Week 2, Lab A – Collections

# Lab Intro & Prep

This lab will serve as an introduction to using data structures to store data. Such data structures could be storing primitive types (e.g. int, double, char), or storing objects – or more technically, storing *references* to objects.

## Learning Objectives

* Manipulate a collection of data values using an array
* Passing arrays to methods
* You may find the **Week 2 Reference Sheet** (located under this lab sheet on Moodle) helpful to refer to how we can perform common array operations
* Begin by launching **Eclipse**
* Create a new Project by selecting **New > Java Project.**
* Name the project **Week 2** – this will serve as your workspace (folder) for all of the programs you make during this week.

# Exercise 1 – Statistics (Array of Primitive doubles)

Create a class called **Statistics** with a main method. This class will prompt the user to enter some numerical data and calculate some statistics on the data entered. Recall from the lecture **(slide 20)** that you can dynamically allocate the size of an array at runtime by fetching an integer from the keyboard and use that as the size of the array.

## Part 1 – Data Entry

* Prompt the user to enter the number of data points (how large the array should be)
* Allocate an array of **doubles** of the correct length to store the data
* Prompt the user for each value in the data set, storing each value in the array

## Part 2 – Calculating Statistics

Now add a static method to your class called **printStatistics** which takes an array of doubles, calculates and prints out the number of data points, the mean, variance and standard deviation. Call the method in your main program. The method header is:

**public** **static** **void** printStatistics( **double**[] data)

The variance is the mean squared difference of the data values from the overall mean. So, to calculate the variance, you need to know the mean first. The squared difference of data point i is given by **(data[i] – meanValue) \* (data[i] – meanValue)**. Your program should calculate the total of these values for each i (in a for loop), then divide this total by the number of values to obtain the variance. The standard deviation is the square root of the variance. A sample session (and test case) is provided in Fig 1.

**Hints:**

* The first thing your method should do is find how many values are in the array. You can then store this length in a variable (e.g. int numValues = data.length;).
* The java function for square root is Math.sqrt( value )

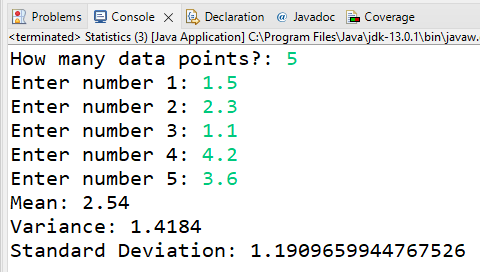


Fig 1 – Exercise 1 Example Output

# Exercise 2 – Array of (Fish) Objects

Create a new class named **Fish** – do **not** give this class a main method, as it will be used to represent a Fish data type. Replace the automatically generated code with the code in Listing 1.

Create a class named **FishDemo** with a main method and create a few Fish objects. A [quick google search](https://www.google.com/search?q=fish+weight&rlz=1C1CHBF_en-GBGB917GB917&oq=fish+weight&aqs=chrome..69i57.1039j0j7&sourceid=chrome&ie=UTF-8#cobssid=s) yields some great example data of fish you can create. Recall from Listing 1 that to create a fish we need two things as defined in the constructor – the species of the fish, and the weight (in kilograms).

Listing 1 - Exercise 2 Fish class

|  |
| --- |
| **public** **class** Fish {  String species;  **double** weight; // weight in kg    Fish(String species, **double** weight) {  **this**.species = species;  **this**.weight = weight;  }    } |

In the **FishDemo** class, create an ArrayList named **pond** that is capable of storing references to Fish objects. Add the previously created Fish objects to this ArrayList by using the ArrayList’s **add** method.

We are now going to create a static method that returns the total weight of all fish in our pond (hopefully you don’t have a 19,000 kg whale shark in your pond). The method header for this static method is provided below:

**static** **double** totalFishWeight( ArrayList<Fish> fishes )

This method should loop though the array list that is supplied and calculate the total weight of all the fish objects in the ArrayList and return it as a double. Once this has been implemented, call the method in your main method.

For example, if your ArrayList of fish contains a “Haddock” that is 9.2kg, and a “Mahi Mahi” that is 22kg, the output should resemble that of Fig 2.

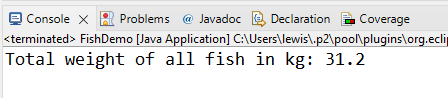


Fig 2 - FishDemo Example Output