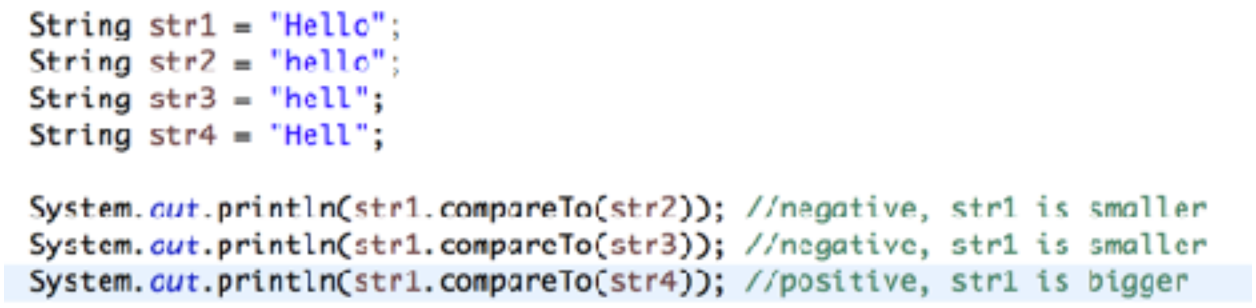
APCS Things to Memorize

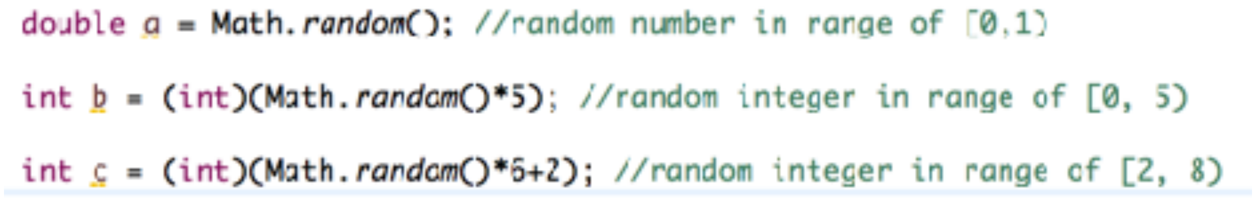
- *compareTo* method
  - Ascii Table (Numbers < Uppercase Letters < Lowercase Letters)
  - Example:



```java
String str1 = "Hello";
String str2 = "hello";
String str3 = "hell";
String str4 = "Hell";

System.out.println(str1.compareTo(str2)); //negative, str1 is smaller
System.out.println(str1.compareTo(str3)); //negative, str1 is smaller
System.out.println(str1.compareTo(str4)); //positive, str1 is bigger
```

- *random* method
  - Generates random *double* type number within range of [0, 1)
    - including 0 but not including 1
  - Application:

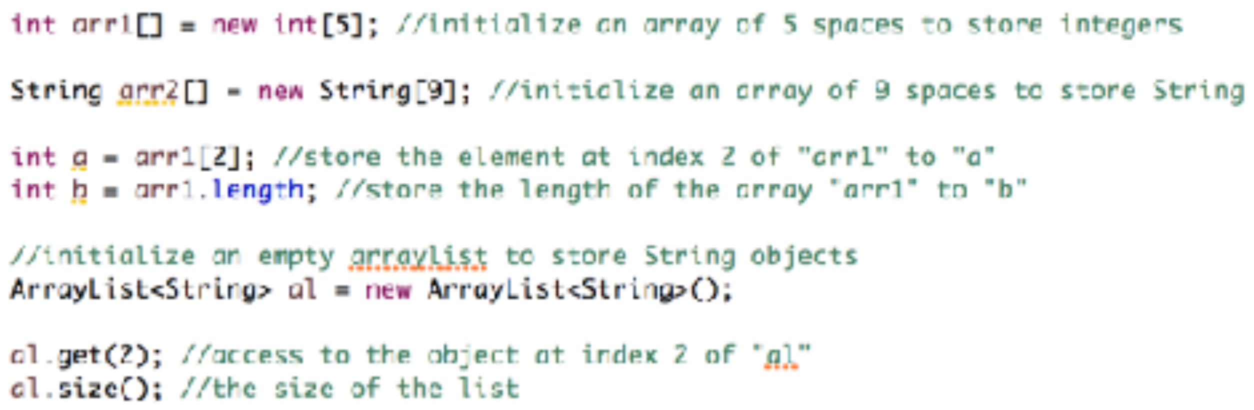


```java
double a = Math.random(); //random number in range of [0,1)

int b = (int)(Math.random()*5); //random integer in range of [0, 5)

int c = (int)(Math.random()*6+2); //random integer in range of [2, 8)
```

- Array & ArrayList



```java
int arr1[] = new int[5]; //initialize an array of 5 spaces to store integers

String arr2[] = new String[9]; //initialize an array of 9 spaces to store String

int a = arr1[2]; //store the element at index 2 of "arr1" to "a"
int b = arr1.length; //store the length of the array "arr1" to "b"

//initialize an empty arraylist to store String objects
ArrayList<String> al = new ArrayList<String>();

al.get(2); //access to the object at index 2 of "al"
al.size(); //the size of the list
```

- Class
  - Create a class called Cookie
    - ***public class Cookie***
  - *Square* class inherit *Rectangle* class
    - ***public class Square extends Rectangle***
  - *Rectangle* class implements *Shape* interface
    - ***public class Rectangle implements Shape***

- Interface
  - Interface does not have variables, constructors
  - It has methods but without implementations
  - It cannot be used to create objects
  - Classes that *implements* the interface must implement the methods

- Abstract class
  - It can have abstract method which has no implementations
  - Class that *extends* the abstract class must implement the abstract method, otherwise, it has to be abstract class as well
  - It cannot be used to create objects

- Method
  - A method that will return a *String* type variable and takes no parameters
    - ***public String Hello()***
  - A method that will return nothing and takes two parameters, one *int,* one *double*
    - ***public void doSomething(int first, double second)***

- Sorting and Searching
  - Sorting always takes more time than searching
  - Generally, *MergeSort* is faster than *InsertionSort* and is faster than *SelectionSort*

- *InsertionSort*
  - Best case: It is already sorted
  - Worst case: It is reversely sorted

- Generally, *BinarySearch* is faster than *SequentialSearch*
- *BinarySearch* (the list has to be sorted)
  - Best case: the target is at the middle
- *SequentialSearch*
  - Best case: the target is at the beginning
  - Worst case: the target is at the end