Basic Input & Output

Silviya Brayanova, Ivan St. Ivanov
FMI
November 2014

Съдържание

- Input & Output Streams
- File Streams
- Byte Streams
- Character Streams
- Buffered Streams
- Data Streams
- Object Streams

Входни и изходни потоци
Байтов вход и изход

Супер класове:

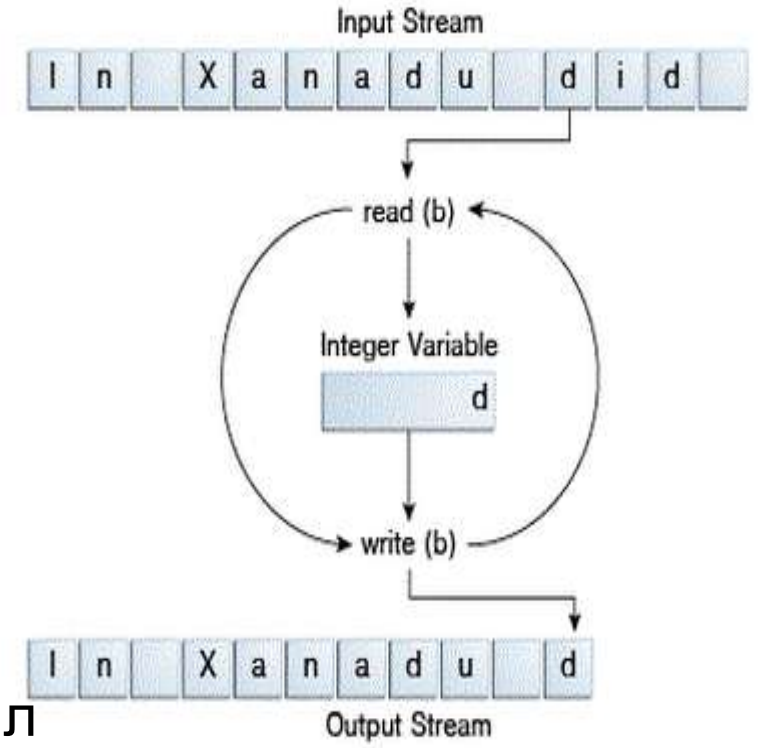
InputStream – четене

OutputStream – писане

Четене от и писане във файл:

FileInputStream – четене от файл

FileOutputStream – писане във файл



Simple byte stream input and output.

Вход и изход във файлове
Инициализиране на потоците

Четене от файл:

FileInputStream – четене от файл

```
FileInputStream in = null;

try {
    in = new FileInputStream(
            "xanadu.txt");
} finally {
    if (in != null) {
        in.close();
    }
}
```

Писане във файл:

FileOutputStream – писане във файл

```
FileOutputStream out = null;

try {
    out = new FileOutputStream(
            "outagain.txt");
} finally {
   if (out != null) {
       out.close();
   }
}
```

Вход и изход във файлове

Четене и писане

```
FileInputStream in = null;
FileOutputStream out = null;

try {
    in = new FileInputStream("xanadu.txt");
    out = new FileOutputStream("outagain.txt");
    int c;

    while ((c = in.read()) != -1) { // четене
        out.write(c);               // писане
    }
} catch (IOException io) {
    // обработване
} finally {
    // затваряне
}
```

Затваряне на поток
Трудният начин

```
FileInputStream in = null;

try {
   ...
} finally {
    if (in != null) {
        try {
            in.close();
        } catch (IOException ioe) {
            // обработване
        }
    }
}
```

Затваряне на поток

Autoclosable и try with resources

„Заемане“ на ресурси

```
try (FileInputStream in =
        new FileInputStream("xanadu.txt")) {
    in.read();
    ...
} catch (IOException ioe) {
    // обработване
}
```

java.lang.AutoCloseable

Байтови и символни потоци

- Unicode представяне
- Конвертиране при четене и писане
- Символните потоци правят това автоматично
- Интернационализация

Символни потоци

Супер класове:

Reader – четене

Writer – писане

Четене от и писане във текстов файл:

FileReader – четене от файл

FileWriter – писане във файл

Вход и изход в текстови файлове

Инициализиране на потоците

Четене от файл:

FileReader – четене от файл

```
try(FileReader in = new
  FileReader ("xanadu.txt")) {
      // четене
} catch (IOException ioe) {
      // обработване
}
```

Писане във файл:

FileWriter – писане във файл

```
try(FileWriter out = new
  FileWriter ("outagain.txt")){
      // писане
} catch (IOException ioe) {
      // обработване
}
```

Буферирани потоци
Защо?

- Четенето/писането се изпълнява от Операционната Система
  - Достъп до диска
  - Мрежова активност
- Небуферираното четене/писане е бавно
- Удобни методи

Буферирани потоци
Инициализация

Буферирани байтови потоци:

BufferedInputStream
BufferedOutputStream

```
BufferedInputStream bis =
    new BufferedInputStream(
        new FileInputStream(
            "xanadu.txt"));

BufferedOutputStream bos =
    new BufferedOutputStream(
        new FileOutputStream(
            "outagain.txt"));
```

Буферирани символни потоци:

BufferedReader
BufferedWriter

```
BufferedReader br =
    new BufferedReader(
        new FileReader(
            "xanadu.txt"));

BufferedWriter bw =
    new BufferedWriter(
        new FileWriter(
            "charoutput.txt"));
```

Други потоци

- PrintStream, PrintWriter
  - Форматиране на изход
  - Извеждане на редове
- Служебни потоци
  - System.in
  - System.out и System.err
  - Console и System.console

Четене с java.util.Scanner

```
try (Scanner inputScanner =
        new Scanner(System.in);
     Scanner fileScanner =
         new Scanner(new FileReader(“f.txt"));
     Scanner stringScanner =
         new Scanner("someString")) {

    while (inputScanner.hasNext()) {
        String line = inputScanner.nextLine();
        ...
    }

}
```

Потоци за данни (Data Streams)

- Поддържат четене и писане на цели обекти
- Атрибутите – примитивни или String
- Програмистът се грижи за реда на писане и четене
- Хвърлят EOFException при край

Писане и четене с DataStream

Класът

```
public class Person {
    String name;
    int age;
    boolean isMale;
}

Person person = new Person();
```

Писане и четене с DataStream

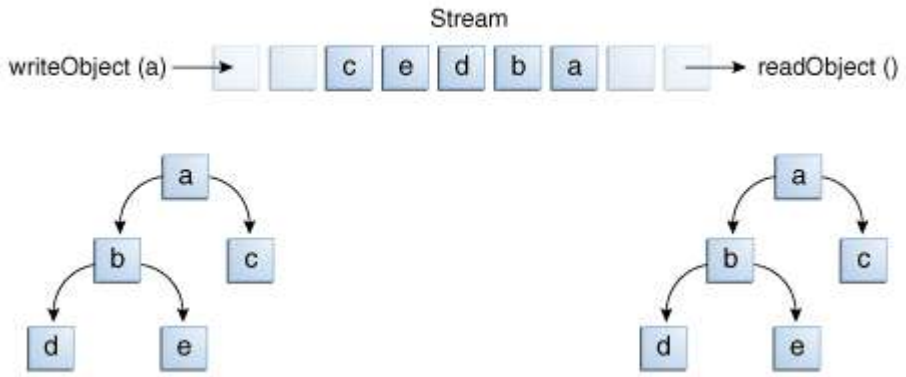
Манипулиране

```
DataOutputStream dos = new DataOutputStream(
        new FileOutputStream("data.bin"));
dos.writeChars(person.getName());
dos.writeInt(person.getAge());
dos.writeBoolean(person.isMale());

DataInputStream dis = new DataInputStream(
        new FileInputStream("data.bin"));
person.setName(dis.readUTF());
person.setAge(dis.readInt());
person.setMale(dis.readBoolean());
```

Потоци за обекти (Object Streams)

- Наследяват Data streams
- Може да записват цели обекти
- Serializable маркер интерфейс
- Поддържат йерархии



Писане и четене с ObjectStream

```
public class Person implements Serializable {
    String name;
    int age;
    boolean isMale;
}
Person person = new Person();

ObjectOutputStream ous = new ObjectOutputStream(
        new FileOutputStream("data.bin"));
ous.writeObject(person);

ObjectInputStream ois = new ObjectInputStream(
        new FileInputStream("data.bin"));
person = (Person) ois.readObject();
```

Благодаря за вниманието!

За контакти:

silviya.brayanova@sap.com
ivan.ivanov@sap.com
SAP Labs Bulgaria
София 1618
бул. Цар Борис III, 136А
тел: 02 91 57 690