Assignment Questions 6

Q1.What is Collection in Java?

ans:-

Collection can simply be described as an Interface. With the help of Collection, we can easily group various objects into a single unit. Collection forms the root or head of the hierarchy of interfaces. If you are well aware of the C++ language, consider the concept of Collection to be very similar to Containers in C++ language. The other sub interfaces of this hierarchy are Sets, Lists, Maps, Queue, Deque, etc.

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

ans:-

Collection->

i. It is an interface.
ii. It is used to represent a group of individual objects as a single unit.
iii. The Collection is an interface that contains a static method since java8. The Interface can also contain abstract and default methods.

Collections->

i. It is a utility class.
ii. It defines several utility methods that are used to operate on collection.
iii. It contains only static methods.

```
import java.io.*;
import java.util.*;

class Test {

	public static void main (String[] args)
	{
	List<String> arrlist = new ArrayList<String>();
	arrlist.add("geeks");
	arrlist.add("for");
	arrlist.add("geeks");
	System.out.println("Elements of arrlist before the operations:");
	System.out.println(arrlist);
	System.out.println("Elements of arrlist after the operations:");
	Collections.addAll(arrlist, "web", "site");
	System.out.println(arrlist);
	Collections.sort(arrlist);
	System.out.println(arrlist);

	}
}
```

Q3. What are the advantages of the Collection framework?

ans:-

1.We need not to learn multiple ad hoc collection APIs.

2.It provides a standard interface for collections that fosters software reuse and also provides algorithms to manipulate them.

3.Reduces the effort required to design and implement APIs by eliminating the need to produce ad hoc collections APIs.

4.It provides useful data structures and algorithms that reduces programming effort due to which we need not to write them ourselves.

5.It provides high-performance implementations of useful data structures and algorithms that increases the performance.

6.Helps in establishing a common language to pass collections back and forth that provides interoperability between unrelated APIs.

7.Collection is resizable and can grow.

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

ans:-

The Collection Interface

It is at the top of collection heirarchy and must be implemented by any class that defines a collection. Its general declaration is,

interface Collection <E>

The List Interface

It extends the Collection Interface, and defines storage as sequence of elements. Following is its general declaration,

interface List <E>

Allows random access and insertion, based on position.

It allows Duplicate elements.

List Interface Methods

Apart from methods of Collection Interface, it adds following methods of its own.

Methods Description

Object get(int index) Returns object stored at the specified index

Object set(int index, E obj) Stores object at the specified index in the calling collection

int indexOf(Object obj) Returns index of first occurrence of obj in the collection

int lastIndexOf(Object obj) Returns index of last occurrence of obj in the collection

List subList(int start, int end) Returns a list containing elements between start and end index in the collection

The Set Interface

This interface defines a Set. It extends Collection interface and doesn't allow insertion of duplicate elements. It's general declaration is,

interface Set <E>

It doesn't define any method of its own. It has two sub interfaces, SortedSet and NavigableSet.

SortedSet interface extends Set interface and arranges added elements in an ascending order.

NavigabeSet interface extends SortedSet interface, and allows retrieval of elements based on the closest match to a given value or values.

The Queue Interface

It extends collection interface and defines behaviour of queue, that is first-in, first-out. It's general declaration is,

interface Queue <E>

Queue Interface Methods

There are couple of new and interesting methods added by this interface. Some of them are mentioned in below table.

Methods Description

Object poll() removes element at the head of the queue and returns null if queue is empty

Object remove() removes element at the head of the queue and throws NoSuchElementException if queue is empty

Object peek() returns the element at the head of the queue without removing it. Returns null if queue is empty

Object element() same as peek(), but throws NoSuchElementException if queue is empty

boolean offer(E obj) Adds object to queue.

The Dequeue Interface

It extends Queue interface and implements behaviour of a double-ended queue. Its general declaration is,

interface Dequeue <E>

Since it implements Queue interface, it has the same methods as mentioned there.

Double ended queues can function as simple queues as well as like standard Stacks.

Q5.Differentiate between List and Set in Java.

ans:-

List

1. The List is an indexed sequence.

2. List allows duplicate elements

3. Elements by their position can be accessed.

4. Multiple null elements can be stored.

5. List implementations are ArrayList, LinkedList, Vector, Stack

Set

1. The Set is an non-indexed sequence.

2. Set doesn't allow duplicate elements.

3. Position access to elements is not allowed.

4. Null element can store only once.

5. Set implementations are HashSet, LinkedHashSet.

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

ans:-

Iterator

i. Can traverse elements present in Collection only in the forward direction.
ii. Helps to traverse Map, List and Set.
iii. Indexes cannot be obtained by using Iterator.

iv. Cannot modify or replace elements present in Collection
v. Cannot add elements and it throws ConcurrentModificationException.
vi. Certain methods of Iterator are next(), remove() and hasNext().

ListIterator

i. Can traverse elements present in Collection both in forward and backward directions.
ii. Can only traverse List and not the other two.
iii. It has methods like nextIndex() and previousIndex() to obtain indexes of elements at any time while traversing List.
iv. We can modify or replace elements with the help of set(E e)
v. Can easily add elements to a collection at any time.
vi. Certain methods of ListIterator are next(), previous(), hasNext(), hasPrevious(), add(E e).

Q7.What is the Differentiate between Comparable and Comparator

ans:-

Comparable

i. 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
ii. Comparable affects the original class, i.e., the actual class is modified.
iii. Comparable provides compareTo() method to sort elements.
iv. Comparable is present in java.lang package.
v. We can sort the list elements of Comparable type by Collections.sort(List) method.

Comparator

i. 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

ii. Comparator doesn't affect the original class, i.e., the actual class is not modified.
iii. Comparator provides compare() method to sort elements.
iv. A Comparator is present in the java.util package.
v. We can sort the list elements of Comparator type by Collections.sort(List, Comparator) method.

Q8.What is collision in HashMap?

ans:-

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:-

HashMap

i. HashMap implements Map<K, V>, Cloneable and Serializable interface. It extends AbstractMap<K, V> class. It belongs to java.util package.
ii. HashMap contains value based on the key.
iii. It may have a single null key and multiple null values.
iv. HashMap does not maintain order while iterating.
v. It contains unique elements.
vi. It works on the principle of hashing.

TreeMap

i. TreeMap class extends AbstractMap<K, V> class and implements NavigableMap<K, V >, Cloneable, and Serializable interface. TreeMap is an example of a SortedMap. It is implemented by the Red-Black tree, which means that the order of the keys is sorted.
ii. TreeMap also contains value based on the key.
iii. TreeMap is sorted by keys.
iv. It contains unique elements.
v. It cannot have a null key but have multiple null values.
vi. Keys are in ascending order.
vii. It stores the object in the tree structure.

Q10.Define LinkedHashMap in Java

ans:-

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.

Important Features of a LinkedHashMap are listed as follows:

i. A LinkedHashMap contains values based on the key. It implements the Map interface and extends the HashMap class.
ii. It contains only unique elements.
iii. It may have one null key and multiple null values.
iv. It is non-synchronized.

It is the same as HashMap with an additional feature that it maintains insertion order. For example, when we run the code with a HashMap, we get a different order of elements.