INF100 H17 Assignment 5

Due on Friday, October 27, 2017

Albin Severinson, Dag Haugland

Abstract

For this assignment you will reproduce the program I use to assign graders to the assignments you hand in each week. This will involve reading lists of students and graders from files and writing a new file where each student has been assigned a grader. As usual we will write one method in each problem and connect all of them at the end. It is therefore very important to make sure the method works properly before moving on to the next problem. We will only grade (pass/fail) the program for the final problem of the assignment. If you have questions specific to this assignment you should send them to me at `albin.severinson@uib.no`.

Submission

1. Add your program file to a `.zip` archive named `ex5_firstname_lastname.zip`. You do this by right clicking on the file ending with `.java` that contains your program code and choosing compress or add to archive (exactly what it is called varies by operating system and version). You should only add your final program, i.e., the one for problem 3. Remember that the file name must match the class name and that you therefore can not rename the `.java` file! Ask your group leader if you are having issues with this. Note that handing in the wrong file may cause you to fail the assignment!
2. If you complete the voluntary getting ahead part of the assignment, it should be handed in separately. If you complete this part you thus need to hand in two files (the `.java` file for problem 3 and the `.java` file for the getting ahead part). Put both files into the same `.zip` archive.
3. Log in to `https://mitt.uib.no` and press the box marked with "INF100" to enter the page for this course.
4. Press "Oppgåver" on the left. Next, select "Assignment 5".
5. Press "Lever oppgåve" and upload the `.zip` file you created in step 1.
6. We grade assignments in one week and semester assignments in two weeks.

General Guidelines

These are general guidelines that will (hopefully) help you maintain your sanity through your programming endeavours.

- Have your computer indent your code for you. You do this in DrJava by selecting all of your code, right clicking, and selecting `indent lines`.
- You should make a copy of all your code after completing each problem so that you can roll back to a previous version when you inadvertently break something. Even the developers at Google sometimes have to roll back.
- Set aside time to clean up your code. The number that gets bandied around on the internet is that the best programmers spend roughly half their time cleaning up and rewriting their code, i.e., you are halfway done when you get it working.
- You save time by making sure you properly understand your current program before moving on.
- You should probably be spending a significant part of your time on stackoverflow.
- Google is your friend when you want to figure out how a standard class or method works (`Scanner`, say). For example, searching for "java Scanner" gives you its Java docs page as the first result.
- When in doubt, look at the Google Java style guide to figure out how to write something.

Problem 1

Consider that you are a PhD student tasked with assigning a grader to each of the about 400 assignments that have been handed in. It is currently 16:45 on a Friday. Do you spend an hour manually assigning graders to assignments? The much better strategy is, of course, to write a short program that does the work for you!

For this assignment you have been given two text files, `students.csv` and `graders.csv`. Both files are lists of names with one name on each line. For example,

```
Stian
Thomas
Anders
Bendik
Rikke
Aziz
Tora
Salar
Kristian
Emilia
Maren
Joakim
Luai
```

Write a method that takes a filename as its argument and returns the list of names stored as an `ArrayList<String>`.

```java
public static ArrayList<String> namesFromFilename(String filename)
    throws FileNotFoundException {

}
```

Remember that you need to import `java.io.FileNotFoundException`. Furthermore, you should catch the exception and print it to the terminal in your main method.

Call this method from your `main()` method with `filename="students.csv"` and `filename="graders.csv"` to load the students and graders respectively. Print both lists of names to the terminal to verify that the method works as intended. Note that when printing an `ArrayList` with `System.out.println()` it does not print a newline between each of the elements. Compare the printout with the file contents to verify that you successfully printed all names to the terminal (there is no need to check all 400 names though!). After making sure that the method works as intended you may remove the printouts.

Note that the files you have been given hold more names than this. We will use these shorter examples to explain how the code should work. You should run the code on the longer files that came with the assignment.

Problem 2

The next step is to assign graders to students. Write a method that takes the lists of students and graders as arguments and returns a single `ArrayList` where each element has the format `student, grader`. The students should be split as evenly as possible among the graders. For example, you may assign 7 students to one grader and 6 to another, but not 8 students to one grader and 5 to another. How to achieve this is up to you. If, for example, the list of students is

```
Stian
Thomas
Anders
Bendik
Rikke
Aziz
Tora
Salar
Kristian
Emilia
Maren
Joakim
Luai
```

and the list of graders is

```
Vemund
Sondre
```

the returned `ArrayList` should have format

```
Stian, Vemund
Thomas, Vemund
Anders, Vemund
Bendik, Vemund
Rikke, Vemund
Aziz, Vemund
Tora, Vemund
Salar, Sondre
Kristian, Sondre
Emilia, Sondre
Maren, Sondre
Joakim, Sondre
Luai, Sondre
```

Furthermore, the method should print the number of students assigned to each grader to the terminal. In this case the method would write the following to the terminal:

```
Assigned 7 students to Vemund
Assigned 6 students to Sondre
```

Note that because the number of students was not divisible by 2, Vemund was assigned 7 students and Sondre 6. Assigning 6 students to Vemund and 7 to Sondre would also be a valid solution. The method you write should have signature

```java
public static ArrayList<String> assignGraders(ArrayList<String> students,
                                              ArrayList<String> graders) {


}
```

Problem 3

The final step is to write the `ArrayList` returned from `assignGraders()` to a new file. Specifically, write a method that creates a new file and prints each element of the provided `ArrayList` to a separate line. The method should have the following signature

```java
public static void writeAssignment(ArrayList<String> assignment,
                                   String filename)
        throws FileNotFoundException {

}
```

The resulting file should look like the example from the previous problem. Remember that you need to flush the writer before returning from the method. The filename is arbitrary. At this point, the following tasks should be performed by your main method:

1. Get `ArrayList` of students using `namesFromFilename()`.
2. Get `ArrayList` of graders using `namesFromFilename()`.
3. Use `assignGraders()` to create an assignment `ArrayList`.
4. Write the assignment to a text file with `writeAssignment`.

Getting Ahead

This assignment illustrates the use of programming as a personal productivity tool. The piece of software that you just wrote solves a real problem that I have and it saves me a significant chunk of time every week. I am sure there will be plenty of opportunity for similar optimizations throughout your own life!

For the getting ahead part of this assignment we will look at objects. Objects is an efficient way of representing something (such as a student) that has various properties (such as a name). Here is a primer https://www.youtube.com/watch?v=MK2SMJZbUmU. This is what people mean when they talk about object oriented programming. Interestingly, this programming paradigm was invented by a Norwegian computer scientist (see https://en.wikipedia.org/wiki/Simula)!

This part is meant as preparation for the coming course material and will not be graded. However, we will still look at it and comment it! Furthermore, you never have to catch up if you are always getting ahead. This part should be handed in separately. If you complete this part you thus need to hand in two files (the `.java` file for problem 3 and the `.java` file for this part). Put both files into the same `.zip` archive.

Write a class `Student` to represent students. The constructor should take the name of the student as its single argument. Furthermore, the class should have an instance method `assignGrader()` that takes the name of the grader as its single argument and stores the name of the grader within the `Student` object. Add a `toString()` method to `Student` (see https://www.youtube.com/watch?v=l0N6WvIVoUI for an explanation) as well. Now, recreate the functionality of the above program using lists of `Student` objects rather than of strings. Use the `toString()` method you wrote when writing the results to file. If you can do this you are already quite advanced!