Įvedimas/išvedimas

Lekt. dr. Pijus Kasparaitis
pkasparaitis@yahoo.com

2008-2009 m. m. pavasario semestras

Duomenų šaltiniai

- Failai
- Konsolė
- Tinklas
- ...

Įvedimo/išvedimo būdai

- Nuoseklus (angl. sequential)
- Pasirinktinis (angl. random-access)
- Buferizuotas
- Baitais
- Simboliais
- Eilutėmis
- ...

Klasė File

```
1 import java.io.*;

2 public class DirList {
3   public static void main(String[] args) {
4     File path = new File(".");
5     String[] list = path.list();
6     for(int i = 0; i < list.length; i++)
7       System.out.println(list[i]);
8   }
9 }
```

Klasė File

```
 1 import java.io.*;
 2 public class RenameFiles {
 3   private static void fileData(File f) {
 4     System.out.println(
 5       "Absolute path: " + f.getAbsolutePath() +
 6       "\n Can read: " + f.canRead() +
 7       "\n Can write: " + f.canWrite() +
 8       "\n getName: " + f.getName() +
 9       "\n getParent: " + f.getParent() +
10       "\n getPath: " + f.getPath() +
11       "\n length: " + f.length() +
12       "\n lastModified: " + f.lastModified()+
13       "\n It's a file: " + f.isFile()+
14       "\n It's a directory: " + f.isDirectory());
15  }
```

Klasė File

```
16 public static void main(String[] args) {
17    File old = new File(args[0]),
18         rname = new File(args[1]);
19    old.renameTo(rname);
20    fileData(old);
21    fileData(rname);
22  }
23}
```

Klasė File

```
 1 import java.io.*;
 2 public class MakeDirectories {
 3   private static void fileData(File f) { ... }
 4   public static void main(String[] args) {
 5     for(int i=0; i<args.length; i++) {
 6       File f = new File(args[i]);
 7       if(f.exists()) {
 8         System.out.println(f + " exists");
 9         fileData(f);
10       } else {
11         f.mkdirs();
12         System.out.println("created " + f);
13         fileData(f);
14         System.out.println("deleting..." + f);
15         f.delete();
16       }
17     }
18 }}
```

Baitinio įvedimo/išvedimo bazinės klasės

- InputStream
- OutputStream

Baitinio įvedimo klasės

- ByteArrayInputStream
- StringBufferInputStream
- FileInputStream
- FilterInputStream
  - DataInputStream
  - BufferedInputStream
  - LineNumberInputStream

Įvedimo klasių metodai

- available() throws IOException
- close() throws IOException
- mark(int readlimit)
- read(), read(byte[]) throws IOException
- reset() throws IOException
- skip() throws IOException

Baitinio išvedimo klasės

- ByteArrayOutputStream
- FileOutputStream
- FilterOutputStream
  - DataOutputStream
  - PrintStream
  - BufferedOutputStream

Išvedimo klasių metodai

- close() throws IOException
- flush() throws IOException
- write(byte), write(byte[]) throws IOException

Filtrų panaudojimas

- Klases, paveldėtas iš FilterXXXStream vadinsime filtrais (angl. decorator)
- Jos naudojamos apgaubti kitas klases pvz., FilterXXXStream(FileXXXStream)
- Klasės DataXXXStream skirtos įvesti/ išvesti paprastus duomenų tipus
- Kitos filtrų klasės tik nurodo įvedimo būdą

DataInputStream metodai

- readBoolean() throws IOException
- readByte() throws IOException
- readChar() throws IOException
- readDouble() throws IOException
- readFloat() throws IOException
- readInt() throws IOException
- readLong() throws IOException
- readShort() throws IOException
- readUTF() throws IOException

DataOutputStream metodai

- size()
- writeBoolean(boolean) throws IOException
- writeByte(int) throws IOException
- writeBytes(String) throws IOException
- writeChar(int) throws IOException
- writeChars(String) throws IOException
- writeDouble(double) throws IOException
- writeFloat(float) throws IOException
- writeInt(int) throws IOException
- writeLong(long) throws IOException
- writeShort(int) throws IOException
- writeUTF(String) throws IOException

DataOutputStream ir PrintStream skirtumai

- DataOutputStream skirtas išvesti tokiu formatu, kad DataInputStream galėtų nuskaityti
- PrintStream skirta išvesti pasižiūrėjimui

PrintStream metodai

- append(char)
- checkError()
- print(...)
- printf(String format, Object)
- println(...)

(negeneruoja IOException,
reikia tikrinti su checkError())

Baitinio ir simbolinio įvedimo/išvedimo klasių atitinkamybė

InputStream	Reader (InputStreamReader)
OutputStream	Writer (InputStreamWriter)
FileInputStream	FileReader
FileOutputStream	FileWriter
StringBufferInputStream	StringReader
(nėra)	StringWriter
ByteArrayInputStream	CharArrayReader
ByteArrayInputStream	CharArrayWriter

Baitinio ir simbolinio įvedimo/išvedimo klasių atitinkamybė

- FilterInputStream
- FilterOutputStream
- BufferedInputStream
- BufferedOutputStream
- DataInputStream
- PrintStream
- LineNumberInputStream

- FilterReader
- FilterWriter
- BufferedReader
- BufferedWriter
- DataInputStream
- PrintWriter
- LineNumberWriter

RandomAccessFile

- Turi tiek DataInput, tiek DataOutput interfeisą
- Dar turi metodus
  - getFilePointer()
  - seek()
  - length()
- Konstruktoriuje reikalingas ir antras parametras, kurio reikšmė “r” arba “rw”

Tipiški įvedimo/išvedimo naudojimo pavyzdžiai

```
1 import java.io.*;
2 public class IOStreamDemo {
  // Throw exceptions to console:
3   public static void main(String[] args)
4   throws IOException {
```

Skaitymas po eilutę iš failo

```
 5  BufferedReader in = new BufferedReader(
 6        new FileReader("IOStreamDemo.java"));
 7  String s, s2 = new String();
 8  while((s = in.readLine())!= null)
 9        s2 += s + "\n";
10  in.close();
```

Skaitymas iš standartinio įvedimo

```
11  BufferedReader stdin = new BufferedReader(
12          new InputStreamReader(System.in));
13  System.out.print("Enter a line:");
14  System.out.println(stdin.readLine());
```

Skaitymas iš atminties

```
15  StringReader in2 = new StringReader(s2);
16  int c;
17  while((c = in2.read()) != -1)
18      System.out.print((char)c);
```

Formatuotas skaitymas iš atminties

```
19  try {
20    DataInputStream in3 = new DataInputStream(
21        new ByteArrayInputStream(s2.getBytes()));
22    while(true)
23        System.out.print((char)in3.readByte());
24  } catch(EOFException e) {
25    System.err.println("End of stream");
26  }
```

Rašymas į failą

```
27  try {
28    BufferedReader in4 = new BufferedReader(
29      new StringReader(s2));
30    PrintWriter out1 = new PrintWriter(
31      new BufferedWriter(new FileWriter("IODemo.out")));
32    int lineCount = 1;
33    while((s = in4.readLine()) != null )
34      out1.println(lineCount++ + ": " + s);
35    out1.close();
36  } catch(EOFException e) {
37    System.err.println("End of stream");
38  }
```

Duomenų įrašymas ir nuskaitymas

```
39  try {
40    DataOutputStream out2 = new DataOutputStream(
41      new BufferedOutputStream(
42        new FileOutputStream("Data.txt")));
43    out2.writeDouble(3.14159);
44    out2.writeUTF("That was pi");
45    out2.close();
46    DataInputStream in5 = new DataInputStream(
47      new BufferedInputStream(
48        new FileInputStream("Data.txt")));
      // Must use DataInputStream
49    System.out.println(in5.readDouble());
      // Only readUTF() will recover the Java-UTF String
50    System.out.println(in5.readUTF());
51  } catch(EOFException e) {throw new RuntimeException(e);}
```

Skaitymas/rašymas į pasirinktinio priėjimo failą

```
52  RandomAccessFile rf =
53      new RandomAccessFile("rtest.dat", "rw");
54  for(int i = 0; i < 10; i++)
55      rf.writeDouble(i*1.414);
56  rf.close();
57  rf = new RandomAccessFile("rtest.dat", "rw");
58  rf.seek(5*8);
59  rf.writeDouble(47.0001);
60  rf.close();
61  rf = new RandomAccessFile("rtest.dat", "r");
62  for(int i = 0; i < 10; i++)
63    System.out.println("Value " + i + ": " +
64       rf.readDouble());
65  rf.close();
66  }}
```

Standartinis įvedimas/išvedimas

- Klasėms System.out ir System.err jau uždėtas filtras PrintStream, todėl jas jau galima naudoti
- System.in veikia kaip InputStream be filtro, todėl prieš naudojant patartina uždėti filtrą, pvz.,

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