CS 1400 - Lab #3

Maximum Points: 10 pts.

Topics

- Familiarization with basic data types.
- Using the Scanner Class with input validation.
- Understanding order of operations.
- Getting familiar with the basic flow control in Java.

Use the following Coding Guidelines

1) Download the template file Lab3.java from Blackboard and fill-in-the-blanks to create your Java program.
2) Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
3) Keep identifiers to a reasonably short length.
4) Use uppercase for constants. Use upper camel case for classes. Use lower camel case for all other identifiers (variables, methods, objects).
5) Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches, and loops. Be consistent with the number of spaces or tabs that you use to indent.
6) Use white space to make your program more readable.
7) Use comments to explain how the parts of your program work.

Problem Description: Final Weighted Total Calculator

Design a program which takes 3 user inputs as homework grade, midterm exam grade, and final exam grade. Then, use the following formula to calculate the weighted total:

$$Total_{weighted} = \left(Exam_{final} / 200 \times 50\right) + (Exam_{midterm} \times 0.25) + (Homework \times 0.25)$$

If the weighted total is greater or equal to 50 (>=50), show *"Student PASSED the class"*. Otherwise, show *"Student FAILED the class"*.

Your program has to validate the input values. A homework grade and midterm exam score should be in the range [0, 100]. The final exam grade should be in the range [0, 200]. If an input is invalid, you have to let the user retry immediately until it is a valid input.

Note: To get full marks, you must use only one loop.

Sample Output

Sample Run 1 (The user passed the class):

```
Enter your HOMEWORK grade: 100

Enter your MIDTERM EXAM grade: 76

Enter your FINAL EXAM grade: 80

[INFO] Student's Weighted Total is 64.00

[INFO] Student PASSED the class
```

Sample Run 2 (The user failed the class):

```
Enter your HOMEWORK grade: 30

Enter your MIDTERM EXAM grade: 40

Enter your FINAL EXAM grade: 35

[INFO] Student's Weighted Total is 26.25

[INFO] Student FAILED the class
```

Sample Run 3 (Input validation, minimum requirement)

```
Enter your HOMEWORK grade: 100

Enter your MIDTERM EXAM grade: 76

Enter your FINAL EXAM grade: 8000
[ERR] Invalid input. A final grade should be in [0, 200].

Enter your FINAL EXAM grade: 40

[INFO] Student's Weighted Total is 54.00

[INFO] Student PASSED the class
```

A Challenge (not required, not part of the lab grade)

Limit the number of chances that the user can retry to 3. It means when one input is invalid, the user will have 3 extra times to retry. If a user has retried three times consecutively and fails, the program loop should terminate immediately and show the error message: *"You have retried 3 times. Please, restart your program."*. In total, a user can input four times in a sequence with the first three inputs invalid. The counter will be reset to 0 when an input is correct. Your program's output messages should change according to the user input states.

Sample Run for Challengers

```
Enter your HOMEWORK grade: 100

Enter your MIDTERM EXAM grade: 76

Enter your FINAL EXAM grade: 1000
[ERR] Invalid input. A final grade should be in [0,200].

Enter your FINAL EXAM grade (2 chance(s) left): -1293
[ERR] Invalid input. A final grade should be in [0,200].

Enter your FINAL EXAM grade (1 chance(s) left): 8023
[ERR] Invalid input. A final grade should be in [0,200].

Enter your FINAL EXAM grade (0 chance(s) left): 12342314
[ERR] Invalid input. A final grade should be in [0, 200].


[ERR] You have retried 3 times. Please restart your program.
```

Step 1: Get started with the header

Create a class called Lab3. Use the same setup for setting up your class and main method as you did in previous labs and assignments. Be sure to name your file Lab3.java.

At the beginning of each programming assignment you must have a comment block with the following information:

```
/*-------------------------------------------------------------
// AUTHOR: your name.
// FILENAME: title of the source file.
// SPECIFICATION: your own description of the program.
// FOR: CS 1400- Lab #3
// TIME SPENT: how long it took you to complete the assignment.
//-------------------------------------------------------------*/
```

Step 2: Declare variables you need

```
// This scanner is prepared for you to get inputs Scanner
scanner = new Scanner(System.in);
// Declare three variables for HW, midterm, and final exam grades
// -->
// Declare a loop control variable i
// -->
```

Since your program will ask the user for the homework score, midterm exam score, and final exam score. Please declare three **double** variables to save these input values.

In addition, you might need a variable to control your while loop. Please declare an **integer** i for this purpose. You can declare extra variables if you think they are needed.

Step 3: Use a while loop with the right condition

```
while (/* Put in the condition involving i */) {
}
```

We hope the program will restart automatically when the user gives you an invalid input. Intuitively, we should use a loop structure to make it restarts if a certain condition is not correct. Please, first make a while loop statement and one loop control variable i. Then design the loop condition you need. Refer the template file if you are not sure what to do.

Hint: It might be challenging to design such a condition sometimes. To help you think about what it should be, here is a simple logic we hope the program to follow:

1. Inside this while loop, the program asks three inputs from the user.
2. If any of the three inputs is not valid, the loop will restart and let the user input that number again (for example, if the midterm grade is invalid, let the user type the midterm grade again. Do NOT ask the user start from the homework grade.)
3. The while loop will stop when all inputs are done and valid.

Step 4: Use if-else statements to control input flow

```
if (i == 0) {
// Ask the user for homework grade
// -->
// ...
}
```

Inside the while loop, the program asks the user for the three grades. Since we are using only one while loop to do three inputs at the same time, we need a proper flow control to make the loop asks for the right input in different moments. To do so, please construct three if-else statements involving your loop variable i, specifically:

1. if i is 0, asks for the homework grade
2. if i is 1, asks for the midterm exam grade
3. if i is 2, asks for the final exam grade

Step 5: Use if-else statements to validate the input values

```
// Do input validation
// -->
if (/* the HW grade is not valid */) {
// Show the error message
// -->
} else {
// Update the loop variable
// -->
}
```

Following Step 4, inside each if-else statement the program asks for the user input. To ensure the user gives you a valid number, you have to design another if-else statement to validate the input. For your reference, the valid ranges are

1. [0, 100] for a homework grade
2. [0, 100] for a midterm exam grade
3. [0, 200] for a final exam grade

Hint: You should update the loop control variable i when any of these inputs is valid, so as to make your program approach the terminal condition of the loop.

Step 6: Calculate the weighted total and show the value

```
double weighted_total = ...;
```

After the while loop, please calculate the weighted total by the formula provided above. The result should be saved into another variable and showed in your console.

Step 7: Show the student passed or failed

```
if (/* ... */) {
// Print "the student PASSED the class."
} else {
// Print "the student FAILED the class."
}
```

Finally, according to the weighted total, the program determines whether the user passed the class or not. Please design another if-else statement to do so. The conditions are

1. if the weighted total is greater than or equal to 50, show "[INFO] The student PASSED the class"
2. Otherwise, show "[INFO] The student FAILED the class".

Step 8: Submit Your Lab3.java to Gradescope

Please submit **ONLY** the file Lab3.java to the "Lab 3" link on Gradescope. Make sure it is compiling and producing the expecting outputs. You are done.