PPT JAVA ASSIGNMENT-6

Q.1 What is Collection in Java?

A Collection represents a single unit of objects, i.e., a group.

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).

Q.2 Differentiate between Collection and collections in the context of Java.

Collection: Collection is a interface present in java.util.package. It is used to represent a group of individual objects as a single unit. The collection is considered as the root interface of the collection framework. It provides several classes and interfaces to represent a group of individual objects as a single unit.

The List, Set, and Queue are the main sub-interfaces of the collection interface. The map interface is also part of the java collection framework, but it doesn't inherit the collection of the interface. The add(), remove(), clear(), size(), and contains() are the important methods of the Collection interface.

Collections: Collections is a utility class present in java.util.package. It defines several utility methods like sorting and searching which is used to operate on collection. It has all static methods. These methods provide much-needed convenience to developers, allowing them to effectively work with Collection Framework. For example, It has a method sort() to sort the collection elements according to default sorting order, and it has a method min(), and max() to find the minimum and maximum value respectively in the collection elements.

Q.3 What are the advantages of the Collection framework?

- Collection is re-sizable or dynamically draw-able memory.
- Provides useful data structures in the form of predefined classes that reduces programming affords.
- It support to store heterogeneous elements or object.
- Collections accept both Homogeneous and Heterogeneous objects.
- It provides higher performance.
- If you want to store the group of individual objects into a single entity then we should go for collections.
- We no need to provide fixed size to store data into the collections means collections are resizable in nature.
- Every collection class having particular underlying data structure which is useful to delete, insert, and retrieve elements present in collections without providing any external code.
- By using the concept of generics we can avoid typesafe problem.
- It provides Extendability (depends on incoming flow of data,if the size of collection framework variable is increasing than the collection framework variable is containing Extendability feature).
- It provides adaptability facility(The process of adding the content of one collection framework variable to another collection framework either in the beginning or in the ending or in the middle in known as adaptability).
- It is one of the algorithmic oriented.
- It provides in-built sorting technique.
- It provides in-built searching technique.
- It provides higher preliminary concepts of Data Structure such as:- Stack,Queue,LinkedList,Trees ..etc.

- Reduces programming effort
- Increases program speed and quality

Q.4 Explain the various interfaces used in the Collection framework.

The core collection interfaces within the Java Collection framework are as follows:

- **List:** The List interface extends the Collection interface and represents an ordered collection of elements. Lists allow duplicate elements and maintain the insertion order. Common implementations of List include ArrayList, LinkedList, and Vector.
- **Set:** The Set interface, also an extension of the Collection interface, represents a collection that does not allow duplicate elements. Sets typically do not maintain a specific order of elements. Notable implementations of Set are HashSet, TreeSet, and LinkedHashSet.
- **Queue:** The Queue interface defines a collection that represents a waiting area, where elements are inserted at one end and removed from the other. Queues follow the First-In-First-Out (FIFO) principle. Notable implementations of Queue include LinkedList and PriorityQueue.
- **Deque:** The Deque interface extends the Queue interface and represents a double-ended queue, allowing elements to be inserted and removed from both ends. Deques support operations at both ends, enabling flexibility in data handling. Common implementations of Deque include ArrayDeque and LinkedList.
- **Map:** The Map interface represents a mapping between unique keys and corresponding values. It does not extend the Collection interface but is an important part of the Java Collection framework. Maps do not allow duplicate keys and are commonly

used for key-value pair associations. Notable implementations of Map include HashMap, TreeMap, and LinkedHashMap.

Q.5 Differentiate between List and Set in Java.

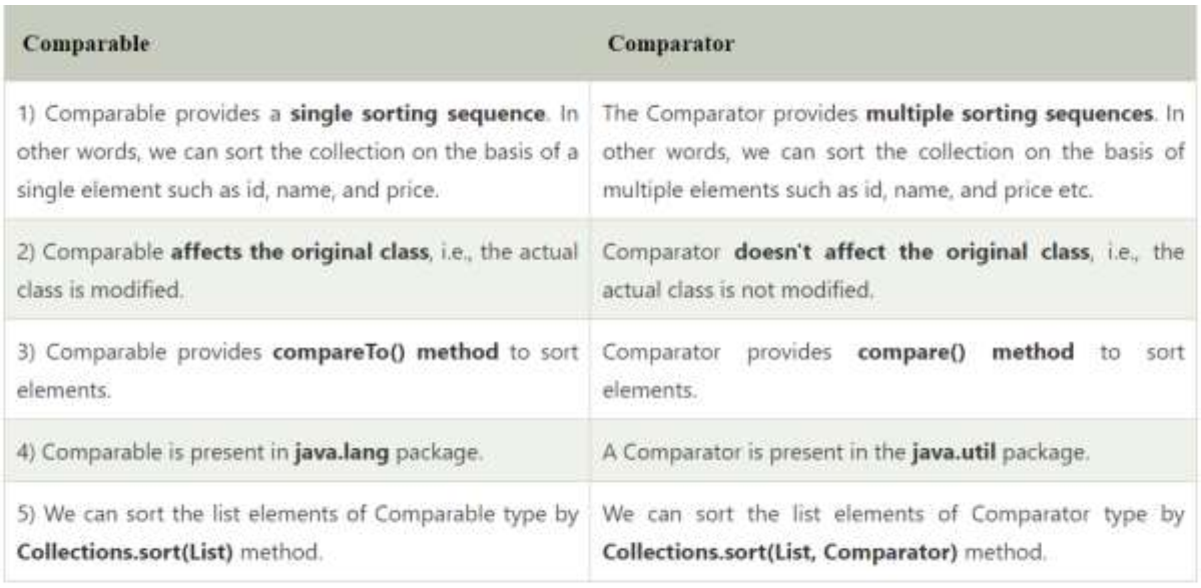
1.	The list implementation allows us to add the same or duplicate elements.	The set implementation doesn't allow us duplicate elements.
2.	The insertion order is maintained by the List.	It doesn't maintain the insertion order of e
3.	List allows us to add any number of null values.	Set allows us to add at least one null value
4.	The List implementation classes are LinkedList and ArrayList.	The Set implementation classes are Tre LinkedHashSet.
5.	We can get the element of a specified index from the list using the get() method.	We cannot find the element from the Set because it doesn't provide any get methoc
6.	It is used when we want to frequently access the elements by using the index.	It is used when we want to design a c elements.
7.	The method of List interface listiterator() is used to iterate the List elements.	The iterator is used when we need to itera

Q.6 What is the Differentiate between Iterator and ListIterator in Java.

Iterator	ListIterator
Can traverse elements present in Collection only in the forward direction.	Can traverse elements present in Collection both in forward and backward directions.
Helps to traverse Map, List and Set.	Can only traverse List and not the other two.
Indexes cannot be obtained by using Iterator.	It has methods like nextIndex() and previousIndex() to obtain

Iterator	ListIterator
	indexes of elements at any time while traversing List.
Cannot modify or replace elements present in Collection	We can modify or replace elements with the help of set(E e)
Cannot add elements and it throws ConcurrentModificationException.	Can easily add elements to a collection at any time.
Certain methods of Iterator are next(), remove() and hasNext().	Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

Q.7 What is the Differentiate between Comparable and Comparator



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.

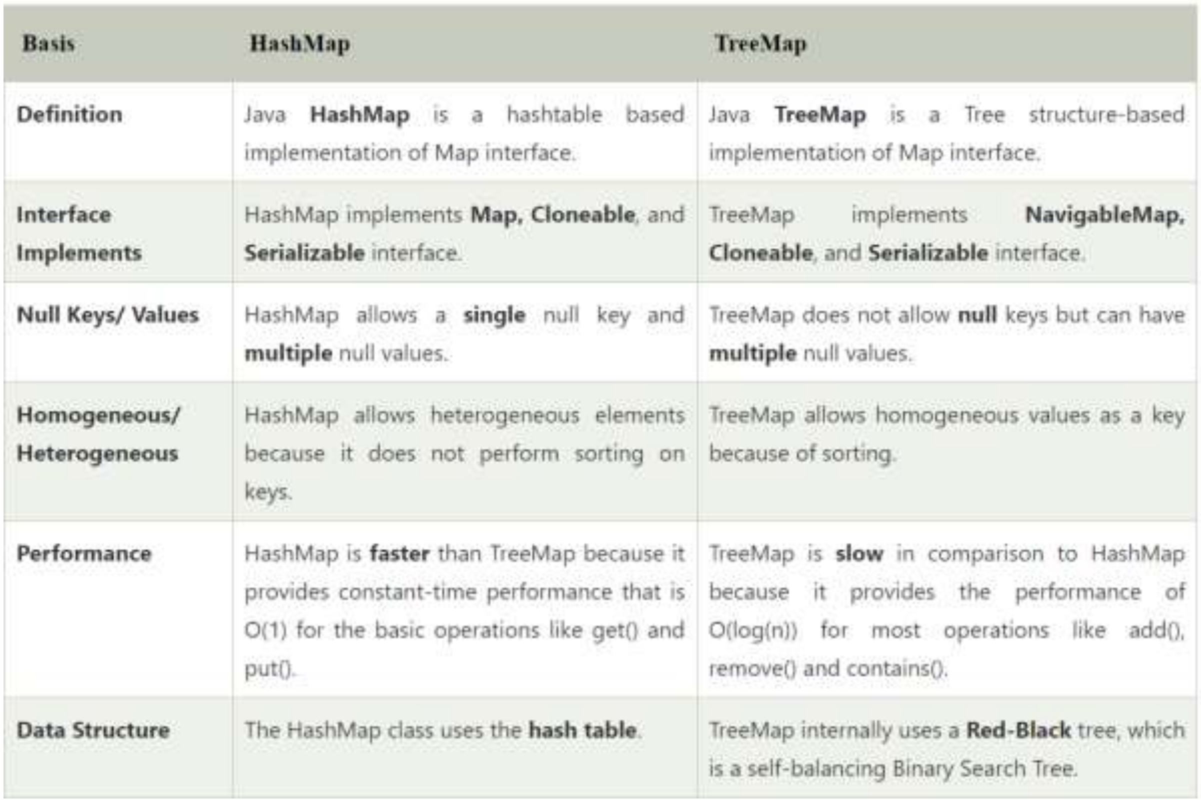
Q.8 What is collision in HashMap?

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.

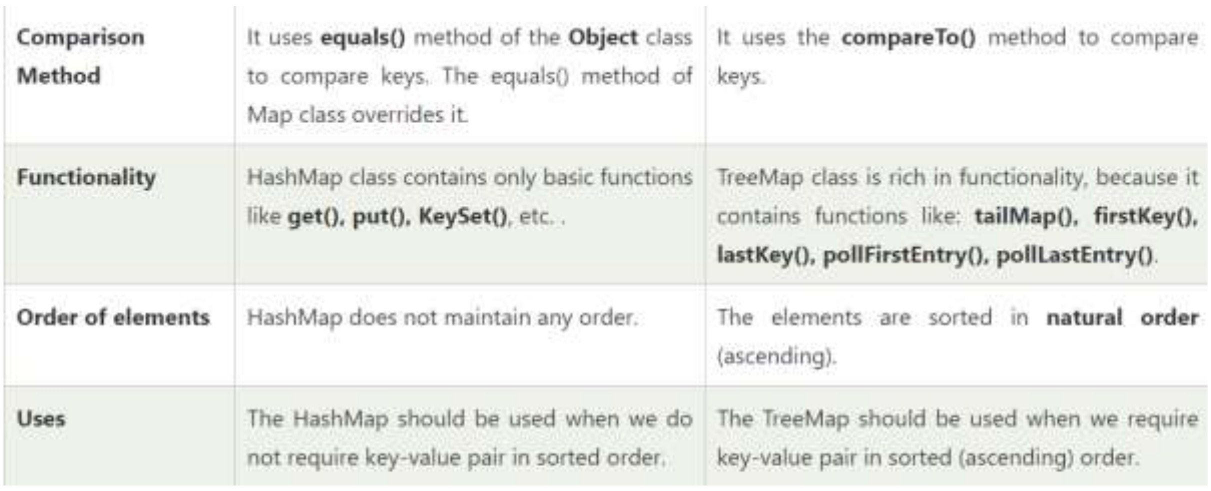
If the hash function return a slot that is already occupied there is a collision.

Collision happens when multiple keys hash to the same bucket. In that case, you need to make sure that you can distinguish between those keys.

Q.9 Distinguish between a hashmap and a Treemap.



Basis	HashMap	TreeMap
Definition	Java **HashMap** is a hashtable based implementation of Map interface.	Java **TreeMap** is a Tree structure-based implementation of Map interface.
Interface Implements	HashMap implements **Map, Cloneable**, and **Serializable** interface.	TreeMap implements **NavigableMap, Cloneable**, and **Serializable** interface.
Null Keys/ Values	HashMap allows a **single** null key and **multiple** null values.	TreeMap does not allow **null** keys but can have **multiple** null values.
Homogeneous/ Heterogeneous	HashMap allows heterogeneous elements because it does not perform sorting on keys.	TreeMap allows homogeneous values as a key because of sorting.
Performance	HashMap is **faster** than TreeMap because it provides constant-time performance that is O(1) for the basic operations like get() and put().	TreeMap is **slow** in comparison to HashMap because it provides the performance of O(log(n)) for most operations like add(), remove() and contains().
Data Structure	The HashMap class uses the **hash table**.	TreeMap internally uses a **Red-Black** tree, which is a self-balancing Binary Search Tree.



Comparison Method	It uses **equals()** method of the **Object** class to compare keys. The equals() method of Map class overrides it.	It uses the **compareTo()** method to compare keys.
Functionality	HashMap class contains only basic functions like **get(), put(), KeySet()**, etc. .	TreeMap class is rich in functionality, because it contains functions like: **tailMap(), firstKey(), lastKey(), pollFirstEntry(), pollLastEntry()**.
Order of elements	HashMap does not maintain any order.	The elements are sorted in **natural order** (ascending).
Uses	The HashMap should be used when we do not require key-value pair in sorted order.	The TreeMap should be used when we require key-value pair in sorted (ascending) order.

Q.10 Define LinkedHashMap in Java.

The LinkedHashMap Class is just like HashMap with an additional feature of maintaining an order of elements inserted into it. HashMap provided the advantage of quick insertion, search, and deletion but it never maintained the track and order of insertion, which the LinkedHashMap provides where the elements can be accessed in their insertion order.