Java

Curs 6

Danciu Gabriel Mihail
Noiembrie 2018

java.io

BufferedInputStream	FileWriter	PipedOutputStream
BufferedOutputStream	FilterInputStream	PipedReader
BufferedReader	FilterOutputStream	PipedWriter
BufferedWriter	FilterReader	PrintStream
ByteArrayInputStream	FilterWriter	PrintWriter
ByteArrayOutputStream	InputStream	PushbackInputStream
CharArrayReader	InputStreamReader	PushbackReader
CharArrayWriter	LineNumberReader	RandomAccessFile
Console	ObjectInputStream	Reader
DataInputStream	ObjectInputStream.GetField	SequenceInputStream
DataOutputStream	ObjectOutputStream	SerializablePermission
File	ObjectOutputStream.PutField	StreamTokenizer
FileDescriptor	ObjectStreamClass	StringReader
FileInputStream	ObjectStreamField	StringWriter
FileOutputStream	OutputStream	Writer
FilePermission	OutputStreamWriter	
FileReader	PipedInputStream	

Interfețe java.io

Closeable	FileFilter	ObjectInputValidation
DataInput	FilenameFilter	ObjectOutput
DataOutput	Flushable	ObjectStreamConstants
Externalizable	ObjectInput	Serializable

Clasa File

```
class FileDemo {
   static void p(String s) {
      System.out.println(s);
   }
   public static void main(String args[]) {
      File f1 = new File("/java/COPYRIGHT");
      p("File Name: " + f1.getName());
      p("Path: " + f1.getPath());
      p("Abs Path: " + f1.getAbsolutePath());
      p("Parent: " + f1.getParent());
      p(f1.exists() ? "exists" : "does not exist");
      p(f1.canWrite() ? "is writeable" : "is not writeable");
      p(f1.canRead() ? "is readable" : "is not readable");
      p("is " + (f1.isDirectory() ? "" : "not" + " a directory"));
      p(f1.isFile() ? "is normal file" : "might be a named pipe");
      p(f1.isAbsolute() ? "is absolute" : "is not absolute");
      p("File last modified: " + f1.lastModified());
      p("File size: " + f1.length() + " Bytes");
   }
}
```

```
File Name: COPYRIGHT
Path: \java\COPYRIGHT
Abs Path: C:\java\COPYRIGHT
Parent: \java
exists
is writeable
is readable
is not a directory
is normal file
is not absolute
File last modified: 1542705291620
File size: 9 Bytes
```

sau

```
File Name: COPYRIGHT
Path: \java\COPYRIGHT
Abs Path: C:\java\COPYRIGHT
Parent: \java
does not exist
is not writeable
is not readable
is not a directory
might be a named pipe
is not absolute
File last modified: 0
File size: 0 Bytes
```

File -> Directory

```
String dirname = "/java";
File f1 = new File(dirname);
if (f1.isDirectory()) {
    System.out.println("Directory of " + dirname);
    String s[] = f1.list();
    for (int i = 0; i < s.length; i++) {
        File f = new File(dirname + "/" + s[i]);
        if (f.isDirectory()) {
            System.out.println(s[i] + " is a directory");
        } else {
            System.out.println(s[i] + " is a file");
        }
    }
} else {
    System.out.println(dirname + " is not a directory");
}
```

Directory of /java
copyright1 is a file

FilenameFilter

```
class OnlyExt implements FilenameFilter {
    String ext;
    public OnlyExt(String ext) {
        this.ext = "." + ext;
    }
    public boolean accept(File dir, String name)
    {
        return name.endsWith(ext);
    }
}
```

```
class FileDemo {
    static void p(String s) {
        System.out.println(s);
    }
    public static void main(String args[]) {
        String dirname = "/java";
        File f1 = new File(dirname);
        FilenameFilter only = new OnlyExt("html");
        String s[] = f1.list(only);
        for (int i = 0; i < s.length; i++) {
            System.out.println(s[i]);
        }
    }
}
```

Interfețele AutoCloseable, Closeable, și Flushable

- **AutoCloseable:** *void close( ) throws Exception*
- **Closeable** extends **AutoCloseable**
- **Flushable:** *void flush( ) throws IOException*

Excepții I/O

- **IOException**
- **FileNotFoundException**
- **SecurityException**

Cum se lucrează cu orice resursa de tip stream

```
//declare the file
try {
    // open and access file
} catch (Exception e) {
    // log e
} finally {
    // close the file
}
```

Nota: *file* poate fi înlocuit cu orice resursă asemănătoare: stream, conexiune la baza de date, etc

Clasele Stream

- **InputStream**
- **OutputStream** → byte

Stream Class	Meaning
BufferedInputStream	Buffered input stream
BufferedOutputStream	Buffered output stream
ByteArrayInputStream	Input stream that reads from a byte array
ByteArrayOutputStream	Output stream that writes to a byte array
DataInputStream	An input stream that contains methods for reading the Java standard data types
DataOutputStream	An output stream that contains methods for writing the Java standard data types
FileInputStream	Input stream that reads from a file
FileOutputStream	Output stream that writes to a file
FilterInputStream	Implements **InputStream**
FilterOutputStream	Implements **OutputStream**
InputStream	Abstract class that describes stream input → System.in
ObjectInputStream	Input stream for objects
ObjectOutputStream	Output stream for objects
OutputStream	Abstract class that describes stream output
PipedInputStream	Input pipe
PipedOutputStream	Output pipe
PrintStream	Output stream that contains **print( )** and **println( )** → System.out
PushbackInputStream	Input stream that supports one-byte "unget," which returns a byte to the input stream
SequenceInputStream	Input stream that is a combination of two or more input streams that will be read sequentially, one after the other

- **Reader**
- **Writer** → char

Stream Class	Meaning
BufferedReader	Buffered input character stream
BufferedWriter	Buffered output character stream
CharArrayReader	Input stream that reads from a character array
CharArrayWriter	Output stream that writes to a character array
FileReader	Input stream that reads from a file
FileWriter	Output stream that writes to a file
FilterReader	Filtered reader
FilterWriter	Filtered writer
InputStreamReader	Input stream that translates bytes to characters
LineNumberReader	Input stream that counts lines
OutputStreamWriter	Output stream that translates characters to bytes
PipedReader	Input pipe
PipedWriter	Output pipe
PrintWriter	Output stream that contains **print( )** and **println( )**
PushbackReader	Input stream that allows characters to be returned to the input stream
Reader	Abstract class that describes character stream input
StringReader	Input stream that reads from a string
StringWriter	Output stream that writes to a string
Writer	Abstract class that describes character stream output

Input de la consolă

```
public static void main(String args[]) throws IOException {
        char c;
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        System.out.println("Enter characters, 'q' to quit.");
        // read characters
        do {
            c = (char) br.read();
            System.out.println(c);
        } while (c != 'q');
    }
```

Citirea unui String de la consolă

```
 public static void main(String args[]) throws IOException {
// create a BufferedReader using System.in
        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
        String str;
        System.out.println("Enter lines of text.");
        System.out.println("Enter 'stop' to quit.");
        do {
            str = br.readLine();
            System.out.println(str);
        } while (!str.equals("stop"));
    }
```

Scrierea la consolă

```
import java.io.PrintWriter;


class FileDemo {


    public static void main(String args[]) {
        int b;
        b = 'A';
        System.out.write(b);
        System.out.write('\n');
        System.out.flush();
        PrintWriter pw = new PrintWriter(System.out, true);
        pw.println("This is a string");
        int i = -7;
        pw.println(i);
        double d = 4.5e-7;
        pw.println(d);
    }
}
```

Citirea Fișierelor

```
public static void main(String args[]) {
    int i;
    FileInputStream fin;
//incercare de a deschide fisierul file.txt
    try {
       fin = new FileInputStream("file.txt");
    } catch (FileNotFoundException e) {
       System.out.println("Cannot Open File");
       return;
    }
    try {
       do {
          i = fin.read();  //incercare de a citi fisierul
          if (i != -1) {
             System.out.print((char) i);
          }
       } while (i != -1);
    } catch (IOException e) {
       System.out.println("Error Reading File");
    } finally {
       try {
          fin.close(); //incercare de a inchide fisierul
       } catch (IOException e) {
          System.out.println("Error Closing File");
       }
    }
}
```

Scrierea in fișiere

```
public static void main(String args[]) {
    int i=0;
    FileOutputStream fout;
    try {
       fout = new FileOutputStream("file.txt");
    } catch (FileNotFoundException e) {
       System.out.println("Cannot Open File");
       return;
    }
    String str = "ceva de afisat in fisier";
```

```
try {
        do {
           fout.write(str.charAt(i++));
        } while (i < str.length());
    } catch (IOException e) {
        System.out.println("Error Writing File");
    } finally {
        try {
           fout.close();
        } catch (IOException e) {
           System.out.println("Error Closing File");
        }
    }
}
```

Fisiere ca resursa

```
public static void main(String args[]) {
        //exemplul de citire din fisier - simplificat
        int i=0;
        try (FileInputStream fin = new FileInputStream("file.txt"))
        {
            do {
                i = fin.read();  //incercare de a citi fisierul
                if (i != -1) {
                    System.out.print((char) i);
                }
            } while (i != -1);
        } catch (IOException e) {
            System.out.println(e);
        }
    }
```

Continuare

```
 String source = "Now is the time for all good men\n"
                 + " to come to the aid of their country\n"
                 + " and pay their due taxes.";
         byte buf[] = source.getBytes();
// Use try-with-resources to close the files.
         try (FileOutputStream f0 = new FileOutputStream("file1.txt");
                 FileOutputStream f1 = new FileOutputStream("file2.txt");
                 FileOutputStream f2 = new FileOutputStream("file3.txt")) {
// write to first file
             for (int i = 0; i < buf.length; i += 2) {
                 f0.write(buf[i]);
             }
// write to second file
             f1.write(buf);
// write to third file
             f2.write(buf, buf.length - buf.length / 4, buf.length / 4);
         } catch (IOException e) {
             System.out.println("An I/O Error Occurred");
         }
```

ByteArrayInputStream

```
String tmp = "abc";
byte b[] = tmp.getBytes();
ByteArrayInputStream in = new ByteArrayInputStream(b);
for (int i = 0; i < 2; i++) {
    int c;
    while ((c = in.read()) != -1) {
        if (i == 0) {
            System.out.print((char) c);
        } else {
            System.out.print(Character.toUpperCase((char) c));
        }
    }
    System.out.println();
    in.reset();
}
```

ByteArrayOutputStream

```
 ByteArrayOutputStream f = new
ByteArrayOutputStream();
        String s = "This should end up in the
array";
        byte buf[] = s.getBytes();
        try {
            f.write(buf);
        } catch (IOException e) {
            System.out.println("Error Writing to
Buffer");
            return;
        }
        System.out.println("Buffer as a string");
        System.out.println(f.toString());
        System.out.println("Into array");
        byte b[] = f.toByteArray();
        for (int i = 0; i < b.length; i++) {
            System.out.print((char) b[i]);
        }
        System.out.println("\nTo an
OutputStream()");
```

```
// Use try-with-resources to manage the file
stream.
        try (FileOutputStream f2 = new
FileOutputStream("test.txt")) {
            f.writeTo(f2);
        } catch (IOException e) {
            System.out.println("I/O Error: " + e);
            return;
        }
        System.out.println("Doing a reset");
        f.reset();
        for (int i = 0; i < 3; i++) {
            f.write('X');
        }
        System.out.println(f.toString());
```

PushbackInputStream

```
String s = "if (a == 4) a = 0;\n";
        byte buf[] = s.getBytes();
        ByteArrayInputStream in = new ByteArrayInputStream(buf);
        int c;
        try (PushbackInputStream f = new PushbackInputStream(in)) {
            while ((c = f.read()) != -1) {
                switch (c) {
                    case '=':
                        int newc;
                        if ((newc = f.read()) == '=') {
                            System.out.print(".eq.");
                        } else {
                            System.out.print("<-");
                            f.unread(newc);
                        }
                        break;
                    default:
                        System.out.print((char) c);
                        break;
                }
            }
        } catch (IOException e) {
            System.out.println("I/O Error: " + e);
        }
```

SequenceInputStream

```
class InputStreamEnumerator implements
Enumeration<FileInputStream> {


    private Enumeration<String> files;


    public InputStreamEnumerator(Vector<String>
files) {
        this.files = files.elements();
    }


    public boolean hasMoreElements() {
        return files.hasMoreElements();
    }


    public FileInputStream nextElement() {
        try {
            return new
FileInputStream(files.nextElement().toString());
        } catch (IOException e) {
            return null;
        }
    }
}
```

Main:

```
int c;

Vector<String> files = new Vector<String>();

files.addElement("file1.txt");

files.addElement("file2.txt");

files.addElement("file3.txt");

InputStreamEnumerator ise = new InputStreamEnumerator(files);

InputStream input = new SequenceInputStream(ise);

try {

    while ((c = input.read()) != -1) {

        System.out.print((char) c);

    }

} catch (NullPointerException e) {

    System.out.println("Error Opening File.");

} catch (IOException e) {

    System.out.println("I/O Error: " + e);

} finally {

    try {

        input.close();

    } catch (IOException e) {

        System.out.println("Error Closing SequenceInputStream");

    }

}
```

PrintStream

```
System.out.println("Here are some numeric values "
        + "in different formats.\n");
System.out.printf("Various integer formats: ");
System.out.printf("%d %(d %+d %05d\n", 3, -3, 3, 3);
System.out.println();
System.out.printf("Default floating-point format: %f\n",
        1234567.123);
System.out.printf("Floating-point with commas: %,f\n",
        1234567.123);
System.out.printf("Negative floating-point default: %,f\n",
        -1234567.123);
System.out.printf("Negative floating-point option: %,(f\n",
        -1234567.123);
System.out.println();
System.out.printf("Line up positive and negative values:\n");
System.out.printf("% ,.2f\n% ,.2f\n",
        1234567.123, -1234567.123);
```

BufferedReader

```
try {
    File f = new File("file2.txt");

     BufferedReader b = new BufferedReader(new FileReader(f));

     String readLine = "";

     System.out.println("Reading file using Buffered Reader");

     while ((readLine = b.readLine()) != null) {
         System.out.println(readLine);
     }

 } catch (IOException e) {
     e.printStackTrace();
 }
```

BufferedWriter

```
try {

            File fout = new File("out.txt");
            FileOutputStream fos = new FileOutputStream(fout);

            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));

            for (int i = 0; i < 10; i++) {
                bw.write("line no " + i);
                bw.newLine();
            }

            bw.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
```

Serialization

```
class MyClass implements Serializable {


    String s;
    int i;
    double d;


    public MyClass(String s, int i,
double d) {
        this.s = s;
        this.i = i;
        this.d = d;
    }


    public String toString() {
        return "s=" + s + "; i=" + i + ";
d=" + d;
    }
}
```

Main:

```
// Object serialization
        try (ObjectOutputStream objOStrm
                = new ObjectOutputStream(new
FileOutputStream("serial"))) {
            MyClass object1 = new MyClass("Hello", -7, 2.7e10);
            MyClass object2 = new MyClass("Some String", 18,
2.34);
            objOStrm.writeObject(object1);
            objOStrm.writeObject(object2);
        } catch (IOException e) {
            System.out.println("Exception during serialization: "
+ e);
        }
// Object deserialization
        try (ObjectInputStream objIStrm
                = new ObjectInputStream(new
FileInputStream("serial"))) {
            MyClass object1 = (MyClass) objIStrm.readObject();
            MyClass object2 = (MyClass) objIStrm.readObject();
            System.out.println("object1 : " + object1);
            System.out.println("object2 : " + object2);
        } catch (Exception e) {
            System.out.println("Exception during deserialization:
" + e);
        }
```

Exercitii

- Se dă clasa Punct cu membrii de tip int: x,y,z.
  - Salvați colecția de puncte [(2,3,7), (-1,3,4),(5,4,0), (4,-2,6)] într-un fișier de tip CSV (comma separated value)
  - Citiți în alt program acest fișier și afișați punctele
  - Introduceți la finalul fișierului un nou punct (1,4,1)
  - Introduceți în fișier dupa un anumit punct (exemplu dupa (5,4,0)) noul punct (3,-6,2)
- Încercați toate cerințele de mai sus folosind un fișier binar (serializare)
- Extra: Încercați toate cerințele de mai sus folosind un fișier XML sau JSON