# Lab #4
# CSE110 - Arizona State University

## *Topics*

- Using conditional (`if`, `if-else`) statements - Chapter 3
- Using loops - Chapter 4

## *Coding Guidelines*

- 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.

## *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 #4
// TIME SPENT: how long it took you to complete the assignment
//-----------------------------------------------------------------------*/
```

## *Getting Started*

Create a class called `Lab4`. Use the same setup for setting up your class and main method as you did for the previous assignments. Be sure to name your file `Lab4.java`.

## *Hints*

- Example of using a loop to read in sentinel values can be found in Section 4.5 of the book as well as the example videos for the week. The instructions below follow the example in the sample video.
- Examples of loops for various tasks can be found in Section 4.7 of the book as well as the videos for the week.

## *Part 0: Task Overview*

The overall goal of this lab is to continually read in integers, one at a time, from the user until a 0 is entered. 0 should not be counted as an input, but rather is a sentinel used to stop the loop. At this point, if the user entered any values (besides 0) the following output should be displayed:

- A line stating the smallest integer the user entered
- A line stating the largest integer the user entered
- A line stating the number of integers entered
- A line stating the number of even integers entered
- A line stating the number of odd integers entered
- A line stating the average value of all of the integers that the users entered

Otherwise a warning message should be displayed. ***IMPORTANT: Please do not include the 0 in these calculations!*** The basic outline for the code logic is given below:

## *Part 1: Before the while*

These steps should be carried out in your main method prior to starting the while loop.

1. Set up the `Scanner`.
2. Declare and initialize variables needed for the calculations in the loop.
   - Note: It is important to declare these before the loop since they need to be printed out after the loop. Any variable declared in the loop is only usable inside of the loop.
   - The following variables will be needed (You can see the coding sample video with sentinel values and averages for an example of initializing a couple variables before a loop. Here we just initialize more):
     - largest number (see below about initializing these)
     - smallest number (see below about initializing these)
     - a counter for how many integers have been entered (this would be initialized to 0, since before the loop no integers have been entered yet)
     - number of even integers (how many have been entered before the loop is started?)
     - number of odd integers (how many have been entered before the loop is started?)
     - sum of all the values entered needed for the average (what is the sum before any value is entered?)
     - a variable to hold the input. Prompt the user to `"Enter a series of values (0 to quit):"`
4. Read in the first value with the `Scanner` and store this in the variable used to hold the input.
5. Initialize the variable that holds the largest and smallest values to the first input just read in.
   - Note: These must be initialized to the first value entered to work properly. If they were just initialized to 0 then if the user never entered any negative numbers, the minimum would remain 0. Similarly, if the user never entered any positive numbers, the maximum would remain 0.

## *Part 2: Setting up the while*

Now we are ready for the loop.

- Set up a while loop to continue while the input is not equal to 0, the sentinel value.
  - For this sentinel problem we will use a while loop (though other loops could be used, particularly the do while with an extra if).
  - We will have multiple statements inside the while loop. These must go in braces. For example:

```
while (input not equal to 0) {
    // Put the instructions below in here
}
```

## *Part 3: Inside the while loop*

This is where we will calculate all the needed values. The order these are done inside the loop is not important. You should add these one at a time and check to make sure the result is working correctly before adding more.

1. First find the maximum and minimum values.
   - The idea is given in the if statements in the while loops in Section 4.7.5. For the maximum (the minimum can be found with a small change):

```
if (input greater than maximum)
    maximum = input // update max
```

2. Find the number of even integers entered.
   - To find if the current input value is even or odd you can use the remainder operator with 2 (`input % 2`). If the result is 0, the number is even. Otherwise it is odd. The logic for this part then is:

```
if (input is even) // that is input % 2 is 0
    increase number of evens by 1
otherwise
    increase number of odds by 1
```

3. Calculate the updated sum.
   - Add the current input to the sum and save the result back in the sum. You can get an idea from section 4.7.1 or the sentinel average program example.
4. Increase the counter by one since one more input has been seen.
   - This is as done in the Average example in section 4.7.1 or the sample video with the counter line.
5. Finally, read in the next input value.
   - This updates the loop control variable to be ready to process the next input.

## *Part 4: After the Loop*

This section goes after the close brace `'}'` of the while loop. Here we will print the results if any data has been entered or print a message that `"No data was entered"`. The Average example in 4.7.1 and the Sentinel Average video sample show checking to make sure input was entered.

1. If any data has been entered (What would the count be if only 0 was entered?)

- Print out the results stating the max, min, number of integers entered, number even and odd. (See sample output below)
- Calculate the average, by dividing the sum by the count and display the results. Be careful of integer division if both the sum and count are integer variables.

2. Otherwise, only the sentinel was entered, so output a warning that `"No data was entered"`.

## *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 a series of integers (zero to quit):
**3**
**-4**
**5**
**12**
**-7**
**0**

The smallest integer is -7
The largest integer is 12
Total number of integers entered is 5
Total even numbers entered is 2
Total odd numbers entered is 3
The average value is 1.8

Sample Run 2:
Enter a series of integers (zero to quit):
**-10**
**-2**
**-6**
**0**

The smallest integer is -10
The largest integer is -2
Total number of integers entered is 3
Total even numbers entered is 3
Total odd numbers entered is 0
The average value is -6.0

# *Submission*

Submit your `Lab4.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 Lab4 from the dropdown box, click on the browse button and find where you saved your `Lab4.java` file (and not the `Lab4.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 Lab4. 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.