




Vstupy a výstupy v Javě.

Obsah

Vstupy a výstupy v Javě	1
Koncepce vstupně/výstupních operací v Javě	1
Práce se soubory	2
Třída File 	3
Třída File  (2)	4
Třída File  (3)	5
Práce s adresáři	5
Práce s binárními proudy	5
Vstupní binární proudy	6
Důležité neabstraktní třídy odvozené od InputStream 	6
Další vstupní proudy	7
Práce se znakovými proudy	7
Výstupní proudy	8
Konverze: znakové <-> binární proudy	9
Serializace objektů	9
Odkazy	10

Vstupy a výstupy v Javě




- Koncepce I/O proudů v Javě, skládání (obalování vlastnostmi)
- Práce se soubory a adresáři, třída File
- Binární proudy, abstraktní třídy InputStream, OutputStream
- Znakové proudy, abstraktní třídy Reader, Writer
- Serializace objektů

Koncepce vstupně/výstupních operací v Javě

založeny na v/v proudech

plně **platformově nezávislé**



V/V proudy jsou

- **znakové** (Reader 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader>]/Writer
 [<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Writer>])
nebo
- **binární** (InputStream/OutputStream 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream/OutputStream>]
)

koncipovány jako "stavebnice" - lze vkládat do sebe a tím přidávat vlastnosti, např.


```
is = new InputStream(...);  
bis = new BufferedInputStream(is);
```


Téměř vše ze vstupních/výstupních tříd a rozhraní je v balíku java.io.

počínaje Java 1.4 se rozvíjí alternativní balík - java.nio 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.nio>] (*New I/O*), zde se
ale budeme věnovat klasickým I/O z balíku java.io 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io>].

Blíže viz dokumentace API balíků java.io
[<http://java.sun.com/j2se/1.5/docs/api/java/io/package-summary.html>],
[<http://java.sun.com/j2se/1.5/docs/api/java/nio/package-summary.html>].

Práce se soubory


vše je opět v balíku java.io 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io>]

základem je třída java.io.File 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io.File>] - nositel jména
souboru, jakási "brána" k fyzickým souborům na disku.

používá se jak pro soubory, tak adresáře, linky i soubory identifikované UNC jmény
(\\počítač\adresář...)

opět plně platformově nezávislé

na odstínění odlišností jednotlivých systémů souborů lze použít vlastností (uvádíme jejich hodnoty pro
JVM pod systémem MS Windows):

- `File.separatorChar` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.separatorChar>] - ja-

- ko char 

[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=char>]
- `File.separator` 

[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.separator \ \]](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.separator%20) - jako `String` 

[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=String>]
- `File.pathSeparatorChar` ; 

[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.pathSeparatorChar ;](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.pathSeparatorChar%20)] - jako `char` 

[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=char>]
- `File.pathSeparator` ; 

[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.pathSeparator ;](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.pathSeparator%20)] - jako `String` 

[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=String>]
- `System.getProperty("user.dir")` 

[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.getProperty\("user.dir"\)](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.getProperty%28%22user.dir%22%29)] - adresář uživatele, pod jehož UID je proces JVM spuštěn

Třída `File`

[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File>]

Vytvoření konstruktorem - máme několik možností:

- | | |
|--|---|
| <code>new File(String filename)</code> | vytvoří v aktuálním adresáři soubor s názvem <i>filename</i> |
| <code>new File(File baseDir, String filename)</code> | vytvoří v adresáři <i>baseDir</i> soubor s názvem <i>filename</i> |
| <code>new File(String baseDirName, String filename)</code> | vytvoří v adresáři <i>se jménem baseDirName</i> soubor s názvem <i>filename</i> |
| <code>new File(URL url)</code> | vytvoří soubor se (souborovým - file:) URL <i>url</i> |

Testy existence a povahy souboru:

- | | |
|-------------------------------|---|
| <code>boolean exists()</code> | test na existenci souboru (nebo adresáře) |
|-------------------------------|---|

boolean **isFile()** test, zda jde o soubor (tj. ne adresář)

boolean **isDirectory()** test, zda jde o adresář


Test práv ke čtení/zápisu:


boolean **canRead()** test, zda lze soubor číst

boolean **canWrite()** test, zda lze do souboru zapisovat

Třída **File** [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File] (2)


Vytvoření souboru nebo adresáře:


boolean **createNewFile()** (pro soubor) vrací true 
 [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=true], když se podaří *soubor* vytvořit

boolean **mkdir()** (pro adresář) vrací true 
 [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=true], když se podaří adresář vytvořit

boolean **mkdirs()** navíc si dotvoří i příp. neexistující adresáře na cestě

Vytvoření dočasného (temporary) souboru:

static File **createTempFile**(String *prefix*, String *suffix*) vytvoří dočasný soubor ve standardním pro to určeném adresáři (např. c:\temp 
 [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=c:\temp]) s uvedeným prefixem a sufixem názvu

static File **createTempFile**(String *prefix*, String *suffix*, File *directory*) dtto, ale vytvoří dočasný soubor v adr. *directory* 
 [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=directory]

Zrušení:

boolean **delete()** zrušení souboru nebo adresáře

Přejmenování (ne přesun mezi adresáři!):

boolean **renameTo**(File *dest*) přejmenuje soubor nebo adresář


Třída **File** [<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File>] (3)

Další vlastnosti:


long length ()	délka (velikost) souboru v bajtech
long lastModified ()	čas poslední modifikace v ms od začátku éry - podobně jako systémový čas vrácený <code>System.currentTimeMillis()</code> .
String getName ()	jen jméno souboru (tj. poslední část cesty)
String getPath ()	celá cesta k souboru i se jménem
String getAbsolutePath ()	absolutní cesta k souboru i se jménem
String getParent ()	adresář, v němž je soubor nebo adresář obsažen

Bližší viz dokumentace API třídy File [<http://java.sun.com/j2se/1.5/docs/api/java/io/File.html>].

Práce s adresáři

Klíčem je opět třída 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File>] - použitelná i pro adresáře


Jak např. získat (filtrovaný) seznam souborů v adresáři?


pomocí metody `File[] listFiles(FileFilter ff)` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File>] `listFiles(FileFilter ff)`
nebo podobné

`File[] listFiles(FileNameFilter fnf):`

`FileFilter` je rozhraní s jedinou metodou `boolean accept(File pathname)`, obdobně `FileNameFilter`, viz Popis API `java.io.FileNameFilter` [<http://java.sun.com/j2se/1.5/docs/api/java/io/FileNameFilter.html>]

Práce s binárními proudy

Vstupní jsou odvozeny od abstraktní třídy `InputStream` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream>]

Výstupní jsou odvozeny od abstraktní třídy `OutputStream` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=OutputStream>]

Vstupní binární proudy

Uvedené metody, kromě `abstract byte read()`, nemusejí být nutně v neabstraktní podtřídě překryty.


<code>void close()</code>	uzavře proud a uvolní příslušné zdroje (systémové "file handles" apod.)
<code>void mark(int readlimit)</code>	poznačí si aktuální pozici (později se lze vrátit zpět pomocí <code>reset()</code>)...
<code>boolean markSupported()</code>	...ale jen když platí tohle
<code>abstract int read()</code>	přečte bajt (0-255 pokud OK; jinak -1, když už není možné přečíst)
<code>int read(byte[] b)</code>	přečte pole bajtů
<code>int read(byte[] b, int off, int len)</code>	přečte pole bajtů se specifikací délky a pozice plnění pole <code>b</code>
<code>void reset()</code>	vrátí se ke značce nastavené metodou <code>mark(int)</code>
<code>long skip(long n)</code>	přeskočí zadaný počte bajtů

Důležité neabstraktní třídy odvozené od `InputStream`




[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream>]

`java.io.FilterInputStream` - je bazová třída k odvozování všech vstupních proudů přidávajících vlastnost/schopnost filtrovat poskytnutý vstupní proud.

Příklady filtrů (ne všechny jsou v `java.io`) 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io!>):

<code>BufferedInputStream</code>	proud s vyrovnávací pamětí (je možno specifikovat její optimální velikost)
<code>java.util.zip.CheckedInputStream</code>	proud s kontrolním součtem (např. CRC32)

<code>javax.crypto.CipherInputStream</code>	proud dešifrující data ze vstupu
<code>DataInputStream</code>	má metody pro čtení hodnot primitivních typů, např. <code>readFloat()</code>  [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=float_readFloat()]
<code>java.security.DigestInputStream</code>	počítá současně i haš (digest) čtených dat, použitý algoritmus lze nastavit
<code>java.util.zip.InflaterInputStream</code>	dekomprimuje (např. GZIPem) zabalený vstupní proud (má ještě specializované podtřídy)
<code>LineNumberInputStream</code>	doplňuje informaci o tom, ze kterého řádku vstupu čteme (zavrhovaná - <i>deprecated</i> - třída)
<code>ProgressMonitorInputStream</code>	přidává schopnost informovat o průběhu čtení z proudu
<code>PushbackInputStream</code>	do proudu lze data vracet zpět


Další vstupní proudy

Příklad rekonstrukce objektů ze souborů









```
FileInputStream istream = new FileInputStream("t.tmp");
ObjectInputStream p = new ObjectInputStream(istream);
int i = p.readInt();
String today = (String)p.readObject();
Date date = (Date)p.readObject();
istream.close();
```

<code>java.sound.sampled.AudioInputStream</code>	vstupní proud zvukových dat
<code>ByteArrayInputStream</code>	proud dat čtených z pole bajtů
<code>PipedInputStream</code>	roura napojená na "protilehlý" <code>PipedOutputStream</code>
<code>SequenceInputStream</code>	proud vzniklý spojením více podřízených proudů do jednoho virtuálního
<code>ObjectInputStream</code>	proud na čtení serializovaných objektů

Práce se znakovými proudy

základem je abstraktní třída `Reader`

[\[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader\]](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader), konkrétními imple-





mentacemi jsou:

- `BufferedReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=BufferedReader>], `CharArrayReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=CharArrayReader>], `InputStreamReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStreamReader>], `PipedReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=PipedReader>], `StringReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=StringReader>]
- `LineNumberReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=LineNumberReader>], `FileReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=FileReader>], `PushbackReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=PushbackReader>]

Výstupní proudy



nebudeme důkladně probírat všechny typy

principy:

- jedná se o protějšky k vstupním proudům, názvy jsou konstruovány analogicky (např. `FileReader` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=FileReader>] -> `FileWriter` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=FileWriter>])
- místo generických metod `read` 
[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=read>] mají `write(...)` 
[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=write\(...\)](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=write(...))]



Příklady:

`PrintStream` poskytuje metody pro pohodlný zápis hodnot primitivních typů a řetězců - příkladem

jsou `System.out` 
[\[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.out\]](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.out)
a `System.err` 
[\[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.err\]](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.err)


`PrintWriter` poskytuje metody pro pohodlný zápis hodnot primitivních typů a řetězců

Konverze: znakové <-> binární proudy

Ze vstupního binárního proudu `InputStream` 
[\[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream\]](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream) (čili každého) je možné vytvořit znakový `Reader` 
[\[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader\]](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader) pomocí

```
// nejprve binární vstupní proud - toho kódování znaků nezajímá
InputStream is = ...
```

```
// znakový proud isr
// použije pro dekódování standardní znakovou sadu
Reader isr = new InputStreamReader(is);
```

```
// sady jsou definovány v balíku java.nio  \[http://cs.wikipedia.org/wiki/Charset\]
Charset chrs = java.nio.Charset.forName("ISO-8859-2");
```

```
// znakový proud isr2
// použije pro dekódování jinou znakovou sadu
Reader isr2 = new InputStreamReader(is, chrs);
```

Podporované názvy znakových sad naleznete na webu IANA `Charsets` [\[http://www.iana.org/assignments/character-sets\]](http://www.iana.org/assignments/character-sets).

Obdobně pro výstupní proudy - lze vytvořit `Writer` z `OutputStream`.

Serializace objektů

- nebudeme podrobně studovat, zatím stačí vědět, že:
 - **serializace objektů** je postup, jak z objektu vytvořit sekvenci bajtů persistentně uložitelnou na paměťové médium (disk) a později restaurovatelnou do podoby výchozího javového objektu.
 - **deserializace** je právě zpětná rekonstrukce objektu
- aby objekt bylo možno serializovat, musí implementovat (prázdňé) rozhraní `Ja-`



[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io.Serializable>]

- proměnné objektu, které nemají být serializovány, musí být označeny modifikátorem - klíčovým slovem `transient`



[<http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=transient>]

- pokud požaduje "speciální chování" při de/serializaci, musí objekt definovat metody
 - `private void readObject(java.io.ObjectInputStream stream) throws IOException, ClassNotFoundException`



[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=private void readObject\(java.io.ObjectInputStream stream\) throws IOException, ClassNotFoundException](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=private%20void%20readObject(java.io.ObjectInputStream%20stream)%20throws%20IOException,%20ClassNotFoundException)]

- `private void writeObject(java.io.ObjectOutputStream stream) throws IOException`



[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=private void writeObject\(java.io.ObjectOutputStream stream\) throws IOException](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=private%20void%20writeObject(java.io.ObjectOutputStream%20stream)%20throws%20IOException)]

- metody:

- `DataOutputStream.writeObject(Object o)`



[[http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=DataOutputStream.writeObject\(Object o\)](http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=DataOutputStream.writeObject(Object%20o))]

Odkazy

Tutoriály k Java I/O: kapitola z Sun Java Tutorial [<http://java.sun.com/docs/books/tutorial/essential/io/>]

Demo programy na serializaci (z učebnice): Serializace objektů [<http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/serializace/>]