

COMP 305, Lab 2

Outline

The goal of this lab is to create a Java application that reads data from a .csv file and formats it as a table in a .html web   
page. We will use the grades.csv file available from [https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html.](https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html) This file   
contains a header row followed by sixteen data rows. Each data row consists of a first and last name, a student number, four test marks, a final mark, and a grade (as shown in screenshot below).

The web page will appear as follows:

The details of Steps 1-9 will be covered during the lab. (Only the key coding steps are shown below. The working code will be developed in the lab).

Step 1 - Project Setup

• Open VS Code and create a new Java project folder named with your last name.

• Create two new folders, src and res, inside the project folder.

• Create a App Java file in the src folder.

• Copy the grades.csv file into the res folder.

• Initialize a local git repository.

• Commit the initial project files to the local repo with “Initial commit” message. ----> IMPORTANT

Step 2 - Open .csv File

This step will require a Scanner object that opens a File object that gives access to the grades.csv file:

**Scanner fileScanner = null;**

**fileScanner = new Scanner(new File("res/grades.csv"));**

Step 3 - Read Header Line from .csv File

After the file has been opened, use the Scanner to read the first line of input

String headerLine = fileScanner.nextLine();

Step 4 - Separate Header Line into Fields

After the header line has been read into a string, it must be cleaned up (extra quotation marks removed) and separated   
into fields.

headerLine = headerLine.replaceAll("\"", "");// replace all quotes with empty string.

String[] headers = headerLine.split(","); // break line (at commas) into array of field names

Step 5 - Open Output File

Create the output (.html) file in the res folder.

FileWriter fileWriter = new FileWriter ("res/grades.html");

Step 6 - Add try/catch Block

The above file operations will potentially throw IOExceptions, so we need to wrap some of the code inside a try block.

try {

fileWriter = new FileWriter("res/grades.html")

fileScanner = new Scanner(new File("res/grades.csv"));   
…

}

catch { …

}

Step 7 - Format Field Names as <thх Elements and Write to Output File

At this point we have an array of field names (from the grades.csv file) that we can format as the top row of an HTML table. We can use a StringBuilder to create the table and the row.

StringBuilder builder = new StringBuilder();

builder.append("<table>").append("\n");builder.append("<tr>");

for (int i = 0; i < headers.length; i++) {

builder.append("<th>").append(headers[i]).append("</th>").append("\n");

}

builder.append("</tr>").append("\n");

**builder.append("</table>");**

Step 8 - Write String to Output File

fileWriter.append(builder.toString()); fileWriter.close();

Step 9 - View the grades.html File in a Browser and Commit to Repo

At this point you can run the application. This will create a new grades.html file in the res folder. You can open the file in the VS Code editor to verify that it contains an HTML fragment consisting of a <table> with one <tr> with nine <td> elements that contain the field names from the first line of the grades.csv file. You can also open the grades.html file in a browser and it will show a row containing the field names.

If the content of the grades.html file is correct, commit your files to the local repo.    IMPORTANT

Step 10- Add Code to Format the Remaining Lines of Input File

There are 16 more lines of data to be read from the grades.csv file. 15 of these lines contain a correctly formatted set of   
grades (one line is formatted incorrectly). Each of these correctly formatted lines can be separated into an array of field   
names using code similar to that used in Step 4. Each array of field names can be reformatted as a table row of <td>   
elements. The incorrectly formatted line (missing commas) should be detected and omitted from the output file.

The code required to read the remaining lines from the input file is:

while (fileScanner.hasNextLine()) {

String line = fileScanner.nextLine();   
…

}

Step 11 - Add Styling to the table

Important: Make sure you have committed your repo

Step 10 (and beyond) will be continued during Lab 3.