CSE 142, Autumn 2006
Programming Assignment #3: Grades (20 points)
Due: Tuesday, October 17, 2006, 2:00 PM

Program Description:

This assignment will give you practice with parameters, returning values, and interactive programs. Turn in a Java class named `Grades` in a file named `Grades.java`, which will be submitted electronically on the course web site. You will be using a `Scanner` object for console input, so you will need to `import java.util.*;` into your program.

This program uses a student's grades on homework and exams to compute an overall course grade. The following is an example log of execution of the program (user input is underlined):

```
This program accepts your homework scores and
scores from two exams as input and computes
your grade in the course.

Homework and Exam 1 weights? 50 20

Number of homework assignments? 3
Assignment 1 score and max score? 14 15
Assignment 2 score and max score? 16 20
Assignment 3 score and max score? 19 25
Number of sections attended? 6
Weighted score = 41.88

Exam 1 score? 81
Exam 1 curve? 0
Weighted score = 16.2

Exam 2 score? 73
Exam 2 curve? 5
Weighted score = 23.4

Course grade = 81.48
```

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 and multiplied by that category's weight.

Part of the homework score is determined by how many discussion sections the student attended. Each section is worth 3 points, up to a maximum of 20 possible points.

In the above execution, the course has 50% weight for homework, 20% weight for exam 1, and 30% weight for exam 2. There are 3 homework assignments worth 15, 20, and 25 points respectively. The student received homework scores of 14, 16, and 19; attended 6 sections; received an exam 1 score of 81; and earned an exam 2 score of 73 (which was curved by +5 points to make 78). The following calculations produce the student's course grade:

$$Grade = WeightedHomeworkScore + WeightedExam1Score + WeightedExam2Score$$

$$Grade = \left(\frac{14+16+19+(6\times 3)}{15+20+25+20}\times 50\right) + \left(\frac{81}{100}\times 20\right) + \left(\frac{73+5}{100}\times 30\right)$$

$$Grade = 41.88 + 16.2 + 23.4$$

$$Grade = 81.48$$

(Note that the preceding equations are 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 Details:

The program asks the user for the weights of the homework and exam 1. Using this information, the program can deduce the weight of exam 2 as 100 minus the other two weights.

In general, you may assume that the user enters valid input. For example, assume that the user enters a number of homework assignments no less than 1, and that the sum of category weights entered will be no more than 100. The weight of a particular category (homework, exam 1, or exam 2) will be non-negative but could be 0.

You should, however, handle the following two special cases of user input:

- A student might receive extra credit on a particular assignment, so for example 22 / 20 is a legal assignment score. But the overall homework points are capped at the maximum possible. For example if a student receives 63 total homework points out of a maximum of 60, the total homework score is treated as a 60 / 60.
- The maximum score for an exam is 100. If the curved exam score exceeds 100, a score of 100 is used.

Notice that all weighted scores and grades printed by the program are shown with no more than 2 digits after the decimal point. To achieve this, you may place the following method in your program and call it 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;
    }
```

See the provided logs of execution on the course web site for examples of expected output. Your program's output should match these examples exactly when the same input is typed. Please note that there are some blank lines between sections of output and that input values typed by the user appear on the same line as the corresponding prompt message.

The code to compute the student's homework scores requires you to use a cumulative sum. See the lecture slides and section 4.1 of the textbook for more information on this technique.

Stylistic Guidelines:

For this assignment you are limited to the language features in Chapters 1 through 3; you are not allowed to use more advanced features (such as `if`/`else` statements) to solve the problem. You should use the `Math.max` and `Math.min` methods to constrain numbers to a given range. 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 3 non-trivial methods other than `main` and `round2` in your program.** In grading, we will examine your `main` method to make sure that it is concise and represents a summary of the program's overall behavior. To fully achieve this goal, some 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.

When handling numeric data, you are expected to choose appropriately between types `int` and `double`.

You should also properly indent your code and use whitespace properly to make your program more readable. Give meaningful names to methods and variables in your code. Follow Java's naming and capitalization standards. Localize variables whenever possible; that is, declare them in the shortest possible scope.

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