• Assignment 1 is due tonight at 11:59pm

• Assignment 2 is due next Thursday at 11:59pm

• Reading for Monday’s class 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

How are arrays implemented?
Generally, as contiguous blocks of memory
We can derive many of the properties of arrays by recalling this fact

Scores

0 1 2 3 4 5 6 7 8 9

79 87 93 58 88 95 75 91 87 66

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

int[] arrayB = new int[5]; // both
char[] lettersArray = new char[5];
String[] words = 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 **indexed** and **mutable** collections.
- We can directly access and change elements at any index.
- Arrays are **homogeneous** collections.
  - All the elements must have the same type.
- Arrays have a **fixed length**.
  - After creation, array length can't be changed.
- Java Arrays are **objects**.
  - The Array class provides an interface for accessing the underlying contiguous block of memory.
- For array `a` its length is given by `a.length`.

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

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

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

- Given a sentence (as a String), how do you count how many times each letter appears?
  - (Hint: You can access any char using `charAt(i)`)
  - Where would you keep track of all the counters?
import java.util.Scanner;

public class LetterCount {
    public static void main (String[] args) {
        final int NUMCHARS = 26;
        Scanner scan = new Scanner (System.in);
        int[] upper = new int[NUMCHARS];
        int[] lower = new int[NUMCHARS];
        char current;
        // the current character being processed
        int other = 0;
        // counter for non-alphabetics
        System.out.println("Enter a sentence:");
        String line = scan.nextLine();
        // Count the number of each letter occurrence
        for (int ch = 0; ch < line.length(); ch++) {
            current = line.charAt(ch);
            if (current >= 'A' && current <= 'Z')
                upper[current-'A']++;
            else
                if (current >= 'a' && current <= 'z')
                    lower[current-'a']++;
                else
                    other++;
        }
        // Print the results
        for (int letter=0; letter < upper.length; letter++) {
            System.out.print ((char) (letter + 'A') + " : " + upper[letter] + "  	" + (char) (letter + 'a') + " : " + lower[letter] + "
        System.out.println("Non-alphabetic characters: " + other);
    }
}

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

What happens here?
arrayA = arrayB;

What is printed here?
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?
The elements of an array can be object references. E.g. references to `String` objects:

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

Initially an array of objects holds `null` references:

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

The above line doesn’t throw a `NullPointerException`. Why?

NullPointerExceptions are caused only when null values are used, or dereferenced. Dereference is just a technical term for “follow”.

```java
words[1] = "loyalty";
words[0] = "friendship";
words[2] = "honor";
```

An entire array can be passed as a parameter to a method – it’s an object!

As with any other object, the reference to the array is passed, making the formal and actual parameters aliases of each other.

Therefore, changing an array slot of the array within a method changes the original (called “by reference”)

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

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.print("Welcome to the game ");
        System.out.println(player1 + " and " + player2);
    }
}
```

```java
java PlayGame Jack Jill
```

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

// Driver which uses the above code
```java
int[] myData = {1, 2, 3, 4, 5};
int result = sumElements(myData);
```
//Compute the sum of the contents of an int[]
public static void incrementElt(int[] numArray) {
    for (int i = 0; i < numArray.length; i++) {
        numArray[i]++;
    }
}

// Driver which uses the above code
int[] myData = {1, 2, 3, 4, 5};
int result = incrementElt(myData);
for (int num : myData) {
    System.out.println(num); // What is printed?
}

//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;
}

// code in the driver
int[] arrayC = createNumArray(20);

*/
* Demonstrates the use of an array of objects
* Author Java Foundations
*/
public class Tunes {
    /*
    * Creates a CDCollection object and adds some CDs to it. Prints
    * reports on the status of the collection.
    */
    public static void main(String[] args) {
        CDCollection music = new CDCollection();
        music.addCD("Storm Front", "Billy Joel", 14.95, 10);
        music.addCD("Come On Over", "Shania Twain", 14.95, 16);
        music.addCD("Soundtrack", "Les Miserables", 17.95, 33);
        music.addCD("Graceland", "Paul Simon", 13.90, 11);
        System.out.println(music);
        System.out.println(music.getTotalCost());
        music.addCD("Double Live", "Garth Brooks", 19.99, 26);
        music.addCD("Greatest Hits", "Jimmy Buffet", 15.95, 13);
        System.out.println(music);
    }
}
import java.text.NumberFormat;

/**
 * Represents a compact disc (an ancient technology for recording sound)
 * @author Java Foundations
 */
public class CD {
    private String title, artist;
    private double cost;
    private int tracks;

    /**
     * Create a new CS with the specified information
     * @param name album title
     * @param singer artist name
     * @param price cost of CD in floating point
     * @param numTracks number of tracks on CD
     */
    public CD(String name, String singer, double price, int numTracks) {
        title = name;
        artist = singer;
        cost = price;
        tracks = numTracks;
    }

    toString() method and getter, setter omitted

    /**
     * Adds a CD to the collection, increasing the size of the
     * collection if necessary.
     * @param name album title
     * @param artist artist's name
     * @param price cost of CD in floating point
     * @param numTracks number of tracks on CD
     */
    public void addCD(String title, String artist, double cost, int tracks) {
        if (count == collection.length)  
            increaseSize();
        collection[count] = new CD(title, artist, cost, tracks);
        totalCost += cost;
        count++;
    }

    /**
     * Return string reporting on the CD collection
     */
    public String toString() {
        NumberFormat fmt = NumberFormat.getCurrencyInstance();
        String report = "-----------------------------------------------------\n";
        report += "My CD Collection\n";
        report += "Number of CDs: " + count + "\n";
        report += "Total cost: " + fmt.format(totalCost) + "\n";
        report += "Average cost: " + fmt.format(totalCost/count) + "\n";
        report += "\n\nCD List:\n";
        for (int cd = 0; cd < count; cd++)
            report += collection[cd].toString() + "\n";
        return report;
    }
}
A two-dimensional array can be thought of as a table of elements, with rows and columns.

In Java a two-dimensional array is **an array of arrays**.

It is **declared** by specifying the size of each dimension separately:

```java
int[][] scores = new int[12][50];
```

An array element is **referenced** using two index values:

```java
value = scores[3][6]
```

The array stored in any one row can be specified using one index!