

# Chapter 8 Lab

## Arrays

### Objectives

- Be able to declare and instantiate arrays
- Be able to fill an array using a for loop
- Be able to access and process data in an array
- Be able to write a sorting method
- Be able to use an array of objects

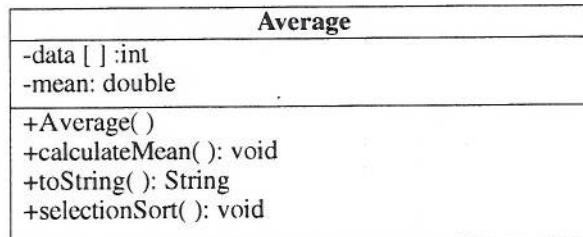
### Introduction

Everyone is familiar with a list. We make shopping lists, to-do lists, assignment lists, birthday lists, etc. Notice that though there may be many items on the list, we call the list by one name. That is the idea of the array, one name for a list of related items. In this lab, we will work with lists in the form of an array.

It will start out simple with a list of numbers. We will learn how to process the contents of an array. We will also explore sorting algorithms, using the selection sort. We will then move onto more complicated arrays, arrays that contain objects.

## Task #1 Average Class

Create a class called Average according to the UML diagram.



This class will allow a user to enter 5 scores into an array. It will then rearrange the data in descending order and calculate the mean for the data set.

### Attributes:

- **data[ ]**—the array which will contain the scores
- **mean**—the arithmetic average of the scores

### Methods:

- **Average**—the constructor. It will allocate memory for the array. Use a for loop to repeatedly display a prompt for the user which should indicate that user should enter score number 1, score number 2, etc. Note: The computer starts counting with 0, but people start counting with 1, and your prompt should account for this. For example, when the user enters score number 1, it will be stored in indexed variable 0. The constructor will then call the **selectionSort** and the **calculateMean** methods.
- **calculateMean**—this is a method that uses a for loop to access each score in the array and add it to a running total. The total divided by the number of scores (use the length of the array), and the result is stored into mean.
- **toString**—returns a String containing data in descending order and the mean.
- **selectionSort**—this method uses the selection sort algorithm to rearrange the data set from highest to lowest.

## Task #2 Average Driver

1. Create an **AverageDriver** class. This class only contains the main method. The main method should declare and instantiate an Average object. The Average object information should then be printed to the console.
2. Compile, debug, and run the program. It should output the data set from highest to lowest and the mean. Compare the computer's output to your hand calculation using a calculator. If they are not the same, do not continue until you correct your code.

### Task #3 Arrays of Objects

1. Copy the files *Song.java* (code listing 8.1), *CompactDisc.java* (code listing 8.2) and *Classics.txt* (code listing 8.3) from [www.aw.com/cssupport](http://www.aw.com/cssupport) or as directed by your instructor. *Song.java* is complete and will not be edited. *Classics.txt* is the data file that will be used by *CompactDisc.java*, the file you will be editing.
2. In *CompactDisc.java*, there are comments indicating where the missing code is to be placed. Declare an array of Songs, called *cd*, to be of size 6.
3. Fill the array by creating a new song with the title and artist and storing it in the appropriate position in the array.
4. Print the contents of the array to the console.
5. Compile, debug, and run. Your output should be as follows:

```
Contents of Classics
Ode to Joy by Bach
The Sleeping Beauty by Tchaikovsky
Lullaby by Brahms
Canon by Bach
Symphony No. 5 by Beethoven
The Blue Danube Waltz by Strauss
```