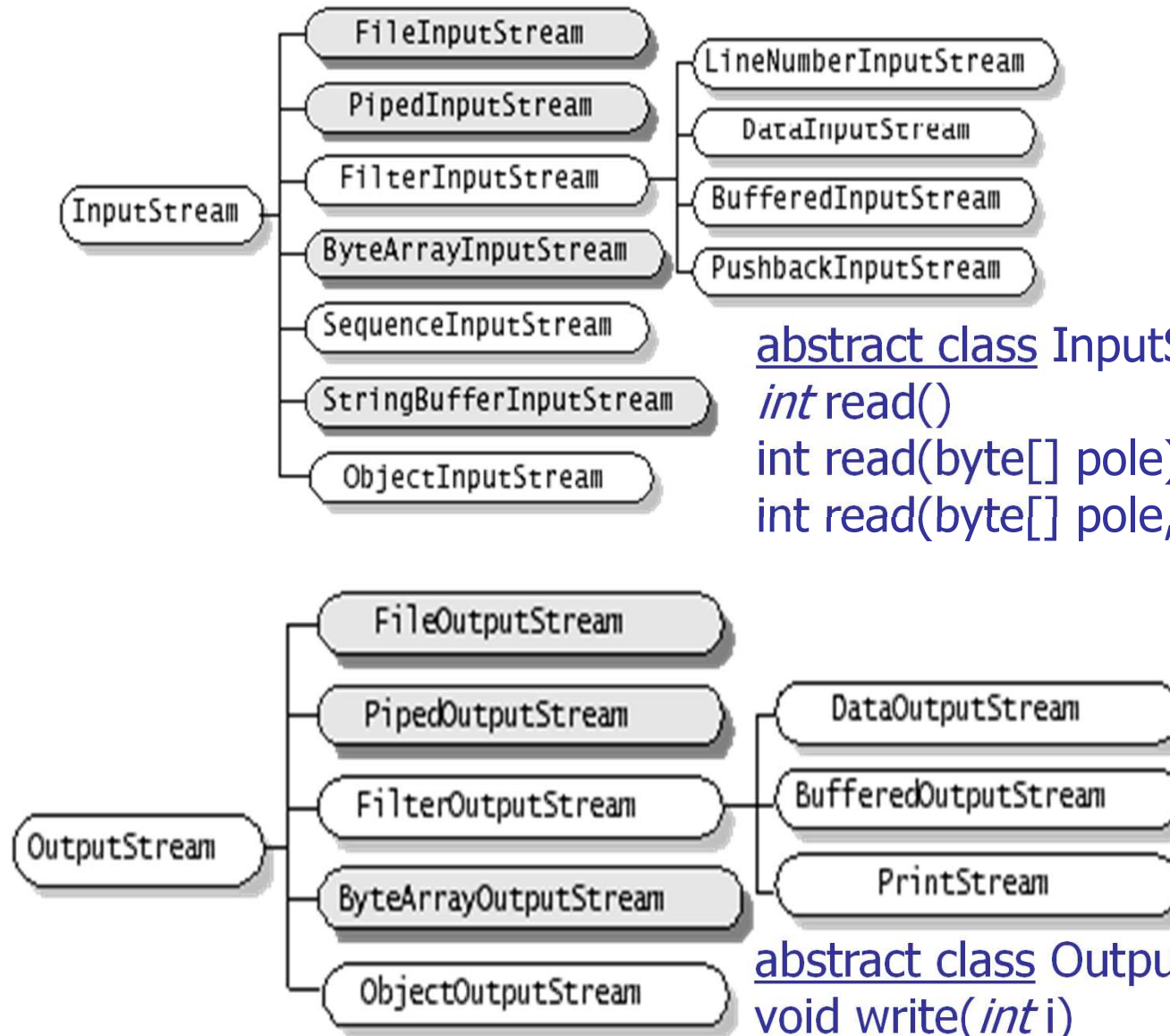


**InputStream/OutputStream** je jednosmerný sekvenčný tok bajtov (8bits)  
zdrojom/cieľom môže byť

- pole bajtov → [ByteArrayInputStream](#)(byte[] buf)
- String → [StringBufferInputStream](#)([String](#) s)
- súbor → [FileInputStream](#)([File](#) file)  
[FileInputStream](#)([String](#) name)
- pipe → [PipedInputStream](#)([PipedOutputStream](#) src)
- sekvencia iných zret'azených streamov (len pre InputStream) → [SequenceInputStream](#)([InputStream](#) s1, [InputStream](#) s2)  
[SequenceInputStream](#)([Enumeration](#) <? extends [InputStream](#) > e)

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

# InputStream/OutputStream (byte)



abstract class **InputStream**:

*int* read()

*int* read(*byte*[] *pole*)

*int* read(*byte*[] *pole*, *int* *offset*, *int* *length*)

abstract class **OutputStream**:

*void* write(*int* *i*)

*void* write(*byte*[] *pole*)

*void* write(*byte*[] *pole*, *int* *offset*, *int* *length*)

# Writer/Reader

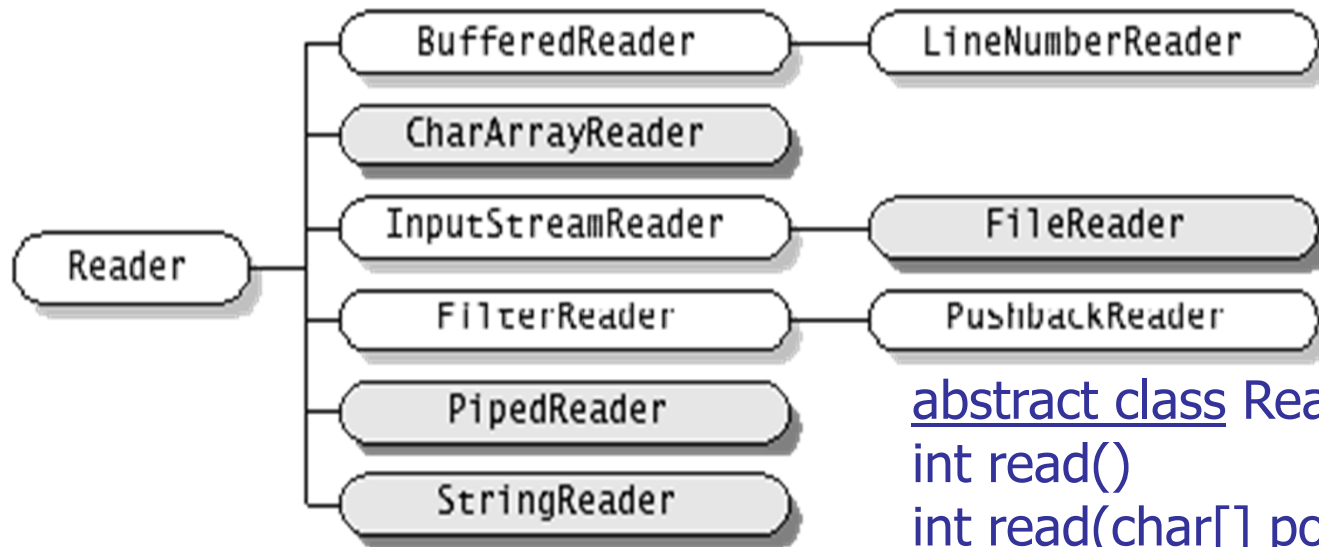
(char 16-bit stream)

**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)
- pipe → [OutputStreamWriter](#)([OutputStream](#) out, [String](#) charsetName)
- súbor → [PipedWriter](#)([PipedReader](#) snk)
- súbor → 

[FileWriter](#)([String](#) fileName)  
[FileWriter](#)([File](#) file)

# Character (unicode) Streams (16bit)

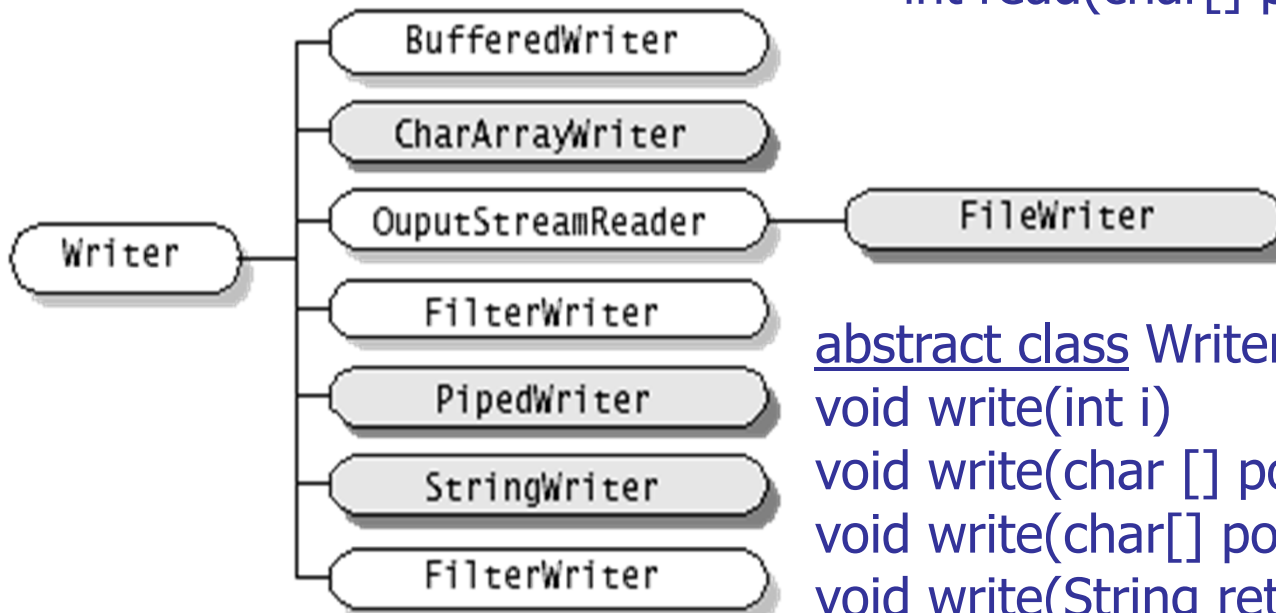


abstract class Reader

int read()

int read(char[] pole)

int read(char[] pole, int offset, int pocet)



abstract class Writer

void write(int i)

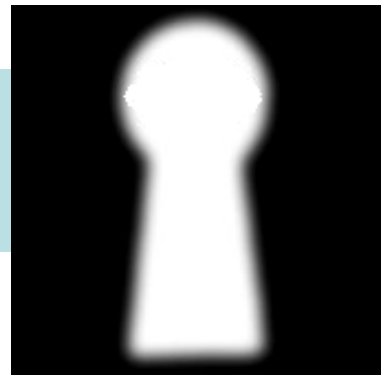
void write(char [] pole)

void write(char[] pole, int offset, int pocet)

void write(String retazec)

void write(String retazec, int offset, int pocet)

# 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);

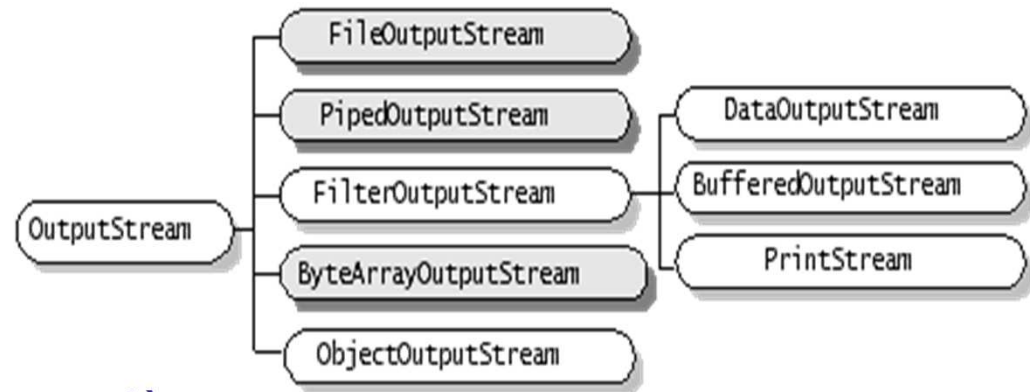
Súbory: InputChar.java, InputLine.java



# OutputStream vs. PrintStream

kým OutputStream poskytuje:

- `void write(int i)`
- `void write(byte[] pole)`
- `void write(byte[] pole, int offset, int pocet)`
- `close()` – uzavrie out/input stream
- `flush()` – vyprázdni buffer do streamu



## System.out.println()



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

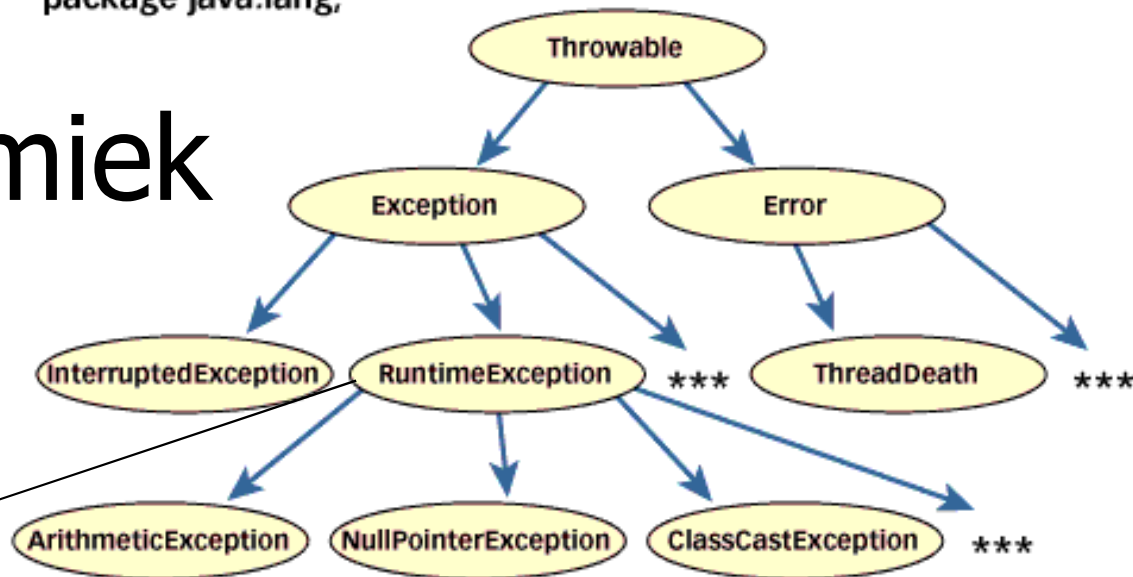
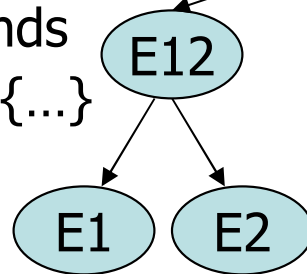
```
public static void citaj() throws IOException {  
    . . . .  
    try {  
        char c = (char)System.in.read ();  
    } catch (IOException e) { // odchytenie výnimky  
        System.err.println ("chyba pri citani");  
        throw e;             // propagovanie (vyvolanie) výnimky  
    }  
    . . . .  
}  
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** {...}



```

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");
    }
}
  
```

**Exception 2**  
**finally**

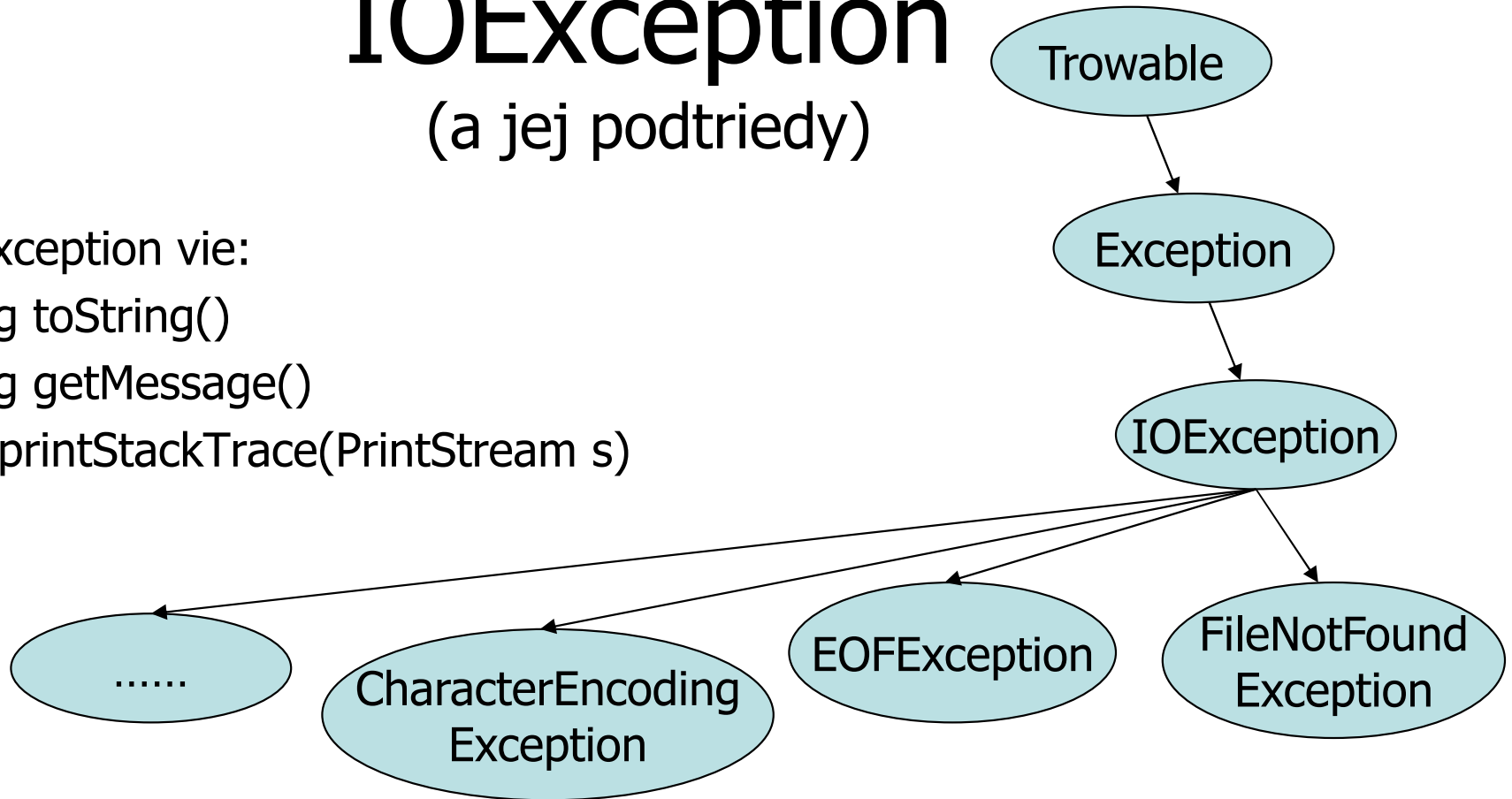
**Exception 12**  
**finally**

# IOException

(a jej podtriedy)

Každá exception vie:

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



**Zlozvyk (Bad practice) :**

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

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

# Kopírovanie súboru

```
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();
```

```
                                                    // zatvor oba streamy
```

```
        int c;
```

```
        while ((c = fr.read()) != -1)
```

```
            fw.write(c);
```

```
    }  
}
```

# Kopírovanie súboru

```
try {  
    FileInputStream fr = null;  
    FileOutputStream fw = null;  
    File frName = new File("a.txt");  
    try {  
        fr = new FileInputStream(frName);  
        fw = new FileOutputStream(new File("b.txt"));  
        int c;  
        while ((c = fr.read()) != -1) fw.write(c);  
    } catch (FileNotFoundException e) {  
        System.err.println(e.getMessage()); e.printStackTrace();  
    } finally {  
        if (fr != null) fr.close();  
        if (fw != null) fw.close();  
    }  
} catch (IOException e) {  
    System.err.println(e.getMessage()); e.printStackTrace();  
}
```

# try-with-resources

(Java 7)

- interface **Closeable**/**AutoCloseable** predpisuje jedinú metódu **close()**
- všetky triedy à la stream, reader, writer, ... implementujú **AutoCloseable**
- syntax (od JDK7) **try-with** resources zjednodušuje konštrukciu bez finally

```
File frName = new File("a.txt");
```

```
try (  
    FileInputStream fr = new FileInputStream(frName);  
    FileOutputStream fw = new FileOutputStream(new File("b.txt"))  
    ) {  
    int c;  
    while ((c = fr.read()) != -1)  
        fw.write(c);  
    }  
}
```

- tento kód môže skončiť s **FileNotFoundException**, resp. **IOException**
- preto to musí niekto odchytiť a ošetriť



# try-with-resources

(jdbc)

```
String sql = "select * from books";
```

```
List list = new ArrayList();
```

```
try (
```

```
    Connection con = ds.getConnection(); // AutoCloseable
```

```
    PreparedStatement ps = con.prepareStatement(sql) ) { // AutoCloseable
```

```
try ( ResultSet rs = ps.executeQuery() ) { // AutoCloseable
```

```
    while (rs.next()) {
```

```
        list.add(rs.getInt("id"));
```

```
    }
```

```
}
```

```
} catch (SQLException e) {
```

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

```
}
```

# try-catch-catch-catch-catch-catch-catch-...

(Java 7)

```
try {  
    ...  
} catch (IOException e) {  
    System.err.println(e.getMessage()); e.printStackTrace(); throw e;  
} catch (SQLException e) {  
    System.err.println(e.getMessage()); e.printStackTrace(); throw e;  
}
```

v Java 7:

```
try {  
    ...  
} catch (IOException | SQLException e) {  
    System.err.println(e.getMessage());  
    e.printStackTrace();  
    throw e;  
}
```

# Obálkovanie

(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-D)  
        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);  
}
```

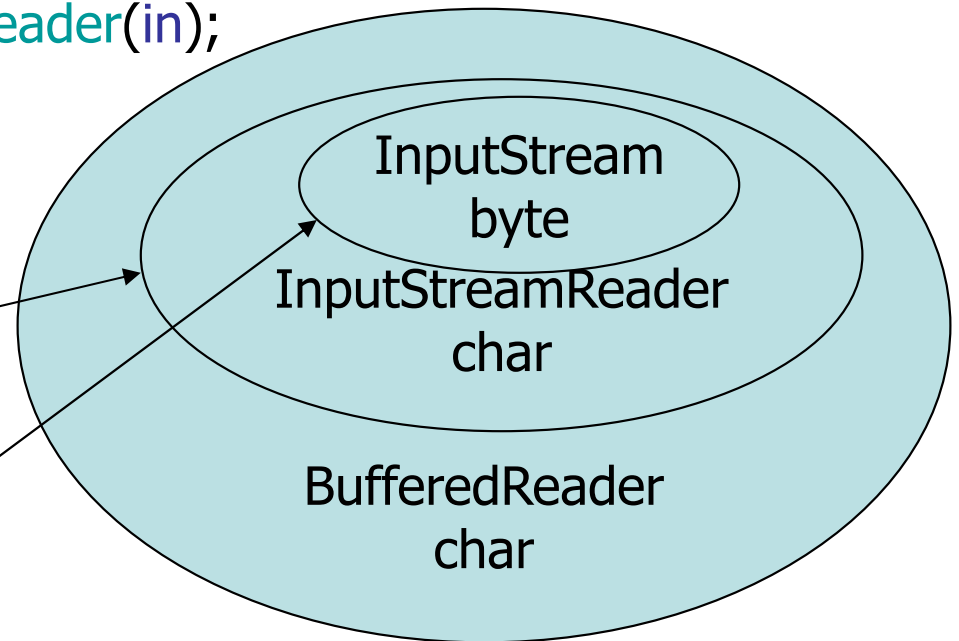
# InputStream → BufferedReader

```
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();
```

# java.nio

(stream API)

java.nio.file.Files ponúka readAllLines() na prečítanie riadkov celého súboru

```
try {  
    List<String> lines = Files.readAllLines(Paths.get("subor.txt"));  
  
    for (String line : lines) System.out.println(line);  
  
    lines.forEach(System.out::println);  
  
    System.out.println(  
        lines.stream().mapToInt(line -> line.length()).sum());  
    lines.stream().mapToInt(String::length).sum();  
  
    System.out.println(  
        lines.stream().mapToLong(  
            line -> line.chars().filter(ch -> ch == '*').count()  
        ).sum());  
} catch (IOException e) {  
    e.printStackTrace();  
}
```

# 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
```

Súbor: Scanner.java

# Príklad Histogram

```
Scanner scan = new Scanner(  
    new FileInputStream(  
        new File("lenBody.txt")));  
  
Map<Double,Integer> tmap =   
    TreeMap<>();  
  
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*
```



# Príklad Histogram

```
try {  
    List<String> lines = Files.readAllLines(  
        Paths.get("lenBody.txt"));  
    var histo = lines.stream()  
        .mapToDouble(  
            line -> Math.floor(Double.valueOf(line)))  
        .boxed()  
        .collect(Collectors.groupingBy(d -> d));  
  
    for(double i=0; i<31; i++)  
        System.out.println(  
            i + "\t" + "*" .repeat((histo.get(i) == null)?0:histo.get(i).size()));  
  
    IntStream.range(0,31).boxed().forEach(i->  
        System.out.println(  
            i + "\t" + "*" .repeat((histo.get((double)i) == null)?  
                0:histo.get((double)i).size()))  
        );  
} catch (Exception e) {  
    System.out.println ("Mismatch exception:" + e );  
}
```

```
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*
```

# Príklad Oddel'ovač

AlPsbeta;Bachronφkovß;19.6;7;0.9;3.5;4;4.2  
Patrik;BaraniÜin;19.1;7;0.9;4.5;2.5;4.2  
KBroly;Belokostolsk²;13.9;2.5;4.9;2.5;4;0

```
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);
```

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

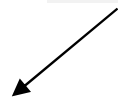
1.priklad ma priemer: 17.888524590163936  
2.priklad ma priemer: 5.049180327868853  
3.priklad ma priemer: 3.106557377049181  
4.priklad ma priemer: 3.30327868852459  
5.priklad ma priemer: 2.8524590163934427  
6.priklad ma priemer: 3.577049180327869

Súbor: Oddelovac.java

# Príklad Oddel'ovač

```
Scanner scan_csv = new Scanner(  
    new FileInputStream(  
        new File("body.csv")));
```

AlPsbeta;Bachronφkovß;19.6;7;0.9;3.5;4;4.2  
Patrik;BaraniÜin;19.1;7;0.9;4.5;2.5;4.2  
Kßroly;Belokostolsk²;13.9;2.5;4.9;2.5;4;0



```
while(scan_csv.hasNextLine()) {  
    String line = scan_csv.nextLine();  
    if (line == null) break;  
    String[]parts = line.split(";");  
    ... = Double.valueOf(parts[i].trim());  
}
```

# 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 (prienik)	\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 vôbec 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]|1[2][0-9]|3[01])

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

# Príklady reg.výrazov

```
Scanner sc = new Scanner(new StringReader("java"));
```

```
System.out.println(
```

```
    sc.hasNext("java")
```

```
"java"
```

```
    sc.hasNext("java.")
```

```
"java9"
```

```
    sc.hasNext("j.*")
```

```
"java9"
```

```
    sc.hasNext(".*v.*")
```

```
"java9"
```

```
    sc.hasNext("[a-z0-9]+")
```

```
"java9"
```

```
    sc.hasNext("[^python]+")
```

```
"java9"
```

```
    sc.hasNext("[A-F[0-9]]+")
```

```
"FF00FF"
```

```
    sc.hasNext("[A-Q&&[K-Z]]+")
```

```
"KLM"
```

```
    sc.hasNext("[A-Z&&[^F-H]]+")
```

```
"KLM"
```

```
    sc.hasNext("\\d{3}")
```

```
"158"
```

```
    sc.hasNext("(\\d)\\1\\d")
```

```
"558"
```

```
    sc.hasNext("(\\d\\d\\d)\\1")
```

```
"567567"
```

```
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]|1[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  
 mail: peter@gmail.com  
 meno: Jano  
 psc: 034 21  
 pevna: 033/232329  
 mail: jano@maznet11.com

# Príklady reg.výraz



`string.matches(regex)` vráti **true**

Regexp:

```
"java"
".*java.*"
"^java.*"
"[a-z0-9]+"
"^python)+"
"[A-F[0-9]]+"
"[A-Q&&[K-Z]]+"
"[A-Z&&[^F-H]]+"
"\\d{3}"
"(\\d)\\1\\d"
"(\\d\\d\\d)\\1"
"[A-Z][a-zA-Z]*"
"\\d{3}\\s\\d{2}"
"09\\d{2}\\s\\d{3}\\s\\d{3}"
"[0]\\d+[/-]\\d+"
"(muz|zena)"
"[a-zA-Z0-9_]+@[a-zA-Z0-9_]+"
"(19|20)\\d\\d[- /.](0[1-9]|1[012])[- /.](0[1-9]|1[2][0-9]|3[01])"
```

string->true:

```
"java"
"python java kotlin"
"java python kotlin"
"java9"
"java9"
"FF00FF"
"KLM"
"KLM"
"158"
"558"
"567567"
"Peter"
"821 06"
"0905 819 123"
"02/2517293"
"muz"
"jano.mrkva@.com"
"1982-12-26"
"2021-02-29" ☹️
```

**false**

string->>false:

```
"_java " // totálna zhoda
"python ja_va kotlin" // niekde v reťazci
"_java python kotlin" // na začiatku riadku
"" // malé písmena, cifry, >=1x
"kotlin" // okrem p,y,t,h,o,n
"ff00ff" // A-F zjednotenie 0-9
"SWISS" // A-Q prienik K-Z
"FRANCE" // A-Q rozdiel K-Z
"1234" // tri cifry
"123" // xxd - rovnaké prvé 2cifry
"567576" // xyzxyz-dve rovnaké trojice
"peter" // meno začína veľkým písm.
"82106" // PSČ
"4212 777 333" // mobil
"02 2517293" // pevná
"dieta" // pohlavie
"jano/mrkva@com" // ≈email
"26.12.1982" // yyyy-mm-dd
```

# Grupy

Grupy v regexpe zatvoríme do ( ... )

```
String regexp = "(\\d{4})(\\d{2})(\\d{2})/(\\d{4}|\\d{3})";
Pattern pattern = Pattern.compile(regexp);
Matcher matcher = pattern.matcher("20200225/123");
if (matcher.matches()) {
    if (matcher.groupCount() > 0) {
        for (int g = 0; g <= matcher.groupCount(); g++) {
            System.out.println("group " + g + " is " + matcher.group(g));
        }
    }
}
```

group 0 is 20200225/123  
group 1 is 2020  
group 2 is 02  
group 3 is 25  
group 4 is 123

- grupa 0 je vždy celý reťazec

matcher.groupCount() podgrúp

for (int g = 0; g <= matcher.groupCount(); g++)



# 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**

# 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 adresármí
- 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 – zretáženie viacerých vstupov do jedného streamu – trieda `SequenceInputStream`,
- komprimované súbory – triedy `ZipInputStream`, `ZipOutputStream`,

# Serializácia

## 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•G%> •L•roott  
    L•AVLNode;xpsr  
AVLNode•Ù=œö½â•I•xL•leftq~ L•rightq~  
  xp ssq~• /sq~•  
sq~• ppsq~• ppsq~• Qsq~• @ppsqsq~•  
~• bppsqsq~• •sq~• „ppsqsq~• |psq~•  
•pp
```

## ObjectInputStream

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

Objekt musí byť serializovateľný: Serializable Interface

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

# 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));
        PiskyStav ps =(PiskyStav)is.readObject();
        is.close();
        return ps;
    }

    public void save(String fileName) throws Exception {
        ObjectOutputStream fs = new ObjectOutputStream(new FileOutputStream(fileName));
        fs.writeObject(this);
        fs.close();
    }
}
```

# Byte-Stream Input/Output

Čítanie EXIF formátu z JPEG obrázku

Čítanie .exe súboru

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

Čítanie .class súboru

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

Čítanie a zápis komprimovaného súboru (váš .zyp)

- [http://en.wikipedia.org/wiki/ZIP\\_file\\_format](http://en.wikipedia.org/wiki/ZIP_file_format)

Čítanie zo zaheslovaného .zip súboru

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");`**

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

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

```
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());
```

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 "file 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 ...

(Java 8)

```
String[] mena2 =  
    aktDir.list((dir, name) -> name.lastIndexOf(".java") > 0 );  
  
File[] subory2 =  
    aktDir.listFiles((dir, name) ->  
        new File(dir, name).length() > 2048 );
```

# Filtrovanie adresára podľa veľkosti

```
class FilterVelkosti implements FilenameFilter {  
    int velkost;  
    FilterVelkosti(int velkost) {                // zapamätaj si veľkosť súborov  
        this.velkost = velkost;  
    }  
    public boolean accept(File dir, String name) {  
        File f = new File(dir, name);           // ako sa dostať k veľkosti 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();
    }
}

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

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);
        }
    }
}

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

# 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();
```

# Sekvencovaný vstup

Trieda `SequenceInputStream` slúži na zretáženie 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();
```