Lab 4
Using a file access policy

Objective: To gain an understanding of the organization of protection in the Java platform through access rights files.

Objective: To learn how to get and edit access rights to a file in the file system using the Java API.

Work order:

1. Open any IDE
2. Create a project
3. Create a class in the project and re-type the code from Listing 1 below into it
4. Compile the code, and get information about the access rights for the file on the console output
5. Reprint the code in Listing 2 into your project. You can delete the old code file
6. Compile the code, get information about the access rights for the file on the console output and make sure that the access rights have been successfully changed, according to our code
7. Check on a real file
8. Save

Ход выполнения работы:

1. Read supporting information.

The list of permissions for a file depends greatly on the operating system used: different operating systems (*nix/Linux/MacOS, Windows FAT32 & NTFS) allow you to define different rights. However, Java defines a set of rights, which are common to all operating systems, and a set of methods for working with them.

Check for access rights:

- File.canExecute(); — returns true if the file can be executed;
- File.canRead(); — returns true if the file is readable;
- File.canWrite(); — returns true if the file is editable.

Set access right:

- File.setExecutable(Boolean); — make the file executable;
- File.setReadable(Boolean); — make file readable;
- File.setWritable(Boolean); — make file writable.

2. Reprint the code below. Check its performance on any file.

```
package ru.mydesignstudio.file;
import java.io.File;

public class TestPermissions {
public static void main(String[] args) {
File testFile = new File("./PATH TO YOUR FILE ");
```

```
    if (testFile.exists()) {
    System.out.println("Reading Allowed: " + testFile.canRead());
    System.out.println("Change allowed: " + testFile.canWrite());
    System.out.println("Run allowed: " + testFile.canExecute());
        }
      }
    }
```

3. Create a new class and re-type the code below into it.

```
package ru.j4web.examples.java.io;

    import java.io.File;
    import java.io.IOException;
    import java.util.logging.Level;
    import java.util.logging.Logger;

    public class FilePermissionsExample {

        private static final String FILENAME = "/home/administrator/myScript.sh";

        public static void main(String[] args) {

          File file = new File(FILENAME);
             System.out.println("File name: " + FILENAME);

             if (file.exists()) {
              System.out.println("File exists.");
              System.out.println("The file is available for execution: "
                      + file.canExecute());
                 System.out.println("The file is readable: " + file.canRead());
              System.out.println("File is writable: " + file.canWrite());
          }

             System.out.println();
          System.out.println("Setting file permissions.");

             file.setExecutable(true);
             file.setReadable(true);
             file.setWritable(false);

             System.out.println("The file is available for execution: "
                     + file.canExecute());
             System.out.println("The file is readable: " + file.canRead());
             System.out.println("File is writable: " + file.canWrite());

             try {
                 if(file.createNewFile()) {
                     System.out.println("File created.");
                 } else {
                     System.out.println("Failed to create file.");
                 }
             } catch (IOException ex) {
                 Logger.getLogger(FilePermissionsExample.class.getName())
                         .log(Level.SEVERE, null, ex);
             }
        }
    }
```

4. Check by means of windows whether the rights to the file have really changed.