Vhod in izhod (delo z datotekami)

Programiranje 2, Tomaž Dobravec

Vhod/izhod

- vhod/izhod (angl. input/output ali I/O)
- program na vhodu dobi podatke na izhod pa jih zapiše
- vrste vhodov in izhodov:

VHOD	IZHOD
standardni vhod (običajno: tipkovnica)	standardni izhod (običajno: zaslon)
datoteka	datoteka
spomin	spomin
miš	tiskalnik
mrežna povezava	mrežna povezava
...	...

Vhod/izhod

- Java predvideva ENOTEN način dela z vsemi vhodi in vsemi izhodi.
- Podatki se berejo iz vhoda in pišejo na izhod s pomočjo vhodno/izhodnih tokov.
- Za delo z vhodom in izhodom potrebujemo paket `java.io`

```
import java.io.*;
```

Znaki in števila

- Računalnik v osnovi “razume” le števila
- Črka je le vidna reprezentacija nekega števila
- Java zna ‘pretvarjati’ med znaki in števili

- Primer: Kaj bo izpisal spodnji program?

```
int  ai = 97;   char ac = (char) ai;
char bc = 'A';  int  bi = (int) bc;
System.out.println(ai); System.out.println(ac);
System.out.println(bc); System.out.println(bi);
```

ASCII tabela

- Osnova za preslikavo med črkami in števili se imenuje ASCII tabela.
- Osnovna ASCII tabela je 7-bitna (2^7=128 mest v tabeli).
- V ASCII tabeli je zakodiranih 95 "vidnih" znakov ...
  - črke in številke ter znaki "# $%&()*+'-./:;<=>?@[\]^_`{|}

  ... ter 33 kontrolnih znakov (zgodovina)
  - LF (10), CR(13), bell(7), DEL (127), TAB (9), BACKSPACE (8), ...
- Nekatere ASCII kode:
  - 'A' = 65, 'B' =66, ... 'a' =97, 'b' =98, ...
  - '0' = 48, '1' =49, ...
  - ' ' (presledek) = 32

ASCII tabela

- Težava: 128 mest je premalo za vse znake, ki jih uporabljamo
- Delna rešitev: razširjena ASCII tabela je 8-bitna (2^8=256 mest v tabeli).
- ASCII tabelo najdeš na

  http://www.lookuptables.com/

Domača naloga: izpis vseh znakov ASCII tabele

Unicode

- Tudi razširjena ASCII tabela (256 znakov) ni dovolj za predstavitev vseh znakov, ki jih potrebujemo.
- Manjkajo čššČŠŽ, grške črke, cirilica, ...
- Rešitev?

  Unicode
- Velikost osnovne Unicode tabele je 2^{16} (= 65536 znakov).
- ASCII tabela je vsebovana v Unicode tabeli, osnovni znaki so v obeh tabelah enako oštevilčeni.

Unicode

- Unicode je razdeljen na sklope po 127 znakov
- Vsak sklop ima svoje ime
- Za nas zanimivi sklopi so:
  - Basic Latin (kode med 0x0000 in 0x007F)
  - Latin Extended-A (kode med 0x0100 in 0x017F)

    tu so med drugim definirani znaki čšžČŠŽ

Unicode

- Kode slovenskih znakov:

č = \u010d	Č = \u010c
š = \u0161	Š = \u0160
ž = \u017e	Ž = \u017d

- Java v celoti podpira Unicode (znotraj Jave je za vsak znak rezerviranih 16bitov)
- Podrobnosti o Unicode: http://www.unicode.org/charts

Primer: Unicode v Javi

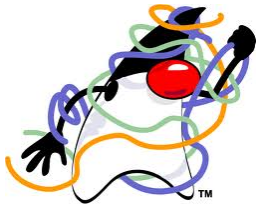
Kodiranje znakov

Kako zapisati Unicode znake v datoteko?

- Star (ASCII) način:

  Vsakemu znaku pripada en byte (8-bitov)

- Nove zahteve:
  - Znaki so dolgi dva byte-a (16-bitov)
  - Težave:
    - za zapis v datoteko potrebujemo dvakrat več prostora kot nekoč
    - nekompatibilnost s starimi datotekami



Kodiranje znakov

- Rešitev (ki reši večino omenjenih težav):

Kodiranje znakov

- **Pisanje v datoteko:** izberemo enega od kodirnih standardov
- **Branje datoteke:** vedeti moramo, kateri od kodirnih standardov je bil uporabljen za zapis datoteke
- Večina datotek nima predvidenega (standardnega) mesta, kjer bi pisalo, kako je bila zakodirana

Najpogostejši kodirni standardi

ASCII	- Star način - vsakemu znaku pripada en byte - Zapis znakov s kodo nad 255 ni mogoč
UTF-16	- Vsakemu znaku pripada 16bitov (16 bitni Unicode zapis brez kodiranja)
UTF-8	- znaki s kodo <128 so zapisani z enim byte-om - znaki s kodo >=128 zasedejo dva ali več byte-ov Prednosti: – popolna ASCII kompatibilnost v eno smer (ASCII datoteka je tudi UTF-8 datoteka, obratno ne velja) – racionalna poraba prostora (več kot en byte zasedejo samo znaki, ki se jih res ne da zakodirati z enim byte-om)
cp1250	8-bitni način kodiranja; kode do 127 so rezervirane za ASCII znake, kode >127 pa ISO-1250 (Latin2) znake

Kodiranje znakov in Java

- Privzeto Javino kodiranje: odvisno od sistema.

- Prevajamo lahko tudi datoteke, ki so zapisane z drugim kodiranjem (npr. cp1250)

```
javac –encoding cp1250 ImeRazreda.java
```

Kodiranje znakov in Java

US-ASCII	Seven-bit ASCII, a.k.a. ISO646-US, a.k.a. the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, a.k.a. ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byte order
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byte order
UTF-16	Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark

Zlogovne in znakovne datoteke v Javi

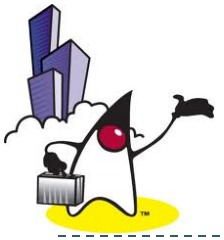
- Java razlikuje med
  - zlogovnimi (byte) in
  - znakovnimi (char) datotekami
- Pri znakovnih datotekah Java avtomatsko skrbi za pretvorbo v Unicode
- Programer lahko določi način (de)kodiranja (sicer se uporabi privzet način)

Razred `java.io.File`

- Razred `File` omogoča sistemsko neodvisen dostop do datotečnega sistema.

- **Objekt razreda `File` uporabljamo za delo z datotekami in NE za delo z vsebino datotek!**

- Objekt razreda File lahko predstavlja
  - datoteko ali
  - direktorij.



Kako ustvarim objekt razreda `File`

Primeri:

```
File f = new File("c:\\AUTOEXEC.BAT");
```

- Dva ekvivalentna načina:

```
File pd = new File("/delo/datoteka.txt");
```

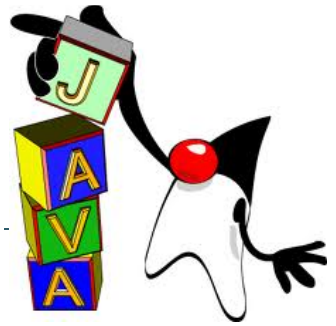
```
File p = new File("/delo");
File pd = new File(p, "datoteka.txt");
```

Nekatere metode objektov razreda File

- `public boolean` **`canRead`**`()`
- `public boolean` **`canWrite`**`()`
- `public boolean` **`exists`**`()`
- `public boolean` **`isDirectory`**`()`
- `public long` **`lastModified`**`()`
- `public long` **`length`**`()`
- `public boolean` **`delete`**`()`
- `public boolean` **`mkdir`**`()`
- `public boolean` **`mkdirs`**`()`
- `public boolean` **`renameTo`**`(File dest)`
- `public String[]` **`list`**`()`

Naloga

Izpis velikosti datoteke

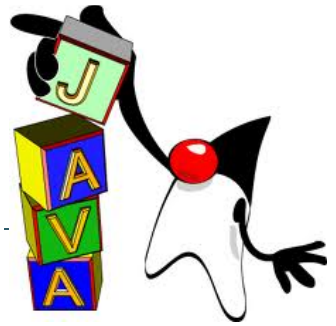


`VelikostDatoteke.java`

- Napiši program, ki izpiše velikost datoteke, ki je podana kot prvi argument ob klicu programa.

Naloga

Izpis drevesa datoteke



`TreeA.java, TreeB.java`

Napiši program, ki izpiše drevo datotek, kot ga izpiše program `tree` v Linux lupini.

Verzija A

```
build
build/classes
build/classes/Dn1.class
build/classes/Dn2.class
build/classes/Dn3.class
build/classes/Dn4.class
build/classes/Dn5.class
build/classes/KvadratneEnacbe.class
build/classes/p2
build/classes/p2/testi
build/classes/p2/testi/Test.class
```

Verzija B

```
|____build
| |____classes
| | |____Dn1.class
| | |____Dn2.class
| | |____Dn3.class
| | |____Dn4.class
| | |____Dn5.class
| | |____KvadratneEnacbe.class
| | |____p2
| | | |____testi
| | | | |____Test.class
```

Tokovi

- Tok je zaporedje podatkov.
- V Javi se tokovi uporabljajo za branje/pisanje podatkov iz vhoda/na izhod
- Vsak vhodni tok ima svoj izvor, vsak izhodni tok svoj ponor
- Razlika med tabelo podatkov in tokom podatkov?
  - tabela: v vsakem hipu enako hitro dobimo katerikoli podatek
  - tok: v nekem hipu lahko beremo le “trenutni” podatek
- Primerjava toka in avdio kasete
  - poslušam samo skladbo, ki je trenutno na vrsti
  - nekateri kasetofoni ne dovolijo previjanja, prav tako tega ne dovolijo nekateri javanski tokovi

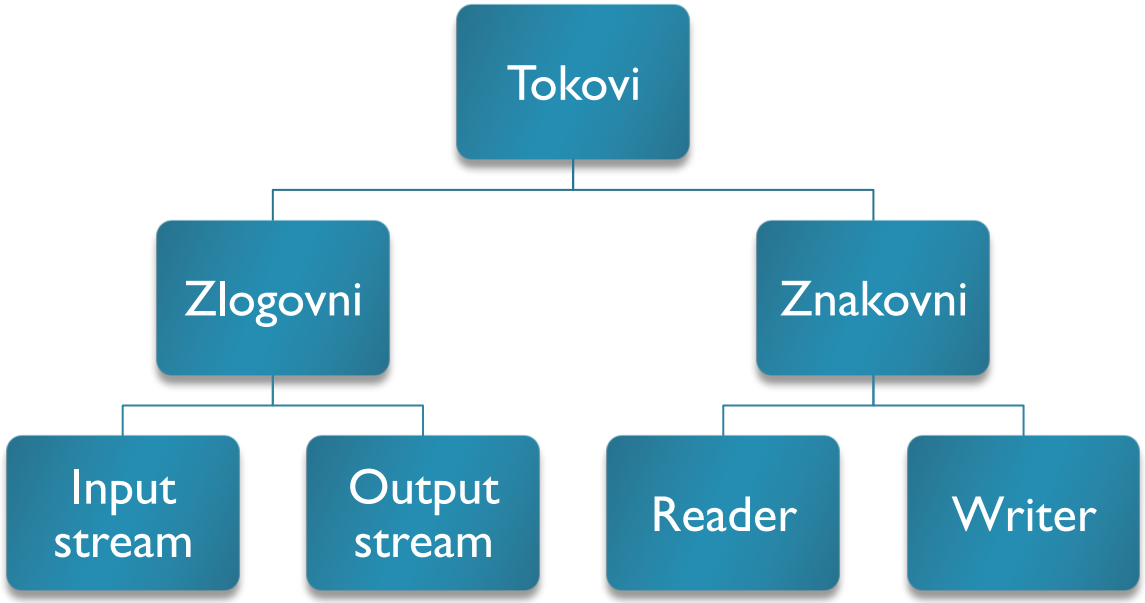
Tokovi

- Tok moramo
  - pred uporabo odpreti (open)
  - po uporabi zapreti (close)
- Java pozna več tipov tokov
- Groba delitev: vhodni in izhodni tokovi
  - skoraj vsak vhodni javanski tok ima svoj izhodni ekvivalent
- Podrobnejša delitev: glede na način uporabe toka

Tokovi v Javi

- Java loči med znakovnimi in zlogovnimi tokovi.

- Osnovni podatkovni tip za zlogovne tokove je zlog (byte – 8 bitov), za znakovne pa znak (char – 16 bitov).



Flitri

- Posebno skupino tokov predstavljajo t.i. filtrirni tokovi ali *filtri*.
- Filter vedno uporabimo v kombinaciji z nekim drugim (osnovnejšim) tokom.
- Filter ovije osnovnejši tok in mu da dodatno funkcionalnost

Filtri

Data filter

Omogoča branje in pisanje javanskih primitivnih tipov (`byte, int, char, ...`) v zlogovne datoteke.

Buffered filter

Omogoča branje in pisanje podatkov v paketih (posledica: veliko večja hitrost!)

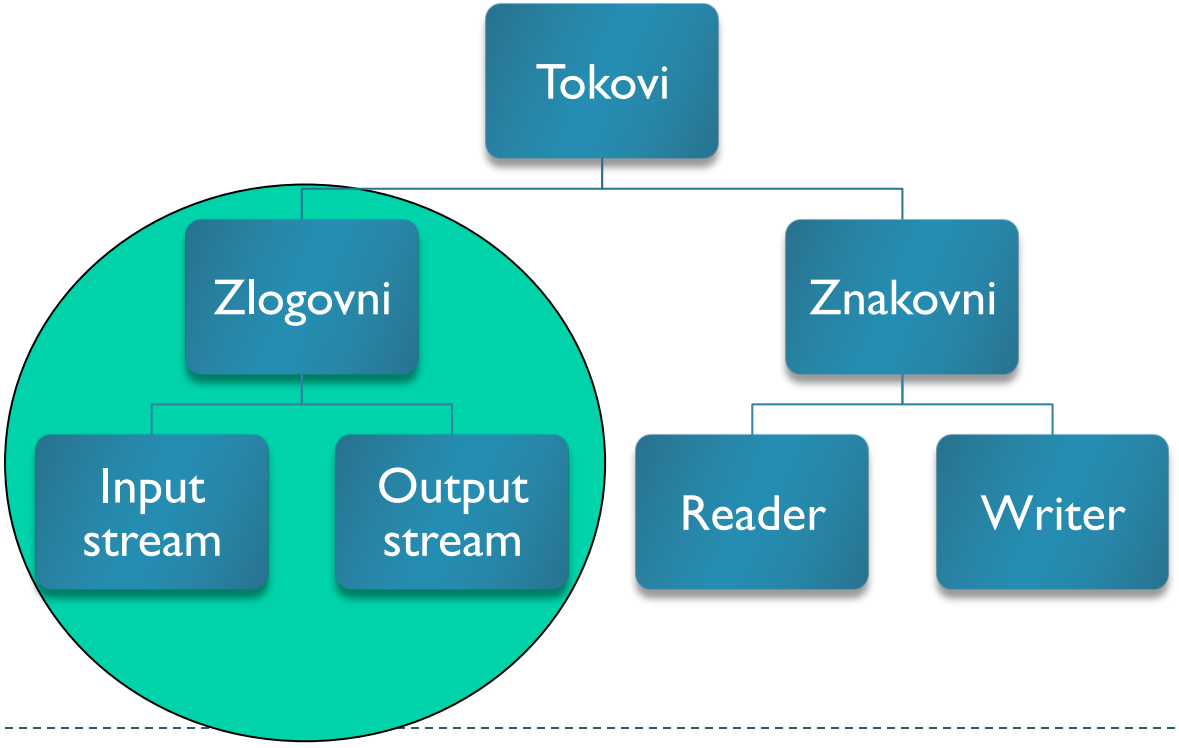
Print filter

Omogoča izpis javanskih primitivnih tipov v tekstovne datoteke.

Zlogovni tokovi

Zlogovni tokovi se uporabljajo za branje/pisanje podatkov

Primer: v gif datoteki je slika opisana z zaporedjem zlogov



Vhodni zlogovni tokovi

- Vhodni zlogovni tok se imenuje *input stream*
- Vsi javanski vhodni zlogovni tokovi so nasledniki abstraktnega razreda InputStream
- Nekatere metode razreda InputStream

```
public abstract int read()
public int read(byte[] b)
public int read(byte[] b, int off, int len)
```

Vse omenjene metode ob napaki vržejo izjemo `IOException`

Izhodni zlogovni tokovi

- Izhodni zlogovni tok se imenuje output *stream*
- Vsi javanski izhodni zlogovni tokovi so nasledniki abstraktnega razreda `OutputStream`
- Nekatere metode razreda OutputStream

```
public abstract void write(int b)
public void write(byte[] b)
public void write(byte[] b, int off, int len)
```

Vse omenjene metode ob napaki vržejo izjemo `IOException`

Nekatere Implementacije razredov InputStream in OutputStream

Vhodni podatkovni tokovi

```
java.lang.Object
 |
 +--java.io.InputStream
     |
     +--java.io.FileInputStream
     |
     +--java.io.FilterInputStream
     |    |
     |    +--java.io.DataInputStream
     |    |
     |    +--java.io.BufferedInputStream
     |
     |
     |
     +--java.io.ObjectInputStream
```

Izhodni podatkovni tokovi

```
java.lang.Object
 |
 +--java.io.OutputStream
     |
     +--java.io.FileOutputStream
     |
     +--java.io.FilterOutputStream
     |    |
     |    +--java.io.DataOutputStream
     |    |
     |    +--java.io.BufferedOutputStream
     |    |
     |    +--java.io.PrintStream
     |
     +--java.io.ObjectOutputStream
```

FileInputStream in FileOutputStream

- Osnovna razreda za branje/pisanje zlogovnih datotek
- Nekateri konstruktorji

```
public FileInputStream(File file)
public FileInputStream(String name)

public FileOutputStream(File file)
public FileOutputStream(String name)
public FileOutputStream(String name, boolean append)
```

Vsi omenjeni konstruktorji ob napaki vržejo izjemo `FileNotFoundException`

FileInputStream in FileOutputStream

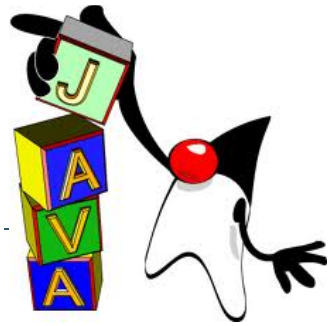
- `FileInputStream`
  - če datoteka, s katero želimo povezati tok, ne obstaja, konstruktor vrže izjemo `FileNotFoundException`

- `FileOutputStream`
  - če datoteka že obstaja, se bo ob klicu konstruktorja resetirala (vsebina se pobriše)
  - pri nekaterih konstruktorjih lahko izberemo opcijo *append* (dodajanje podatkov v datoteko)
  - če želimo pisati v datoteko, pa za to nimamo pravic, konstruktor vrže izjemo `SecurityException`

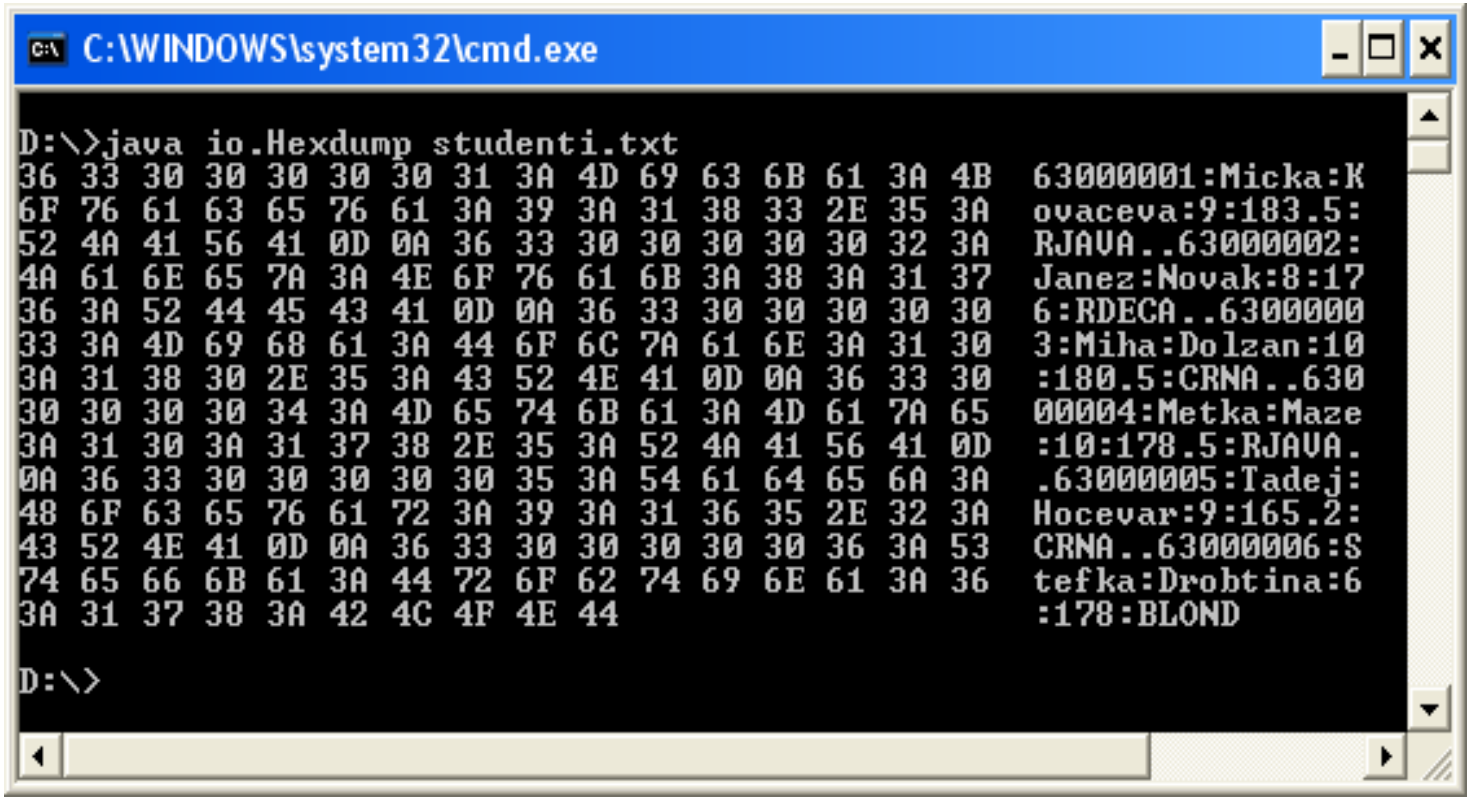
Naloga

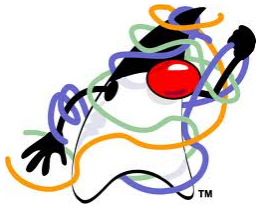
Izpis datoteke po zlogih (hexdump)

io/Hexdump.java



Napiši program za izpis zlogov (šestnajstiške kode) datoteke.





DataInputStream in DataOutputStream

Branje in pisanje javanskih osnovnih podatkovnih tipov (`boolean, char, byte, short, int, long, float, double, String`) v zlogovni obliki

Pisanje:

- boolean se zapiše z enim zlogom kot 0 ali 1
- char se zapiše z dvema zlogoma
- int se zapiše s štirimi zlogi
- ...

Branje:

- obratno od pisanja

DataInputStream in DataOutputStream

- `DataInputStream` in `DataOutputStream` sta ovojna tokova (filtra) – vedno jih uporabimo v kombinaciji z nekim drugim tokom

- Konstruktorji:

```
public DataInputStream(InputStream in)
public DataOutputStream(OutputStream out)
```

DataOutputStream - uporaba

1. Ustvarimo FileOutputStream

```
FileOutputStream fos = new FileOutputStream("podatki.dat");
```

2. Ustvarimo DataOutputStream

```
DataOutputStream dos = new DataOutputStream(fos);
```

3. Zapišemo podatke

```
dos.writeBoolean(true);
dos.writeByte(32);
dos.writeChar('A');
dos.writeChar('\u01D7');
dos.writeChars("abcd");
dos.writeUTF("abcd");
dos.writeInt(32);
```

4. Zapremo tok

```
dos.close();
```

01 20 00 41 01 D7 00 61 00 62 00 63 00 64 00 04 61 62 63 64 00 00 00 20

DataInputStream - uporaba

1. Ustvarimo FileInputStream

```
FileInputStream fis = new FileInputStream("podatki.dat");
```

2. Ustvarimo DataInputStream

```
DataInputStream dis = new DataInputStream(fis);
```

3. Preberemo podatke

```
boolean b = dis.readBoolean();
byte    z = dis.readByte();
char    c1 = dis.readChar();
char    c2 = dis.readChar();
String  s1 = dis.readLine();
String  s2  = dis.readUTF();
int     i = dis.readInt();
```

4. Zapremo tok

```
dis.close();
```

`BufferedInputStream` in `BufferedOutputStream`

- `BufferedInputStream` in `BufferedOutputStream` sta ovojna tokova (filtra) – vedno jih uporabimo v kombinaciji z nekim drugim tokom
- Namen *buffered* filtrov – branje in pisanje podatkov v večjih sklopih (namesto zlog po zlogu)
- Uporabi se medpomnilnik (buffer), zato jih imenujemo tudi medpomnilniški tokovi
- Zaradi uporabe medpomnilnika se bistveno poveča hitrost.

Uporaba medpomnilniških tokov - primer

1. Ustvarimo FileInputStream

```
FileInputStream fis = new FileInputStream("podatki.dat");
```

2. Ustvarimo BufferedInputStream

```
BufferedInputStream bif = new BufferedInputStream(fis);
```

3. Ustvarimo DataInputStream

```
DataOutputStream dis = new DataOutputStream(bif);
```

- Tok *dis* uporabimo na enak način, kot smo ga uporabili v primeru pri `DataInputStream`

- Na enak način uporabimo BufferdOutputStream

Serializacija in deserializacija objektov

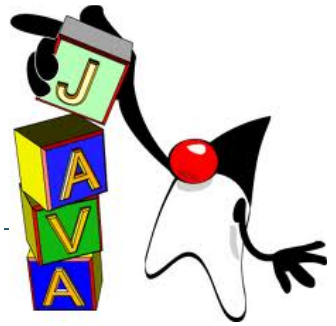
- Serializacija objektov je postopek, ki omogoča pretvorbo objekta v zaporedje zlogov.
- Deserializacija je postopek za generiranje objektov iz (pravilnega) zaporedja zlogov
- Objekt, ki ga želimo serializirati, mora implementirati vmesnik *Serializable.*

ObjectInputStream in ObjectOutputStream

- Tok `ObjectOutputStream` omogoča pisanje objektov v datoteko (pred zapisom se izvede serializacija).
- Tok `ObjectInputStream` omogoča branje objektov iz datoteke (po branju se izvede deserializacija).
- `ObjectXXXStream` pozna tudi metode za branje/pisanje javanskih osnovnih tipov.
- `ObjectXXXStream` lahko uporabimo v kombinaciji z BufferedXXXStream.

Naloga

Pisanje in branje objektov

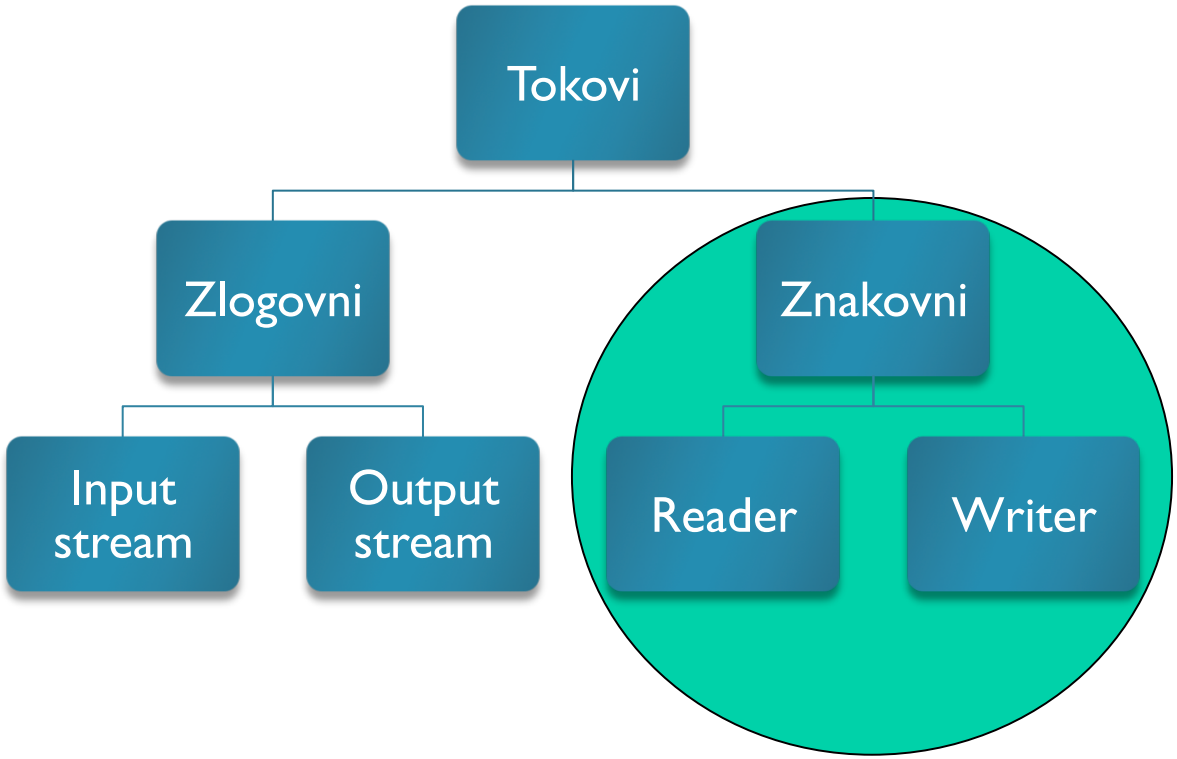


io/Imenik.java

- Napiši razred `Oseba`, v katerem hraniš podatke (ime, priimek in starost) o neki osebi. Napiši razred `Imenik`, ki omogoča pisanje in branje objektov tipa `Oseba`.

Znakovni tokovi

- Znakovni tokovi se uporabljamo za branje človeku razumljivih znakov (črke)



Znakovni tokovi

- Osnovne smernice:
  - Za branje in pisanje znak po znaku uporabim `InputStreamReader` in `OutputStreamWriter`
  - Za formatirano branje uporabim `Scanner`
  - Za formatiran izpis uporabim `PrintWriter`

InputStreamReader in OutputStreamWriter

- Tokova `InputStreamReader` in `OutputStreamWriter` uporabljamo za branje/pisanje znakov iz datoteke.
- Za kodiranje poskrbi java.
- Če uporabnik kodiranja ne določi eksplicitno, se uporabi sistemsko privzet način.

InputStreamReader

- Nekateri konstruktorji:

```
public InputStreamReader(InputStream in)

public InputStreamReader(InputStream in,
                         String charsetName)
```

- Nekatere metode

```
public String getEncoding()
public int read()
public boolean ready()
```

- Priporočena je uporaba `BufferedReader`-ja

OutputStreamWriter

- Nekateri konstruktorji:

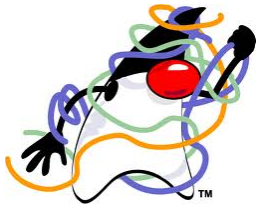
```
public OutputStreamWriter(OutputStream out)

public OutputStreamWriter(OutputStream out,
                          String charsetName)
```

- Nekatere metode

```
public String getEncoding()
public void write(int c)
public void close()
```

- Priporočena je uporaba `BufferedWriter`-ja



InputStreamReader in OutputStreamWriter

- Uporaba InputStreamReaderja z medpomnilnikom

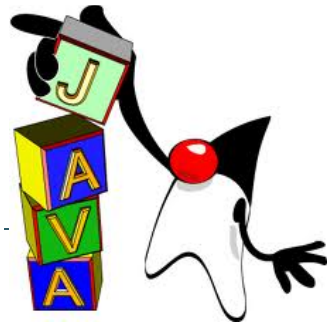
```
BufferedReader vhod  = new BufferedReader(
          new InputStreamReader(System.in));
```

- Uporaba OutputStreamWriter-ja z medpomnilnikom

```
BufferedWriter izhod = new BufferedWriter(
          new OutputStreamWriter(System.out));
```

Naloga

Pretvorba kodirnih sistemov



io/Pretvornik.java

Napiši program, ki omogoča pretvorbo iz enega v drug kodirni sistem. Ime vhodne in izhodne datoteke ter imena obeh kodirnih sistemov so podani kot argumenti ob klicu programa.

FileReader in FileWriter

- Razreda `FileReader` in `FileWriter` predstavljata poenostavitev razredov `InputStreamReader` in `OutputStreamWriter`:
  - uporabljata privzet način kodiranja
  - avtomatsko se povežeta z datoteko

- Preprosta konstruktorja

```
public FileReader(String fileName)
public FileWriter(String fileName)
```

Scanner

- Razred Scanner se uporablja za branje besedila in za razbijanje prebranega besedila na posamezne dele
- Pri razbijanju besedila se uporablja ločilo (delimiter)
- Privzeta ločila: tabulator, presledek, nova vrsta
- Posamezni deli besedila se lahko avtomatsko pretvorijo v javanske primitivne tipe.

- Primer branja celega števila iz standardnega vhoda:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

Scanner

- Nekateri konstruktorji:

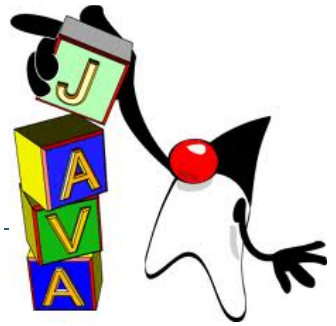
```
public Scanner(File source)
public Scanner(File source, String charsetName)
public Scanner(InputStream source)
public Scanner(String source)
```

- Nekatere metode:

```
public boolean hasNextInt()
public int nextInt()
public boolean hasNextBoolean()
public boolean nextBoolean()
```

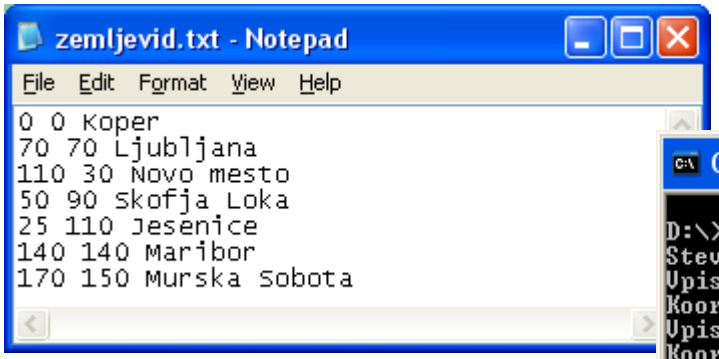
Naloga

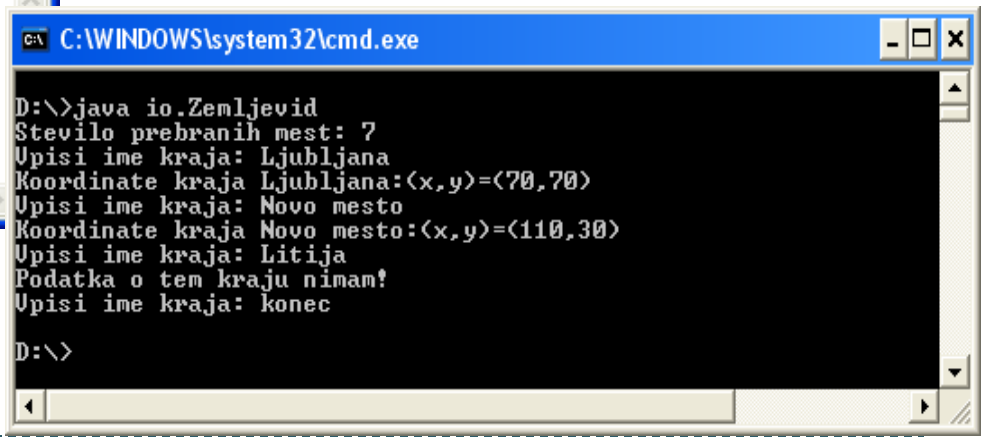
Scanner - primer



io/Zemljevid.java

V datoteki *zemljevid.txt* so podani podatki o koordinatah slovenskih mest. Napiši program, ki to datoteko prebere, nato pa od uporabnika v zanki zahteva vpis imena mesta; za vsako prebrano mesto naj program izpiše koordinate.





PrintWriter

- `PrintWriter` je ovojni tok, ki se uporablja za formatiran izpis besedila
- `PrintWriter` izpisuje primitivne javanske tipe v tekstovni obliki
- Za kodiranje podatkov poskrbi oviti tok
- Nekateri konstruktorji:

```
public PrintWriter(OutputStream out)
public PrintWriter(Writer out)
```

PrintWriter

- Nekatere metode
  - `public void print(char c)`
  - `public void print(int i)`
  - `public void print(double d)`
  - `...`

- Primer uporabe:

```
PrintWriter pw = new PrintWriter("C:\\test.txt");
    pw.print(true);
    pw.println(32);
    pw.println("Besedilo");
pw.close();
```

true
32
Besedilo

Branje iz tipkovnice – standardna ročica `System.in`

Za branje iz standardnega vhoda (tipkovnice) uporabljamo tok `System.in`

Primer: datoteko odpremo, na primer, takole

```
FileInputStream fis = new FileInputStream("datoteka.txt");
BufferedReader br = new BufferedReader(new InputStreamReader(fis));
```

tipkovnico pa takole:

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

- Najlažji način branje iz tipkovnice: uporaba razreda `Scanner`

```
Scanner sc = new Scanner(System.in);
```

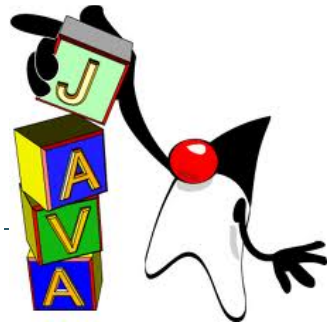
Branje iz drugih virov

- Za branje podatkov lahko poleg tipkovnice in datoteke uporabljamo tudi druge vire (niz, spomin, internet, ...)

- Primer branja podatkov iz internetne strani:

```
URL yahoo = new URL("http://www.yahoo.com");
Scanner sc = new Scanner(yahoo.openStream());
while (sc.hasNextLine()) {
   System.out.println(sc.nextLine());
}
sc.close();
```

Naloga Branje iz drugih virov



io/Temperatura.java

Napiši program, ki izpiše trenutno temperaturo v Ljubljani. Podatke o temperaturi preberi na strani `http://www.vreme-on.net/temperature-veter.html`


```
70  <td bordercolor="#000000" style="border: thin
    solid rgb(0,0,0)" ><font face="verdana, Arial,
71  Helvetica"><strong><font
    color=Brown><small>Temperatura</font></strong><br>
    </small></font></td>
72  <td width="150" bordercolor="#000000"
    style="border: thin solid rgb(0,0,0)" ><font
    face="verdana, Arial,
73  Helvetica"><strong><small><font
    color="#3366FF">19.7C</font><br></small></strong><
    /font>
```