A couple of lectures on arrays...
## Arrays

An array of size $N$ is indexed from 0 to $N-1$.

Not every array entry needs to have contents (but it is convenient if it has).

The entire array has a single name.

### scores

<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>

Each value has a numeric index.
Declaration, Memory Allocation, Initialization

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];

// declaration and initialization
int[] arrayC = {1, 2, 3, 4};

char[] letterGrades = {'A', 'B', 'C', 'D', 'F'}

String[] wordArray = {“CS230”, “Data”, “Structures”};
Array Properties

- Array is 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.

- For array a its length is given by `a.length`
Arrays and for-loops

```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]);
```
Counting letter frequencies

Counting the number of occurrences of each letter in a sentence
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?
<table>
<thead>
<tr>
<th>ASCII control characters</th>
<th>ASCII printable characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>NULL (Null character)</td>
</tr>
<tr>
<td>01</td>
<td>SOH  (Start of Header)</td>
</tr>
<tr>
<td>02</td>
<td>STX  (Start of Text)</td>
</tr>
<tr>
<td>03</td>
<td>ETX  (End of Text)</td>
</tr>
<tr>
<td>04</td>
<td>EOT  (End of Trans.)</td>
</tr>
<tr>
<td>05</td>
<td>ENQ  (Enquiry)</td>
</tr>
<tr>
<td>06</td>
<td>ACK  (Acknowledgement)</td>
</tr>
<tr>
<td>07</td>
<td>BEL  (Bell)</td>
</tr>
<tr>
<td>08</td>
<td>BS   (Backspace)</td>
</tr>
<tr>
<td>09</td>
<td>HT   (Horizontal Tab)</td>
</tr>
<tr>
<td>10</td>
<td>LF   (Line feed)</td>
</tr>
<tr>
<td>11</td>
<td>VT   (Vertical Tab)</td>
</tr>
<tr>
<td>12</td>
<td>FF   (Form feed)</td>
</tr>
<tr>
<td>13</td>
<td>CR   (Carriage return)</td>
</tr>
<tr>
<td>14</td>
<td>SO   (Shift Out)</td>
</tr>
<tr>
<td>15</td>
<td>SI   (Shift In)</td>
</tr>
<tr>
<td>16</td>
<td>DLE  (Data link escape)</td>
</tr>
<tr>
<td>17</td>
<td>DC1  (Device control 1)</td>
</tr>
<tr>
<td>18</td>
<td>DC2  (Device control 2)</td>
</tr>
<tr>
<td>19</td>
<td>DC3  (Device control 3)</td>
</tr>
<tr>
<td>20</td>
<td>DC4  (Device control 4)</td>
</tr>
<tr>
<td>21</td>
<td>NAK  (Negative acknowledge)</td>
</tr>
<tr>
<td>22</td>
<td>SYN  (Synchronous idle)</td>
</tr>
<tr>
<td>23</td>
<td>ETB  (End of trans. block)</td>
</tr>
<tr>
<td>24</td>
<td>CAN  (Cancel)</td>
</tr>
<tr>
<td>25</td>
<td>EM   (End of medium)</td>
</tr>
<tr>
<td>26</td>
<td>SUB  (Substitute)</td>
</tr>
<tr>
<td>27</td>
<td>ESC  (Escape)</td>
</tr>
<tr>
<td>28</td>
<td>FS   (File separator)</td>
</tr>
<tr>
<td>29</td>
<td>GS   (Group separator)</td>
</tr>
<tr>
<td>30</td>
<td>RS   (Record separator)</td>
</tr>
</tbody>
</table>
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

        (more...)
System.out.println("Enter a sentence:");
String line = scan.nextLine();

// Count the number of times each letter occurs
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'));
    System.out.print(": " + upper[letter]);
    System.out.print(" \t" + (char) (letter + 'a'));
    System.out.println(": " + lower[letter]);
}
System.out.println("Non-alphabetic characters: " + other);
Copying and Comparing Arrays

When you access 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");
```

How do we check if two arrays contain the same info?

What happens here?

```java
arr1 = arr2;
```

How do we copy the contents of `arr1` into `arr2`?
Arrays of Strings (Objects)

- 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].length());
  ```

- At this point, the above line would throw a `NullPointerException`

- Each object must be instantiated separately

  ```java
  words[1] = "loyalty";
  words[0] = "friendship";
  words[2] = "honor";
  ```
What about those **args** in `main()`?

- 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 contents of array `args` (argument 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 PlayGame Jack Jill
Arrays as parameters in methods for input and output

- 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!
Methods can have arrays as input

//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;
}

//code in the driver (e.g. inside main() method)

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;
}

// code in the driver (e.g., inside main() method)

int[] arrayC = createNumArray(20);
Arrays of Objects

A basic but powerful data structure to manage a collection of objects
CS230 is about software structures that allow us to manage collections of data.

We already saw that when we used an array of String objects:

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

We could use an array to store and manage a collection of any other objects, e.g.
- of CD objects,
- of Student objects,
- of Rectangle objects,
- of ...
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 ommitted
A collection of CD objects

collection[0] = new CD("Storm Front", "Billy Joel", 14.95, 10);
collection[1] = new CD("Come On Over", "Shania Twain", 14.95, 16);
collection[2] = new CD("Soundtrack", "Les Miserables", 17.95, 33);
7.3 – Arrays of Objects

- A UML diagram for the Tunes program

```
A UML diagram for the Tunes program
```
```java
/**
 * 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);

    music.addCD ("Double Live", "Garth Brooks", 19.99, 26);
    music.addCD ("Greatest Hits", "Jimmy Buffett", 15.95, 13);

    System.out.println (music);
}
```
import java.text.NumberFormat;
/**
 * Represents a collection of compact discs.
 * @author Java Foundations
 */

public class CDCollection {
    private CD[] collection;
    private int count;
    private double totalCost;

    /**
     * Constructor: Creates an initially empty collection.
     */
    public CDCollection () {
        collection = new CD[100];
        count = 0;
        totalCost = 0.0;
    }
}
/**
 * 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 tracks 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++;
}
```java
/**
 * @return string reporting on the CD collection
 */

public String toString() {
    NumberFormat fmt = NumberFormat.getCurrencyInstance();

    String report = "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    My CD Collection\n\n    Number of CDs: " + count + "\n    Total cost: " + fmt.format(totalCost) + "\n    Average cost: " + fmt.format(totalCost/count);

    report += "\n\n    CD List:\n    
    for (int cd = 0; cd < count; cd++)
        report += collection[cd].toString() + "\n    
    return report;
```
/**
 * Increases the capacity of the collection by creating a
 * larger array and copying the existing collection into it.
 */

private void increaseSize () {
    CD[] temp = new CD[collection.length * 2];

    for (int cd = 0; cd < collection.length; cd++)
        temp[cd] = collection[cd];

    collection = temp;
}
A 2D array is just an array of arrays.
(A 3D array is an array of 2D arrays ;)

Multidimensional Arrays
7.6 – Two-Dimensional Arrays

- 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[][] table = new int[12][50];
    ```
- An array element is referenced using two index values:
  ```java
  value = table[3][6]
  ```
- The array stored in any one row can be specified using one index!
## 7.6 – Two-Dimensional Arrays

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>table</code></td>
<td><code>int[][]</code></td>
<td>array of integer arrays (2D array)</td>
</tr>
<tr>
<td><code>table[3]</code></td>
<td><code>int[]</code></td>
<td>array of integers</td>
</tr>
<tr>
<td><code>table[3][2]</code></td>
<td><code>int</code></td>
<td>Integer</td>
</tr>
<tr>
<td><code>table[3].length</code></td>
<td><code>int</code></td>
<td>Integer (= 6)</td>
</tr>
<tr>
<td><code>table.length</code></td>
<td><code>int</code></td>
<td>Integer (= 5)</td>
</tr>
</tbody>
</table>
```java
/**
 * Demonstrates the use of a two-dimensional array.
 * @author Java Foundations
 */

public class TwoDArray {
    /**
     * Creates a 2D array of integers, fills it with increasing
     * integer values, then prints them out.
     */

    public static void main (String[] args) {
        int[][] table = new int[5][10];

        // Load the table with values
        for (int row=0; row < table.length; row++)
            for (int col=0; col < table[row].length; col++)
                table[row][col] = row * 10 + col;

        // Print the table
        for (int row=0; row < table.length; row++) {
            for (int col=0; col < table[row].length; col++)
                System.out.print (table[row][col] + "\t");
            System.out.println();
        }
    }
}
```