

CMPS 12a Lab Assignment 9 - Draft  
David Helmbold, Winter 2004  
Due by 11:59 pm Wednesday March 10

**Introduction:** This program assignment will give you practice with basic Java I/O operations and classes. We suggest that you do following steps:

- Understand the problem and write the specification.
- Design the program by listing the objects, variables and operations.
- Write the program bottom-up, testing and redesigning if needed.

Submit your working program as described below. The program will have at least two classes, one called `Course` and another class `Student`.

This is a Major Lab, so work only with your designated partner on this assignment. Feel free to consult with the tutors and TAs. If you want to discuss Java details with another person, use a different problem (perhaps from one of the text's exercises) for your discussion. Under no circumstances you should take, give, borrow, lend, or transfer Java program code related to this assignment to a student other than your partner.

**The problem:** You are asked to design a program for reading student scores in a course, calculating class statistics, and generating reports.

**Getting ready:** Create a new directory called `hw9` or `assignment9` as a subdirectory of your `cs12a` directory. For this assignment you are required to use standard Java methods and classes for I/O operations.

**IMPORTANT:** do NOT use `tio` package to read the input.

**The specification:** Your program must create a class `Student` to represent each student's information.

Each instance of the `Student` class must contain the following information:

- the student's name (a `LastName` and a `FirstName`);
- midterm score;
- final score;
- scores for five assignments.

Each score is an integer value between 0 and 100.

The `Student` class should also provide:

- a constructor for a `Student` object with the fields above as parameters;
- a method for retrieving the weighted average of the 5 assignments;
- a method for retrieving the student's total score.

The total score is a weighted average of the midterm (20%), the final (30%) and the assignments (10% each).

Your program should read the student information (names and scores) for a course from a file called `students.txt`. The format of the input will be:

```
NumStudents
midterm final hw1 hw2 hw3 hw4 hw5 LastName, FirstName
midterm final hw1 hw2 hw3 hw4 hw5 LastName, FirstName
midterm final hw1 hw2 hw3 hw4 hw5 LastName, FirstName
...
```

where `NumStudents` is an integer that specifies the number of students in the class. It is followed by `NumStudents` lines of student information, each of which contains 7 integers followed by the last and first names of the student. A simple way of reading information from a file is given in `FileIO.java`, one of the sample programs off the class webpage.

Your program must first display the information read from the file. After reading in the information for all the students in the class, your program must generate the following output (see the example run):

- average of the total scores of the class;
- a numerically sorted list of the class (in decreasing order by the total score of each student).

To start, open (create) in your favorite editor (`pico`, `emacs`, `vi` etc.) two new files `Course.java` and `Student.java`. First lines of your programs should always have the file name and the names and computer accounts of the program authors in comments.

```
// Course.java
// Author: Firstname Lastname account
// Author: Firstname Lastname account
// Date:
// lab9 for the class cmps12a.
// Short description what this program does.
```

**The Submission:** To submit this assignment run the `submit` command out of your working directory.

```
submit cmps012a-dh.w04 hw9 Course.java Student.java
```

Submit only your Java source code. Do not submit Java `.class` files. You can submit multiple times, only last version will be saved and graded. To see your files use:

```
peek cmps012a-dh.w04 hw9
```

**Grading:** This program will be graded on a 8 point scale:

- 3 points for human readability - good variable names, good comments, consistent and meaningful indentation;
- 3 points for correctly solving the problem;
- 2 points for efficiently splitting the code into functions and classes.
- 1 extra-credit point for successfully implementing the optional part.

Programs that do not make a substantial effort towards solving the problem will earn fewer points even if they are well documented and compile cleanly. Programs using `tio` (instead of standard Java IO) will be penalized 2 points.

The optional part is to list the class a second time sorted in alphabetical order by last name (see the example run).

**Example Run:** Feel free to modify the dialog so long as the essential usage and statistics information is clearly presented.

```
> javac Course.java
> java Course
```

```
INFORMATION read from students.txt
```

```
=====
```

```
4
79 87 70 80 40 90 95 Red, John
20 30 45 12 46 39 40 Orange, Maria
40 50 20 50 30 20 10 White, Jim
49 68 30 40 10 80 90 Brown, Ana
```

```
NUMERICAL LIST
```

```
=====
```

```
Final Score  Name
-----
      81.15    Red, John
      57.2     Brown, Ana
      36.8     White, Jim
      32.42    Orange, Maria
```

```
CLASS AVERAGE: 51.89250000000001
```

For the extra credit point, you must follow this output with a nicely formatted (i.e. everything lined up) listing of the students and all their scores sorted alphabetically by last name, like that shown below.

```
ALPHABETICAL LIST
```

```
=====
```

```
Final Score  Name                mid fin hw1 hw2 hw3 hw4 hw5
-----
      57.2     Brown, Ana           49 68 30 40 10 80 90
      32.42     Orange, Maria        20 30 45 12 46 39 40
      81.15     Red, John            79 87 70 80 40 90 95
      36.8      White, Jim           40 50 20 50 30 20 10
```