\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Homework: Week 1 Day 2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Research the following Java datatypes and explain the following:

1. List
   1. Is a Java interface that extends from the Collection class. Elements can be inserted or removed from the list and elements in a list are by default “sorted” by insertion order. Lists can be iterated and its basic methods (it has more) are add, addAll, remove, get and set.
2. ArrayList
   1. An ArrayList is part of the Collections framework of Java. This class implements the List interface so basically it has access to every method the list has and it is also iterable. What is important about ArrayList is that it behaves like an Array, but it is more powerful. You can add or remove elements from the Array at any given existing index and it will growth or reduce its size depending on the given action.
3. Difference in List vs ArrayList
   1. List is an Interface that some of the Collections classes use (like ArrayList, LinkedList, Vector, Stack), It has its own methods and of course we can use it as a data structure. On the other hand, ArrayList is an implementation of the List interface, it has the List’s methods, but it also has its own methods to which lists have to access.
4. HashMap
   1. It is a data structure that gives access to its content via a match of a key with a value. The key is used to “name” an element, while the value is the element itself. Hashmapsare not ordered, so the its elements are not guaranteed to be returned in the same order they were inserted. Also, Hashmaps are not synchronized which means that more that one thread can access them at the same time. Hashmap also accepts null elements for keys and for values.
5. HashTable
   1. HashTable is a similar data structure to HashMap. However, HashTable is a synchronized class. That means that when a thread is using a HashTable, no other thread can access it because it is locket to the thread using it. This class cannot have null elements, is iterable and is not ordered. Like the HashMap, HashTable also uses pairs of key-value.
6. Differences in HashMap and HashTable
   1. The main differences between these two classes is that HashMap is not synchronized so it can be accessed by more than one thread simultaneously while HashTable is thread safe, this means that when one thread is accessing it, it locks itself so no other thread can access it while it is being accessed. Another difference is that HashMap can have a null key and null values whilst HashTable can only have non null keys and values.
7. Set
   1. A set is a data structure (it is an interface in Java) that is just a “container for data”. It is unlike stacks or queues. Sets cannot have duplicated entries. Sets are the representation of a mathematical finite set. The set class is not an ordered class so the elements will not be returned in the same order they were inserted.
8. HashSet
   1. Is an implementation of the Set interface which is backed by a HashMap instance. Because of this, the order of its elements is not guaranteed, also, HashSets are not synchronized, they can be accessed by more than one thread simultaneously. And HashSets allow to have null elements. Like a set, HashSets do not allow duplicated elements.
9. CocurrentHashMap
   1. ConcurrentHashMap is a data structure that came to fix the problems with HashMaps and HashTables. As we saw on the previous definitions, these two kinds of collections (HashMap and HashTable) have their advantages but also have some flaws. While HashMaps can contain a null key and elements, they can present concurrency errors. On the other hand, HashTables allows thread safety but they only can have non-null elements, and this thread safety comes to a price. When elements from a HashTable has to be accessed many times, it shows a worst performance compared with how a HashMap would behave.

In order to fix this flaws, the Java team implemented the ConcurrentHashMap in the language. Now programmer can have thread safe HashTables but with an efficiency almost as good as in a HashMap.

1. What is hashCode and equals methods and how do they differ
   1. The equals() method compare the equality of two objects. Java does this by checking equality of two objects by checking if both point to the same location in memory and if their values are the same.
   2. The hashCode() method is also used to make a comparison between objects. The method returns an ID that is unique to that object if the hash codes does not match, the objects are not equal, but if the do match, an equals() comparison has to be performed to know if the fields and vales of the objects are the same.
   3. The difference is that hashCode could be more efficient when the objects have different hash codes, but when their hash codes match, other comparisons still have to be performed.