Java Chapter 10: Files

Contents

Time required: 60 minutes

Do: Online Tutorials

- Java Files
- Java Create and Write to Files
- Java Read Files
- Java Delete Files

Files

Most of the programs we have seen so far are transient in the sense that they run for a short time and produce some output. When they end, their data disappears. If you run the program again, it starts with a clean slate.

Other programs are persistent: they run for a long time (or all the time); they keep at least some of their data in permanent storage (a hard drive, for example); and if they shut down and restart, they pick up where they left off.

Examples of persistent programs are operating systems, which run pretty much whenever a computer is on, and web servers, which run all the time, waiting for requests to come in on the network.

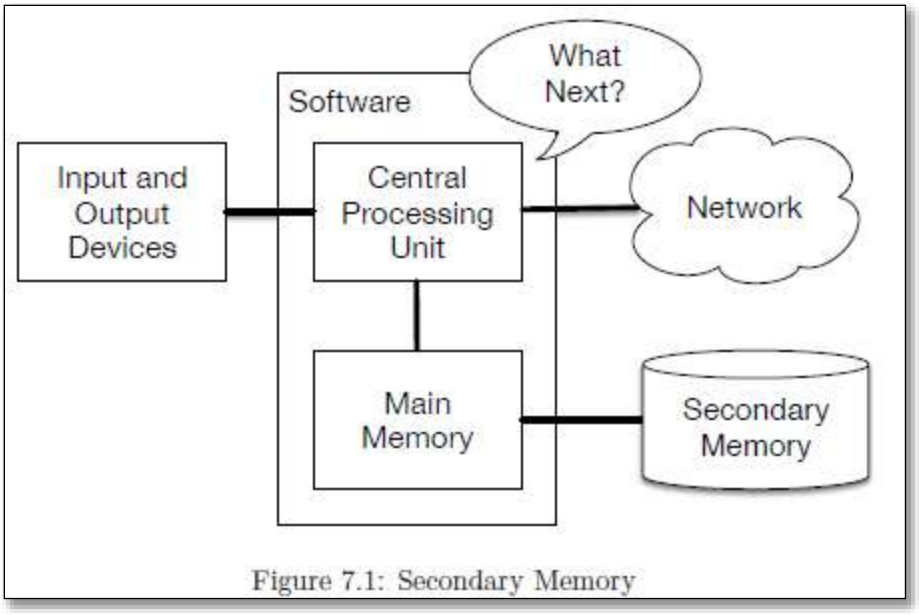

Figure 7.1: Secondary Memory

Secondary memory is not erased when the power is turned off. Or in the case of a USB flash drive, the data we write from our programs can be removed from the system and transported to another system.

Tutorial 8.1: Reading a Text File

In the same folder as the program, create a txt file named **example.txt** that includes 3 lines of text.

Create and test the program as listed. Name it **FileReadFun.java**

```java
/**
 * Name: FileReadFun.java
 * Written by:
 * Written on:
 * Purpose: Read a text file line by line
 */

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;

public class FileReadFun {
    Run | Debug
    public static void main(String[] args) {

        try {
            // Open file handle to operating system
            File infile = new File("example.txt");

            // Scanner reads text input from file, rather than the console
            Scanner input = new Scanner(infile);

            // Read the text file one line at a time
            // while there is still a line of text
            while (input.hasNextLine()) {

                // Get a line of text
                String data = input.nextLine();
                System.out.println(data);
            }

            // Close operating system resource file handle
            input.close();

        } catch (FileNotFoundException ex) {
            System.out.println("Can't find file!");
            System.out.println(ex.getMessage());
        }
    }
}
```

Tutorial 8.2: Writing a Text File

Create and test the following Python program called **FileWriterFun.java**

The **FileWriter** class has two modes of writing to file.

- **false** – Overwrites the previous file
- **true** – appends to the file.

This program uses try: except: to provide error handling. If there is an exception, inform the user.

```java
/**
 * Name: FileWriterFun.java
 * Written by:
 * Written on:
 * Purpose: Write a text file line by line
 */
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

public class FileWriterFun {
    Run | Debug
    public static void main(String[] args) {
        // Filename to write to
        String fileName = "rockstars.txt";

        // Call writeFile method
        writeFile(fileName);
    }
```

```
21      private static void writeFile(String fileName) {
22          try {
23              // Create Filewriter object to open OS file handle
24              // false overwrites the text file, true appends
25              FileWriter fileWriter = new FileWriter(fileName, false);
26
27              // Print text to a file
28              PrintWriter pw = new PrintWriter(fileWriter);
29
30              // Print to file is like printing to the console
31              // Substitute your favorite rock stars
32              pw.println("Dave Grusin");
33              pw.println("Journey");
34              pw.println("Toto");
35
36              // Close OS file handle
37              pw.close();
38
39              System.out.println("File written successfully.");
40
41          } catch (FileNotFoundException ex) {
42              System.out.println(ex.getMessage() + "File doesn't exist.");
43          } catch (IOException ex) {
44              System.out.println(ex.getMessage() +
45                      "Couldn't write to the file!");
46          }
47      }
48  }
```

Example run:

```
File written successfully.
```

You should have a file named: **rockstars.txt** in your current program folder. Double click it to open it up and see if this program worked.

Tutorial 8.3: Writing and Reading Numbers with a Text File

Numbers and other primitive data types can be written to and read from a text file. The data is stored as text, when we read it we convert it to the datatype by using the same Scanner class methods we used with the keyboard.

Create a text file **numbers.txt** with several integers in it with one per line. Don't press the Enter key after the last number, that will cause an error as Java will try to read the Enter key as a number.

```java
/**
 * Name: FileNumberFun.java
 * Written by:
 * Written on:
 * Purpose: Write and read numbers with a text file.
 */

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;
```

```java
public class FileNumberFun {
    Run | Debug
    public static void main(String[] args) {
        String numbers = "numbers.txt";
        String twoTimesNumbers = "twotimesnumbers.txt";
        try {
            File inFile = new File(numbers);

            // Filewriter object set to overwrite the file
            FileWriter fileWriter = new FileWriter(twoTimesNumbers, false);

            // PrintWriter object writes text to a file
            PrintWriter pw = new PrintWriter(fileWriter);

            // Scanner reads text input from file, rather than the console
            Scanner input = new Scanner(inFile);
            int inputNumber;
            int outputNumber;
```

```
33              while (input.hasNextLine()) {
34
35                  // Get a line of text as an int
36                  inputNumber = input.nextInt();
37
38                  // Multiply the number by 2
39                  outputNumber = inputNumber * 2;
40
41                  // Write the new number to file
42                  pw.println(outputNumber);
43                  System.out.println("Input: " + inputNumber +
44                          " * 2  = " + outputNumber + " written to file.");
45              }
46              input.close();
47              pw.close();
48          } catch (FileNotFoundException ex) {
49              System.out.println(ex.getMessage() + "File doesn't exist.");
50          } catch (IOException ex) {
51              System.out.println(ex.getMessage() +
52                      "Couldn't write to the file!");
53          }
54      }
55  }
```

Example run:

```
Input: 12 * 2  = 24 written to file.
Input: 23 * 2  = 46 written to file.
Input: 45 * 2  = 90 written to file.
Input: 66 * 2  = 132 written to file.
Input: 77 * 2  = 154 written to file.
Input: 11 * 2  = 22 written to file.
```

Open each text file to confirm successful operation of the program.

Assignment Submission

1. Attach the program files.
2. Attach screenshots showing the successful operation of the program.
3. Submit in Blackboard.