Core Specific- Java Assignment 6

Q1. What is Collection in Java?

Ans:

The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

Java Collections can achieve all the operations that you perform on a data such as searching, sorting, insertion, manipulation, and deletion.

Java Collection means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet).

Q2. Differentiate between Collection and collections in the context of Java.

Ans:

Collection is the interface where you group objects into a single unit.

Collections is a utility class that has some set of operations you perform on Collection.

Collection does not have all static methods in it, but Collections consist of methods that are all static.

Q3. What are the advantages of the Collection framework?

Ans:

Advantages of collections framework

- Not necessary to learn multiple ad hoc collection APIs.
- It provides a standard interface for collections and also provides algorithms to manipulate them.
- It reduces the programming efforts by providing useful data structures and algorithms.

Core Specific- Java Assignment 6

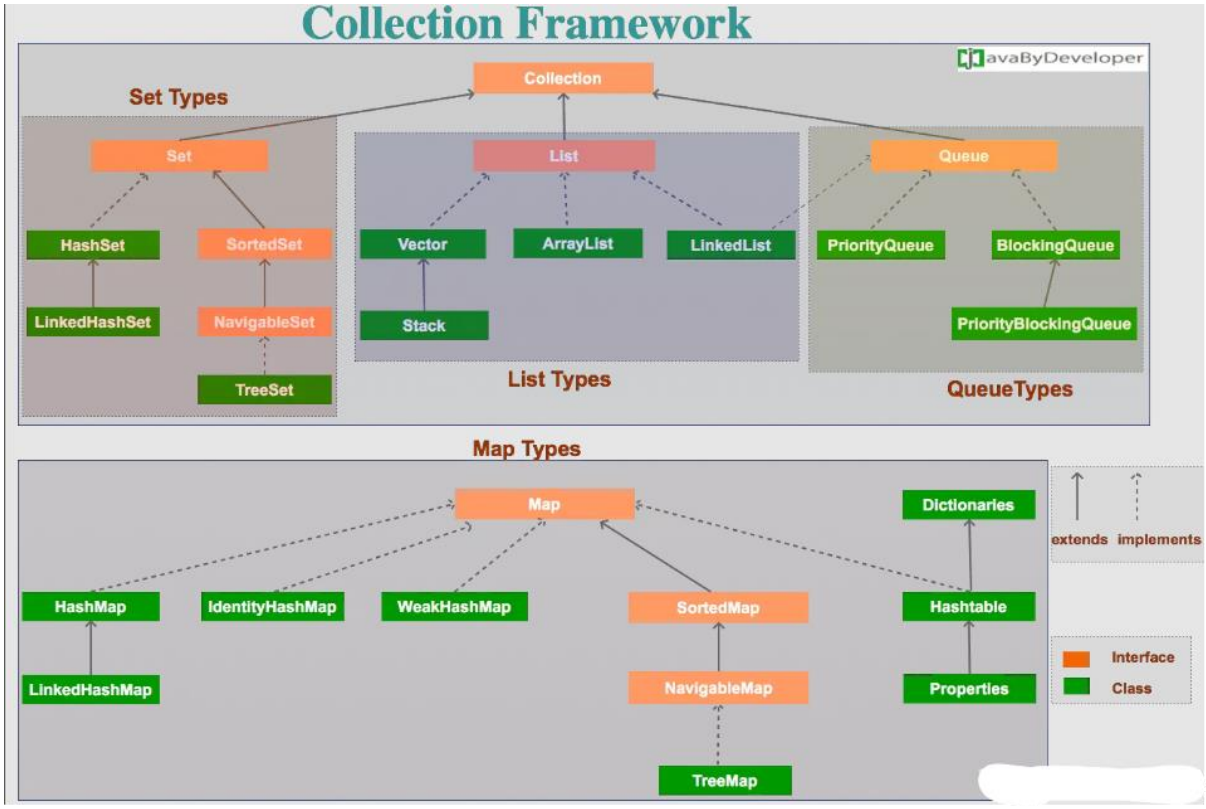
Q4. Explain the various interfaces used in the Collection framework.

Ans:

Collection Framework provides many interfaces, in this tutorial I am going to explain the major Interfaces and these are key Interfaces of Collection Framework. They are:

1. Collection
2. List
3. Set
4. SortedSet
5. NavigableSet
6. Queue
7. Map
8. SortedMap
9. NavigableMap

The following image illustrates the key interfaces of Collection Framework and its Implementation classes hierarchy. Which will give you the high level overview of Collection framework.



Q5. Differentiate between List and Set in Java.

Ans:

Difference between List and Set:

List	Set
1. The List is an indexed sequence.	1. The Set is an non-indexed sequence.
2. List allows duplicate elements	2. Set doesn't allow duplicate elements.
3. Elements by their position can be accessed.	3. Position access to elements is not allowed.
4. Multiple null elements can be stored.	4. Null element can store only once.
5. List implementations are ArrayList, LinkedList, Vector, Stack	5. Set implementations are HashSet, LinkedHashSet.

Q6. What is the Differentiate between Iterator and ListIterator in Java.

Ans:

Property	Iterator	ListIterator
Direction of Traversal	It can traverse elements present in a Collection only in the forward direction.	It can traverse elements present in a Collection both in forward and backward directions.
Traverasability	It is used to traverse Map, List, and Set.	It is used only to traverse List. That is it can't traverse Map and Set.
Addition of elements	Iterator cannot add elements to a Collection.	ListIterator can add elements to the Collection.

Modification Possible	An Iterator cannot modify the elements in the Collection.	A ListIterator can modify the elements in a Collection using a set().
Obtaining Index	An Iterator has no method to obtain an index of the element in a given Collection.	ListIterator can be used to obtain the index of an element in the Collection.
Methods available	It has methods like hasNext(), next(), remove().	It has methods like add(E e), hasNext(), hasPrevious(), next(), nextIndex(), previous(), previousIndex(), remove(), set(E e).

Q7. What is the Differentiate between Comparable and Comparator.

Ans:

Comparable	Comparator
1) Comparable provides a **single sorting sequence**. In other words, we can sort the collection on the basis of a single element such as id, name, and price.	The Comparator provides **multiple sorting sequences**. In other words, we can sort the collection on the basis of multiple elements such as id, name, and price etc.
2) Comparable **affects the original class**, i.e., the actual class is modified.	Comparator **doesn't affect the original class**, i.e., the actual class is not modified.
3) Comparable provides **compareTo() method** to sort elements.	Comparator provides **compare() method** to sort elements.
4) Comparable is present in **java.lang** package.	A Comparator is present in the **java.util** package.
5) We can sort the list elements of Comparable type by **Collections.sort(List)** method.	We can sort the list elements of Comparator type by **Collections.sort(List, Comparator)** method.

Q8. What is collision in HashMap?

Ans: Collisions in the HashMap

A collision, or more specifically, a hash code collision in a HashMap, is a situation where two or more key objects produce the same final hash value and hence point to the same bucket location or array index.

Q9. Distinguish between a hashmap and a Treemap.

Ans:

Basis of comparison	HashMap	TreeMap
Basic	HashMap does not keep track of the order of insertions.	TreeMap preserves insertion order.
Interface Implements	Map, Cloneable, and Serializable interfaces are all ones that are implemented by HashMap.	TreeMap is capable of being Cloned and Serialized, in addition to implementing the NavigableMap interface.
Data Structure	A Hash Table serves as the foundation for HashMap's underlying data structure.	The Red-Black Tree is the foundational data structure that TreeMap is built upon.
Null Keys and Values	The Null key can be used once in HashMap, and the Null value can be used any number of times.	TreeMap does not let the use of a null key, but it does permit the use of a null value any number of times.
Extends and implements	The HashMap class is an extension of the AbstractMap class and an implementation of the Map interface.	The TreeMap class extends the AbstractMap base class and implements the SortedMap and NavigableMap interfaces respectively.

Basis of comparison	HashMap	TreeMap
Performance	HashMap processes operations more quickly.	When compared to HashMap, the operation speed of TreeMap is lower.
Order of elements	HashMap does not keep track of order.	The elements are arranged in their natural order (ascending).
Homogeneous/ Heterogeneous	HashMap supports heterogeneous elements because it does not perform key sorting.	Because of sorting, TreeMap allows homogeneous values as keys.
Uses	When we do not want key-value pairs to be in sorted order, the HashMap data structure should be utilised.	When we need the key-value pair to be in sorted (ascending) order, we should utilise the TreeMap.

Q10. Define LinkedHashMap in Java.

Ans:

A LinkedHashMap contains values based on the key. It implements the Map interface and extends the HashMap class. It contains only unique elements. It may have one null key and multiple null values.