Course 5 - Sprint 1: Store and Manipulate Objects Using Ordered Collections

Learning Consolidation

1. What is the HashSet class in Java and how does it store elements?
    a. java.util.HashSet class is a member of the Java collections framework which inherits the AbstractSet class and implements the Set interface. It implicitly implements a hashtable for creating and storing a collection of unique elements. Hashtable is an instance of the HashMap class that uses a hashing mechanism for storing the information within a HashSet. Hashing is the process of converting the informational content into a unique value that Is more popularly known as hash code. This hashcode is then used for indexing the data associated with the key. The entire process of transforming the informational key into the hashcode is performed internally.

2. Can you add a null element into a TreeSet or HashSet?
    a. In HashSet, only one null element can be added but in TreeSet it can’t be added as it makes use of NavigableMap for storing the elements. This is because the NavigableMap is a subtype of SortedMap that doesn’t allow null keys. So, in case you try to add null elements to a TreeSet, it will throw a NullPointerException.

3. Differentiate between ArrayList and LinkedList.

ArrayList	**LinkedList**
Implements dynamic array internally to store elements	Implements doubly linked list internally to store elements
Manipulation of elements is slower	Manipulation of elements is faster
Can act only as a List	Can act as a List and a Queue
Effective for data storage and access	Effective for data manipulation

4. Differentiate between Iterator and ListIterator.

Iterator	**ListIterator**
Can only perform remove operations on the Collection elements	Can perform add, remove and replace operations the Collection elements
Can traverse List, Sets and maps	Can traverse only Lists
Can traverse the Collection in forward direction	Can traverse the collection in any direction
Provides no method to retrieve the index of the element	Provides methods to retrieve the index of the elements
iterator() method is available for the entire Collection Framework	listIterator() is only available for the collections implementing the List interface

5. Why Collection interface does not extend Cloneable and Serializable interface?
   a. Well, simplest answer is “there is no need to do it”. Extending an interface simply means that you are creating a subtype of interface, in other words a more specialized behavior and Collection interface is not expected to do what Cloneable and Serializable interfaces do.
   b. Another reason is that not everybody will have a reason to have Cloneable collection because if it has very large data, then every unnecessary clone operation will consume a big memory. Beginners might use it without knowing the consequences.
   c. Another reason is that Cloneable and Serializable are very specialized behavior and so should be implemented only when required. For example, many concrete classes in collection implement these interfaces. So, if you want this feature. use these collection classes otherwise use their alternative classes.

6. Which implementation of set would you choose if you want the iterator of set would give you objects in the order in which it was inserted?

    a. LinkedHashSet
    b. TreeSet
    c. HashSet

7. Which of the following implementation will you use if you were to insert elements at any position in the collections?

    a. ArrayList
    b. LinkedList
    c. Vector

8. How do you get immutable object of a collection? For example if you were to write an API which return a List or a Set or a Map when a method is called, but you also want that you don't want the client of your API to add or delete any object in the returned collection?

    a. Use Collections.immutableCollection() method
    b. Use the Collections.unmodifiableXxxx()method with the collection as an argument, which returns an immutable object of specific type.