**The Government of the Russian Federation**

**The Federal State Autonomous Institution of Higher Education "National Research University - Higher School of Economics"**

National Research University «Higher School of Economics»

Faculty of Information Technology and Computer Engineering

Department of Computer Systems and Networks

**Course title**: Network computing

**Practical training № 6. Java - Streams, Files and I/O.**

|  |  |
| --- | --- |
| Senior lecturer | Baybikova T.N. |
| Student: | Lapin A.S. |

Date: 12.01.16

Grade:

Moscow 2016

**Practical training № 6. Java - Streams, Files and I/O.**

**Goal**: Add the possibility of keeping results in files, and the possibility of getting the results from the files to Practical training 4.

**Variant:** 12 Compare two arrays

**Listings and Figures**:

package com.example.sixthlab;  
  
import java.io.\*;  
import java.util.Arrays;  
  
public class Main {  
 public static void main(String[] args) {  
  
 int[] masA = {1, 2, 3, 4, 5};  
 int[] masB = {1, 2, 3, 4, 5};  
  
 try {  
   
 // Операция записи ------------>  
   
 BufferedWriter out = new BufferedWriter(new FileWriter("array.txt"));  
  
 System.*out*.println("File created suc");  
  
 if (Arrays.*equals*(masA, masB)) {  
 out.write("Arrays are equal!");  
 }  
 else {  
 out.write("Arrays are not equal!");  
 }  
  
 out.close();  
  
 // Операция считывания ------------>  
  
 BufferedReader reader = new BufferedReader(new FileReader("array.txt"));  
 String s;  
 int j = 1;  
 int[] masA = new int[0];  
 int[] masB = new int[0];  
 while ((s = reader.readLine()) != null) {  
 String[] buff = s.split(" ");  
 if (j++ == 1) {  
 masA = new int[buff.length];  
 for (int i = 0; i < buff.length; i++) {  
 masA[i] = Integer.*valueOf*(buff[i]);  
 }  
 }  
 else {  
 masB = new int[buff.length];  
 for (int i = 0; i < buff.length; i++) {  
 masB[i] = Integer.*valueOf*(buff[i]);  
 }  
 }  
 }  
  
 reader.close();  
  
 if (Arrays.*equals*(masA, masB)) {  
 System.*out*.println("Arrays are equal!");  
 }  
 else {  
 System.*out*.println("Arrays are not equal!");  
 }  
 }  
 catch (IOException e) {}  
  
 }  
}

This would produce the following result:

Arrays are equal!

Give a description of the used Java - Streams, Files and I/O.

In Java, the main functionality of the streaming is concentrated in the classes from the package *java.io*. In relation to working with files and input-output, we'll talk about the stream, as an abstraction, which is used to read or write data (files, sockets, text console, etc.).

We can be defined stream that is associated with the file, and through which we can keep reading or writing a file. This stream may also be associated with a network socket via which it is possible to send or receive data from the network. All these tasks: reading and writing different files, information exchange network, input-output console we will solve in Java using threads.

The object from which to read the data, called the input stream, and the object to which you can record data - output stream. At the heart of all classes, controls the flow of bytes, there are two abstract classes: InputStream (representing the input streams) and OutputStream (representing the output streams)

To read data from a file is the class *FileInputStream*, which is the successor of class *InputStream* and therefore implements all of its methods. To create an object FileInputStream we can use a number of designers.

Class *FileOutputStream* bytes for recording a file. It is derived from the class *OutputStream*, therefore inherits all of its functionality. To create an object using FileOutputStream constructor that takes as a parameter the path to the file for writing.

As opening or reading the file, an error may occur *IO*, the code reader is placed in the block *try*. And to be sure that the flow in any case will be closed, even when working with it an error, call *close()* method is placed in the unit *finally*. And, as the *close()* method as in the case of an error can throw a *IOException*, then the call is also placed in a nested *try..catch* block.

After the end of the try block in the resource (in this case the object *FileInputStream*) method is automatically called *close()*. When you close the flow release all its resources, such as file. In case the flow will not be closed may be a memory leak.

FileInputStream and FileOutputStream classes designed primarily for writing binary files. Although they can be used also for text files.

**Conclusions:** The application reads two arrays from the file, compares arrays, and writes the result of comparison into the file.

**References:**

1. Baybikova T.N., course of lectures "Network computing", 2016