Lab #2
CSE110 - Arizona State University

Topics

- Problem Solving
- Arithmetic Expressions
- Input/Output

Problem Description: Finding an Average

Your friend Jenny has a class that gives three tests. She would like you to write a program that will take the three test grades as input and tell her what her average test grade is. For this Lab you are required to write a program that will read three test grades from the user and then calculate and print the average of those three test grades.

Step 0: Getting Started

Create a class called `Lab2`. Use the same setup for setting up your class and main method as you did for the previous examples. Be sure to name your file `Lab2.java`.

Assignment/Lab Documentation

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: description of the program
// FOR: CSE 110- Lab #2
// TIME SPENT: how long it took you to complete the assignment
//-----------------------------------------------------------*/
```

Step 1: Setting up a Scanner for Input

Since you are required to read in the three test grades from the user, you will have to use a `Scanner` object. Follow the instructions in the sample video under Chapter 2 Lectures or in the book on page 49 to import the `Scanner` class from the `java.util` library and create a `Scanner` object to get input from the keyboard (`System.in`).

Step 2: Declaring Variables

Examining the problem, we see that we will need three inputs from the user. We will need variables to hold all of the inputs. For this Lab, let's assume that all the test grades will be integers. Therefore, we will need three `int` variables to hold the three test grades. Remember, if you need more than one variable of the same type, you can declare them in the same statement, separated by commas. For example if we needed two double variables, we could declare them like:

```
double var1, var2;
```

Declare three int variables to hold the three test grades. Be sure to give them appropriate names like test1, test2, etc. rather than x, y, z.

Additionally, looking at the problem, we see that we have the number 3 occurring in the problem. Rather than simply using this number in the program when needed, it is preferable to declare a constant variable to hold the number so that when it is used in the program, it will be clear what the 3 refers to. Remember to create a constant you use the keyword `final` in front of the declaration. Also it is customary to use `ALL_CAPS` for the name of the constant. For example if we wanted a constant to hold the value `PI`, we would declare:

```
final double PI = 3.14159;
```

Declare an int constant to hold the value 3, the number of tests. Be sure to give the constant an appropriate name like NUM_TESTS.

Finally, when looking at a problem you may need variables to hold the solution or some intermediary steps. For this problem we need to calculate an average. We will need a variable to hold the average. Usually, the average of values can contain decimal values, so you will need to declare a `double` variable to hold the average.

Step 3: Getting the Input

Now that we have the needed variables declared, we are ready to use the `Scanner` object we created to get the input from the user. Before reading in the input though, it is important to give the user a prompt so the user knows what they are expected to enter. Then we use the `Scanner` object with the appropriate method to read in the value and store it in a variable. For example to prompt and read in the first test score, we would use:

```
System.out.print("Enter the score on the first test: "); // prompt
test1 = in.nextInt(); // read in the next integer (since test1 is an int) and store it in test1
```

where the already declared variable `test1` will hold the score for the first test and in is the `Scanner` object.

Write the prompt and read the input for all three tests.

Step 4: Calculate the Average

After reading the three input values from the user, we can use them to calculate the average. To do so we add up all the values and divide them by the number of tests. Naively, this would be:

```
average = test1 + test2 + test3 / NUM_TESTS;
```

However, due to operator precedence rules Java will do the division, `test3 / NUM_TESTS`, before the addition, which will give the wrong result. To force Java to do the addition first, we have to use parentheses:

```
average = (test1 + test2 + test3) / NUM_TESTS;
```

This will calculate the average, but there is still a problem. Assume the test grades are 90, 90, and 92, then the average will be 90.6666, but Java will give the answer as 90 (You should run the program and print the result to verify). This is because all the variables are integers and so Java does integer division. To force Java to do decimal division, we have to cast one of the variables to a `double`. Remember to cast a value to another type you put the type you want to cast to in parentheses before the value. So, let's cast `NUM_TESTS` to a `double`:

```
average = (test1 + test2 + test3) / (double)NUM_TESTS;
```

We could have cast any of the other variables as well. **Calculate the average test score in your code.**

Step 5: Display Results

Now that we have calculated the result we need to show it to the user. Use a `System.out.println` statement to display the average score to the user. Be sure to have a statement explaining what the number is. That is, don't just print the number. For example, if we wanted to print the first test score, we would use the following statement:

```
System.out.println("Your first test score: " + test1);
```

Use the following Coding Guidelines (You will be graded on this)

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
- Keep identifiers to a reasonably short length.
- Use upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- 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.
- Use white space to make your program more readable.
- Use comments after the ending brace of classes, methods, and blocks to identify to which block it belongs.

Note

- Labs are not graded by a program, so you do not need to spend a large amount of time making the output match perfectly with the sample below. Do, however, make sure your output is reasonable. The goal here is for you to demonstrate that you understand the underlying concepts.
- See the sample output below in the lab for an idea of what your program should output.

Sample Output

Below is an example of what your output should roughly look like when this lab is completed. All text in bold represents user input.

Sample Run 1:
Enter the score on the first test: **90**
Enter the score on the second test: **91**
Enter the score on the third test: **92**
Your average score is: 91.0

Sample Run 2:
Enter the score on the first test: **90**
Enter the score on the second test: **90**
Enter the score on the third test: **92**
Your average score is: 90.6666

Submission

Submit your Lab2.java file to the Submission Server. Go to the Submission Server site located on the course website, login, then click on Lab Submissions in the left frame. Choose Lab2 from the dropdown box, click on the browse button and find where you saved your `Lab2.java` file (and not the `Lab2.class` file) on your computer. Upload the file to the site and then click on the Submit button.

Your file will be submitted and a screen will show up displaying if your program compiled and what your output is when run on some sample input.

You should then check to make sure that the actual file submitted properly and is readable to the grader. To do so click on Grades in the frame on the left of the page and then click on the 0 underneath Lab2. You will again see that your program compiled and the sample output, but you should scroll down to the bottom of the screen and make sure your file is readable as well.

Important Note: You may resubmit as many times as you like until the deadline, but we will only mark your last submission.