Java

Java files IO

Vincent Gerber, Tilman Hinnerichs

Java Kurs

19. Juli 2018

Overview

1. How to use streams

Streams

A stream can be defined as a sequence of data. We got two kinds of streams in Java:

InPutStream	is used to read data from a source
OutPutStream	is used for writing data to a destination

Many implementations can be found in the module **java.io**.

File Streams

Byte streams used to perform input and ouput of 8-bit bytes. The most common streams for file reading and writing are **FileInputStream** and **FileOutputStream**:

How to use file streams

We would like to copy a file "trateotu.txt":

```
1   import java.io.*
2   ... main(String args[]) throws IOException {
3       FileInputStream in = null;
4       FileOutputStream out = null;
5
6       try {
7           in = new FileInputStream("trateotu-input.txt");
8           out = new FileOutputStream("trateotu-output.txt");
9
10          int c;
11          while ((c = in.read()) != -1) {
12              out.write(c);
13          }
14      }
15      finally {
16          if (in != null) {
17              in.close();
18          }
19          if (out != null) {
20              out.close();
21          }
22      }
```

Other useful methods for InputStream

public void close()	releases any system resources associated with the file
protected void finalize()	cleans up connection to the file, closes stream when there are no more references
public int read(int r)	this method reads the specified byte of data
public int available()	returns the amount of bytes which can be read from the input stream

Othe useful methods for OutputStream

public void close()	closes output stream, releases any resources
public void finalize()	see InputStream
public void write(int w)	writes the specified byte

Character Streams

Instead of the previously used Byte streams, we can also use **Character streams**. **Character streams** are able to perform input and output for 16-bit unicode:

```
1          in = new FileReader("ltuae-input.txt);
2          out = new FileReader("ltuae-output.txt);
3
```

User interaction

Now knowing about streams we can also perform command line input/output. Java provides the following three streams, given by your OS:

Standard Input	is used to feed data into you program, is your standard input stream, represented as **System.in**
Standard Output	is used to output the data produced by the user's program, represented as **System.out**
Standard Error	is used to output the error data produced by our program, **System.err**

First Console IO program

We will now read standard input stream until the user types a "q":

```
1   InputStreamReader cin = null;
2
3   try {
4       cin = new InputStreamReader(System.in);
5       System.out.println("Enter characters, 'q' to quit.");
6       char c;
7       do {
8           c = (char) cin.read();
9           System.out.print(c);
10      } while(c != 'q');
11  }
12  finally {
13      if (cin != null) {
14          cin.close();
15      }
16  }
17
```