Vstupy a výstupy v Jav#.

Obsah

Vstupy a výstupy v Jav#

- Koncepce I/O proud# v Jav#, skládání (obalování vlastnostmi)
- Práce se soubory a adresá#i, t#ída File
- Binární proudy, abstraktní t#ídy InputStream, OutputStream
- Znakové proudy, abstraktní t#ídy Reader, Writer
- Serializace objekt#

Koncepce vstupn#/výstupních operací v Jav#

založeny na v/v proudech

pln# **platformov# nezávislé**

V/V proudy jsou

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

)

koncipovány jako "stavebnice" - lze vkládat do sebe a tím p#idávat vlastnosti, nap#.

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

Tém## vše ze vstupních/výstupních t#íd a rozhraní je v balíku java.io.

po#ínaje Java 1.4 se rozvíjí alternativní balík - `java.nio` [http://www.google.com/search?q=java.nio] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= java.nio]*(New I/O)*, zde se ale budeme v#novat klasickým I/O z balíku `java.io` [http://www.google.com/search?q=java.io] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io].

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

Práce se soubory

vše je op#t v balíku `java.io` [http://www.google.com/search?q=java.io] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io]

základem je t#ída `java.io.File` [http://www.google.com/search?q=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 soubor#m na disku.

používá se jak pro soubory, tak adresá#e, linky i soubory identifikované UNC jmény (\\po#íta#\adresá#...)

op#t pln# platformov# nezávislé

na odstín#ní odlišností jednotlivých systém# soubor# lze použít vlastností (uvádíme jejich hodnoty pro JVM pod systémem MS Windows):

- `File.separatorChar` \ [http://www.google.com/search?q=File.separatorChar \ ] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.separatorChar \ ] - jako `char` [http://www.google.com/search?q=char] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=char]
- `File.separator` \ [http://www.google.com/search?q=File.separator \ ] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.separator \ ] - jako `String` [http://www.google.com/search?q=String] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=String]
- `File.pathSeparatorChar` ; [http://www.google.com/search?q=File.pathSeparatorChar ;] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.pathSeparatorChar ;] - jako `char` [http://www.google.com/search?q=char] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=char]
-

`http://www.google.com/search?q=File.pathSeparator ;]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File.pathSeparator ;] - jako `String [http://www.google.com/search?q=String]` [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=String]

- `System.getProperty("user.dir")` `[http://www.google.com/search?q=System.getProperty("user.dir")]` http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.getProperty("user.di [r")] - adresá# uživatele, pod jehož UID je proces JVM spušt#n

T#ída `File` [http://www.google.com/search?q=File] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:S [earch?search=File]

Vytvo#ení konstruktorem - máme n#kolik možností:

new File(String *filename*)	vytvo#í v aktuálním adresá#i soubor s názvem *filename*
new File(File *baseDir*, String *filename*)	vytvo#í v adresá#i *baseDir* soubor s názvem *filename*
new File(String *baseDirName*, String *filename*)	vytvo#í v adresá#i *se jménem baseDirName* soubor s názvem *filename*
new File(URL *url*)	vytvo#í soubor se (souborovým - file:) URL *url*

Testy existence a povahy souboru:

boolean **exists**()	test na existenci souboru (nebo adresá#e)
boolean **isFile**()	test, zda jde o soubor (tj. ne adresá#)
boolean **isDirectory**()	test, zda jde o adresá#

Test práv ke #tení/zápisu:

boolean **canRead**()	test, zda lze soubor #íst
boolean **canWrite**()	test, zda lze do souboru zapisovat

T#ída `File` [http://www.google.com/search?q=File] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:S [earch?search=File] (2)

Vytvo#ení souboru nebo adresá#e:

boolean **createNewFile**()	(pro soubor) vrací `true` [http://www.google.com/search?q=true] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=true], když se poda#í *soubor* vytvo#it
boolean **mkdir**()	(pro adresá#) vrací `true` [http://www.google.com/search?q=true] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=true], když se poda#í adresá# vytvo#it
boolean **mkdirs**()	navíc si dotvo#í i p#íp. neexistující adresá#e na cest#

Vytvo#ení do#asného (temporary) souboru:

static File **createTempFile**(String *prefix*, String *suffix*)	vytvo#í do#asný soubor ve standardním pro to ur#eném adresá#i (nap#. `c:\temp` [http://www.google.com/search?q=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 vytvo#í do#asný soubor v adr. `directory` [http://www.google.com/search?q=directory] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=directory]

Zrušení:

boolean **delete**()	zrušení souboru nebo adresá#e

P#ejmenování (ne p#esun mezi adresá#i!):

boolean **renameTo**(File *dest*)	p#ejmenuje soubor nebo adresá#

T#ída `File` [http://www.google.com/search?q=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**()	#as poslední modifikace v ms od za#átku éry - podobn# jako systémový #as vracený System.currentTimeMillis().
String **getName**()	jen jméno souboru (tj. poslední #ást cesty)
String **getPath**()	celá cesta k souboru i se jménem

String **getAbsolutePath**()	absolutní cesta k souboru i se jménem
String **getParent**()	adresá#, v n#mž je soubor nebo adresá# obsažen

Blíže viz dokumentace API t#ídy File [http://java.sun.com/j2se/1.5/docs/api/java/io/File.html].

Práce s adresá#i

Klí#em je op#t t#ída `File` [http://www.google.com/search?q=File] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=File] - použitelná i pro adresá#e

Jak nap#. získat (filtrovaný) seznam soubor# v adresá#i?

pomocí metody `File[] listFiles(FileFilter ff)` [http://www.google.com/search?q=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), obdobn# 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 t#ídy `InputStream` [http://www.google.com/search?q=InputStream] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream]

Výstupní jsou odvozeny od abstraktní t#ídy `OutputStream` [http://www.google.com/search?q=OutputStream] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=OutputStream]

Vstupní binární proudy

Uvedené metody, krom# abstract byte read(), nemusejí být nutn# v neabstraktní podt#íd# p#ekryty.

void **close**()	uzav#e proud a uvolní p#íslušné zdroje (systémové "file handles" apod.)
void **mark**(int *readlimit*)	pozna#í si aktuální pozici (pozd#ji se lze vrátit zp#t pomocí reset())...
boolean **markSupported**()	...ale jen když platí tohle
abstract int **read**()	p#e#te bajt (0-255 pokud OK; jinak -1, když už není možné p#e#íst)
int **read**(byte[] b)	p#e#te pole bajt#
int **read**(byte[] b, int off, int len)	p#e#te pole bajt# se specifikací délky a pozice pln#ní pole b
void **reset**()	vrátí se ke zna#ce nastavené metodou mark(int)

long **skip**(long n)	p#esko#í zadaný po#te bajt#

D#ležité neabstraktní t#ídy odvozené od `InputStream` [http://www.google.com/search?q=InputStream] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:S [earch?search=InputStream]

java.io.FilterInputStream - je bázová t#ída k odvozování všech vstupních proud# p#idávajících vlastnost/schopnost filtrovat poskytnutý vstupní proud.

P#íklady filtr# (ne všechny jsou v `java.io` [http://www.google.com/search?q=java.io] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io]!):

BufferedInputStream	proud s vyrovnávací pam#tí (je možno specifikovat její optimální velikost)
java.util.zip.CheckedInputStream	proud s kontrolním sou#tem (nap#. CRC32)
javax.crypto.CipherInputStream	proud dešifrující data ze vstupu
DataInputStream	má metody pro #tení hodnot primitivních typ#, nap#. `float readFloat()` [http://www.google.com/search?q=float readFloat()] http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?[search=float readFloat()]
java.security.DigestInputStream	po#ítá sou#asn# i haš (digest) #tených dat, použitý algoritmus lze nastavit
java.util.zip.InflaterInputStream	dekomprimuje (nap#. GZIPem) zabalený vstupní proud (má ješt# specializované podt#ídy)
LineNumberInputStream	dopl#uje informaci o tom, ze kterého #ádku vstupu #teme (zavrhovaná - *deprecated* - t#ída)
ProgressMonitorInputStream	p#idává schopnost informovat o pr#b#hu #tení z proudu
PushbackInputStream	do proudu lze data vracet zp#t

Další vstupní proudy

P#íklad rekonstrukce objekt# ze soubor#

```
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();
```

javax.sound.sampled.AudioInputStream	vstupní proud zvukových dat
ByteArrayInputStream	proud dat #tených z pole bajt#
PipedInputStream	roura napojená na "protilehlý" PipedOutputStream
SequenceInputStream	proud vzniklý spojením více pod#ízených proud# do jednoho virtuálního
ObjectInputStream	proud na #tení serializovaných objekt#

Práce se znakovými proudy

základem je abstraktní t#ída `Reader` [http://www.google.com/search?q=Reader] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader], konkrétními implementacemi jsou:

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

Výstupní proudy

nebudeme d#kladn# probírat všechny typy

principy:

- jedná se o prot#jšky k vstupním proud#m, názvy jsou konstruovány analogicky (nap#. `FileReader` [http://www.google.com/search?q=FileReader] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=FileReader] -> `FileWriter` [http://www.google.com/search?q=FileWriter] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=FileWriter])
- místo generických metod `read` [http://www.google.com/search?q=read] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=read] mají `write(...)` [http://www.google.com/search?q=write(...)] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=write(...)]

P#íklady:

PrintStream — poskytuje metody pro pohodlný zápis hodnot primitivních typ# a #et#zc# - p#íkladem jsou `System.out` [http://www.google.com/search?q=System.out] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.out] a `System.err` [http://www.google.com/search?q=System.err] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=System.err]

PrintWriter — poskytuje metody pro pohodlný zápis hodnot primitivních typ# a #et#zc#

Konverze: znakové <-> binární proudy

Ze vstupního binárního proudu `InputStream` [http://www.google.com/search?q=InputStream] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=InputStream] (#ili každého) je možné vytvo#it znakový `Reader` [http://www.google.com/search?q=Reader] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=Reader] pomocí

```
// nejprve binární vstupní proud -
      toho kódování znak# 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://www.google.com/search?q=java.nio] [http://cs.wikipedia.org/
      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].

Obdobn# pro výstupní proudy - lze vytvo#it Writer z OutputStream.

Serializace objekt#

- nebudeme podrobn# studovat, zatím sta#í v#d#t, že:
  - **serializace objekt#** je postup, jak z objektu vytvo#it sekvenci bajt# persistentn# uložitelnou na pam##ové médium (disk) a pozd#ji restaurovatelnou do podoby výchozího javového objektu.
  - **deserializace** je práv# zp#tná rekonstrukce objektu
- aby objekt bylo možno serializovat, musí implementovat (prázdné) rozhraní `java.io.Serializable` [http://www.google.com/search?q=java.io.Serializable] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=java.io.Serializable]
- prom#nné objektu, které nemají být serializovány, musí být ozna#eny modifikátorem - klí#ovým slovem - `transient` [http://www.google.com/search?q= transient] [http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search= transient]
- pokud požaduje "speciální chování" p#i de/serializaci, musí objekt definovat metody
  - `private void readObject(java.io.ObjectInputStream stream) throws IOException, ClassNotFoundException` [http://www.google.com/search?q=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]

- `private void writeObject(java.io.ObjectOutputStream stream) throws IOException [http://www.google.com/search?q=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]

- metody:
  - `DataOutputStream.writeObject(Object o) http://www.google.com/search?q=DataOutputStream.writeObject(Objec [t o)]`
  http://cs.wikipedia.org/wiki/Speci%C3%A1ln%C3%AD:Search?search=DataOutputStream.write [Object(Object o)]

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 u#ebnice): Serializace objekt# [http://www.fi.muni.cz/~tomp/java/ucebnice/javasrc/tomp/ucebnice/serializace/]