********************** Homework: Week 1 Day 2 *************************
Research the following Java datatypes and explain the following:

1. **List**
   A list in contrast to a set allows the storage of duplicate elements in a collection and also allows the user to specify where it is to be stored. Element in the list can be accessed by index. The list interface extends Collection to define an ordered collection with duplicates allowed. The list interface also adds a positioning operation as a new list iterator that enable the user to traverse the list bidirectionally.

2. **ArrayList**
   An ArrayList is an implementation of resizable-array of the List interface. An array is a fixed-size data structure. Once it is created, it has a fixed size that cannot be changed. We can still use arrays to implement dynamic data structures. We will need to create a larger new array to replace the current array, when the current array cannot hold new elements in the list and then copy all the elements from the current array to the new array. The new array now becomes the current array.

3. **Difference in List vs ArrayList**
   - One of the most important differences between List and ArrayList is that List is an interface and ArrayList is a standard Collection class.
   - List interface extends the Collection framework whereas, the ArrayList extends *AbstractList* Class and it implements *Listinterfaces.*
   - The namespace for List interface is *System.Collection.Generic* whereas, the namespace for ArrayList is *System.Collection.*
   - List interface creates a collection of elements that is stored in a sequence and are identified or accessed by their index number. On the other hand, ArrayList creates an array of objects where the array can dynamically grow when required.

4. **HashMap**
   A HashMap is an efficient data structure that store a large data set. A HashMap is a container that stores the elements along with the keys. In an array the index are integers but in an HashMap the indexes are keys and the keys can be objects. A HashMap can not contain duplicate keys. Each key maps to one value. There are tow other types of maps: LinkedHashMap and TreeMap. The Map interface provides the methods for querying, updating, and obtaining a collection of values and set of keys.

5. **HashTable**
   The classes in the java Collections Framework are not thread -safe; this means that their contents may be corrupted if they are accessed and updated concurrently by multiple threads. Data in the collection can be protected by locking the collection or by using synchronized collections. A HashTable is synchronized container that stores elements along with keys. A HashTable is a synchronized collection that can be safely accessed and modified by multiple thread concurrently.

6. **Differences in HashMap and HashTable**
HashTable implements the map interface and is used in the same way as HashMap the only difference is that HashTable is synchronized.

7. **Set**
The Set interface extends the collection interface. An instance of Set contains no duplicate elements. The Set class endures that no duplicates elements can be added to the set. There are three concrete classes of set: HashSet, LinkedHashSet and TreeSet.

8. **HashSet**
The HashSet class is a concrete class that implements Set. A HashSet can be empty if created using a no argument constructor or can be created from an existing collection. In a HashSet you may specify the initial capacity and load factor or just use the default values.
Load factor determines how full the set is allowed to be before its capacity is increased. When the number of elements exceeds the product of the capacity and load factor the capacity is automatically doubled. HashSet can be used to store duplicate-free elements.

9. **ConcurrentHashMap**
A ConcurrentHashMap is a HashTable that supports full concurrency of retrievals and high expected concurrency for updates. The ConcurrentHashMap class obeys the same functional specification as Hashtable, and includes versions of methods corresponding to each method of Hashtable. But still with all operations thread-safe, retrival operation do not entail locking and there is no support for locking the entire table in a way that prevent all access.

10. **What is hashCode and equals methods and how do they differ?**
HashCode is defined in the object class. The HashCode of two objects must be the same if the two objects are equal. Two different objects may have the same HashCode but you should implement the hashCode method to avoid too many such cases.
- The hashCode method must consistently return the same integer when invoked on the same object more than once during execution of the application if the object is not modified.
- the hashCode method on each of the two objects must produce the same integer result If two objects are equal according to the equals(Object) method.
- It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results

The equals methods indicate if two different objects are equals.
The equals method implements an equivalence relation on non-null object references:

- It is reflexive: for any non-null reference value x, x.equals(x) should return true.
- It is symmetric: for any non-null reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.

- It is transitive: for any non-null reference values x, y, and z,
  if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.
- It is consistent: for any non-null reference values x and y, multiple invocations of x.equals(y) consistently return true or consistently return false, provided no information used in equalscomparisons on the objects is modified.
- For any non-null reference value x, x.equals(null) should return false.