Assignment 2 is due Wednesday 14 February at 11:59pm

Reading for Monday’s lecture is the remainder of Chapter 7 and all of Chapter 8

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

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[] 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]);
```

## ASCII Control Characters

<table>
<thead>
<tr>
<th>ASCII control characters</th>
<th>ASCIlle printable characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>09 Bell</td>
<td>32 space</td>
</tr>
<tr>
<td>01 DLE</td>
<td>10 FF (Form-feed)</td>
</tr>
<tr>
<td>02 DC1</td>
<td>11 LF (Line-feed)</td>
</tr>
<tr>
<td>03 DC4</td>
<td>12 VT (Vertical Tab)</td>
</tr>
<tr>
<td>05 ENQ</td>
<td>13 FF (Form-carriage return)</td>
</tr>
<tr>
<td>06 TGU</td>
<td>14 CR (Carriage return)</td>
</tr>
<tr>
<td>07 BEL</td>
<td>15 SO (Space)</td>
</tr>
<tr>
<td>08 BS</td>
<td>16 DC6 (Device control 6)</td>
</tr>
<tr>
<td>10 DEL</td>
<td>17 DC7 (Device control 7)</td>
</tr>
<tr>
<td>11 DCS</td>
<td>18 DC8 (Device control 8)</td>
</tr>
<tr>
<td>12 MST</td>
<td>19 DLE (Data link escape)</td>
</tr>
<tr>
<td>13 CAN</td>
<td>20 CCH (Cancel)</td>
</tr>
<tr>
<td>14 EM</td>
<td>21 OFF (End of medium)</td>
</tr>
<tr>
<td>15 ESC</td>
<td>22 STR (Sync/clear)</td>
</tr>
<tr>
<td>16 acknowledgement</td>
<td>23 ETB (End of transmission)</td>
</tr>
<tr>
<td>17认可</td>
<td>24 CAN (Cancel)</td>
</tr>
<tr>
<td>18 ENQ</td>
<td>25 EM (End of medium)</td>
</tr>
<tr>
<td>19 STX</td>
<td>26 ESC (Escape)</td>
</tr>
<tr>
<td>20 ETX</td>
<td>27 STX (End of test)</td>
</tr>
<tr>
<td>21 EOT</td>
<td>28 ETX (End of test)</td>
</tr>
<tr>
<td>22 EOD</td>
<td>29 EOT (End of transmission)</td>
</tr>
<tr>
<td>23 EOD</td>
<td>30 ESC (Escape)</td>
</tr>
<tr>
<td>24 ACK</td>
<td>31 US (Unit separator)</td>
</tr>
<tr>
<td>25 NAK</td>
<td>32 RS (Record separator)</td>
</tr>
</tbody>
</table>

Non-alphabetic characters: 45

---

Given a sentence, how do you count how many times each letter appears?

Where would you keep track of all the counters?
Use the charAt(i) function to get characters from a String.

```java
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-alphetics
        System.out.println("Enter a sentence:");
        String line = scan.nextLine();
        System.out.println("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++;
        }
        System.out.println("Print the results");
        for (int letter=0; letter < upper.length; letter++) {
            System.out.print((char) (letter + 'A') + " ");
            System.out.print((char) (letter + 'a') + " ");
        }
        System.out.println("Non-alphabetic characters: " + other);
    }
}
```

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

What happens here?

```java
arrayA = arrayB;
```
When you manipulate an array, you access it through a reference!

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");
```

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

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
- 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";
  words[0] = "friendship";
  words[2] = "honor";
  ```
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);
    }
}
```

```bash
> java PlayGame Jack Jill
```

---

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;
}
```

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

Hand simulate the main() method in
ArrayExercise.java and see what you get