Assignment 2 is due Monday September 18 at 11:59pm
- Task 1 of Assignment 1 should have already been turned into the box outside E116

Reading for next lecture is the remainder of Chapter 7 and all of Chapter 8

An array of size N is indexed from zero to N-1

The entire array has a single name

Each value has a numeric index

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>79</td>
<td>87</td>
<td>93</td>
<td>58</td>
<td>88</td>
<td>95</td>
<td>75</td>
<td>91</td>
<td>87</td>
<td>66</td>
</tr>
</tbody>
</table>

Note that an array name is a pointer to the front element

int[] A; // declaration
A = new int[5]; // memory allocation

int[] arrayB = new int[5]; // both
char[] lettersArray = new char[5];
String[] s = new String[3];

int[] arrayC = {1, 2, 3, 4, 5}; // initialization
char[] letterGrades = {'A', 'B', 'C', 'D', 'F'}
String[] wordArray = {"CS230", "Data", "Struct"};
Arrays are an **indexed** and **mutable** collection - we can directly access and change an element at *any* index.

Arrays are **homogeneous** collections. All the elements of a Java array must have the same type.

Arrays have a **fixed length**. Once an array is created, its length cannot be changed.

If `a` denotes an array, `a.length` refers to its length.

```java
int[] arrayB = new int[5];
for (int i = 0; i < 5; i++) {
    arrayB[i] = 2*i;
}

int[] arrayC = {1, 2, 3, 4, 5};
for (int i = 0; i < arrayC.length; i++) {
    arrayC[i]++;
    System.out.println(arrayC[i]);
}

String[] wordArray = {"CS230", "Data", "Struct"};
wordArray[1] = "Silly ";
System.out.println(wordArray[1] + wordArray[2]);
```

Enter a sentence:
To be, or not to be: that is the question: Whether 'tis nobler in the mind to suffer the slings and arrows of outrageous fortune, or to take arms against a sea of troubles, and by opposing end them?

A: 0   o: 11
B: 0   b: 5
C: 0   c: 0
D: 0   d: 4
E: 0   e: 17
F: 0   f: 5
G: 0   g: 0
H: 0   h: 7
I: 0   i: 8
J: 0   j: 0
K: 0   k: 1
L: 0   l: 13
M: 0   m: 3
N: 0   n: 12
O: 0   o: 18
P: 0   p: 2
Q: 0   q: 1
R: 0   r: 11
S: 0   s: 13
T: 1   t: 18
U: 0   u: 0
V: 0   v: 0
W: 1   w: 1
X: 0   x: 0
Y: 0   y: 2
Z: 0   z: 0

Non-alphabetic characters: 45

When you manipulate an array, you access it through a pointer!

What happens here?

```java
arrayA = arrayB;
```

What is printed here?

```java
int[] arr1 = {1, 2, 3, 4, 5};
int[] arr2 = {1, 2, 3, 4, 5};
if (arr1 == arr2)
    System.out.println("same");
else
    System.out.println("different");
```

How do we copy the contents of `arrayA` into `arrayB`?

How do we check if two arrays contain the same info?
• An entire array can be passed as a parameter to a method

• Like any other object, the **reference** to the array is passed, making the formal and actual parameters **aliases** of each other

• Therefore, changing an array element within the method changes the original (called “by reference”)

• This can also be a source of **errors** – be careful!

```java
//Compute the sum of the contents of an int[]
public static int sumElements (int[] numArray) {
    int sum = 0;
    for (int i = 0; i<numArray.length; i++)
        sum = sum + numArray[i];
    return sum;
}

//... in main
int[] myData = {1, 2, 3, 4, 5};
int result = sumElements(myData);
```

```java
//create an array and fill it up with its indices
public static int[] createNumArray (int size) {
    int[] newArray = new int[size];
    for (int i = 0; i<size; i++)
        newArray[i] = i;
    return newArray;
}

//... in main
int[] arrayC = createNumArray(20);
```

• The elements of an array can be object references

• The following declaration reserves space to store 5 references to String objects

  ```java
  String[] words = new String[5];
  ```

• Initially an array of objects holds **null** references

  ```java
  System.out.println (words[0]);
  ```

• At this point, the above line would throw a **NullPointerException**

• Each object of an array must be instantiated separately

  ```java
  words[1] = “loyalty”;
  ```
The `String[] args` input parameter in the `main()` method is Java's way to communicate with the outside world at the time of invocation. The arguments to the `main()` method are called command-line arguments and are provided when an application is run.

```java
public class PlayGame {
    public static void main(String[] args) {
        String player1 = args[0];
        String player2 = args[1];
        System.out.println("Welcome to the game ");
        System.out.println(player1 + " and " + player2);
    }
}
> java PlayGame Jack Jill
```