Ulazno-izlazni podsistem

Programski jezici II

Tipično korištenje tokova

```
import java.io.*;

public class IOStreamDemo {
  public static void main(String[] args) throws
   IOException {
    // Čitanje liniju po liniju
    BufferedReader in = new BufferedReader(
       new FileReader("IOStreamDemo.java"));
    String s, s2 = new String();
    while((s = in.readLine())!= null)
      s2 += s + "\n";
    in.close();
    // Čitanje sa standardnog ulaza
    BufferedReader stdin = new BufferedReader(
      new InputStreamReader(System.in));
    System.out.print("Enter a line:");
    System.out.println(stdin.readLine());
```

Tipično korištenje tokova

```
// Čitanje iz memorije
   StringReader in2 = new StringReader(s2);
   int c;
   while((c = in2.read()) != -1)
     System.out.print((char)c);

// Čitanje iz memorije - formatirani ulaz
   try {
     DataInputStream in3 = new DataInputStream(
       new ByteArrayInputStream(s2.getBytes()));
     while(true)
       System.out.print((char)in3.readByte());
   } catch(EOFException e) {
     System.err.println("End of stream");
   }
```

Tipično korištenje tokova

```
// 4. Pisanje u datoteku (human-readable)
 in4 = new BufferedReader(
       new StringReader(s2));
     PrintWriter out1 = new PrintWriter(
       new BufferedWriter(new
            FileWriter("IODemo.out")));
     int lineCount = 1;
     while((s = in4.readLine()) != null )
       out1.println(lineCount++ + ": " + s);
     out1.close();
   } catch(EOFException e) {
     System.err.println("End of stream");
   }
```

Tipično korištenje tokova

```
// 5. Smještane i recovery podataka (od strane drugog sistema, nije human-
    readable)
    try {
      DataOutputStream out2 = new DataOutputStream(
        new BufferedOutputStream(
          new FileOutputStream("Data.txt")));
      out2.writeDouble(3.14159);
      out2.writeUTF("That was pi");
      out2.writeDouble(1.41413);
      out2.writeUTF("Square root of 2");
      out2.close();
      DataInputStream in5 = new DataInputStream(
        new BufferedInputStream(
          new FileInputStream("Data.txt")));
      // Must use DataInputStream for data:
      System.out.println(in5.readDouble());
      // Only readUTF() will recover the
      // Java-UTF String properly:
      System.out.println(in5.readUTF());
      // Read the following double and String:
      System.out.println(in5.readDouble());
      System.out.println(in5.readUTF());
    } catch(EOFException e) {
      throw new RuntimeException(e);
    }
```

Tipično korištenje tokova

```
// 6. Čitanje i pisanje u RAF
RandomAccessFile rf =
      new RandomAccessFile("rtest.dat", "rw");
    for(int i = 0; i < 10; i++)
      rf.writeDouble(i*1.414);
    rf.close();
    rf = new RandomAccessFile("rtest.dat", "rw");
    rf.seek(5*8);
    rf.writeDouble(47.0001);
    rf.close();
    rf = new RandomAccessFile("rtest.dat", "r");
    for(int i = 0; i < 10; i++)
      System.out.println("Value " + i + ": " +
        rf.readDouble());
    rf.close();
  }
}
```

Tipično korištenje tokova

```
// čitanje/pisanje sa/na konzolu
import java.io.*;

public class StandardIOReadingWriting {
  public static void main(String[] args)
  throws IOException {
    BufferedReader in = new BufferedReader(new
   InputStreamReader(System.in));
    String s;
    while((s = in.readLine()) != null &&
   s.length() != 0)
      System.out.println(s);
  }
}
```

Tipično korištenje tokova

```
// redirekcija standardnog ulaza i izlaza
import java.io.*;

public class Redirecting {
  public static void main(String[] args) throws IOException {
    PrintStream console = System.out;
    BufferedInputStream in = new BufferedInputStream(
          new FileInputStream("Redirecting.java"));
    PrintStream out = new PrintStream(new BufferedOutputStream(
          new FileOutputStream("test.out")));
    System.setIn(in);
    System.setOut(out);
    System.setErr(out);
    BufferedReader br = new BufferedReader(
          new InputStreamReader(System.in));
    String s;
    while((s = br.readLine()) != null)
      System.out.println(s);
    out.close();
    System.setOut(console);
  }
}
```

Kompresija/dekompresija

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

public class ZipTest {
  public ZipTest() {
    byte[] buffer = new byte [BUFFER_LENGTH];
    try {
      // Kreiramo arhivu "test.zip"
      ZipOutputStream out =
        new ZipOutputStream(
          new FileOutputStream("test.zip"));
      // Pripremimo se za citanje datoteke koju cemo zapakovati.
      BufferedInputStream fin =
        new BufferedInputStream(
          new FileInputStream("ZipTest.java"));
      // Ubacivanje datoteke u arhivu pocinje metodom putNextEntry().
      out.putNextEntry(new ZipEntry("ZipTest.java"));
      // Poslije toga moramo da datoteku procitamo i smjestimo u
    arhivu.
      int read;
      while ((read = fin.read(buffer, 0, BUFFER_LENGTH)) != -1) {
        out.write(buffer, 0, read);
      }
      out.close();
```

Kompresija/dekompresija

```
// Sada cemo da otvorimo arhivu i procitamo iz nje datoteku "ZipTest.java"
      ZipInputStream in = new ZipInputStream(new FileInputStream("test.zip"));
      ZipEntry zipEntry;
      // Prolazimo kroz sve datoteke u arhivi.
      while ((zipEntry = in.getNextEntry()) != null) {
        System.out.println("Extracting file: " + zipEntry.getName());
        int total = 0;
        byte[] accumulator = new byte[MAX_FILE_LENGTH];
        while ((read = in.read(buffer, 0, BUFFER_LENGTH)) != -1) {
          for (int i = 0; i < read; i++)
             accumulator[total+i] = buffer[i];
          total += read;
        }
        // U nizu bajtova "accumulator" nalazi se raspakovana tekuca datoteka.
        String fileText = new String(accumulator, 0, total);
        System.out.println(fileText);
      }
      in.close();
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }

  public static void main(String[] args) {
    ZipTest zp = new ZipTest();
  }

  private static final int BUFFER_LENGTH = 1024;
  private static final int MAX_FILE_LENGTH = 65536;
}
```