# CSC 205 Lab 2 : Array Stats Program & OOP Review

*Due by Friday, September 15th, 2:00 PM*

## Goals

After completing this lab, you should be able to:

* Work with loops and arrays in Java.
* Use and call multiple class methods and understand the value of structured programming.
* Make use of instance methods and understand how both instance methods and class methods can be used within the same class.
* Be able to differentiate between instance variables and class variables.
* Write constructors, getters, and setters.

## Lab Setup

Change into your Labs directory, and let's create and change into a Lab2 directory.

Now, let's copy over a source file I've already started for you (Stats.java) to use for your program along with a sample executable file.

cp /pub/digh/CSC205/Lab2/\* .

**The Array Stats Program**

Two of the measures taken on a group of data are the range of data and the mean of data. The range of a group of data is defined to be the difference of the largest element and the smallest element. The mean is defined to be the average of the group of data. Complete the program Stats so that it will successfully compute the range and mean of a data set that is input by a user.

#### Input and Output

The input to your program will be *n* positive integers one per line from the keyboard, where *1 < n < 100*. The end of all input is indicated by a line with the integer 0.

Following the input set, your program should print out the range of the input as well as the mean with appropriate output prompts.

#### Sample Run

Please Input Your Values (Enter a 0 to Stop) :

4

6

2

20

10

0

The range of your 5 items is : 18

The mean of your 5 items is : 8.4

#### Program Design

Your main method has already been set up for you. You simply need to code the bodies of three methods : fillUp, range, and mean.

##### Method fillUp is used to read all the input values into the array List and return the number of items read in back to the main method.

Method range returns the difference between the maximum value and minimum value in List. The calculation of the maximum and minimum can be done using a single for loop.

Method mean returns the average of the input values. Notice that it returns a double value so you will need to use some typecasting. There is no need to worry about a specific double precision for this lab.

## OOP Review

Take a look at class Person on the attached handout. Each person has instance variables for an id and name associated with them. Notice also the private class variable personCount. We will use this class variable to assign a unique id to each person in the order they are “created”.

Unlike instance variables, which as the name implies come with every instance of a class (i.e., object), class variables are associated with the class instead. This means, only one copy of this variable is shared among all objects of this class and therefore the value of this variable is the same for all of the objects.

To declare a class variable we use the keyword static. Class methods are of course similarly declared. Notice the static method getCount(). Do you see how its value is shared among all possible variables (instances) of type person? This is in contrast to the id, which is different for each person.

Finally, notice how we can invoke the static method. We can call it with in instance of the class, i.e., an object (e.g., sue.getCount()) – or, and this is different, by just referring to the *name of the class*, i.e., Person.getCount()).

Now, in the space below, trace through and give the output of the StaticTest program shown.

#### Embellishing the Person Class

* Let’s overload our constructor by adding a default (no-args) constructor to Person that defines an object to have the name “N/A” and an id of –1. Use the this operator appropriately, and be sure to increment personCount.
* Add a setter method named reset that can be used to reset the two private instance variables of this class to two values passed in as parameters. This will allow your Person class to now be mutable.
* Add getter methods getName and getId that can be used to retrieve these two private variables.

Finally, let’s test our new methods at the end of the StaticTest program. Declare a Person object samplePerson that uses the default constructor. Next, reset it to your own name and the last four digits of your social security number using the reset method. Finally, print out your name and id using your getter methods.

**Person Class**

public class Person

{

private String name;

private int id;

private static int personCount = 0;

// constructor

public Person(String pname)

{

name = pname;

personCount++;

id = 100 + personCount;

}

public String toString()

{

return "name: " + name + " id: " + id

+ " (Person count: " + personCount + ")";

}

// static/class method

public static int getCount()

{

return personCount;

}

}

**StaticTest Program**

public class StaticTest

{

public static void main(String args[])

{

Person tom = new Person("Tom Jones");

System.out.println("Person.getCount(): " + Person.getCount());

System.out.println(tom);

System.out.println();

Person sue = new Person("Susan Top");

System.out.println("Person.getCount(): " + Person.getCount());

System.out.println(sue);

System.out.println("sue.getCount(): " + sue.getCount());

System.out.println();

Person fred = new Person("Fred Shoe");

System.out.println("Person.getCount(): " + Person.getCount());

System.out.println(fred);

System.out.println();

System.out.println("tom.getCount(): " + tom.getCount());

System.out.println("sue.getCount(): " + sue.getCount());

System.out.println("fred.getCount(): " + fred.getCount());

}

}