Arrays, Arrays, Arrays, Arrays, …

When you need to refer to a lot of similar items using a single variable
Arrays stored contiguously

The entire array has a single name

scores

Each value has a numeric index

An array of size N is indexed from 0 to N-1
If you know the address of the 0-th item, you know the address of all
Not every array entry needs to have contents
We usually draw them horizontally or vertically
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 go well with for-loops

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

What happens when you execute this line?

```
arr1 = arr2;
```

How do we copy the contents of `arr1` into `arr2`?
Practice!

How do we check if two arrays `arr1` and `arr2` contain the same info?

Answer:

```java
public static boolean sameArrayInfo(int[] arr1, int[] arr2) {
    if (arr1.length != arr2.length) return false;
    for (int i = 0; i < arr1.length; i++)
        if (arr1[i] != arr2[i]) return false;
    return true;
}
```
How do we copy the contents of arr1 into arr2 that have the same length?

Answer:

```java
public static void copyArr(int [] arr1, int [] arr2){
    // assume arr1.length == arr2.length
    for (int i=0; i<arr1.length; i++)
        arr2[i] = arr1[i]);
}
```
How do we declare and initialize `valuesInCents` to use in a more sophisticated solution of `makeChangeWithOneDenomination()`?

**Answer:**

```c
int [] valuesInCents =
{2000, 1000, 500, 100, 25, 10, 5, 1};
```
Arrays of Strings

- 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";
  ```
Practice! Practice! Practice!

How do we declare and initialize `namesOfBills` to use in a more sophisticated solution of `makeChangeWithOneDenomination()`?

Answer:

```java
String [] namesOfBills = {
    "$20 bill", "$10 bill", "$5 bill", "$1 bill", "quarter", "dimes", "nickel", "cent"};
```
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
import java.util.Scanner;

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!
Practice: Reverse an array!

Given an array arr1 of, say, integers, create a new array arr2 that has its elements in reverse order.
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);
Methods can have arrays as output

//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);
Practice: Compute average and maximum

Given the following array:

```java
int[] arr1 = {31, 52, 13, 24, 45};
```

Write a function `computeAverage(int[] data)`

Write a function `computeMaximum(int[] data)`

Could you do them both at the same time? `computeAvgMax()`

What should your method return?