

# Lab 7

Learn about Single-Dimensional Arrays: how to create, fill, and search through arrays.

*In this Lab, you will get practice with creating arrays and initializing them (filling them with values). Activities also include concepts such as searching through the arrays using both the index position and/or element value. Pay attention to how loops are related to each of these topics.*

## In-Lab Activities

*Don't forget to answer the Guided Inquiry Questions in your **notebooks**. It is also encouraged to use it frequently while working through activities.*


### Warm-Ups

Go to the Google Forms link on the board and complete your Warm-Up Questions. You have 10 minutes to complete this. If you finish early, feel free to begin the Lab Activities, but make sure your partner is ready as well.


### Activity 1 – Intro to Arrays

Write a program that will create an array of ten integers, allow the user to enter those integers, and eventually search through the array based on index position:

1. **Declare** the array. You cannot assign **elements** to an array until it has first been created.
2. Next, run a for loop so that the user can enter an integer to save in each **index position** of the array. Since you this array holds ten **elements**, the **index position** starts at 0 and ends at 9. **Think about this when creating the for loop.** How can you make it so that each loop iteration is specific to **one specific index** of the array?
3. Once the array has been filled with values, allow the user to search for a specific element based on **index position**. Allow them to enter an **index position**, and then display the **element** at that given position. Let the user keep searching until they enter -1.
4. What happens when you search for an index position that doesn't exist, such as 10? Add some code that, if the user tries searching for an index that is **out-of-bounds**, displays a message and asks for a new index.



What is the **index position** of an array?



Briefly describe the **off-by-one** error, as mentioned in the textbook.



### Activity 2 – Searching in Arrays

Write a program which creates an array based on a size which the user gets to choose. Once declared, fill the array with random integers from 1 to the size of the array. Next, it will ask the user to enter an integer to search for – make sure to also tell the user what the range is. The program should search through the array for this value, and then display the index position of that value. If the value cannot be found in the array, print “No element found.”



CHECKPOINT

40 MINUTES



### Activity 3 – Playing with arrays

Write a program that creates and initialize an array of 50 random integers between 1 and 1000 and make the following methods and display the results:

- `getHighest()` – this method will return the **highest element** in the array.
- `getLowest()` – this method will return the **index** of the lowest element in the array. **Notice** how this is different than `getHighest()`.
- `getAverage()` – this method will return the Average of all elements in the array.
- `getAboveAverage()` – this method returns the number of elements in the array above the average value. **Hint:** don't forget that you can call methods from within other methods.
- `getBelowAverage()` – this method returns the number of elements in the array below the average.

How do you **pass an array** to a method? How does this differ from passing a normal variable?

Note: when making these methods, don't forget to think about what the **return type** should be, and what to include for **parameters**.

Include proper **input** and **output** to test each of these methods back in the **main method**.



#### CHECKPOINT

20 MINUTES



#### Brainstorming Individually

(5-7 minutes)

Read the next Activity and start **brainstorming**; think about ways you could start solving this problem and write down any notes. Do this *individually* for 3 to 5 minutes and then come together afterwards to compare, discuss, and code.



### Activity 4 – The Deck of Cards

Case Study 7.4 in the Revel Textbook may be a helpful reference if needed.

In this activity, you will simulate a deck of cards using an array of size 52. Use these hints to guide you:

1. The array will need to be of a **String** data type, and will hold values such as “Queen of Hearts”, “King of Spades”, “4 of Clubs”, “Ace of Diamonds”, etc...
2. You will need to run a **nested for loop**: one loop to determine suits, and one loop for numbers (Ace, Jack, Queen, and King included). *If it gets confusing or values become hard to keep track of, try illustrating it yourself in your notebook.*
3. Since you are using for loops to determine the card's value, what additional variable can you create to keep track of the incrementing **index position** of the array? You can't save each new card in the same position every time. **Remember running totals and/or counters** and how this can be applied.
4. Use proper conditional statements within your loops to determine the card's value. You will probably need more than one, considering both suit *and* value must be determined.

Don't forget to include proper **input** and **output** to test this, such as displaying each card and/or allowing the user to display the element at a specific index.



## Activity 5 – Shuffling The Deck



You will use your code from Activity 4 for this Activity as well. Write a **static method** in the class that will take the original deck (the array) and return a shuffled deck. Before worrying about the **return type**, think about **how arrays as parameters work**. In order to shuffle, generate two random numbers between 0 and 51, and then swap the elements of those two index positions (the two random numbers). Do this at least 100 times to shuffle thoroughly.

Back in the main method, display the decks both before and after the method is called to make sure that the shuffling worked correctly.

Let's say you pass an array as a parameter of a new method, and then edit the contents of that array within said method. If you display all the elements of the array back in the main method, is it the original array, or the new method's edited version? **Explain why.**



### CHECKPOINT

50 MINUTES



#### Brainstorming Individually

(3-5 minutes)

Read the next Activity and start **brainstorming**; think about ways you could start solving this problem and write down any notes. Do this *individually* for 3 to 5 minutes and then come together afterwards to compare, discuss, and code.



## Activity 6 – Comparing Two Cards



Write a method in the class called “compareTo()” which will compare two cards based on their values and **returns the index position of the highest value**. When deciding which card is greatest, note that Aces count as 1. Suits are ranked as follows: Spades > Diamonds > Clubs > Hearts.

Also pay close attention to what **parameter(s)** will be required to accomplish this.

Back in the **main method**, test this by asking the user to enter two index positions. Display the cards at each position, and then use the **compareTo()** method to determine which is greatest. Display both the **index position**, as well as the actual **card** (the String value), with the greatest value.



### CHECKPOINT

30 MINUTES

## Reflection

Go to the Google Forms link given by your TA to work on your Reflection for Lab 7. This counts as a part of your participation grade, and it is important that you put as much detail as possible.