

CSE 142, Summer 2008

Programming Assignment #4: Grades (20 points)

Due Tuesday, July 22, 2008, 11:00 PM

This interactive program focuses on `if/else` statements and returning values. Turn in a Java class `Grades` in a file `Grades.java`. You will use a `Scanner` for console input, so you must `import java.util.*;` in your code.

The program uses a student's grades on homework, a midterm exam, and a final exam to compute an overall course grade as a weighted average. The student's points earned in each category are divided by the total possible points for that category and multiplied by the category's weight. The program either computes the student's course grade (if the student has taken the final exam) or tells the student what score he/she needs to earn on the final to achieve a certain course grade.

Below is an example log of execution from the program. Your program's output should match these examples exactly given the same input. (Be mindful of spacing, such as around input prompts and between output sections.) It is important to note that this program behaves differently depending on the user input. Look at the other example logs on the course web site and on the next page to get more examples of the program's behavior.

```
<< your introduction message here >>
```

```
Exam weights? 20 30
```

```
Homework (weight 50):
```

```
Number of assignments? 3
```

```
Assignment 1 score and max? 14 15
```

```
Assignment 2 score and max? 17 20
```

```
Assignment 3 score and max? 19 25
```

```
Sections attended? 5
```

```
Section points = 15 / 20
```

```
Total points = 65 / 80
```

```
Weighted score = 40.6
```

```
Exam (weight 20):
```

```
Score? 78
```

```
Curve? 3
```

```
Total points = 81 / 100
```

```
Weighted score = 16.2
```

```
Have you taken the final? (1=yes, 2=no) 1
```

```
Exam (weight 30):
```

```
Score? 95
```

```
Curve? 10
```

```
Total points = 100 / 100
```

```
Weighted score = 30.0
```

```
Course grade = 86.8
```

The program begins with an introduction message of your choosing, preferably one that greets the user and explains the program. It asks the user for weights of the two exams and subtracts their sum from 100 to get the homework weight.

Then the user enters how many homework assignments were given. For each assignment, the user enters his/her score and the max score. Use a cumulative sum to compute homework scores; see textbook section 4.1.

Part of the homework score comes from sections the student attended. Each is worth 3 points, up to a maximum of 20.

The user next enters his/her midterm exam score. The maximum exam score is 100. If the exam was curved, these points are added to the user's midterm score.

The program can be run before or after the final exam, so first the user enters whether it has been taken; 1 means yes, 2 means no. If it has, the user enters his/her score and the curve, and the student's grade is shown. If not, it computes the final exam score needed to earn a particular course grade.

If the student can achieve the desired course grade without taking the final (in other words, if a final exam score ≤ 0 is needed), the program shows 0.0 as the needed score. If the needed score is above 100, the program indicates this and shows the highest course grade the student can get (if the student earns 100 on the final exam).

In the log of execution shown, the student earned 14/15, 17/20, and 19/25 on homework assignments. The student attended 5 sections, earning 15/20 section points for doing so. The student received a midterm exam score of 78, curved by 3 to 81. The student has taken the final exam and earned a score of 95; it was curved by 10, but exam scores are capped at 100, so the student gets a 100. The following calculations produce the student's grade from the above log:

$$\text{Grade} = \text{WeightedHomeworkScore} + \text{WeightedMidtermScore} + \text{WeightedFinalExamScore}$$

$$\text{Grade} = \left(\frac{14 + 17 + 19 + (5 \times 3)}{15 + 20 + 25 + 20} \times 50 \right) + \left(\frac{78 + 3}{100} \times 20 \right) + \left(\frac{100}{100} \times 30 \right)$$

$$\text{Grade} = 40.6 + 16.2 + 30.0$$

$$\text{Grade} = 86.8$$

Note that in Java, `81/100` (the midterm score) would evaluate to 0, but the value desired is 0.81.

Implementation Details:

You may assume that the user enters valid input. When prompted to enter a value, the user will enter an integer, and one in a proper range. The user will enter a number of homework assignments ≥ 1 , and the sum of the two exam weights will be no more than 100. The weight of a particular category will be non-negative but could be 0. Curves will be ≥ 0 .

You should handle the following two special cases of user input.

- A student can receive extra credit on an assignment; for example, 22 / 20 is a legal score. But the total homework points are capped at the max possible. If a student receives a total of 63 / 60, your program should cap this to 60 / 60.
- The maximum score for an exam is 100. If the curved exam score exceeds 100, a score of 100 is used.

All weighted scores and grades printed by the program are shown with no more than 1 digit after the decimal point. There are a few ways to do this. You can multiply a number by 10, round it using `Math.round`, then divide it by 10. Or you can use `System.out.printf`. The following code prints a variable `x`, rounded to the nearest tenth:

```
double x = 1.2345;
System.out.printf("x is around %.1f in size.\n", x); // 1.2
```

Additional Output:

The following additional logs demonstrate the program's behavior when the user has not taken the final exam.

Log #2

<< your introduction message here >>

```
Exam weights? 15 25
Homework (weight 60):
Number of assignments? 5
Assignment 1 score and max? 9 10
Assignment 2 score and max? 15 15
Assignment 3 score and max? 15 20
Assignment 4 score and max? 30 20
Assignment 5 score and max? 20 25
Sections attended? 2
Section points = 6 / 20
Total points = 95 / 110
Weighted score = 51.8

Exam (weight 15):
Score? 61
Curve? 12
Total points = 73 / 100
Weighted score = 11.0

Have you taken the final? (1=yes, 2=no) 2
Desired percentage? 80
Score needed = 68.9
```

Log #3

<< your introduction message here >>

```
Exam weights? 33 34
Homework (weight 33):
Number of assignments? 2
Assignment 1 score and max? 18 50
Assignment 2 score and max? 15 25
Sections attended? 8
Section points = 20 / 20
Total points = 53 / 95
Weighted score = 18.4

Exam (weight 33):
Score? 56
Curve? 0
Total points = 56 / 100
Weighted score = 18.5

Have you taken the final? (1=yes, 2=no) 2
Desired percentage? 75
Score needed = 112.1
Sorry, it is impossible to earn this score.
The highest percentage you can get is 70.9.
```

The needed final exam score is computed using a formula such as the following, using the data from Log #2:

$$\text{NeededFinalScore} = 100 * \frac{\text{DesiredCourseGrade} - \text{WeightedHomeworkScore} - \text{WeightedMidtermScore}}{\text{FinalExamWeight}}$$

$$\text{NeededFinalScore} = 100 * \frac{80 - 51.8 - 11.0}{25}$$

$$\text{NeededFinalScore} = 68.9$$

Also note Log #3's output for the case where the needed final exam score is impossible to earn (> 100).

Suggested Development Strategy and Hints:

We offer the following suggestions for approaching this problem:

- Tackle each part of the program (homework, midterm, final exam) one at a time, rather than writing the entire program at once without testing it. Write a bit of code, get the program to compile, and test what you have so far.
- Many students encounter "cannot find symbol" compiler errors related to scope, parameters, and return values. Avoid common mistakes such as forgetting to pass / return a needed value, forgetting to store a returned value into an appropriate variable, and referring to a variable by the wrong name.
- Related to the previous bullet, it is very important that you write small amounts of code and then compile and test this code to make sure it works properly. If you try to write large amounts of code without attempting to compile it, you will encounter a large list of compiler errors and/or bugs and will have great difficulty fixing the program.
- To compute homework scores, you will need to cumulatively sum not only the total points the student has earned, but also the total points possible on all homework assignments. See textbook section 4.1 about cumulative sums.
- To read two input values on the same line, prompt the user once, then call the appropriate `next__` method on the `Scanner` twice in a row to read the two values.
- If you are getting scores of 0 regardless of what data the user types, you may have a problem with integer division. See Chapter 2 about types `int` and `double` and how to avoid integer division problems.
- The final exam code can be tricky, so initially make your program just assume that the student has taken the final. Once this works properly, then add the code for the case where the user hasn't yet taken the final exam.
- Worry about edge cases and cleanup last (such as extra credit, exam scores over 100, and rounding).
- Use `Math.max` and `Math.min` to constrain numbers to within a particular bound.

Stylistic Guidelines:

For this assignment you are limited to Java features from Ch. 1-4. A major part of this assignment is demonstrating that you understand parameters and return values. Use static methods that accept parameters and return values as appropriate for structure and to eliminate redundancy. **For full credit, use at least 3 non-trivial methods other than `main`.**

Unlike on previous assignments, you *can* have some `println` statements in your `main` method on this program. However, your `main` method should still be short and should represent a summary of the overall program; the majority of the behavior should come from other methods. To fully achieve this goal, some of your methods will need to return values. Each method should perform a coherent task and should not do too large a share of the overall work. You will not receive full credit if your solution has lengthy "chains" of method calls, where each method calls the next, no values are returned, and control does not come back to `main`. For reference, our solution is roughly 110 lines long, with 5 methods other than `main`, though you do not need to match this.

When handling numeric data, you are expected to choose appropriately between types `int` and `double`. For example, you may lose points if you always use type `double` for your variables even where type `int` would be more appropriate.

Some of your code will use conditional execution with `if` and `if/else` statements. Part of your grade will come from using these statements properly. Review book sections 4.2-4.3 about nested `if/else` statements and factoring them.

Give meaningful names to methods and variables, and use proper indentation and whitespace. Follow Java's naming standards as specified in Chapter 1. Localize variables when possible; declare them in the smallest scope needed. Include meaningful comment headers at the top of your program and at the start of each method.