

# JAVA PROGRAMMING COURSE

## EXERCISE

### FILE MANAGEMENT IN JAVA

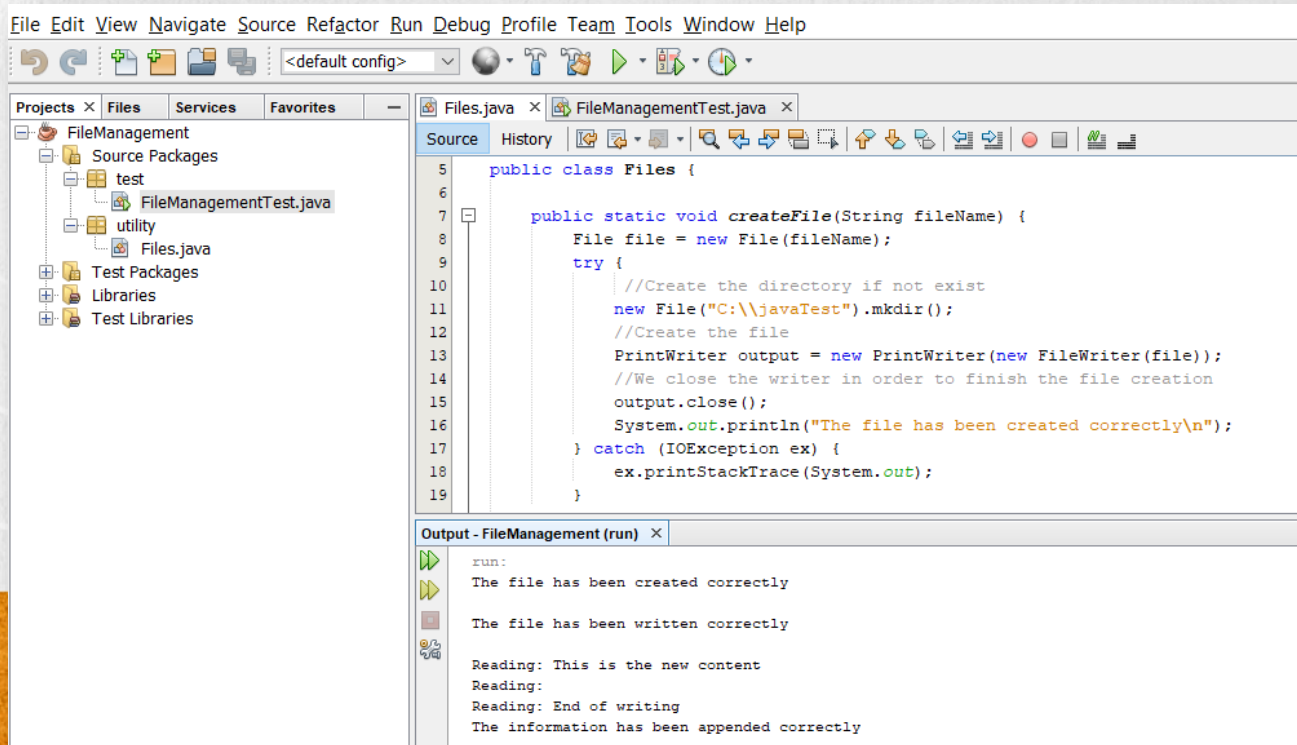


JAVA PROGRAMMING COURSE

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# EXERCISE OBJECTIVE

Implement the concept of file management in Java. At the end we should observe the following:



The screenshot displays an IDE window with the following components:

- Menu Bar:** File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help.
- Toolbar:** Includes icons for undo, redo, save, run, and other development actions.
- Project Explorer (Left):** Shows a project named 'FileManagement' with a 'test' package containing 'FileManagementTest.java' and a 'utility' package containing 'Files.java'.
- Source Editor (Center):** Displays the code for 'Files.java'.
- Output Console (Bottom):** Shows the execution results of the program.

```
5 public class Files {  
6  
7     public static void createFile(String fileName) {  
8         File file = new File(fileName);  
9         try {  
10             //Create the directory if not exist  
11             new File("C:\\\\javaTest").mkdir();  
12             //Create the file  
13             PrintWriter output = new PrintWriter(new FileWriter(file));  
14             //We close the writer in order to finish the file creation  
15             output.close();  
16             System.out.println("The file has been created correctly\\n");  
17         } catch (IOException ex) {  
18             ex.printStackTrace(System.out);  
19         }  
20     }  
21 }
```

**Output - FileManagement (run)**

```
run:  
The file has been created correctly  
  
The file has been written correctly  
  
Reading: This is the new content  
Reading:  
Reading: End of writing  
The information has been appended correctly
```

# 1. CREATE A NEW PROJECT

Create a new project:

**New Java Application**

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



## 2. CREATE A NEW CLASS

Create a new class:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name: Files

Project: FileManagement

Location: Source Packages

Package: utility

Created File: C:\Courses\JavaProgramming\Lesson22\FileManagement\src\utility\Files.java

< Back Next > **Finish** Cancel Help

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

### 3. MODIFY THE CODE

#### Files.java:

```
package utility;
import java.io.*;
public class Files {

    public static void createFile(String fileName) {
        File file = new File(fileName);
        try {
            //Create the directory if not exist
            new File("C:\\\\javaTest").mkdir();
            //Create the file
            PrintWriter output = new PrintWriter(new FileWriter(file));
            //We close the writer in order to finish the file creation
            output.close();
            System.out.println("The file has been created correctly\n");
        } catch (IOException ex) {
            ex.printStackTrace(System.out);
        }
    }
}
```

### 3. MODIFY THE CODE

#### Files.java:

```
public static void writeFile(String fileName, String content) {  
    File file = new File(fileName);  
    try {  
        try (PrintWriter output = new PrintWriter(new FileWriter(file))) {  
            output.println(content);  
            output.println();  
            output.println("End of writing");  
        }  
        System.out.println("The file has been written correctly\n");  
    } catch (IOException ex) {  
        ex.printStackTrace(System.out);  
    }  
}
```

### 3. MODIFY THE CODE

#### Files.java:

```
public static void readFile(String fileName) {  
    File file = new File(fileName);  
    try {  
        try (BufferedReader input = new BufferedReader(new FileReader(file))) {  
            String reading;  
            reading = input.readLine();  
            while (reading != null) {  
                System.out.println("Reading: " + reading);  
                reading = input.readLine();  
            }  
        }  
    } catch (IOException ex) {  
        ex.printStackTrace(System.out);  
    }  
}
```

### 3. MODIFY THE CODE

#### Files.java:

```
public static void appendFile(String fileName, String content) {  
    File file = new File(fileName);  
    try {  
        try (PrintWriter output = new PrintWriter(new FileWriter(file, true))) {  
            output.println(content);  
            output.println();  
            output.println("End of append");  
        }  
        System.out.println("The information has been appended correctly\n");  
    } catch (IOException ex) {  
        ex.printStackTrace(System.out);  
    }  
}
```



# PASO 4. CREATE A NEW CLASS

Create a new class:

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# 5. MODIFY THE CODE

## FileManagementTest.java:

```
package test;

import static utility.Files.*;

public class FileManagementTest {

    //Note: The folder on which you are going to work must already be created
    //And if necessary write permissions must be assigned to the folder
    private static final String FILE_NAME = "c:\\javaTest\\javaFile.txt";

    public static void main(String[] args) {

        //Create a file
        createFile(FILE_NAME);

        //Write to a file
        writeFile(FILE_NAME, "This is the new content");

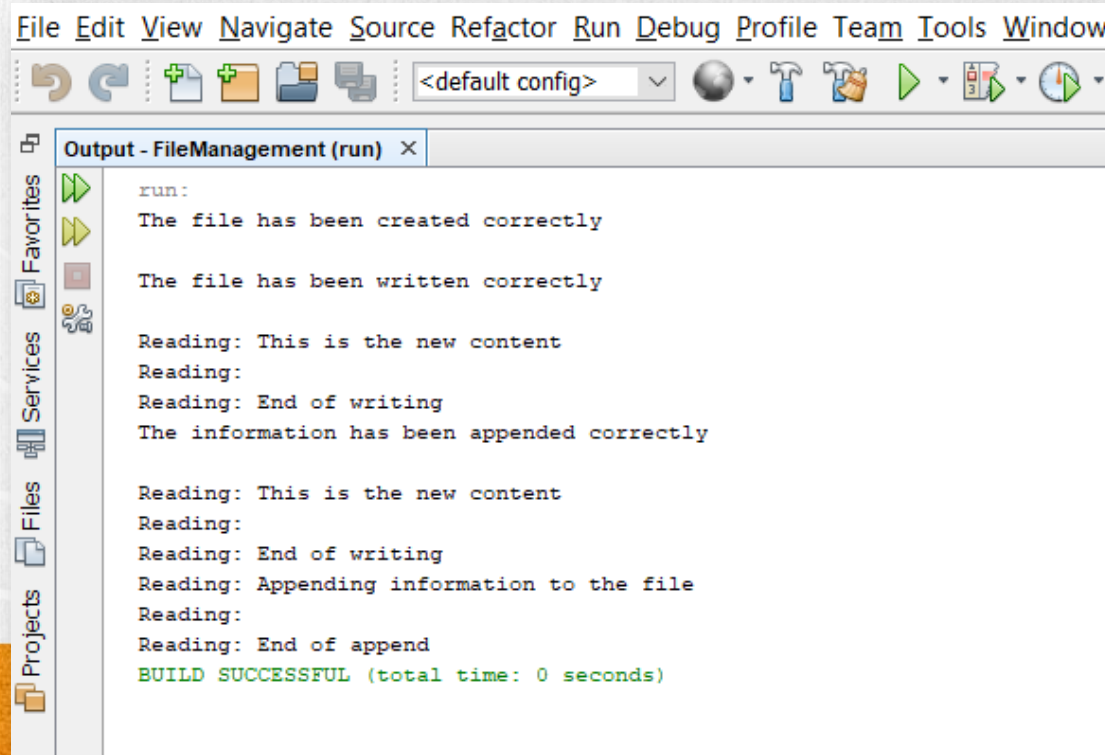
        //Read from a file
        readFile(FILE_NAME);

        //Append to a file
        appendFile(FILE_NAME, "Appending information to the file");

        //Leer de un archivo
        readFile(FILE_NAME);
    }
}
```

## 6. EXECUTE THE PROJECT

The result is as follows:



The screenshot shows an IDE's Output window titled "Output - FileManagement (run)". The window displays the output of a Java application. The output text is as follows:

```
run:
The file has been created correctly

The file has been written correctly

Reading: This is the new content
Reading:
Reading: End of writing
The information has been appended correctly

Reading: This is the new content
Reading:
Reading: End of writing
Reading: Appending information to the file
Reading:
Reading: End of append
BUILD SUCCESSFUL (total time: 0 seconds)
```

# EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of File Management in Java.
- We have seen several operations, such as creating, writing, reading and append information to a file. With this, we have the bases to understand the operation of file management from Java.



**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)



**ONLINE COURSE**

# **JAVA PROGRAMMING**

---

By: Eng. Ubaldo Acosta



**JAVA PROGRAMMING COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)