



Vstup a výstup

Peter Borovanský
KAI, I-18

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

I/O stream nemá nič spoločné so StreamAPI

dnes bude:

- úvod do package java.io
- **stream** (ako jednosmerný sekvenčný tok dát) uvidíme v dvoch podobách
 - byte (8-bit) stream (tryasy InputStream/OutputStream),
 - 16-bit (unicode/character) stream (tryasy Reader/Writer),
- **try-catch-finally** výnimky (znova a podrobnejšie),
- obálkovanie (wrapovanie),
- formátovaný vstup a výstup (scan, printf),
- regulárne výrazy (bez nedeterministických automatov :D,
- serializácia,
- práca so súbormi a adresármí

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/naukte-se-javu-prace-se-vstupy-a-vystupy-1/>,
- <http://interval.cz/clanky/naukte-se-javu-prace-se-vstupy-a-vystupy-2/>,

Buffre

Vstup:

- ked' sme sa učili čítať, najprv sme zvládli jednotlivé písmená, 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.

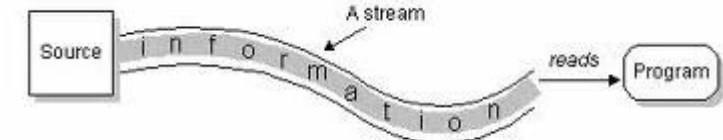
aj súbory (alias rozprávky) preto vieme čítať po:

- 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)

캐리어리에드카

InputStream/OutputStream

(byte stream)

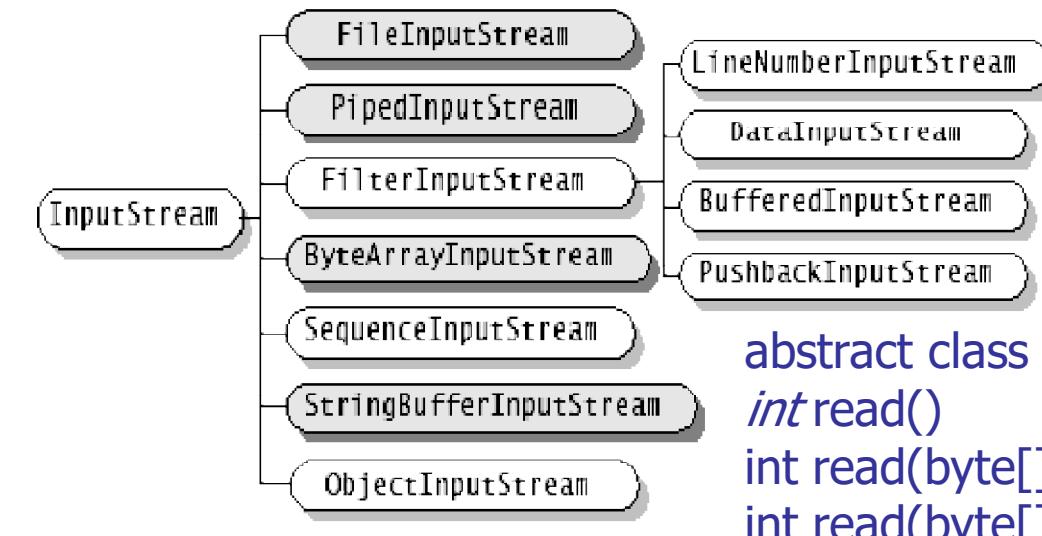


InputStream/OutputStream je jednosmerný sekvenčný tok bajtov (8bits) zdrojom/cielom môže byť

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

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

InputStream/OutputStream (byte)

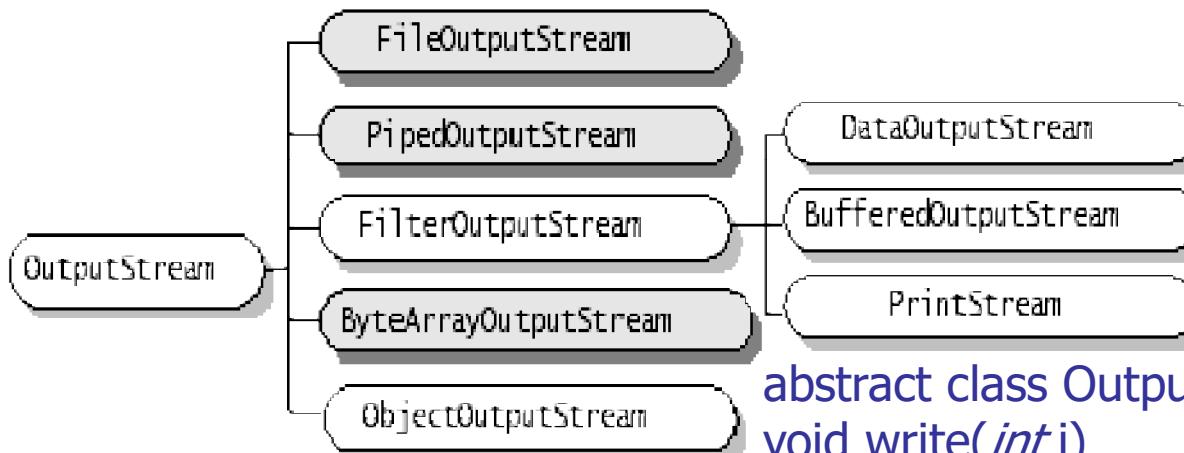


abstract class InputStream:

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)

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

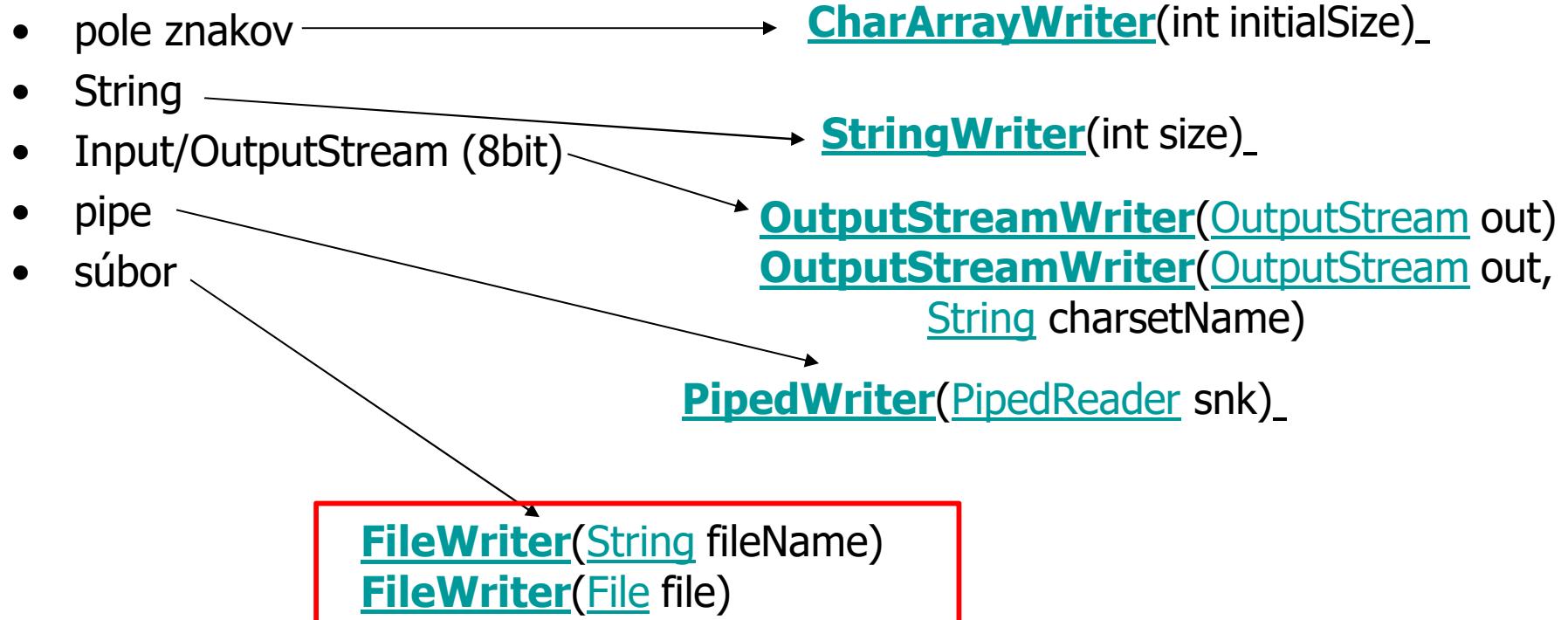
Čítanie a zápis komprimovaného súboru (váš .zyp)

- http://en.wikipedia.org/wiki/ZIP_%28file_format%29

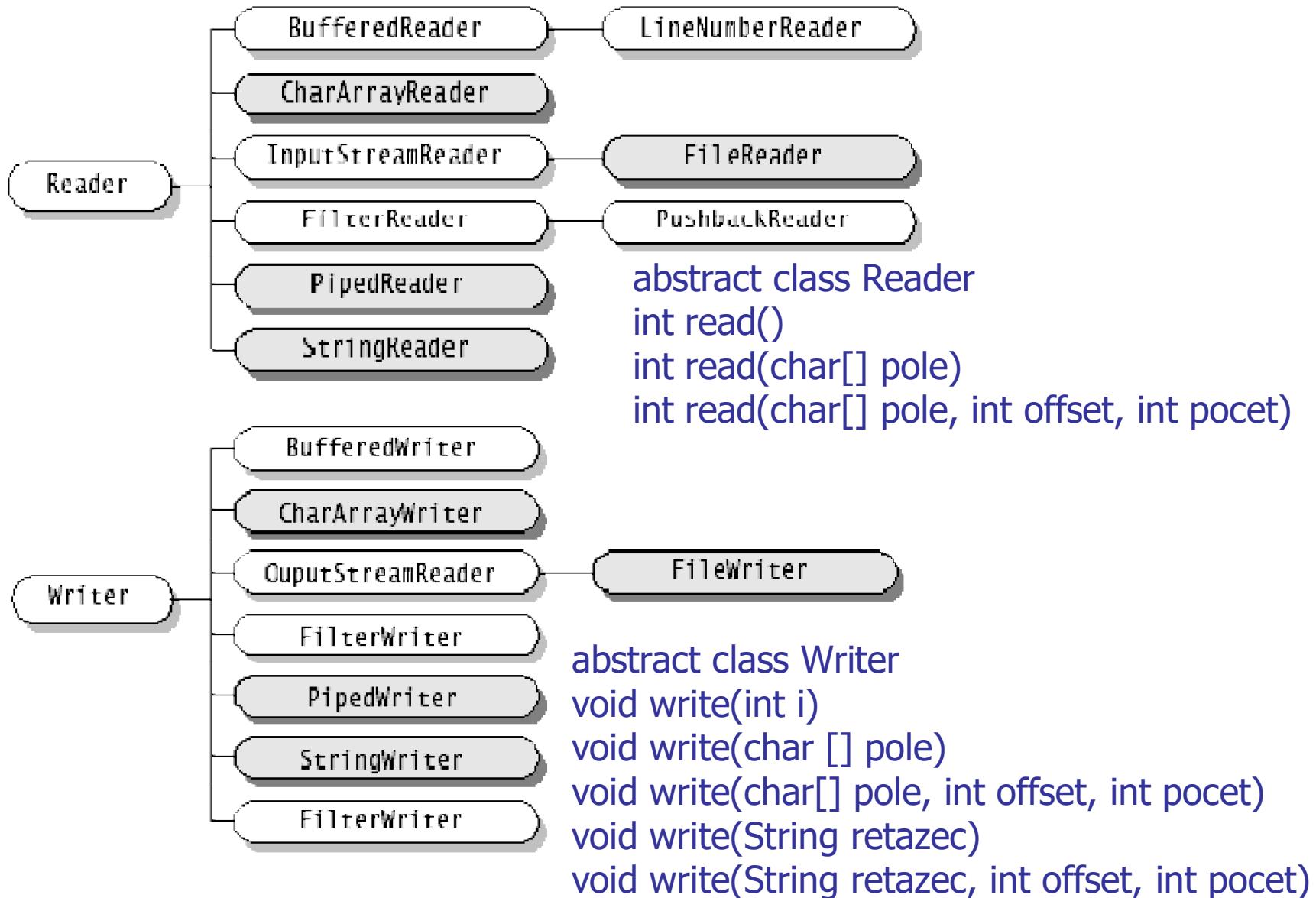
Writer/Reader

(char stream)

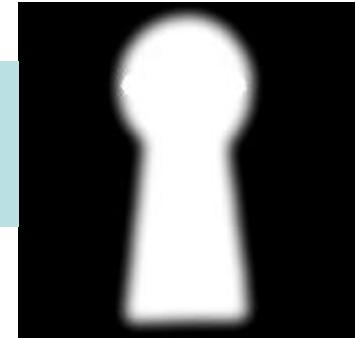
Reader/Writer je jednosmerný sekvenčný tok znakov/charakterov/ (16bits) zdrojom/cieľom môže byť



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("Klepni 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;

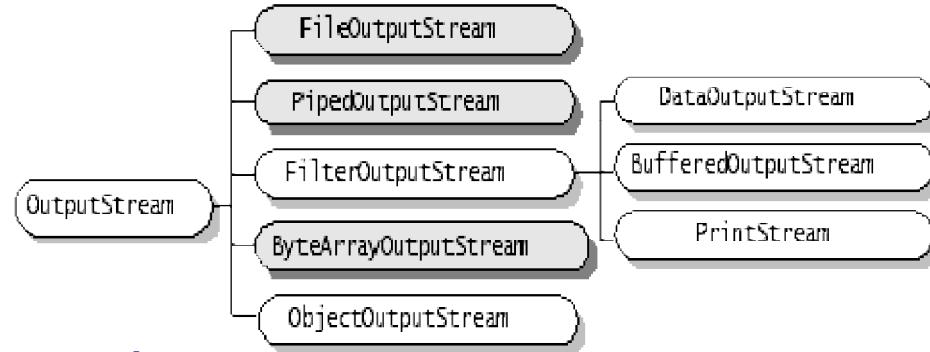
// čítanie riadku s echom
System.out.println("Napis riadok");
String inputLine = "";
while (true) {
    try {
        int tmp = System.in.read ();
        if (tmp == -1) break;
        if (tmp == '\n') break;
        char c = (char) tmp;
        inputLine = inputLine + c;
    } catch (IOException e) {
        System.err.println("chyba");
    }
}
System.out.println("echo: "+inputLine);
```

Súbory: `InputChar.java`, `InputLine.java`

OutputStream vs. PrintStream

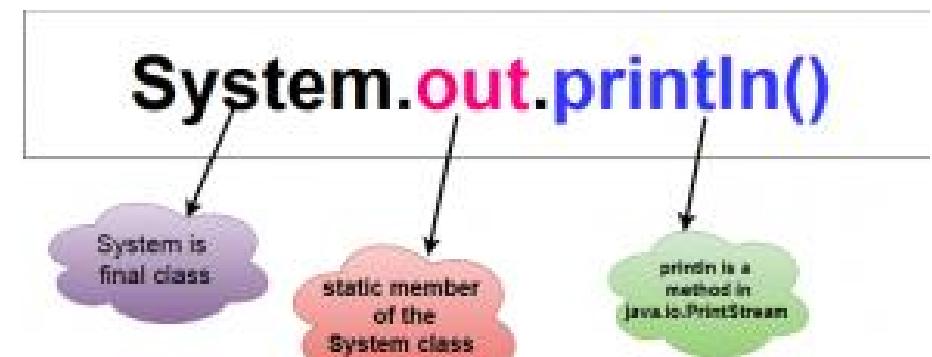
kým OutputStream poskytuje:

- 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



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 vzniknúvšia 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

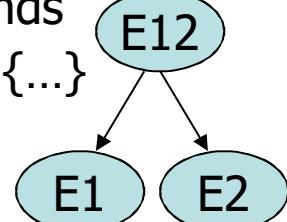
```
public static void citaj() throws IOException {  
    ....  
    try {  
        char c = (char)System.in.read();  
    } catch (IOException e) { // odchycenie výnimky  
        System.err.println ("chyba pri citani");  
        throw e;           // propagovanie (vyvolanie) výnimky  
    }  
  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

v mieste odkiaľ voláme metódu citaj():

```
try {  
    citaj();  
} catch (IOException e) { // opäťovné odchycenie výnimky  
    System.err.println("rachlo to");  
}
```

Hierarchia výnimiek

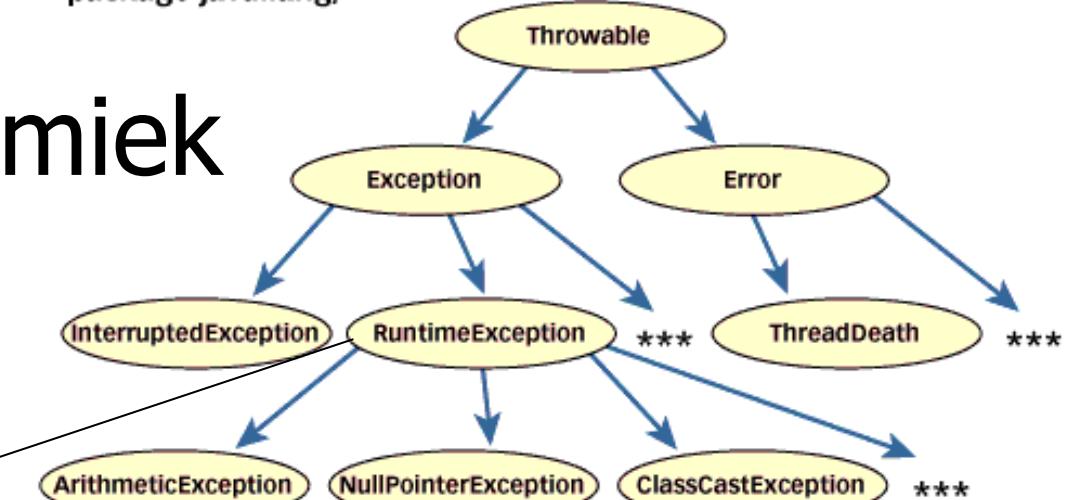
```
class Exception12 extends  
    RuntimeException {...}  
  
class Exception1 extends  
    Exception12 {...}  
  
class Exception2 extends  
    Exception12 {...}
```



**Exception 2
finally**

**Exception 12
finally**

```
package java.lang;
```



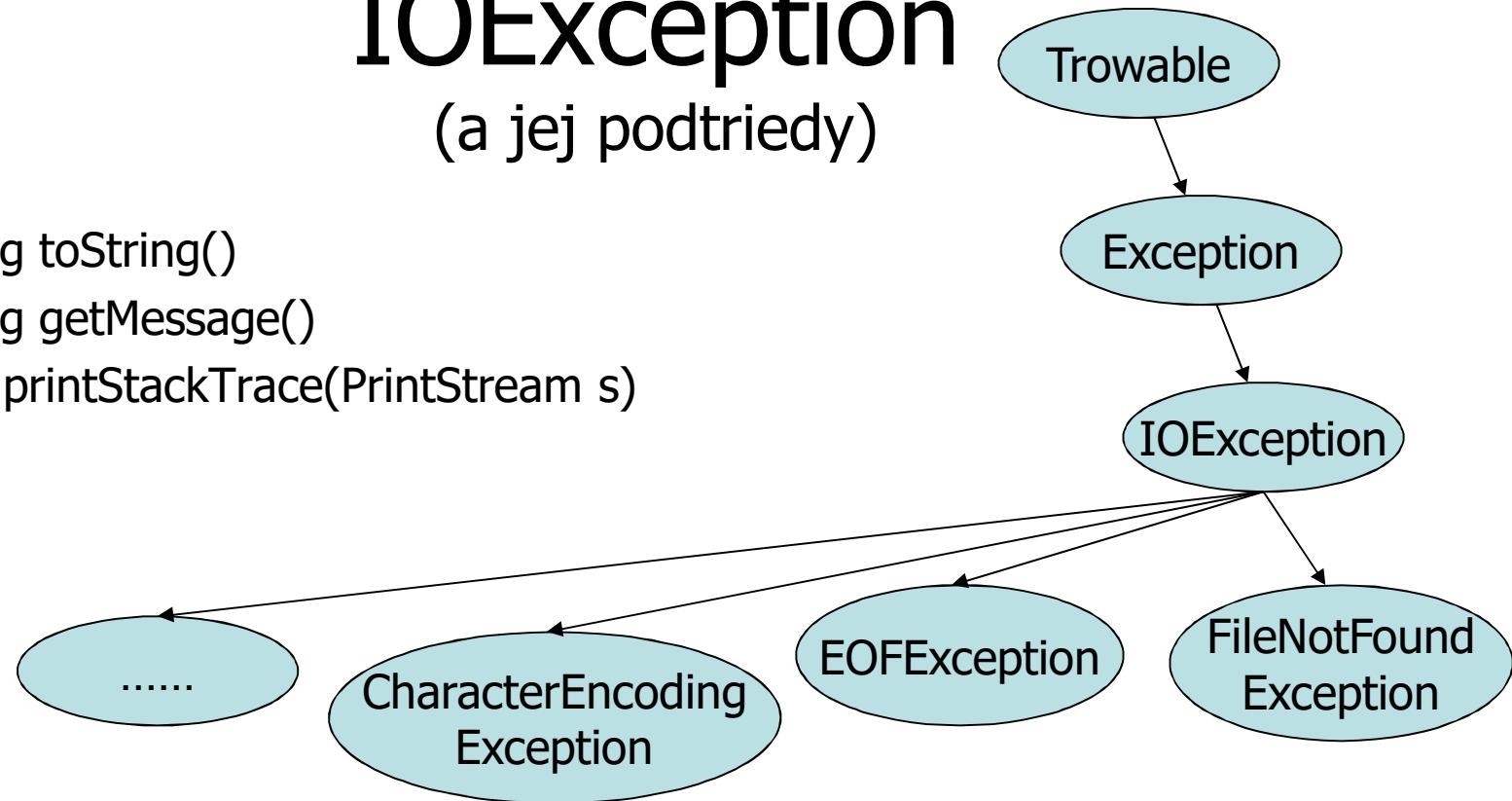
```
try { // kód produkujúci Ex1, Ex2 resp. Ex12  
switch((int)(3*Math.random())) {  
    case 0: throw new Exception12();  
    case 1: throw new Exception1();  
    case 2: throw new Exception2(); }  
} catch (Exception1 e1) {  
    System.out.println("Exception 1");  
} catch (Exception2 e2) {  
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

Súbor: DoException.java

IOException

(a jej podtrydy)

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



Zloživýk (Bad practice) :

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

Kopírovanie súboru

```
import java.io.*;  
  
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  
  
        FileInputStream fr = new FileInputStream(frName);  
                                         // vytvor InputStream (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
```

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-catch-finally

(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 odchytiať a ošetriť

try-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)

```
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++)
            sum += (inputLine.charAt(j)=='*')?1:0;
    }
    System.out.println("pocet *=" +sum);
} catch (IOException e) {
    System.err.println ("IO exception = " + e);
}
```

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

InputStream

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

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

InputStreamReader konwertuje
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()`

BufferedReader

InputStream
byte
InputStreamReader
char

BufferedReader
char

`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();
```

Formátovaný vstup

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

```
import java.util.Scanner;
// obalíme InputStream
// triedou Scanner
// nasleduje int ?
// nasleduje float ?
// nasleduje boolean ?
```

```
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: Scaner.java

Príklad Histogram

```
Scanner scan = new Scanner( // scanner na vstupnom súbore
    new FileInputStream(
        new File("lenBody.txt")));
TreeMap<Double, Integer> tmap = // interná tabuľka
    new TreeMap<Double, Integer>(); // výskytov/početnosti

while(scan.hasNextDouble()) { // kým sú ...
    double val = Math.floor(
        scan.nextDouble()); // čítanie realov
    if (tmap.get(val)==null) tmap.put(val,1); // update tmap
    else tmap.put(val,1+tmap.get(val));
}

for(double i=0; i<50; i++) // tlač
    System.out.println(i+"\t"+
        "*****\n".
        substring(0,(tmap.get(i)==null)?0:tmap.get(i)));



```

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

```
Scanner scan_csv = new Scanner(
```

```
    new FileInputStream(
```

```
        new File("body.csv")));
```

```
double sucty[] = {0,0,0,0,0,0};
```

// súčty bodov za jednotl.príklady

```
int pocet = 0;
```

// počet študentov

```
scan_csv.useDelimiter("[;\\r\\n]");
```

// oddel'ovač .csv

```
while(scan_csv.hasNextLine()) {
```

// kým nie je posledný riadok

```
    int i = 0;
```

// nastav index cvičenia

```
    /*String meno = */ scan_csv.next();
```

// preskoč meno

```
    /*String priezvisko = */ scan_csv.next();
```

// preskoč priezvisko

```
    while(scan_csv.hasNextDouble())
```

// kým sú body

```
        sucty[i++] += scan_csv.nextDouble();
```

// čítaj body

```
        scan_csv.nextLine(); pocet++;
```

// na nový riadok

```
}
```

```
for(int i=0; i<sucty.length; i++)
```

// tlač priemerov

```
    System.out.println((i+1)+".priklad ma priemer: "+sucty[i]/pocet);
```

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

Regulárne výrazy

[abc]	a, b, c	.	\ub.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-zA-Z0-9]+")	"java9"
sc.hasNext("[^python]+")	"java9"
sc.hasNext("[A-F0-9]+")	"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;
```

```
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;
}
```

Príklad Regular

```
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: janو@maznet11.com
```

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

Formátovaný výstup

```
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 ();
```

Text

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

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

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

String.format

%[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 obálkovania (napr. Buffered..., Scanner s reg.výrazmi),

Pokračovanie:

- serializácia – spôsob ako ľubovoľný objekt pretransformovať na postupnosť bajtov tak, aby sme ho vedeli späťne reštaurovať. Využitie: zápis a čítanie objektu do/z súboru, pajpy, ...
- práca s adresármami
- 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 – zretťazenie 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"); // zapiš String
fs.writeObject(s); // AVLTree
fs.flush(); // fs.close();
```

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 byt' serializovateľný: Serializable Interface

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

Súbor: AVLTree.java

Ulož a prečítaj konfiguráciu

Ak máme uložiť a opäťovne 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();  
    }  
}
```

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 súbroru:

```
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());
```

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

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

Filtrovanie adresára podľa prípony

```

class FilterPripony implements FilenameFilter {
    String maska;
    FilterPripony(String maska) {           // zapamätaj si %de 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
 Scaner.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 vel'kosti

```
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(); // zaujímajú ma len adresáre  
    }  
}  
  
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); // chod' do podadresára  
        }  
    }  
}
```

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

Súbor: RecDirList.java

Priamy prístup

priamy prístup k dátam v súbore – najčastejšie v binárnom súbore, keď
pristupujeme 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 InputStreamov 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();
```