CSE 142, Spring 2007
Programming Assignment #4: Grades (20 points)
Due: Tuesday, April 24, 2007, 4:00 PM

Program Description:

This assignment will give you practice with parameters, returning values, `if`/`else` 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 two exams to compute an overall course grade and minimum numerical grade (see syllabus for details). The following is one example log of execution of the program (user input is underlined).

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

Homework and Exam 1 weights? 50 20
Using weights of 50 20 30

Homework:
Number of assignments? 3
Assignment 1 score and max? 14 15
Assignment 2 score and max? 16 20
Assignment 3 score and max? 19 25
Sections attended? 4
Total points = 65 / 80
Weighted score = 40.63

Exam 1:
Score? 81
Curve? 0
Total points = 81 / 100
Weighted score = 16.2

Exam 2:
Score? 95
Curve? 10
Total points = 100 / 100
Weighted score = 30.0

Course grade = 86.83
You will get at least a 3.0
```

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 4 points, up to a maximum of 20 possible section points.

In the log of execution shown, 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, and attended 4 sections (earning 16 points for doing so). The student received an exam 1 score of 81. The student earned an exam 2 score of 95; the exam was curved by +10 points, but exam scores are capped at 100, so the student was given 100 for exam 2.

The following calculations produce the student's course grade from the above log of execution:

$$Grade = WeightedHomeworkScore + WeightedExam1Score + WeightedExam2Score$$

$$Grade = \left(\frac{14+16+19+(4\times 4)}{15+20+25+20}\times 50\right) + \left(\frac{81}{100}\times 20\right) + \left(\frac{100}{100}\times 30\right)$$

$$Grade = 40.63 + 16.2 + 30.0$$

$$Grade = 86.83$$

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.

The program behaves differently depending on the user input it receives; you should look at all of the example logs of execution on the course web site to get a more comprehensive example of the program's behavior.

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.

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 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 total points for homework are capped at the maximum possible. For example if a student receives 63 total points out of a maximum of 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.

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 type the following method into 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;
}
```

The following is an example usage of this method to print a variable named `x`:

```
System.out.println("The rounded value of x is " + round2(x));
```

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. If you use DrJava, copy your output from the "Console" pane (it is located right next to the "Interactions" pane) when using the Output Comparison Tool.

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

Stylistic Guidelines:

A major part of this assignment is demonstrating that you understand parameters and return values. Therefore, use static methods that accept parameters and return values where appropriate for structure and to eliminate redundancy. **For full credit, you must use at least 3 non-trivial methods other than `main` and `round2`.**

Unlike previous assignments, you can have `println` statements in your `main` method. However, your `main` method should still 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 back to their caller. Each method should perform a coherent task and should not do too large a share of the overall work. For reference, our solution is 112 lines long with 5 methods other than `main` and `round2`.

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

For this assignment you are limited to the language features in Chapters 1 through 4; you are not allowed to use more advanced features not covered in class or the textbook to solve the problem.

You should properly indent your code and use whitespace to make your program readable. Give meaningful names to methods and variables in your code. Follow Java's naming and capitalization standards as described in Section 1.2 of the book. Localize variables whenever possible; 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.