University of Washington
Computer Science & Engineering 142 (Computer Programming I), Summer 2005

Programming Assignment #4 (Grades)
Due: Tuesday, 7/19/2005, 6:00 PM

Problem Description:

This assignment will give you practice with interactive programs, `if/else` statements, and methods that return values. Turn in a Java class named `Grades` in a file named `Grades.java`. You will be using a `Scanner` object for console input, so you will need to download `Scanner.java` from the course web site and place it in the same folder as your program.

This program uses a student's grades on homework, a midterm exam, and a final exam to compute an overall course grade. The course grade is a weighted average. To compute a weighted average, the student's point scores in each category are divided by the total points for that category, then multiplied by that category's weight. (The sum of all categories' weights should be 100.)

$$Grade = WeightedHomeworkScore + WeightedMidtermScore + WeightedFinalExamScore$$

$$Grade = \left(\frac{HomeworkEarned}{HomeworkPossible} \times HomeworkWeight\right) + \left(\frac{MidtermEarned}{MidtermPossible} \times MidtermWeight\right) + \left(\frac{FinalEarned}{FinalPossible} \times FinalWeight\right)$$

Example: A course has 50% weight for homework, 20% weight for its midterm, and 30% weight for its final. There are 3 homework assignments worth 15, 20, and 25 points respectively. The student received homework scores of 10, 16, and 19, a midterm score of 81, and a final exam score of 73 (which was curved by +5 points to make a curved score of 78).

$$Grade = WeightedHomeworkScore + WeightedMidtermScore + WeightedFinalExamScore$$

$$Grade = \left(\frac{HomeworkEarned}{HomeworkPossible} \times HomeworkWeight\right) + \left(\frac{MidtermEarned}{MidtermPossible} \times MidtermWeight\right) + \left(\frac{FinalEarned}{FinalPossible} \times FinalWeight\right)$$

$$Grade = \left(\frac{10+16+19}{15+20+25} \times 50\right) + \left(\frac{81}{100} \times 20\right) + \left(\frac{73+5}{100} \times 30\right)$$

$$Grade = 37.5 + 16.2 + 23.4$$

$$Grade = 77.1$$

Note that the above math is written as real math and not Java math; in Java, an integer expression such as 81/100 would evaluate to 0, but above the value intended is 0.81.

Program Behavior:

The following section describes the program's behavior in detail. You can also get a good idea of the program's behavior by looking at the logs of execution that will follow, but you should read this section completely to make sure you see all of the cases and behaviors your program should have.

The program asks the user to enter his/her grades on homework, an optional midterm exam, and a final exam. First, the program prints a header message describing itself.

Second, the program asks the user for homework information. The user enters the weight of the homework assignments, which can be any number between 0 and 100. The user enters how many homework assignments were completed. For each of these assignments, the user enters his/her score along with the maximum possible score, separated by spaces. Your code should work for any number of assignments greater than or equal to 1.

Third, the program asks the user for midterm exam information. The user enters its weight from 0 to 100 and his/her score. (The user does not enter the maximum score for the midterm, because this is assumed to be 100.) The user is asked whether the midterm was curved; a response of 1 means yes, and 2 means no. If there was a curve, the user is asked how many points were added. These points are added to the user's midterm score. **No exam's score can be above 100, so if the curve would have made the user's score exceed 100, a score of 100 is used.**

Fourth, the program asks the user for final exam information. The program can be run before or after the final exam, so first the user enters whether the final exam has been completed; a response of 1 means yes, and 2 means no.

If the final exam has been completed, the program will help the user show his/her overall course grade. The user enters the exam weight and his/her score on the final exam. The user is asked whether the final exam was curved; a response of 1 means yes, and 2 means no. If there was a curve, the user is asked how many points were added. These points are added to the user's midterm score. No exam's score can be above 100, so if the curve would have made the user's score exceed 100, a score of 100 is used. Lastly, the user's overall course grade is printed.

If the final exam has not yet been completed, the program will help the user see what final exam score he/she needs to earn a particular overall grade in the course. The final exam weight and the user's desired course grade are entered. The program computes and shows what score the student needs on the final to achieve the desired course grade.

If the student can achieve the desired course grade without taking the final exam (in other words, if a final exam score of 0 or less is required), the program shows 0.0 as the needed final exam score. If the required score for the student to get the desired course grade is greater than 100, it should still be printed as normal, and then the user should receive a message indicating that the desired score cannot be achieved, and a message showing the highest course grade that the student can get (which is the grade that would result if the student earns 100 on the final exam). See the provided logs of execution on the course web site for an example of this output.

The following logs of execution indicate the exact format of the output that you should reproduce. Your program's output should match these samples exactly when the same input is typed. Please note that there are some blank lines between sections of output and that some lines of output are indented by four spaces. Also note that input values typed by the user appear on the same line as the corresponding prompt message.

First log of execution (user input underlined)

```
This program accepts your homework and exam
scores as input, and computes your grade in
the course or indicates what grade you need
to earn on the final exam.

Homework:
    What is its weight (0-100)? 50
    How many homework assignments were there? 3
    Homework 1 score and max score: 14 15
    Homework 2 score and max score: 18 20
    Homework 3 score and max score: 19 25
    Weighted homework score: 42.5

Midterm exam:
    What is its weight (0-100)? 20
    Exam score: 81
    Was there a curve? (1 for yes, 2 for no) 2
    Weighted exam score: 16.2

Final exam:
    Have you taken the final exam yet? (1 for yes, 2 for no) 2
    What is its weight (0-100)? 30
    What percentage would you like to earn in the course? 80

    You need a score of 71.0 on the final exam.
```

Second log of execution (user input underlined)

```
This program accepts your homework and exam
scores as input, and computes your grade in
the course or indicates what grade you need
to earn on the final exam.

Homework:
    What is its weight (0-100)? 40
    How many homework assignments were there? 4
    Homework 1 score and max score: 21 30
    Homework 2 score and max score: 11 20
    Homework 3 score and max score: 28 50
    Homework 4 score and max score: 5 10
    Weighted homework score: 23.64

Midterm exam:
    What is its weight (0-100)? 30
    Exam score: 95
    Was there a curve? (1 for yes, 2 for no) 1
    How much was the curve? 10
    Weighted exam score: 30.0

Final exam:
    Have you taken the final exam yet? (1 for yes, 2 for no) 1
    What is its weight (0-100)? 30
    Exam score: 63
    Was there a curve? (1 for yes, 2 for no) 1
    How much was the curve? 5
    Weighted exam score: 20.4

    Your course grade is 74.04
```

Third log of execution (user input underlined)

```
This program accepts your homework and exam
scores as input, and computes your grade in
the course or indicates what grade you need
to earn on the final exam.

Homework:
    What is its weight (0-100)? 50
    How many homework assignments were there? 2
    Homework 1 score and max score: 10 50
    Homework 2 score and max score: 12 25
    Weighted homework score: 14.67

Midterm exam:
    What is its weight (0-100)? 25
    Exam score: 56
    Was there a curve? (1 for yes, 2 for no) 2
    Weighted exam score: 14.0

Final exam:
    Have you taken the final exam yet? (1 for yes, 2 for no) 2
    What is its weight (0-100)? 25
    What percentage would you like to earn in the course? 90

    You need a score of 245.33 on the final exam.
    Sorry, it is impossible to achieve this percentage.
    The highest percentage you can get is 53.67.
```

You do not have to perform any error checking on user input. You may assume that the user types legal values of the proper type and in the appropriate range. For example, when prompted for a number, assume that the user does enter a number and not a String. When prompted for a number within an expected range, such as a weight or exam score between 0 and 100, the user will enter a valid number in that range and not a number outside the range. When prompted for the number of homework assignments, the user will enter a number no less than 1. The sum of the weights the user will enter will always be 100.

Notice that all real numbers output by the program are printed with no more than 2 digits after the decimal point. To achieve this, you may use code such as the following method to round a `double` value to the nearest hundredth:

```
// Returns the given double value rounded to the nearest hundredth.
public static double round2(double number) {
    return Math.round(number * 100.0) / 100.0;
}
```

Only round numbers as you are about to print them, such as:

```
System.out.println("Weighted exam score: " + round2(weightedExamScore));
```

Stylistic Guidelines:

For this assignment you are limited to the language features in Chapters 1 through 5; you are not allowed to use more advanced features to solve the problem. Please do not use Java features that are not covered in lecture or the textbook.

Structure your solution and eliminate redundancy in the output by using static methods that accept parameters and return values where appropriate. **For full credit, you must have at least 4 non-trivial methods other than `main` in your program.** In grading, we will examine your `main` method to make sure that it is short, and we will look at whether you have broken down the problem into several appropriate static methods that capture the structure of the task. To fully achieve this goal, some or all of your methods should return values to pass information back to their caller. Each method should perform a coherent task and should not do too large a share of the overall work. See a TA or the instructor if you are concerned about your structure and methods.

You are required to properly indent your code and will lose points if you make significant indentation mistakes. See section 2.5.3 of the book for an explanation and examples of proper indentation. You should also use whitespace to make your program more readable, such as between operators and their operands, between parameters, and blank lines between groups of statements or methods.

Give meaningful names to methods and variables in your code. Follow Java's naming standards about the format of ClassNames, methodAndVariableNames, and CONSTANT_NAMES. Localize variables whenever possible -- that is, declare them in the smallest scope in which they are needed.

Include a comment at the beginning of your program with basic information and a description of the program and include a comment at the start of each method.

Submission and Grading:

Name your file `Grades.java` and it them in electronically from the "Assignments" link on the course web page. You do not have to turn in `Scanner.java`. This assignment is worth 20 points total.

Part of your program's score will come from its "external correctness." External correctness measures whether your log of execution matches exactly what is expected, including identical prompting for user input. Occasionally Java's real-number arithmetic causes rounding, which may cause your output to deviate by very small amounts below the decimal point; this is allowed and will not cause a deduction.

The rest of your program's score will come from its "internal correctness." Internal correctness measures whether your source code follows the stylistic guidelines specified in this document. This includes using `if` and `if/else` statements to represent conditional execution, using `for` loops to capture repetition, representing the structure and redundancy of the program using appropriate parameterized static methods with return values as needed, commenting, naming identifiers, and indentation of your source code.