Java in vhodno izhodne operacije

Alenka Kavčič
alenka.kavcic@fri.uni-lj.si

DATOTEKE IN TOKOVI V JAVI: pregled in uporaba v praksi

Alenka Kavčič

alenka.kavcic@fri.uni-lj.si

Pregled vsebine

- Datoteke v Javi
  - branje in pisanje datotek
- Tokovi v Javi
  - pregled
  - primeri uporabe tokov
- Kako beremo besedilo
  - učinkovito in enostavno
  - primer uporabe bralnika

- V paketu java.io najdemo razrede za delo z vhodom in izhodom, tudi z datotekami
  - preko 50 razredov in vmesnikov

Potrebno predznanje

- Osnovni koncepti OO programiranja v Javi
  - razredi, objekti
  - dedovanje
- Izjeme in obravnava izjem
  - kaj so izjeme
  - vrste izjem
    - preverjene (Exception)
    - nepreverjene (Error, RuntimeException)
  - obravnava izjem
    - prestrezanje izjem, proženje izjem
    - razglašanje izjem (posredovanje naprej)

Datoteke

- Kaj je datoteka
  - shranjeni podatki na disku, CD-ju, USB ključku, ...
    - datoteke s podatki uporabnika
      - slike, zapiski, glasba, ...
    - datoteke s programi
      - izvorna koda, prevedeni programi, aplikacije
  - vsaka datoteka
    - zaseda prostor na disku (velikost)
    - ima ime (skupaj s potjo v datotečnem sistemu)
    - čas kreiranja
    - ostali atributi

Razred java.io.File

- Predstavitev datoteke v programu
  - dostop do podatkov o datoteki
    - ali obstaja; lahko beremo/pišemo
    - je navadna datoteka; je direktorij
    - je odprta
    - velikost datoteke
  - <u>ni</u> odpiranja, zapiranja, procesiranja vsebine!

Konstruktor razreda File

- Več različnih konstruktorjev
- Najpogosteje podamo niz z imenom datoteke

```
File datoteka = new File("ime.txt");
```

- Ime lahko vsebuje tudi pot

```
File datoteka = new File("/home/vaje/ime.txt");
```

  – zapis poti na Windows in Linux je različen
  – pozor: znak \ je del ubežne sekvence v Javi

```
File datoteka = new File("C:\\prog2\\vaje\\ime.txt");
```

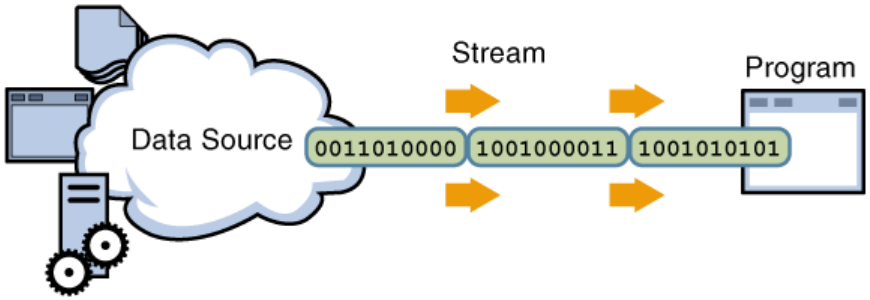
Branje/pisanje datotek

- Zaporeden dostop do podatkov
  - tokovi
  - bajti, znaki
  - formatirano, podatki imajo tipe: int, double, String, ...
- Naključen dostop do podatkov
  - nezaporedna uporaba podatkov
  - datoteke z naključnim dostopom
- Pred uporabo datoteko odpremo
  - ustvarimo nov objekt in z njim povežemo tok
    - klic konstruktorja
- Po uporabi datoteko zapremo
  - sprostimo vire, ki jih zaseda

Vhodno izhodni tokovi

- Za zaporeden dostop do podatkov
  - tok je zaporedje podatkov
  - deluje kot cevovod ali kanal do V/I naprav
    - datoteke, naprave, drugi programi, omrežna vtičnica, ...

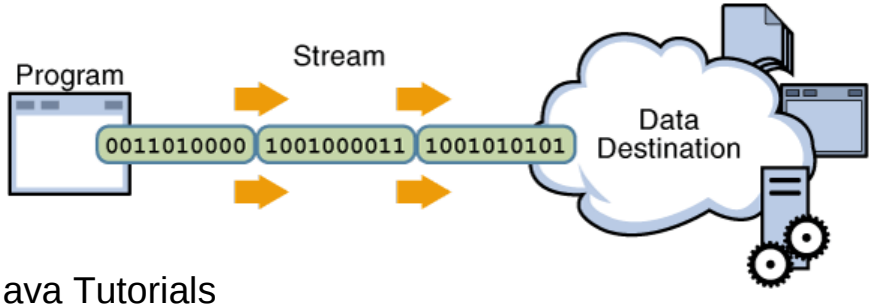
- Vhodne operacije
  - vhodna naprava --> aplikacija



Vhodni tok:
bere iz vira, en element naenkrat

- Izhodne operacije
  - aplikacija --> izhodna naprava

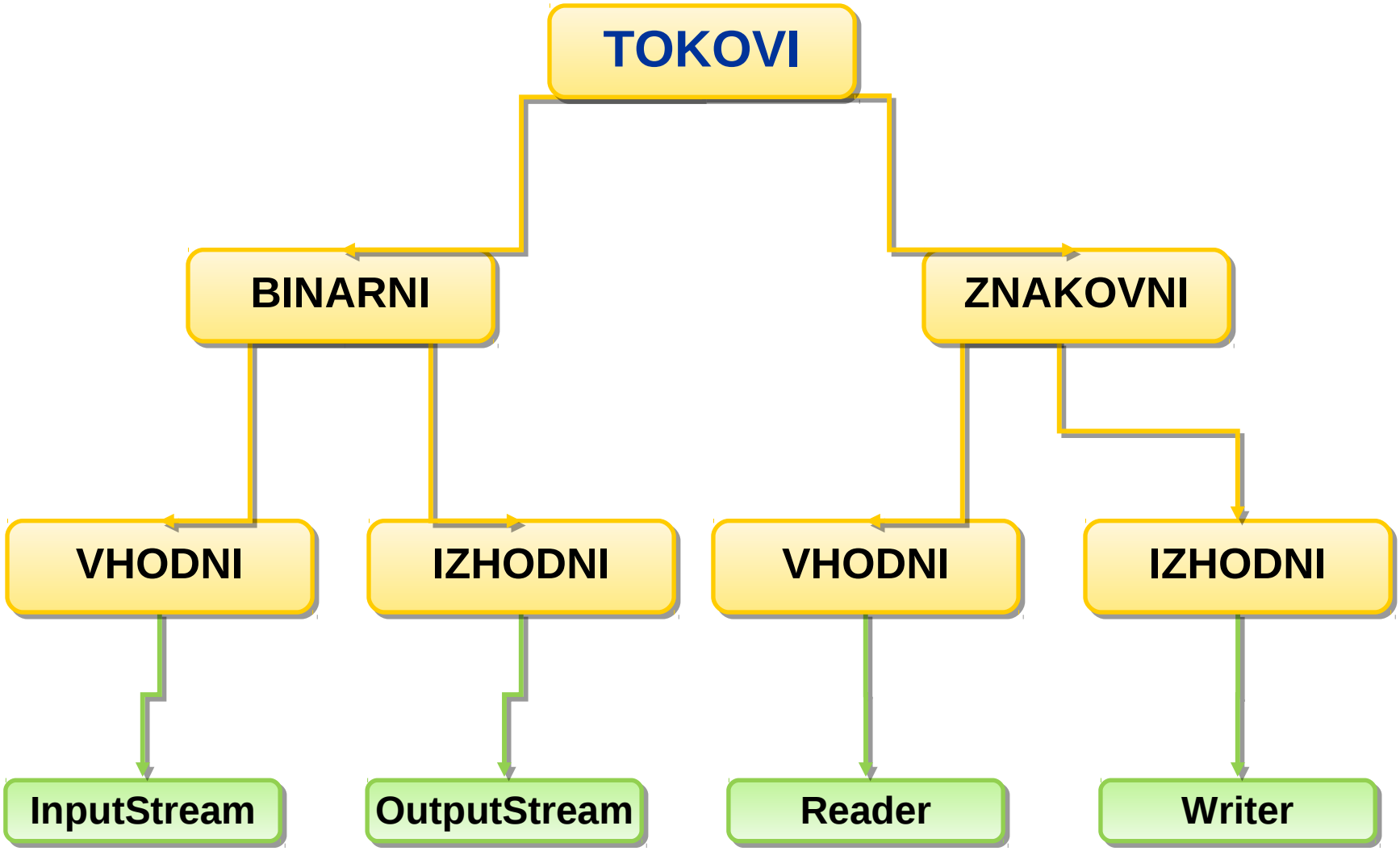
Izhodni tok:
zapisuje na ponor, en element naenkrat



vir slik: The Java Tutorials

Tokovi (streams)

- Tok je objekt
  - metode omogočajo akcije
    - odpiranje, zapiranje
    - branje, pisanje, izpiranje (*flushing*)
    - za različne vrste podatkov
      - bajti, primitivni podatkovni tipi, lokalizirani znaki, objekti
- Večina tokov teče le v eno smer
  - ali vhodni ali izhodni
- Aplikacija ima lahko hkrati odprtih več tokov
- Datoteke z naključnim dostopom
  - tokovi tečejo v obe smeri
    - razred implementira vmesnika za vhodni in izhodni tok


TOKOVI
BINARNI
ZNAKOVNI
VHODNI
IZHODNI
VHODNI
IZHODNI
InputStream
OutputStream
Reader
Writer

Binarni tokovi

- Temeljni tokovi
  - vsi ostali tipi tokov temeljijo na njih
- Prenašajo se bajti
  - skladno z vsebino tipa byte
  - 8 bitov
- Osnovna razreda InputStream, OutputStream
  - za datoteke: FileInputStream, FileOutputStream

Primer: Kopiranje datoteke

```
import java.io.*;

public class Kopiraj {

  public static void main(String[] args) throws IOException
{
    if(args.length < 2) {
      System.out.println("Uporaba: java Kopiraj <izvor>
<ponor>");
      System.exit(1);
    }
    InputStream vhod = new FileInputStream(args[0]);
    OutputStream izhod = new FileOutputStream(args[1]);
    int bajt;
    while( (bajt = vhod.read()) != -1 )
      izhod.write(bajt);
    vhod.close();
    izhod.close();
  }
}
```

DEMO

Vhodno izhodne izjeme

- Razred IOException
  - preverjene izjeme
- Izjeme se lahko sprožijo
  - branje iz toka, pisanje v tok
  - odpiranje (klic konstruktorja), zapiranje toka
- Izjemo moramo obravnavati
  - ujamemo (try-catch-finally)
  - posredujemo naprej (throws)

Izjeme

```
public void metoda() throws Izjema {
   ...
}
```

```
public void metoda() {
   ...
   try {
      // klic metode, ki lahko sproži
izjemo
   }
   catch(Izjema e) {
      // ujamemo in obravnavamo izjemo e
   }
   finally {
      // zaključimo
   }
   ...
}
```

Binarni tok (Byte stream)

```
import java.io.FileOutputStream;
import java.io.IOException;

public class CopyBytes {
    public static void main(String[] args) throws IOException {
        FileInputStream in = null;
        FileOutputStream out = null;
        try {
            in = new FileInputStream("xanadu.txt");
            out = new FileOutputStream(“izhod.txt");
            int c;

            while ((c = in.read()) != -1) {
                out.write(c);
            }

        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        }
    }
}
```

Demo

Znakovni tokovi

- Prenašajo se znaki
  - znaki v Javi so shranjeni v Unicode (16-bitni kod)
  - samodejno se prevedejo v lokalni nabor znakov
  - skladno z vsebino tipa char
- Osnovna razreda Reader in Writer
  - za datoteke: FileReader, FileWriter
- Bajt-znak most:
  - razreda InputStreamReader, OutputStreamWriter
  - za pretvorbo bajtov iz toka v znakovni tok
    - znakovni tokovi uporabljajo binarne tokove za branje/pisanje (fizični V/I) ter le spremenijo bajte v znake

Primer 2: Kopiranje besedila

```
import java.io.*;

public class Kopiraj {

  public static void main(String[] args) throws IOException
{
    if(args.length < 2) {
      System.out.println("Uporaba: java Kopiraj <izvor>
<ponor>");
      System.exit(1);
    }
    FileReader vhod = new FileReader(args[0]);
    FileWriter izhod = new FileWriter(args[1]);
    int znak;
    while( (znak = vhod.read()) != -1 )
      izhod.write(znak);
    vhod.close();
    izhod.close();
  }
}
```

Tok znakov (Character stream)

```
import java.io.FileWriter;
import java.io.IOException;

public class CopyCharacters {
    public static void main(String[] args) throws IOException {
        FileReader inputStream = null;
        FileWriter outputStream = null;

        try {
            inputStream = new FileReader("vhod.txt");
            outputStream = new FileWriter("izhod.txt");

            int c;
            while ((c = inputStream.read()) != -1) {
                outputStream.write(c);
            }
        } finally {
            if (inputStream != null) {
                inputStream.close();
            }
            if (outputStream != null) {
                outputStream.close();
            }
        }
    }
}
```

Demo

Branje in izpis cele vrstice

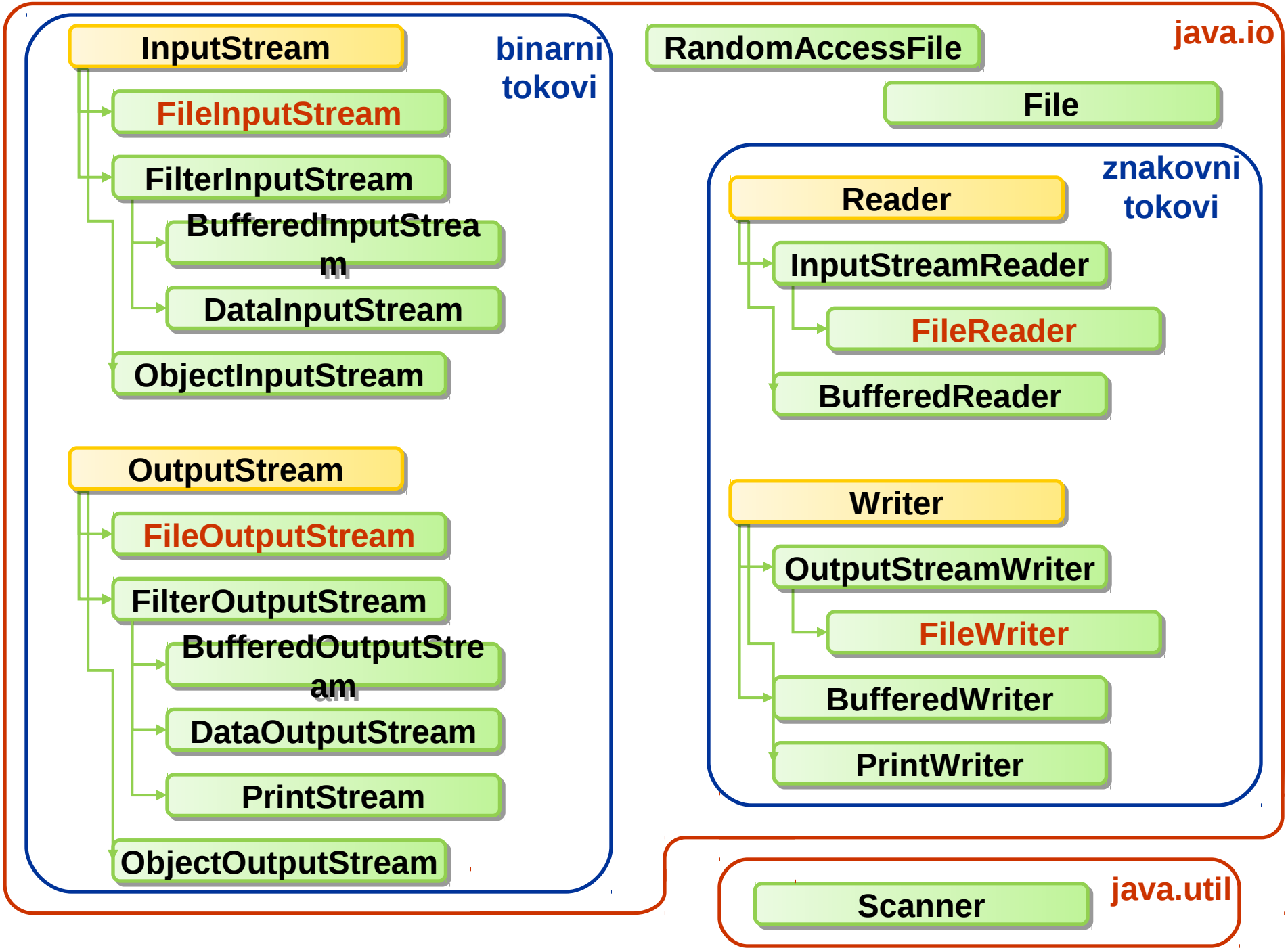
```
public class CopyLines {
    public static void main(String[] args) throws IOException {
        BufferedReader inputStream = null;
        PrintWriter outputStream = null;
        try {
            inputStream =
                new BufferedReader(new FileReader("vhod.txt"));
            outputStream =
                new PrintWriter(new FileWriter("izhod.txt"));

            String l;
            while ((l = inputStream.readLine()) != null) {
                outputStream.println(l);
            }
        } finally {
            if (inputStream != null)          inputStream.close();
                if (outputStream != null)   outputStream.close();
      }
    }
}
```

Demo

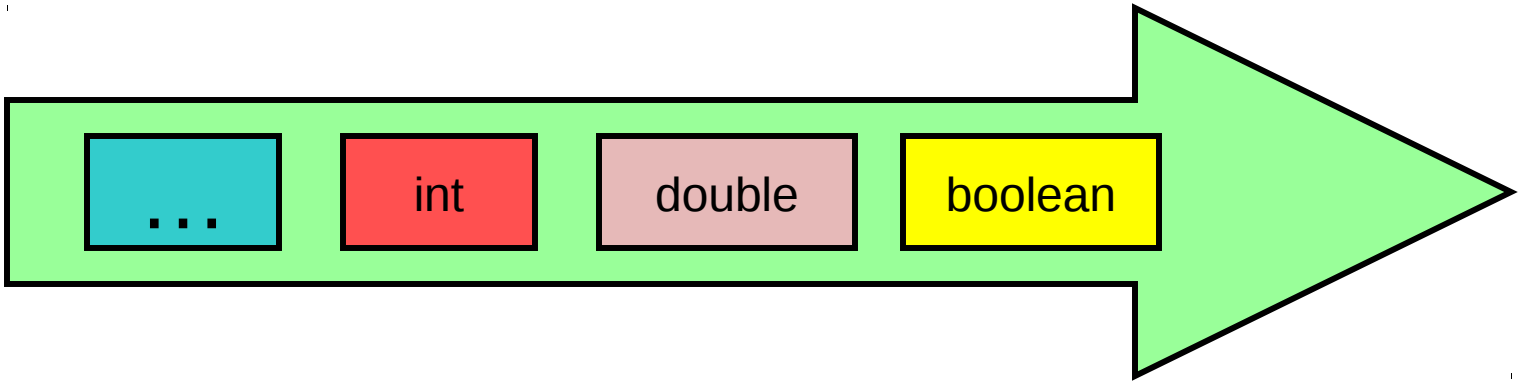
Tokovi

- Za branje/pisanje datotek
  - File...
- Za filtriranje podatkov ob branju/pisanju
  - Filter... in njegovi podrazredi
    - formatirano: Data...
      - podatkov ponavadi ne beremo po bajtih (znakih)
      - imajo tipe: int, double, String, ...
    - preko medpomnilnika: Buffered...
- Za branje/pisanje poljubnih objektov
  - Object...
  - zaporedna predstavitev (serializacija) objektov
    - objekt se razčleni na zaporedje bajtov
    - binarni tok


java.io
binarni tokovi
InputStream
FileInputStream
FilterInputStream
BufferedInputStream
DataInputStream
ObjectInputStream
OutputStream
FileOutputStream
FilterOutputStream
BufferedOutputStream
DataOutputStream
PrintStream
ObjectOutputStream
RandomAccessFile
File
znakovni tokovi
Reader
InputStreamReader
FileReader
BufferedReader
Writer
OutputStreamWriter
FileWriter
BufferedWriter
PrintWriter
java.util
Scanner

Podatkovni tokovi (Data Streams)

- Branje primitivnih tipov iz datoteke
- DataInputStream
- DataOutputStream




```
InputStream basic = new FileInputStream("podatki.dat");
DataInputStream data = new DataInputStream(basic);

int i           = data.readInt();
boolean b = data.readBoolean();
double d  = data.readDouble();
```

Primer s podatkovnimi tokovi

Datoteka z naročilom vsebuje binarno kodirano:

19.99	12	Kumarice
9.99	8	Pivo
28.99	1	Kruh
…		

```
try{
        while (true) {
                price = data.readDouble();
                unit = data.readInt();
                desc = data.readUTF();
                System.out.println("Narocili ste: " + desc);
                total = total + unit * price;
        }
} catch (EOFException e) {
                System.out.println("Skupen znesek : $" + total);
} catch (IOException e) {}
```

Demo

Tokovi objektov (object streams)

Branje in zapisovanje kompleksnih objektov

```
Object ob = new Object();
out.writeObject(ob);
out.writeObject(ob);
```

```
Object ob1 = in.readObject();
Object ob2 = in.readObject();
```

Demo

Ovijanje tokov

- Sestavljanje ali povezovanje tokov
  - dveh ali več tokov
  - obstoječi tok podtaknemo konstruktorju drugega toka
  - zanimajo nas metode zadnjega toka v verigi
- Branje preko medpomnilnika (*buffer*)

```
FileReader tok = new FileReader("dat.txt");
BufferedReader medpom = new BufferedReader(tok);
String vrstica = medpom.readLine();
```

- Formatirano branje/pisanje

```
FileInputStream tok = new FileInputStream("stevila.dat");
BufferedInputStream medpom = new BufferedInputStream(tok);
DataInputStream podatki = new DataInputStream(medpom);
int stevilo = podatki.readInt();
```

Standardni tokovi

- Standardni izhod (System.out)
  - izhod na sistemsko standardno izhodno napravo
    - privzeto zaslon
  - razred PrintStream
- Standardni vhod (System.in)
  - vhod s sistemske standardne vhodne naprave
    - privzeto tipkovnica
  - razred BufferedInputStream
    - preko medpomnilnika; uporaba tipk Enter, Backspace
- Standardni izhod za napake (System.err)

Primer: Izpis tekstovne datoteke na zaslon po vrsticah

```
...

BufferedReader vhod =
  new BufferedReader(new FileReader(new File(args[0])));

String vrstica;
while( (vrstica = vhod.readLine()) != null )
  System.out.println(vrstica);
vhod.close();

...
```

Primer 4: Branje standardnega vhoda in zapis v datoteko

```
...

BufferedReader vhod =
     new BufferedReader(new
InputStreamReader(System.in));
BufferedWriter izhod =
     new BufferedWriter(new FileWriter(args[0]));

String vrstica;
while( (vrstica = vhod.readLine()) != null ) {
  izhod.write(vrstica);
  izhod.newLine();
  // izhod.flush();
}
izhod.close();

...
```

Vsako vrstico moramo zaključiti z Enter, vnos pa zaključimo s Ctrl+Z (EOF).

Program za vajo (doma)

- Napiši program Echo, ki bere standardni vhod (s tipkovnice) in prebrano zapisuje na standardni izhod (na zaslon)

```
java Echo
```

- Program Echo dopolni tako, da bo njegovo delovanje odvisno od podanih argumentov:
  - kot argument podano ime datoteke
    - program bere iz te datoteke
  - kot argument podano -o in ime datoteke
    - program piše v to datoteko

```
java Echo ime.txt
java Echo -o ime.txt
java Echo ime1.txt -o ime2.txt
```

Formatiran vhod/izhod

- Formatirano branje/pisanje podatkov
  - DataInputStream, DataOutputStream
- Kaj uporabiti v praksi?
  - enostavna uporaba
  - prilagodljivost, moč
- Tekstovne datoteke
  - PrintWriter (java.io)
  - Scanner (java.util)

Razred java.io.PrintWriter

- Formatirano predstavitev objekta zapiše v tekstovni izhodni tok
- Metode za izpis
  - implementira vse metode razreda PrintStream
    - print, println, printf, ...
    - poznamo od System.out
- Konstruktorji z različnimi argumenti
  - String (ime datoteke), File
  - OutputStream, Writer
  - vključimo lahko tudi samodejno izpiranje

Razred java.util.Scanner

- Java 1.5+
- Preprost bralnik teksta
  - besedilo razčleni
  - izlušči primitivne podatkovne tipe in nize znakov
  - s pomočjo regularnih izrazov
- Vhod razbije na elemente
  - z uporabo ločitvenega vzorca
  - privzeto prazen prostor (*whitespace*)
- Ob branju elemente pretvori v različne tipe
  - različne metode za branje

Konstruktorji razreda Scanner

- Konstruktor
  - podan argument je vir podatkov
    - File, InputStream
    - String (to ni ime datoteke!)
      - bere iz podanega niza
- Na koncu moramo bralnik zapreti
  - close()

Metode razreda Scanner

- Ali je še kakšen element na vhodu
  - hasNext()
  - hasNextInt(), hasNextDouble(), hasNextLine(), ...
- Preberi naslednji element
  - next()
  - nextInt(), nextDouble(), nextLine(), ...
- Sprememba ločila med elementi
  - useDelimiter(String vzorec)
  - vzorec določa ločilo (regularni izraz)
    - ":", "\\s", "\\n", "\\t", "[^\\w]+", "a|b", ...

Skeniranje vhoda

```
import java.io.*;
import java.util.Scanner;

public class ScanInput {
    public static void main(String[] args) throws IOException {
        Scanner s = null;
        try {
            s = new Scanner(new BufferedReader(new FileReader("vhod.txt")));

            while (s.hasNext()) {
                System.out.println(s.next());
            }
        } finally {
            if (s != null) {
                s.close();
            }
        }
    }
}
```

Demo

Skeniranje številčnih podatkov

```
import java.io.FileReader;
import java.io.BufferedReader;
import java.io.IOException;
import java.util.Scanner;
import java.util.Locale;

public class ScanStevila {
    public static void main(String[] args) throws IOException {
        Scanner s = null;
        double sum = 0;
        try {
            s = new Scanner(   new BufferedReader(new FileReader("stevila.txt")));
            s.useLocale(Locale.US);

            while (s.hasNext()) {
                if (s.hasNextDouble()) {
                        sum += s.nextDouble();
                    } else { s.next();  }
            }
        } finally {  s.close();  }
        System.out.println(sum);
    } }
```

Demo

Izpis spremenljivk na zaslon

```
public class KvadratniKoren {
    public static void main(String[] args) {
        int i = 2;
        double r = Math.sqrt(i);

        System.out.print("Kvadratni koren od  ");
        System.out.print(i);
        System.out.print(" je ");
        System.out.print(r);
        System.out.println(".");

        i = 5;
        r = Math.sqrt(i);
        System.out.println("Kvadratni koren od  " + i + " je " + r + ".");
    }
}
```

Demo

Formatirani izpis

```
public class KvadratniKoren2 {
    public static void main(String[] args) {
        int i = 2;
        double r = Math.sqrt(i);

        System.out.format("Kvadratni kored od  %d je %f.%n", i, r);
    }
}
```

Demo

Java in printf

Metoda printf( ) avtomatsko uporablja Formatter za tvorbo formatiranega niza. Metoda printf( ) je definirana za PrintStream in PrintWriter.

Za PrintStream, printf( ) ima naslednji obliki:

```
PrintStream printf(String fmtString, Object ... args)
PrintStream printf(Local loc, String fmtString, Object ... args)
```

Prva oblika piše argumente na standardni izhod v formatu, ki ga definira fmtString, in pri tem uporablja privzeti (default) locale. Druga oblika omogoča specificiranje lokala.

Demo

Primer: Branje podatkov o študentih in zapis v drugo datoteko

- Datoteka studenti.txt ima podatke o študentih
  - vsak študent v svoji vrstici
  - posamezni podatki ločeni s podpičjem

```
63070279;Ime;Priimek;10
63070264;Ana Marija;Ficko;10
63060043;Janez;Novak Erhar;9
63060053;Mojca;Pokraculja;8
63070284;Zelo Slabo;Znanje de Neznanje;5
63060047;Dobra;Vila;6
63070223;Bedanc;Grozni;7
63070260;Kekec;Samsvoj;9
63070281;Franci;Nekdo;5
63060561;Mirko;Stegnar;10
63060058;Janko;Novak;5
```

Kombiniran primer (Scanner in PrintWriter)

```
import java.util.*;
...

  Scanner bralnik = new Scanner(new File(args[0]));
  PrintWriter izhod = new PrintWriter(args[1]);
  bralnik.useDelimiter(";|\\r\\n"); // za Windows okolje!
  double vsota = 0.0;  int skupaj = 0;
  while( bralnik.hasNext() ) {
    String vpisna = bralnik.next();
    bralnik.next(); // ime ignoriramo
    bralnik.next(); // priimek ignoriramo
    int ocena = bralnik.nextInt();
    vsota += ocena;  skupaj++;
    izhod.printf("%s %2d%n", vpisna, ocena);
  }
  bralnik.close();
  izhod.close();
  System.out.printf("Skupaj je pisalo %d studentov,
povprecna
                      ocena je %.2f%n", skupaj,
vsota/skupaj);
```

DEMO