* Arguments in Java, unlike C++ cannot be passed by reference(**Reference is what they call the location of an object in Java**). However, Java copies and passes the reference by value. Thus passed objects can be altered, but a swap function might fail.
* Array declaration is as below-
  + int[] arr = new int[5];
  + String[] arr = new String[5];

as opposed to- int arr[5] = {1,2,3,4,5} or string arr[5] in C++.

* All data structures are defined under the collections framework, which like C++’s STL provide a similar use and work manner for different data structures.
* The Collections framework is implemented through following architectural layers-
  + Interfaces - Abstract data types that represent collections.
  + Implementations, i.e. Classes - Concrete representations of the interfaces.
  + Algorithms - Methods that perform useful computations on objects that implement collection interfaces.
* Following are the interfaces in the Collections framework-
  + Collection interface
  + List interface
  + Set interface
  + SortedSet interface
  + Map interface
  + Map.Entry interface (see <https://www.tutorialspoint.com/java/java_mapentry_interface.htm> for usage)
  + SortedMap interface
  + Enumeration interface
* Following are the classes implementing the different interfaces in the Collections framework-
  + Abstract Collection
  + AbstractList
  + AbstractSequentialList
  + LinkedList\*
  + ArrayList\*
  + AbstractSet
  + HashSet\*
  + LinkedHashSet\*
  + TreeSet\*
  + AbstractMap
  + HashMap\*
  + TreeMap\*
  + WeakHashMap\*
  + LinkedHashMap\*
  + IdentityHashMap\*

(The classes with ‘\*’ are the ones that can be instantiated as they are not abstract.)

* Following are some of the different algorithms that can be applied to the different classes in the Collections framework. These algorithms are defined as static methods within the Collections framework classes.
  + See - <https://www.tutorialspoint.com/java/java_collection_algorithms.htm>
* Using an iterator in Java-
  + <https://www.tutorialspoint.com/java/java_using_iterator.htm>
* Using a comparator in Java-
  + Both TreeMap and TreeSet store elements in a sorted order. This order is determined by a comparator.
  + The Comparator interface defines two methods compare() and equals()-
    - int compare(Object obj1, Object obj2)
    - boolean equals(Object obj)
  + There is another interface similar to the Comparator interface, called the Comarable interface. This interface defines the compareTo(Object obj) method.
  + There is a very clear difference b/w the Comparator and the Comparable interface.
    - Basically, in the Comparable interface the class whose objects to be sorted must implement this Comparable interface. Whereas in the Comparator interface the class whose objects to be sorted do not need to implement this Comparator interface. The Comparator interface is much like the comparators used by the map and set containers in C++.
    - Great explanation for the above concept here- <https://www.linkedin.com/pulse/whats-difference-between-comparable-comparator-java-mourad-benkadour>
* In Java the equivalent of **templates in C++** are **generics.**
  + See <https://www.tutorialspoint.com/java/java_generics.htm> for usga eof generics in functions and classes.
  + A simple use case of generics in a function is as follows-
    - public static < E > void printArray( E[] inputArray ) {  
       // Display array elements  
       for(E element : inputArray) {  
       System.out.printf("%s ", element);  
       }  
       System.out.println();  
       }
    - The above method will print an array of any type.