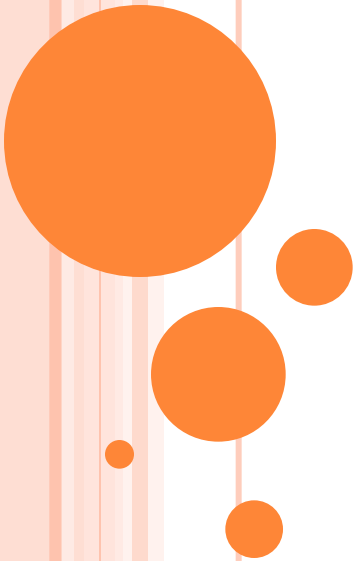
Algoritmi in podatkovne strukture I



Vhod/izhod

Vhod/izhod

- Tokovi (streams) so predmetna predstavitev vhodno/izhodnih operacij.
- Omogočajo branje ter zapisovanje podatkov skozi zaporedje znakov ali bajtov.
  - standardni vhod in izhod, datoteke, omrežna vtičnica…
- Razrede za delo s tokovi ponuja paket java.io.

Binarni tokovi

- Prenašajo se bajti.
  - Skladno s primitivnim tipom *byte* (8 bitov).
- Osnovna razreda InputStream in OutputStream
  - Za datoteke: FileInputStream in FileOutputStream .
- Pri delu z datotekami je koristna uporaba medpomnilnika, kar omogočata razreda BufferedInputStream in BufferedOutputStream.
- Za branje in pisanje podatkov določenega tipa (npr. int, double,…) sta namenjena DataInputStream in DataOutputStream.

Ovijanje tokov

- Povezovanje tokov
  - Obstoječi tok podtaknemo konstruktorju drugega toka

```
FileInputStream tok = new FileInputStream("podatki.dat");
BufferedInputStream med = new BufferedInputStream(tok);
DataInputStream podatki = new DataInputStream(med);

double val = podatki.readDouble();
```

Primer 1 – kopiranje datotek

```
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();
  }
}
```

Primer 2 – branje formatiranih podatkov

```
//primer: v datoteki je najprej zapisano število podatkov tipa double, nato sledijo sami podatki

DataInputStream dis = null;
try {
    dis = new DataInputStream(
            new BufferedInputStream( new FileInputStream("c:\\test\\data.dat"))
    );

    int count = dis.readInt();        //preberemo število podatkov
    for (int i = 0; i < count; i++)
            System.out.println(dis.readDouble());          //beremo in izpisujemo podatke
}
catch (IOException ex) { System.err.println("I/O exception");}
finally {
        try { if (dis != null) dis.close();}
        catch (IOException ex) { System.err.println("I/O exception");}
}
```

Znakovni tokovi

- Prenašajo se znaki kodirani kot 16-bitni simboli.
  - Skladno s primitivnim tipom *char*.
  - Samodejno prevajanje v lokalni nabor znakov.
- Osnovna razreda Reader in Writer.
  - Za datoteke: FileReader in FileWriter.
- Za formatiran izpis besedila je koristen razred PrintWriter.
  - Pozna metodi print() in println().
- Za branje besedila je koristen razred BufferedReader.
  - Pozna metodo readLine().

Primer 3 – uporaba razreda PrintWriter

```
…
PrintWriter p = new PrintWriter(
    new FileWriter(“c:\\test\\izpis.txt”)
);

p.println(100);
p.println(“primer besedila”);

p.close();
…
```

Primer 4 – branje datoteke po vrsticah

```
…
String vrstica;
BufferedReader tok = new BufferedReader(
    new FileReader(“besedilo.txt”)
);

while  ((vrstica = tok.readLine()) != null) {
    //uporabimo vrstico
}

tok.close();
…
```

Razred Scanner

- Za formatirano branje besedila (Java 1.5+).
  - V paketu java.util.
  - Besedilo razčleni z uporabo ločitvenega vzorca
  - Izlušči primitivne podatkovne tipe in nize znakov
- Ali je kakšen element na vhodu?
  - hasNext()
  - hasNextInt(), hasNextDouble(), hasNextLine(), …
- Preberi naslednji element
  - next()
  - nextInt(), nextDouble(), nextLine(), …
- Sprememba ločila med elementi
  - useDelimiter(String vzorec)

Primer 5 – Branje iz konzole

```
double n = 0;
Scanner in = new Scanner(System.in);
in.useLocale(Locale.US);

while (true) {
        System.out.println("Vpisi realno število:");
        if (in.hasNextDouble()) {
                n = in.nextDouble();
                System.out.println("Vnešeno število je " + n);
                break;
        }
        else {
                in.next();
                System.out.println("Nepravilen vnos");
        }
}
```