# Lab 8 – Arrays (16 pts)

## Lab Objectives

- Be able to declare and instantiate arrays
- Be able to fill an array using a 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

## ~~Deliverables~~

~~This lab has three tasks. When you have all tasks done, run the report in Blackboard. The report is a Blackboard test with short-answer, file-response, multiple-answer, and other types of questions.~~

~~In the report, you may be asked to provide code segments, Java source code files (must have extension *.java*), screenshot of program execution, files in PDF format, and your analysis of the results.~~

~~If a short answer question requests a code segment, please ensure that your input is readable: all **new lines and indents** are in place.~~

~~Screenshots in your report **must show a full screen**, so your computer can be identified. Please resize your IDE panels the way that the required dialog or output is visible along with the source code. Show as much source code as possible.~~

~~NOTE:~~

- ~~Use **Blackboard only** to submit your work; **no email** submission unless your instructor directs it.~~
- ~~If Blackboard gives you multiple submission attempts (usually three), the **last one** will be evaluated and graded.~~
- ~~**No late submissions**, **no changes** in your submission after the due date.~~

## Introduction

This lab may need to refer to different parts of the text and lectures: arrays (textbook, Ch.7), files, and objects.

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.

In Task #3, we will then move onto more complicated arrays, arrays that contain objects. This part may require additional time, but you are not a beginner anymore.

**An important comment on the file names:** make sure that your code and files you are working with are all in the same folder (directory). Then `javac` (or IDE Dr. Java) should have no problem with opening files. If you are on IDE Eclipse, the file to open must be in the project directory at the same level as folders *bin* and *src*.

If you are experiencing problems, you may need to pass in the absolute path of the file. On Windows, if file *Classics.txt* is located in your *Desktop* folder, the absolute path could be *C:/Users/YourName/Desktop/Classics.txt*.

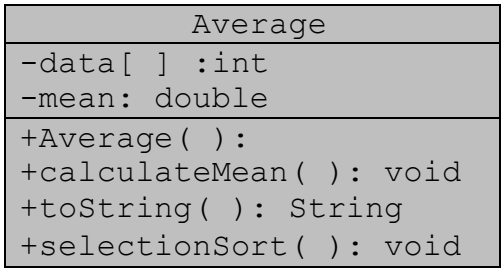
To manage the absolute path if you are hard coding the file name, e.g., `FileReader file = new FileReader("`*file name*`");` you must pass in the absolute file path like this:

- for Windows: `FileReader file = new FileReader("C:\\Users\\YourName\\Desktop\\Classics.txt");`
- for MacOS: `FileReader file = new FileReader("//Users//YourName//Desktop//Classics.txt");`

The absolute path is individual for each file and above examples only illustrate the idea.

## Task #1 Average Class (6 pts)

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 the `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. You may modify code example from section 7.11 *The Selection Sort and the Binary Search Algorithms* of the textbook to write this method.

## Task #2 Average Driver (6 pts)

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 (4 pts)

1. Copy the files *Song.java*, *CompactDisc.java* and *Classics.txt* as directed by your instructor. All files must be in the same directory. *Song.java* describes class `Song`; it 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 `Song` objects, called `cd`, with a size of 6.
    - Add the code to fill the array by creating a new song with the title and artist sequentially read from *Classics.txt* and storing it in the appropriate position in the array.
    - Add the code to print the contents of the array to the console using the appropriate method from the class `Song`.
3. 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