CS 101: Introduction to Computer Science

Project One - Outputting to the stdout

Student Name Mohammad El-Abid

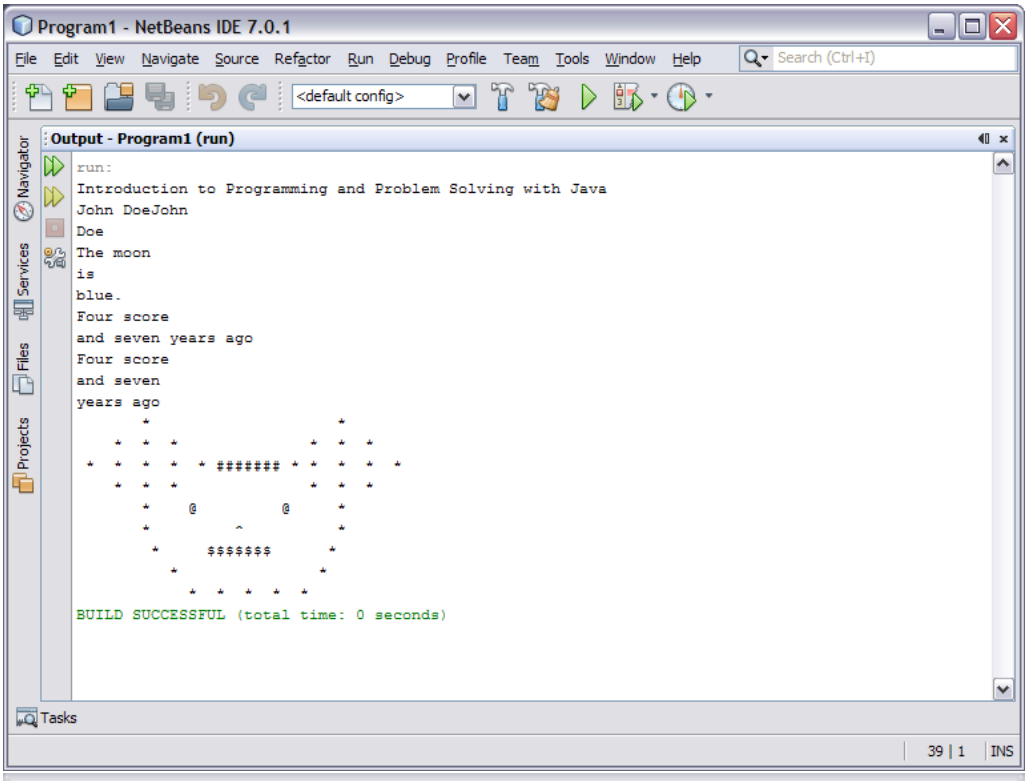
Professor Yasser Elleithy

Date 9/2/2011 with slight modifications thereafter

Table of Contents

Abstract

The goal of this application is to create an output similar to the following, using our name and the System.out.print and System.out.println functions. System.out is a stream that is associated with STDOUT (**St**andar**d out**). I also used exceptions and logging.

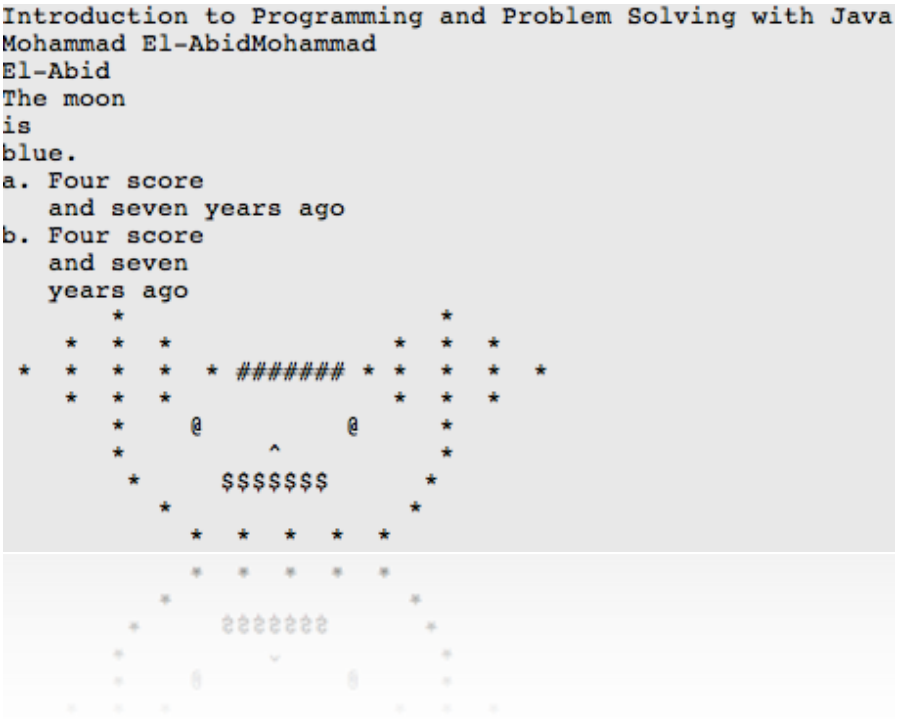


Introduction

Since this is a very basic task, outputting text to the stdout, I decided to take the time to learn Java a bit better. Learning how to use resources (an internal file that is inside the application) and how to organize the code and use the correct Java formatting standards.

Screenshots

Application output to stdout:



Code

File: UniProjectOne.java

```
package uniprojectone;

public class UniProjectOne {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

Helper.saveLogAs("edu.bridgeport.uniprojectone.log");

        Helper.outputBookName();
        Helper.outputNameObsessively();
        Helper.outputBrokenPoem();
        Helper.outputAbeSpeech();
        Helper.outputFace();
    }
}
```

File: Helper.java

```
package uniprojectone;

import java.io.InputStream;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.util.logging.Logger;
import java.util.logging.Level;
import java.util.logging.FileHandler;

public class Helper {
    private final static Logger LOGGER =
Logger.getLogger(Helper.class .getName());

    /**
     * @param fileLocation location to save log file
     */
    public static void saveLogAs(String fileLocation){
        saveLogAs(fileLocation, true);
    }

    /**
     * @param fileLocation location to save log file
     * @param appendFile is it appending or overwriting?
     */
    public static void saveLogAs(String fileLocation,
boolean appendFile){
        try {
            FileHandler logFile = new
FileHandler(fileLocation, appendFile);
            LOGGER.addHandler(logFile);
        } catch(IOException exception){
            println("WARNING: Unable to open log file!");
            println(exception.toString());
        }
    }

    /**
     * @param text to be printed without a break line
     */
    private static void print(String text) {
        System.out.print(text);
    }
```

```
    /**
     * @param text string to output to console, followed by
a break line
     */
    private static void println(String text) {
        System.out.println(text);
    }

    /**
     * @param resourceLocation is string of file to load,
ex: "resources/README"
     */
    public static void outputResource(String
resourceLocation) throws FileNotFoundException {
        InputStream inputStream =
UniProjectOne.class.getResourceAsStream(resourceLocation);
        if(inputStream == null){
            throw new FileNotFoundException("Resource '" +
resourceLocation + "' could not be found.");
        }
        try {
            println(inputStreamToString(inputStream));
        } catch(IOException exception) {
            LOGGER.log(Level.SEVERE, "Access denied (file)
(403)",exception);
            println("The following error was raised while
trying to access: " + resourceLocation);
            println(exception.toString());
        }
    }

    /**
     * @param in InputStream that you want returned as a
String
     * @return String value of in
     */
    public static String inputStreamToString(InputStream
in) throws IOException {
        StringBuilder output = new StringBuilder();
        byte[] b = new byte[4096];
        for (int n; (n = in.read(b)) != -1;) {
            output.append(new String(b, 0, n));
        }
        return output.toString();
    }
```

```
    public static void outputBookName() {
        println("Introduction to Programming and Problem
Solving with Java");
    }

    public static void outputNameObsessively() {
        String firstName = "Mohammad";
        String lastName = "El-Abid";
        print(firstName + " " + lastName);
        println(firstName);
        println(lastName);
    }

    public static void outputBrokenPoem() {
        println("The moon\nis\nblue.");
    }

    public static void outputAbeSpeech() {
        println("a. Four score\n   and seven years ago");
        println("b. Four score\n   and seven\n   years
ago");
    }

    public static void outputFace() {
        try {
            outputResource("resources/face.txt");
        } catch(FileNotFoundException exception) {
            LOGGER.log(Level.SEVERE, "File not found
(404)", exception);
            println(exception.toString());
        }
    }
}
```

File: resources/face.txt

```
    *         *
  * * *       * * *
* * * * *#######* * * * *
  * * *       * * *
    *  @     @   *
    *    ^     *
     *   $$$$$$$   *
      *         *
        * * * * *
```

Conclusion

I've learnt that Java is a versatile and powerful language. That it has the ability to compile resources inside of the application itself, making both development and distribution much easier. I learnt that if you're outputting several strings, that the best thing would be making it a resource since it clutters the script and makes the source code ugly. Realistically this application would read a text file with all of the output already rendered and print that data.

I also learned to use a `StringBuilder` when reading from external outputs since you can not change a string once it's created, because they are static, and that you do not know if it's all on one line which would cause problems if you used `readLine()` because of the amount of characters it would load into RAM.

I began to feel the the main file was getting clustered, so I moved the functions into another class, `Helper`, and also added a `FileNotFoundException` should an invalid resource location be passed into `outputResource(resouceLocation)`. This forced me to add a try/catch block to the main file, so I then moved that block into `Talker.outputFace()` to keep the code organized and clean. I also added a very basic logger for debugging client's crashes (saves to a .log file where the application is ran).

Works Used

[How to Embed Resource Files Using Netbeans]