IO Java

<https://java.vse.cz/pdf/java-vstupy_vystupy.pdf>

https://javatutorialhq.com/java/io/bufferedreader-class-example/read-char-cbuf-int-off-int-len-method-example/

Na prácu so vstupami a výstupami Java poskytuje triedy a metódy, ktoré sú uložené v balíčku java.io. Práca s IO je založená na mechanizmu vstupných a výstupných prúdov(streams). Stream je postupnosť

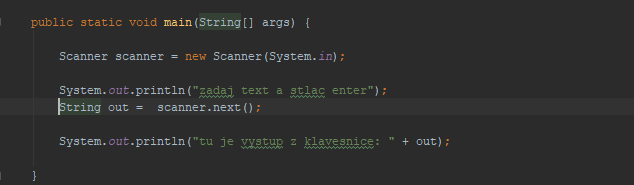
dát. V Jave stream pozostáva y bajtov. Doteraz sme sa stretli s tromi prúdmi. Tieto streamy sú prepojené s konzolou.

**1) System.out:** standard output stream

**2) System.in:** standard input stream

**3) System.err:** standard error stream

Priklad System.out a System.in:

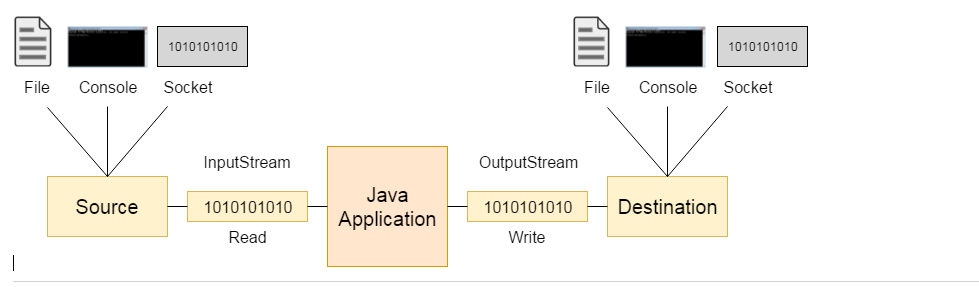


System.err vidime vždy ak sa nám niečo nepodarí. (kto ma viac System.err vyhrá 😊)

InputStream vs OutputStream

Pokiaľ chceme IO ( Input/Output, vstup/výstup) použit inde ako v interakciou s konzolou treba použit triedy Inputstream a OutputStream.   
Pre vysvetlenie pojmov: InputStream sa používa pri čítaní dát do Java aplikácie a OutputStream opačne. Treba názvoslovie brať z pohľadu Java aplikacie. Teda aplikácia číta dáta, ide do aplikácie ( InputStream ).

Ako zdroj/cieľ môžeme brať súbor, zariadenie (sieťová karta) atď.



File.class

Absolutna cesta suboru alebo pomocou

URL fileUrl = getClass().getResource("/subor/subor.txt");

priklad OuputStreamu   
príklad InputSteamu

ByteStreams vs character stream.

Java IO prúdy sa rozdelujú podľa toho aký je zdroj streamu. A to: bajtový alebo znakový zdroj. Streamy zaloťžené na bajtovom zdroji majú v názve tried slovo „stream“ InputStream alebo OutputStream. Tieto streamy zapisujú alebo čítajú neupravené bajty.

Streamy, ktoré pracujú so znakmi tak nazvz tied obsahujú Reader alebo Writer. Streamy ktorých zakladom sú znaky môžme tie znaky priamo čítať-zapisovať. Napr znaky v UNICODE tabuľke.

## Reading Files via Java IO

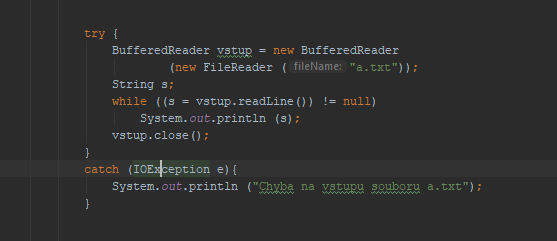
Pokiaľ chceme čítať súbor možme použit [**FileInputStream**](http://tutorials.jenkov.com/java-io/fileinputstream.html) alebo [**FileReader**](http://tutorials.jenkov.com/java-io/filereader.html) podľa toho či chceme čítať binárne alebo textové data. Tieto dve triedy nám umožňujú čítať jeden bajt alebo znak od začiatku suboru po koniec, alebo čítať zdrojové dáta po väčších častiach tak že sa jednotlivé znaky ukladajú do poľa typu char alebo byte. (priklad treba dat)

try (FileReader nacitajZnak = new FileReader("a.txt")) {  
 // a je mozne citat jednotlive znaky  
 int znak = nacitajZnak.read();  
 System.*out*.println((char)znak);  
}catch (IOException ex){  
 System.*out*.println(ex);  
}

Najčastejšie používaným vstupom je textový súbor. Ak chceme prečítať tento súbor po riadkoch, je nutné si pre to vytvoriť vstupný prúd. Pre čítanie po znakoch zo súboru A.TXT si môžeme pripraviť inštanciu triedy FileReader : *FileReader ctiZnak = new FileReader ("a.txt");* . A potom môžme čítať jednotlivé znaky z tohto súboru pomocou metódy read().

*znak = ctiZnak.read ( );*

Väčšinou chceme čítať súbor po riadkoch a pre toto čítanie musíme inštanciu triedy FileReader ešte zabaliť do triedy *BufferedReader.*



A potom môžme použiť metódu readLine().

*String s = radek.readLine();*

Obidve triedy poskytujú metódu close() na zatvorenie súboru. Je dôležité aby sa metóda close() použila vždy po skončení práce so súborom. Dôvod je ten, že ak JVM začne pracovať so súborom tak OS uzamkne súbor pre použitie pre ostatné aplikácie. A je potrebné, aby sa súbor uvoľnil po skončení práce s ním.

Trieda File

Trieda File slúži na manipuláciu so súbormi a adresármi. Inštanciou triedy File môže byt adresár alebo súbor. Pri vytváraní inštancie nie je nutné aby súbor/adresár existoval. Trieda File slúži aj na vytvorenie súboru. Na popis cesty sa používa /.

|  |  |
| --- | --- |
| **príklad použitia** | **význam** |
| File mojSubor = new File ("a.txt") | inštancia mojSubor nastavená na a.txt v aktualnom adresári |
| File mojaPosta = new File("C:"+File.separator+ "posta") | inštancia mojaPosta nastavena na adresár posta na disku C |
| File mojList = new File ("C:/posta/list1.txt") | inštancia mojaPosta nastavena na súbor list1txt v adresári pošta na disku C |
| File mujList = new File ("C:\\posta","list1.txt") | inštancia mojList nastavená na súbor list1.txt a adresari posta |
| File mojList = new File (mojaPosta,"list1.txt") | instance mojList nastavená na súbor list1.txt v adresári posta na disku C, použili jsme instanciu mojaPošta vytvorenú vyššie |

## Trieda File obsahuje metódy isFile() a is Directory(), ktoré zistia či danná inštancia triedy je súbor alebo adresár. Ak chceme zisti, či adresár alebo súbor existuje použijeme metódu exists(). Na vytvorenie adresára sa použie metóda mkdir() a na vytvorenie súboru sa požije metóda createNewFile(). Ďalšie metódy: delete(), renameTo(), list() – výpis obsahu adresára.

## Writing File via Java IO

If you need to write a file from one end to the other you can use a **[FileOutputStream](http://tutorials.jenkov.com/java-io/fileoutputstream.html)** or a **[FileWriter](http://tutorials.jenkov.com/java-io/filewriter.html)** depending on whether you need to write binary data or characters. You can write a byte or character at a time from the beginning to the end of the file, or write arrays of byte and char. Data is stored sequentially in the file in the order they are written.

File class. Co s tym mozeme robit.

public class GetFileInfo {   
 public static void main(String[] args) {

File myObj = new File("filename.txt");

if (myObj.exists()) {

System.out.println("File name: " + myObj.getName());

System.out.println("Absolute path: " + myObj.getAbsolutePath());

System.out.println("Writeable: " + myObj.canWrite());

System.out.println("Readable " + myObj.canRead());

System.out.println("File size in bytes " + myObj.length());

} else {

System.out.println("The file does not exist.");

}

}

}