Vstupy a výstupy v Javě.

December 2, 2013

1 Vstupy a výstupy v Javě

1.1 Koncepce vstupně/výstupních operací v Javě

Založeny na v/v proudechPlně **platformově nezávisléV/V** proudy členíme na

- **znakové** (`Reader/Writer`) a
- **binární** (`InputStream/OutputStream`)

1.2 Koncepce vstupně/výstupních operací v Javě (2)

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

```
is = new InputStream(...);
// bis přidá k proudu is vlastnost vyrovnávací paměti
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` *(New I/O)*, zde se ale budeme věnovat klasickým I/O z balíku `java.io`.Blíže viz dokumentace API balíků 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`).

1.3 Práce s binárními proudy

Vstupní jsou odvozeny od abstraktní třídy `InputStream`Výstupní jsou odvozeny od abstraktní třídy `OutputStream`

1.4 Vstupní binární proudy

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

void close() uzavře proud a uvolní příslušné zdroje (systémové "file handles" apod.)

void mark(int readlimit) poznačí si aktuální pozici (později se lze vrátit zpět pomocí reset())...

boolean markSupported() ...ale jen když platí tohle

abstract int read() přečte bajt (0-255 pokud OK; jinak -1, když už není možné přečíst)

1.5 Vstupní binární proudy - pokračování

int read(byte[b)] přečte pole bajtů

int read(byte[b, int off, int len)] přečte pole bajtů se specifikací délky a pozice plnění pole b

void reset() vrátí se ke značce nastavené metodou mark(int)

long skip(long n) přeskočí zadaný počte bajtů

1.6 Důležité třídy odvozené od InputStream

java.io.FilterInputStream - je bázová třída k odvozování všech vstupních proudů přidávajících vlastnost/schopnost filtrovat poskytnutý vstupní proud.Příklady filtrů (ne všechny jsou v `java.io`!):

BufferedInputStream proud s vyrovnávací pamětí (je možno specifikovat její optimální velikost)

java.util.zip.CheckedInputStream proud s kontrolním součtem (např. CRC32)

javax.crypto.CipherInputStream proud dešifrující data ze vstupu

DataInputStream má metody pro čtení hodnot primitivních typů, např. `float readFloat()`

1.7 Důležité třídy odvozené od InputStream (2)

java.security.DigestInputStream počítá současně i haš (digest) čtených dat, použitý algoritmus lze nastavit

java.util.zip.InflaterInputStream dekomprimuje (např. GZIPem) zabalený vstupní proud (má ještě specializované podtřídy)

LineNumberInputStream doplňuje informaci o tom, ze kterého řádku vstupu čteme (zavrhovaná - *deprecated* - třída)

ProgressMonitorInputStream přidává schopnost informovat o průběhu čtení z proudu

PushbackInputStream do proudu lze data vracet zpět

1.8 Další vstupní proudy

Příklad rekonstrukce objektů ze souborů

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

1.9 Další vstupní proudy (2)

javax.sound.sampled.AudioInputStream vstupní proud zvukových dat

ByteArrayInputStream proud dat čtených z pole bajtů

PipedInputStream roura napojená na "protilehlý" PipedOutputStream

SequenceInputStream proud vzniklý spojením více podřízených proudů do jednoho virtuálního

ObjectInputStream proud na čtení serializovaných objektů

1.10 Práce se znakovými proudy

základem je abstraktní třída `Reader`, konkrétními implementacemi jsou:

- `BufferedReader`, `CharArrayReader`, `InputStreamReader`, `PipedReader`, `StringReader`
- `LineNumberReader`, `FileReader`, `PushbackReader`

1.11 Znakové výstupní proudy

Nebudeme důkladně probírat všechny typy

- jedná se o protějšky k vstupním proudům, názvy jsou konstruovány analogicky (např. `FileReader` → `FileWriter`)
- místo generických metod `read` mají `write(...)`

Příklady:

PrintStream poskytuje metody pro pohodlný zápis hodnot primitivních typů a řetězců - příkladem jsou `System.out` a `System.err`

PrintWriter poskytuje metody pro pohodlný zápis hodnot primitivních typů a řetězců

1.12 Konverze: znakové – binární proudy

Ze vstupního binárního proudu `InputStream` (čili každého) je možné vytvořit znakový `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
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`).Obdobně pro výstupní proudy - lze vytvořit Writer z OutputStream.

1.13 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 (prázdné) rozhraní `java.io.Serializable`

1.14 Serializace objektů (2)

- proměnné objektu, které nemají být serializovány, musí být označeny modifikátorem - klíčovým slovem - `transient`
- pokud požaduje "speciální chování" při de/serializaci, musí objekt definovat metody
  - `private void readObject(java.io.ObjectInputStream stream) throws IOException, ClassNotFoundException`
  - `private void writeObject(java.io.ObjectOutputStream stream) throws IOException`
- metody:
  - `DataOutputStream.writeObject(Object o)`

1.15 Odkazy

Tutoriál essential Java I/O: kapitola z Oracle Java Tutorial (`http://docs.oracle.com/javase/tutorial/essential/io/`)

2 Práce se soubory

2.1 Principy

- Vše je opět v balíku `java.io`
- Základem je třída `java.io.File` - objekt třídy File je de facto nositelem jména souboru, jakási "brána" k fyzickým souborům na disku.
- Používá se jak pro soubory, tak adresáře, linky i soubory identifikované UNC jmény
- Opět plně platformově nezávislé

2.2 Práce se soubory (2)

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

- File.separatorChar - jako char
- File.separator - jako String
- File.pathSeparatorChar ; - jako char
- File.pathSeparator ; - jako String
- System.getProperty("user.dir") - adresář uživatele, pod jehož UID je proces JVM spuštěn

2.3 Třída File - vytvoření objektu

Vytvoření objektu třídy File konstruktorem (NEJEDNÁ SE PŘÍMÉ VYTVOŘENÍ SOUBORU NA DISKU!) - máme několik možností:

new File(String filename) zpřístupní v aktuálním adresáři soubor s názvem *filename*

new File(File baseDir, String filename) zpřístupní v adresáři *baseDir* soubor s názvem *filename*

new File(String baseDirName, String filename) zpřístupní v adresáři *se jménem baseDirName* soubor s názvem *filename*

new File(URL url) zpřístupní soubor se (souborovým - `file:`) URL *url*

2.4 Třída File - různé metody/testy

Testy existence a povahy souboru:

boolean exists() vrátí true, právě když zpřístupněný soubor (nebo adresář) existuje

boolean isFile() test, zda jde o soubor a nikoli adresář

boolean isDirectory() test, zda jde o adresář

2.5 Třída File - testy (2)

Test práv ke čtení/zápisu:

boolean canRead() test, zda lze soubor číst

boolean canWrite() test, zda lze do souboru zapisovat

2.6 Vytvoření souboru/adresáře

Vytvoření souboru nebo adresáře:

boolean createNewFile() (pro soubor) vrací `true`, když se podaří *soubor* vytvořit

boolean mkdir() (pro adresář) vrací `true`, když se podaří adresář vytvořit

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

2.7 Vytvoření dočasného souboru

Vytvoření dočasného (temporary) souboru:

static File createTempFile(String prefix, String suffix) skutečně fyzicky vytvoří dočasný soubor ve standardním, pro to určeném, adresáři (např. `c:/temp`) s uvedeným prefixem a sufixem názvu

static File createTempFile(String prefix, String suffix, File directory) dtto, ale vytvoří dočasný soubor v zadaném adr. `directory`

Zrušení:

boolean delete() zrušení souboru nebo adresáře

Přejmenování (ne přesun mezi adresáři!):

boolean renameTo(File dest) přejmenuje soubor nebo adresář

2.8 Smazání, přejmenování

Zrušení:

boolean delete() zrušení souboru nebo adresáře

Přejmenování (ne přesun mezi adresáři!):

boolean renameTo(File dest) přejmenuje soubor nebo adresář

2.9 Třída File - další vlastnosti

long length() délka (velikost) souboru v bajtech

long lastModified() čas poslední modifikace v ms od začátku éry - podobně jako systémový čas vracený System.currentTimeMillis().

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

Blíže viz dokumentace API třídy File (`http://download.oracle.com/javase/6/docs/api/java/io/File.html`).

2.10 Práce s adresáři

Klíčem je opět třída `File` - použitelná i pro adresářeJak např. získat (filtrovaný) seznam souborů v adresáři?pomocí metody `File[] listFiles(FileFilter ff)` nebo podobnéFile[] 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`)