Lab 10 – File Input

For this lab, you will read from a file.

Part 1

Find the *BlueJ* project folder named `Lab10_Start` in the `Resource` folder (or download and decompress `Lab10_Start.zip` on Blackboard). Copy this to your home directory. Rename your copy to `Lab10_lastname_firstname`. This project contains a program which you will use for the first part of the lab.

Open the `Part1` class in the editor. This is a working program that calculates the average of the student grades. The data is in a text file called `data1.txt`.

A. Observations

1. Open the data file `data1.txt` in Notepad. Notice the format of the data in the file: the first line is the number of students; the remaining lines alternate between being a student's full name and a grade. Note that each data item is on a separate line – the reason for this will become clear as we go along. Close the file and go back to *BlueJ*.

2. Notice the use of the `try` and `catch` blocks to capture any errors produced by opening the file.

3. Look at the next to the last line. This statement closes the input file. Make it a habit to close all files before exiting the program.

4. For the most part the code is identical to what would be done if the input were coming from the keyboard. The reading of a full name is different than anything we have done before. Previously when reading strings they have always been a word, an item without any internal whitespace. A full name has at least one space between the first and last names. Notice the second line in the for loop:

```
studentName = infile.nextLine();
```

The `nextLine` method reads an entire line of input as a String. This is done by reading each character on the line until a *newline* is encountered. The *newline* is **not** included in the input. Since *spaces* and *tabs* are normal characters they will be included in the input. In this way the full name can be read as one item. Notice that since the `nextLine` method terminates when it encounter the *newline* this data item **must** be on a line by itself, which explains the format of the data file. If anything else is on the line with the full name it will be considered part of the name.

5. Run the program. Look at the printed result. Notice that the full name is successfully read in and written out as a single item.

6. Return to the editor and observe the first line in the for loop:

```
infile.nextLine();
```

This should seem a little strange – a line of data is being read but nothing is being done with it!

Comment this line out, compile the program and re-run it. What happens?

The program should abort due to a `java.util.InputMismatchException` and in the editor the line after reading the full name should be highlighted in yellow. To determine what is happening we can use the debugger. To do this in the editor on the highlighted line move the mouse cursor into the far left column and click. This should result in a small red stop sign appearing in this column. This has set a breakpoint at this line and when this line is encountered during execution the debugger will be activate. Re-run the program. This should result in a new window appearing – the Debugger window.

There are five sub-windows in the Debugger, but the most interesting one is the bottom right window – the local variables window. It displays all of the active variables in your program – currently there are five active variables: the `Scanner`, `numberStudents`, `sum`, `count` and `studentName`. Based on the values in these variables, `count = 1` and `sum = 0.0` it should be evident that this is the first iteration of the loop. Notice that `numberStudents` is 5, so it was successfully read, check the data1.txt file if you are in doubt. Having stopped after the read of the full name the `studentName` variable should have a value, but observe that it holds "", this is an empty string. Somehow it has read nothing.

Terminate the debugger using the terminate button – the one with the big red X. In the editor copy the full name read, changing the variable name to studentName1 and add a declaration of this in the variable declaration section. Compile the program, set a breakpoint on the new read line and re-run the program. The debugger should stop with the same situation as before. Execute the new read using the Step button. What are the contents of the studentName1 variable?

They should be the name "John Smith". So why did the first read result in an empty string?

This is because the data file contains *newline* characters at the end of every line. The item before the full name is an integer which is read using the `nextInt` method. This method stops reading an integer when it encounters whitespace, i.e. the *newline* and the terminating character is left in the file. Thus, the next read using the `nextLine` terminates when it encounters a newline, which is immediately so nothing is read! Notice that the second string read we added was successful, so while the *newline* is not included in the String it is removed from the file by `nextLine`.

The moral of the story is: when using mixed reads, i.e. numeric and String, the programmer must account for *newline* characters. This is the reason for the dummy `nextLine` as the first line of the for loop.

Lab 10 – File Input

Part 2

The above program reads in the number of students. However, it is common for the number of elements in a file to be unknown. This is because the only time the amount of data in the file is known is after it has been put into the file. For this situation, we need to detect that the "end of file" has been reached.

Open the program `Part2`. This is the same program as `Part1` and you need to make the following changes:

1. Change the name of the input file from `data1.txt` to `data2.txt`. This file is in the project directory and you should look at it to verify that the number of students has been removed.

2. Remove:
   - the declaration of the `count` variable
   - the line that reads the number of students from the file, but you should initialize the `numberStudents` to zero, as we need to count the number of students to compute the class average.

3. At the bottom of the loop add a line to increment the `numberStudents`

4. Change the `for` loop to a `while` loop. The condition for the while loop should be when there is no more data to be read from the file. What would this condition be?

These changes should result in a working program. Clear the terminal window and run the program.

- Why does the program crash?
- Have you seen this error before?
- Is there any output in the terminal window? If so what is it?

Use the debugger to determine the cause of this logical error. Set a breakpoint on the line that is highlighted in the editor and re-run the program. Observe the values in the variables at the current time. Now step through the program one instruction at a time until you get the program to crash. After executing each variable altering instruction verify that the variable contents have been altered as expected.

Once you have determined the problem with the program fix it.

Lab 10 – File Input

After getting the program to work as expected change the program so that is asks the user for the filename and uses this file for input.

Submission

Submit your work in the usual way.