Структури от Данни и Обектно-Ориентирано Програмиране
спец. Компютърни Науки
Упражнение №5
25.03.2010г.

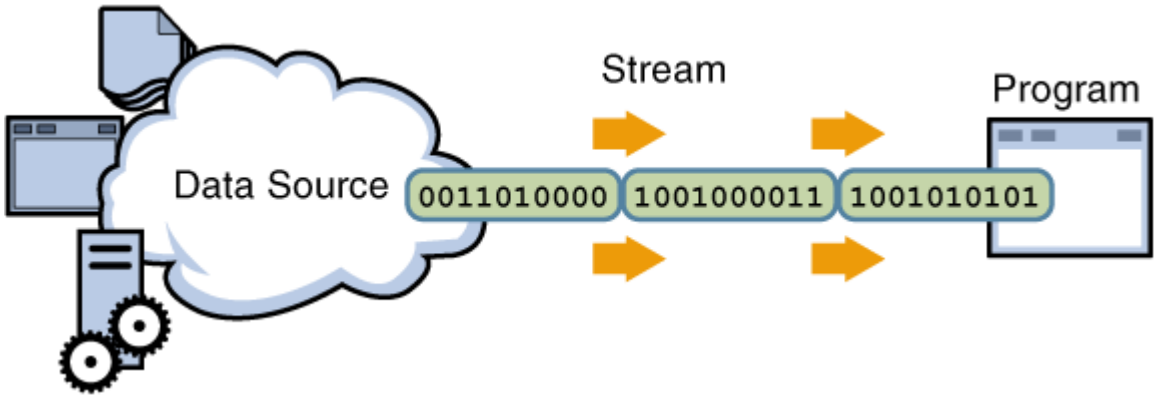
ТЕМА: Файлове

Файл – именувана съвкупност от данни, която се съхранява на външен носител.
Файлова система – организация на съвкупността от файлове с цел да се борави с тях. Всички операции на програмата над файлове минават през ОС.

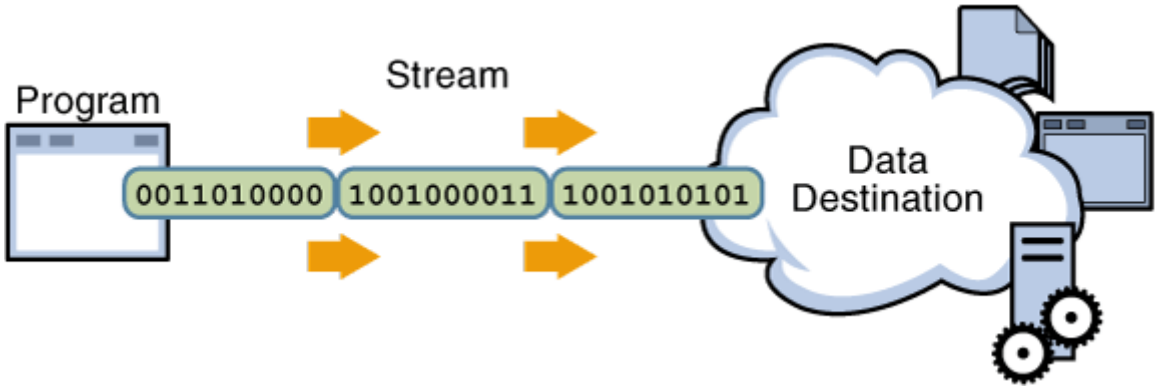
Класът ***File*** предоставя обекти, които се асоциират със съществуващи или новосъздавани файлове, за да се използват възможностите на ОС за тяхното управление.

Потоци – абстракция, произтичаща от входно/изходните устройства с последователен достъп.
Входен поток – източник на данни, които постъпват в програмата.
Изходен поток – контейнер, в който програмата изпраща данни.
Този източник/контейнер може да е от различен характер: дисков файл, устройство, масив в паметта или друга програма.
Данните, които се четат или записват в потоците могат да са различни: байтове, примитивни типове или обекти. Те могат просто да бъдат пренесени от потока, или междувременно да претърпят някаква обработка.

Входният поток е източник, от който програмата получава данните една по една.



Изходният поток е контейнер, в който програмата записва данни една по една.



Видове потоци

- байтови
- символни

Байтови потоци

Байтовите потоци служат за запис и четене на данни в двоичен вид.

В основата на двете йерархии са абстрактните класове ***OutputStream*** и ***InputStream***, чиито основни методи са съответно за записване на един байт в изходен поток или за четене на един байт от входен поток. Данни, чийто размер е повече от един байт, се четат или записват като последователности от съответния брой байтове.
Този вид потоци са най-примитивни и те са в основата на реализацията на останалите видове.

Класовете ***FileOutputStream*** и ***FileInputStream*** представят байтови потоци за запис и четене от дискови файлове.

Класовете ***FilterOutputStream*** и ***FilterInputStream*** са базови за класовете, които обвиват произволен изходен или входен поток съответно, и променят или допълват неговата функционалност.

Класовете ***BufferedOutputStream*** и ***BufferedrInputStream*** увеличават производителността като буферират изхода или входа съответно така, че да не се извършва обръщение към ОС за запис или четене на всеки байт.

Класовете ***DataOutputStream*** и ***DataInputStream*** разширяват функционалността като предоставят методи за запис в изходния поток или четене от входния поток на примитивни типове данни.

Символни потоци

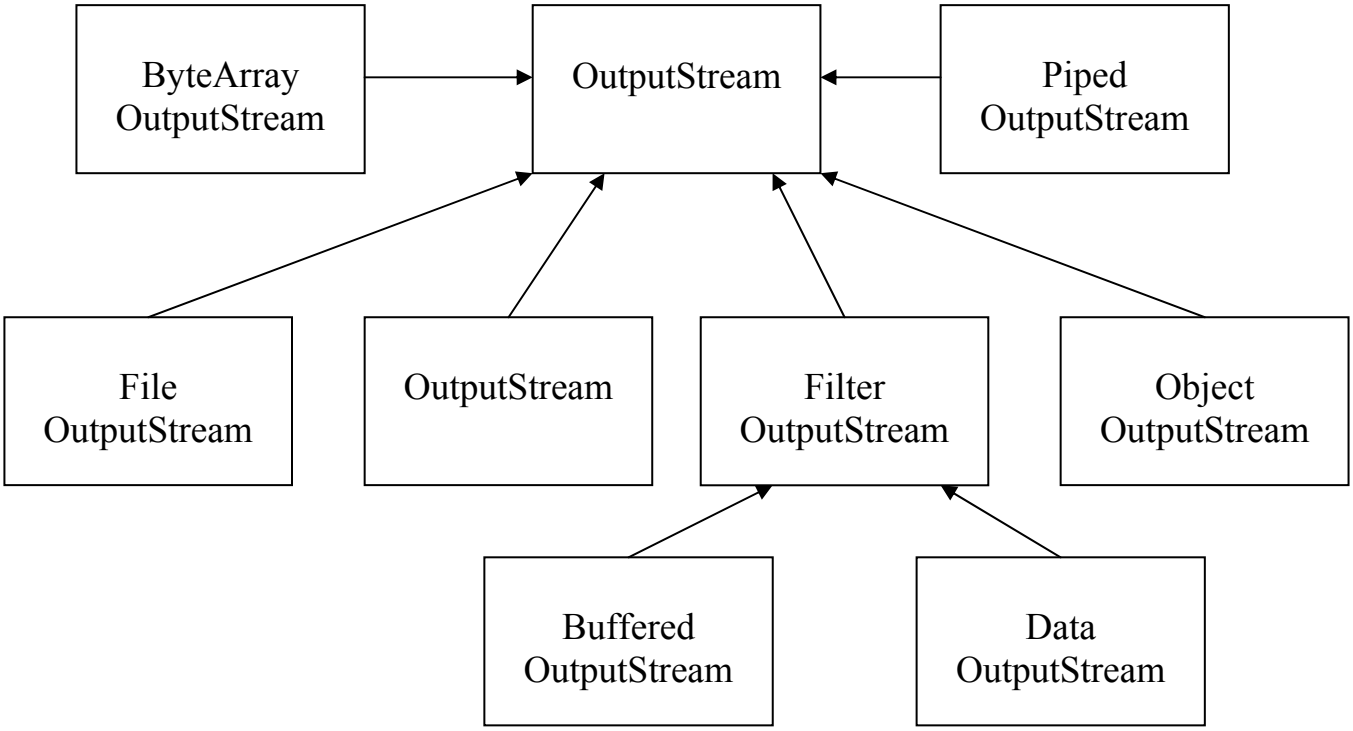
Символните потоци служат за запис и четене на текст (последователност от символи). По премълчаване се подразбира стандартната кодировка на средата (операционната система), но е възможно и изрично указване на външната кодировка. Основните класове, реализиращи символни потоци са показани по-долу.

В корените на йерархиите са абстрактните класове ***Writer***, чийто основен метод *write* извежда като един символ младшите два байта на аргумента си тип *int* и ***Reader*** с основен метод *read*, който въвежда един прочетен символ в младшите два байта на връщаната стойност тип *int*.

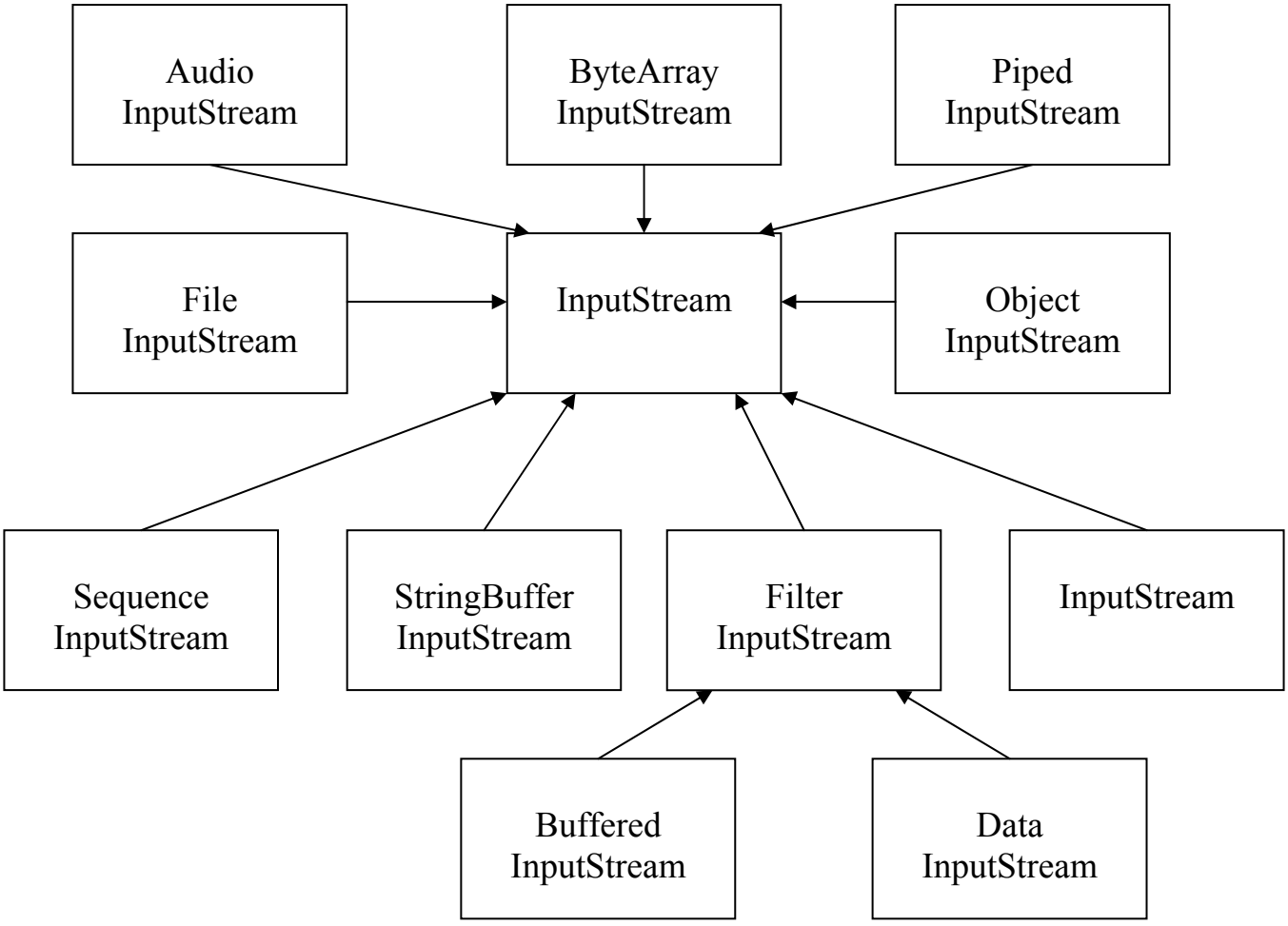
Класът ***InputStreamReader*** служи като мост между байтови и символни потоци. Всяко четене от байтовия поток води до въвеждане на един или два байта и превръщане в символ съгласно дадена кодова таблица. Използва кодировка по премълчаване или явно зададена.
Класът ***OutputStreamWriter*** служи като мост между символни и байтови потоци. Символите, записани в него се кодират в байтове съгласно дадена кодова таблица.

Класовете от пакета java.io, които поддържат байтови потоци

http://java.sun.com/javase/6/docs/api/java/io/OutputStream.html

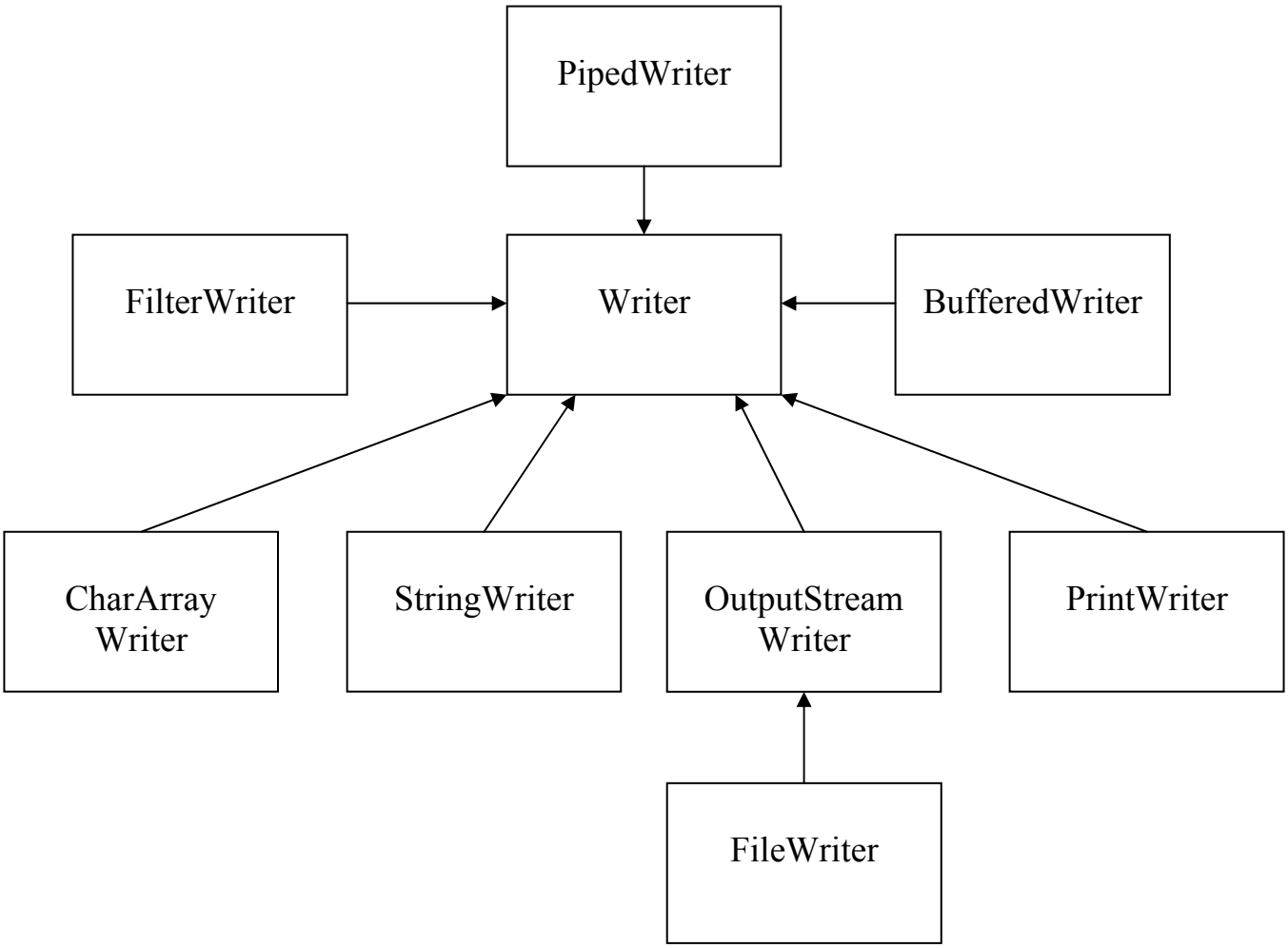


http://java.sun.com/javase/6/docs/api/java/io/InputStream.html

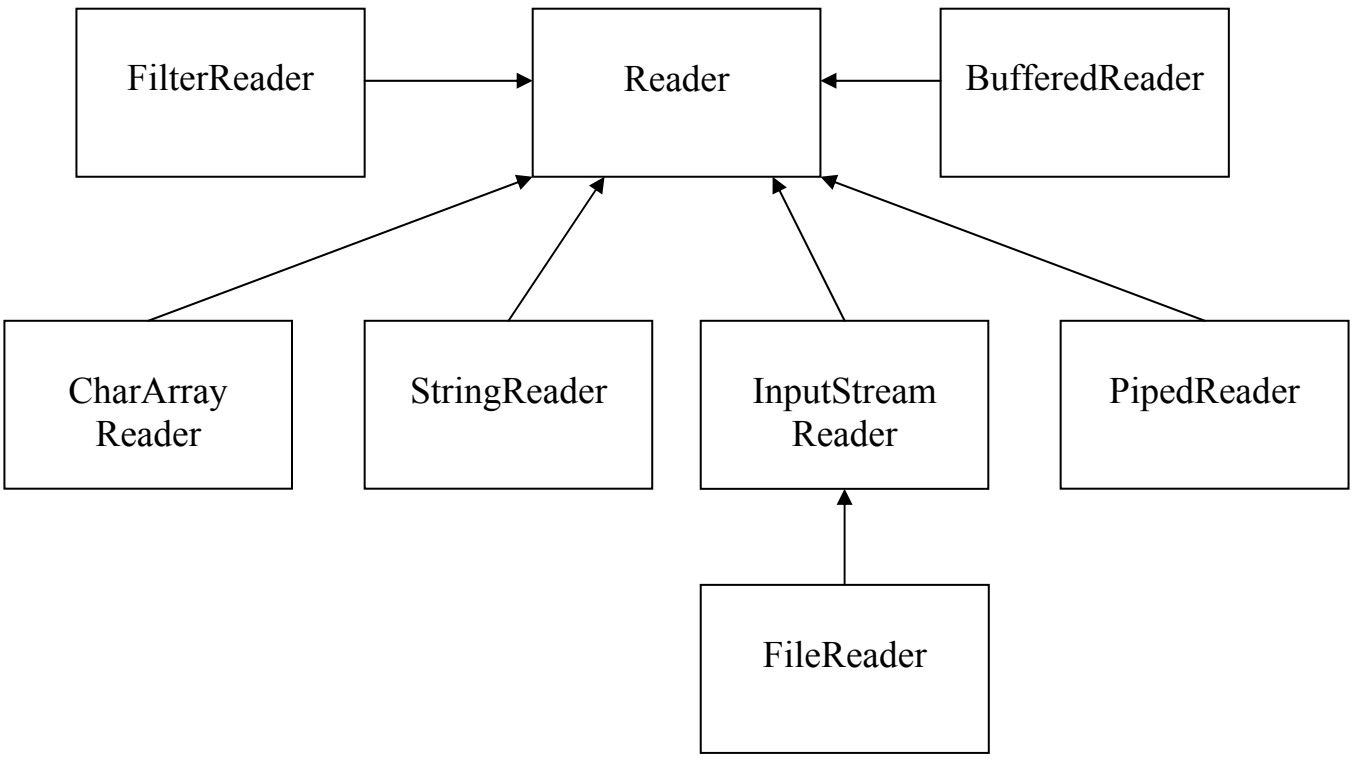


Класовете от пакета java.io, които поддържат символни потоци

http://java.sun.com/javase/6/docs/api/java/io/Writer.html



http://java.sun.com/javase/6/docs/api/java/io/Reader.html



Пример 1:

```
import java.io.*;

public class ByteStreamDemo {

    static void copyFile(String from, String to) throws IOException   {
        InputStream  in  = null;
        OutputStream out = null;
        try {
            in  = new FileInputStream (from);
            out = new FileOutputStream(to);
            int c;
            while((c = in.read())!=-1) {
                out.write(c);
            }
        }
        finally {
            if(in != null)
                in.close();
            if(out != null)
                out.close();
        }
    }

    static boolean testByteDataIO() throws IOException   {
        InputStream  in  = null;
        OutputStream out = null;
        int    i  = 1234567890;
        int    j  = 0;
        double d  = 12345.6789;
        double e  = 0.0;
        boolean result = false;
        try {
            out = new FileOutputStream("data1.txt");
            out.write(i>>24);
            out.write(i>>16);
            out.write(i>> 8);
            out.write(i);
            long ld = Double.doubleToLongBits(d);
            out.write((int)(ld>>56));
            out.write((int)(ld>>48));
            out.write((int)(ld>>40));
            out.write((int)(ld>>32));
            out.write((int)(ld>>24));
            out.write((int)(ld>>16));
            out.write((int)(ld>> 8));
            out.write((int) ld&0xff);
```

```
            out.flush();
            in = new FileInputStream("data1.txt");
            j  = in.read()<<24 |
            in.read()<<16 |
            in.read()<< 8 |
            in.read();
            ld = (long)in.read()<<56 |
            (long)in.read()<<48 |
            (long)in.read()<<40 |
            (long)in.read()<<32 |
            (long)in.read()<<24 |
            (long)in.read()<<16 |
            (long)in.read()<< 8 |
            (long)in.read();
            e = Double.longBitsToDouble(ld);
        }
        finally {
            if(in != null)
                in.close();
            if(out != null)
                out.close();
        }

        if((i == j) && (d == e))
            result = true;
        return result;
}

static boolean testDataIO() throws IOException   {
        DataOutputStream out = null;
        DataInputStream  in  = null;
        int    i  = 1234567890;
        int    j  = 0;
        double d  = 12345.6789;
        double e  = 0.0;
        boolean result = false;
        try {
            out = new DataOutputStream(
                        new FileOutputStream("data2.txt")  );
            out.writeInt(i);
            out.writeDouble(d);
            out.flush();
            in = new DataInputStream(
                        new FileInputStream("data2.txt")   );
            j = in.readInt();
            e = in.readDouble();
        }
        finally {
```

```
                if(in != null)
                        in.close();
                if(out != null)
                        out.close();
        }
        if((i == j) && (d == e))
                result = true;
        return result;
    }

    public static void main(String[] args) throws IOException {
        copyFile("input.txt", "input_copy.txt");
        System.out.println(testByteDataIO());
        System.out.println(testDataIO());
    }

}
```

Пример 2:
Да се прочете текст от клавиатурата и да се запише във файл.

```
import java.util.Scanner;
import java.io.*;
public class CopyInput {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String fileName = "input.txt";
        PrintWriter pr = null;
        try {
            pr = new PrintWriter(fileName);
            String str;
            while((sc.hasNextLine())) {
                str = sc.nextLine();
                pr.println(str);
            }
            pr.flush();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally{
            if(pr != null)
                pr.close();
        }
    }
}
```

Пример 3:
Да се изведе списък на файловете в дадена директория

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

/**
 * This program lists the files in a directory specified by the user.
 * If the name entered by the user is not a directory,
 * a message is printed and the program ends.
 */
public class DirectoryList {

  public static void main(String[] args) {

    String directoryName;  // Directory name entered by the user.
    File directory;        // File object referring to the directory.
    String[] files;        // Array of file names in the directory.
    Scanner sc;       // For reading a line of input from the user.

    sc = new Scanner(System.in);  // scanner reads from standard input.

    System.out.print("Enter a directory name: ");
    directoryName = sc.nextLine().trim();
    directory = new File(directoryName);

    if (!directory.isDirectory()) {
      if (!directory.exists())
        System.out.println("There is no such directory!");
      else
        System.out.println("That file is not a directory.");
    }
    else {
      files = directory.list();
      System.out.println("Files in directory \"" + directory + "\":");
      for (int i = 0; i < files.length; i++)
        System.out.println("   " + files[i]);
    }
  } // end main()
} // end class DirectoryList
```

Пример 4: Да се конкатенират всички файлове, чиито имена се въвеждат и резултатът да се запише във файл с указано име.

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

public class Concatenate {
   public static void main(String[] args) throws IOException {
        Vector<String> listOfFileNames = new Vector<String>();
        String nextFileName;
        Scanner stdin = new Scanner(System.in);
        BufferedWriter outFile;

        nextFileName = stdin.next();
       outFile = new BufferedWriter(new PrintWriter(nextFileName));

        while(stdin.hasNext()) {
                nextFileName = stdin.next();
                listOfFileNames.add(nextFileName);
        }

        String [] fileNames = new String[listOfFileNames.size()];
        listOfFileNames.toArray(fileNames);

        ListOfFiles listOfFiles = new ListOfFiles(fileNames);

       SequenceInputStream s = new SequenceInputStream(listOfFiles);
       InputStreamReader r = new InputStreamReader(s);

       int c;
       while ((c = r.read()) != -1)
          outFile.write(c);

       r.close();
       outFile.flush();
       outFile.close();
   }
}


import java.util.*;
import java.io.*;

public class ListOfFiles implements Enumeration<FileInputStream> {

   private String[] listOfFiles;
   private int current = 0;
```

```
    public ListOfFiles(String[] listOfFiles) {
        this.listOfFiles = listOfFiles;
    }

    public boolean hasMoreElements() {
         return current < listOfFiles.length;
    }

    public FileInputStream nextElement() {
        FileInputStream in = null;

        if (!hasMoreElements())
            throw new NoSuchElementException("No more files.");
        else {
            String nextElement = listOfFiles[current];
            current++;
            try {
                in = new FileInputStream(nextElement);
            } catch (FileNotFoundException e) {
                System.err.println("ListOfFiles: Can't open " + nextElement);
            }
        }
        return in;
    }
}
```

Пример 5: Манипулиране на файлове

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

public class UseFiles {
        /*      How to copy Files */

        // Copy file as character Stream
        public static void copyCharacters (String fromFile, String toFile)
                                                throws IOException {
                File inputFile = new File(fromFile);
                File outputFile = new File(toFile);
                FileReader in = new FileReader(inputFile);
                FileWriter out = new FileWriter(outputFile);
                int c;
```

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

// Copy file as byte Stream
public static void copyBytes (String fromFile,String toFile) throws IOException {
        File inputFile = new File(fromFile);
        File outputFile = new File(toFile);
        FileInputStream  in = new FileInputStream(inputFile);
        FileOutputStream out = new FileOutputStream(outputFile);
        int c;
        while (( c = in.read()) != -1)
                out.write(c);
        in.close();
        out.close();
}

// Copy text file
public static void copyText (String fromFile, String toFile) throws IOException {
        File inputFile = new File(fromFile);
        File outputFile = new File(toFile);
        BufferedReader in = new BufferedReader(new FileReader(inputFile));
        PrintWriter out = new PrintWriter(new BufferedWriter
                                        (new FileWriter(outputFile)));
        String c;
        while ((c = in.readLine()) != null)
                out.println(c);
        in.close();
        out.close();
}

public static void main(String[] args) {
        try {
                String s = File.separator;
                String d1 = "C:"+s+"examples"+s+"students"+s+"source_files";
                File sourceDir = new File(d1);

                String d2 = "C:"+s+"examples"+s+"students"+s+"target_files";
                File targetDir = new File(d2);

                // Copy file in sourceDir
                String source =
                   "C:"+s+"examples"+s+"students"+s+"source_files"+s+"proba1.txt";
                String target =
                   "C:"+s+"examples"+s+"students"+s+"source_files"+s+"proba3.txt";
                copyBytes(source,target);
```

```
                    // List of files in sourceDir
                    File[] files = sourceDir.listFiles();
                    String[] fileNames = sourceDir.list();

                    //Moving files
                    for (int i = 0;i < files.length;i++)
                      files[i].renameTo(new File(targetDir,files[i].getName()));

                  }
            catch (IOException e) {
                    System.err.println(e.toString());
            }
      }
}
```

Пример 6: Да се прочете текстов файл и да се намери броя на символите, бланковете, думите и редовете в него.

```
import java.io.*;

public class WordCount {
      public static void main(String[] args) {
            String fileName = "input.txt";
             if (args.length == 1)
            fileName = args[0];
            try {
                  //FileReader fr = new FileReader(fileName);
                  BufferedReader br = new BufferedReader(new FileReader(fileName));
                  wordCount(br);
            }
            catch (IOException e) {
                  e.printStackTrace();
            }
      }

      static void wordCount(BufferedReader in) {
            int charCount = 0, spaceCount = 0;
            int lineCount = 0, wordCount = 0;
            try {
                  String str;
                  while((str = in.readLine()) != null) {
                        lineCount++;
                        int index = 0;
                        while(index < str.length()) {
                              while(index < str.length() &&
                                    Character.isWhitespace(str.charAt(index))) {
```

```
                        spaceCount++;
                        index++;
                    }
                if(index < str.length())
                    wordCount++;
                while(index < str.length() &&
                        !Character.isWhitespace(str.charAt(index))) {
                    index++;
                    charCount++;
                }
            } // while for the line
        } // for the file
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    System.out.println("Chars = " + charCount);
    System.out.println("Words = " + wordCount);
    System.out.println("Spaces = " + spaceCount);
    System.out.println("Lines = " + lineCount);
  }
}
```

Задача 1: Да се прочете текстов файл и да се сумират всички числа, които се съдържат в текста.

Задача 2: Да се прочете текстов файл и да се кодира, като всеки символ бъде заменен с нов символ, като разликата в кодовете на оригиналния и новия символ е константа.

Задача: Да се прочете текстов файл и да се направи списък на всички различни думи в текста заедно с броя на срещанията на всяка от тях. Списъкът от думите, сортирани лексикографично, да се запази във файл.