Q1.What is Collection in Java?

Ans : Collections in Java is a framework that stores and manipulates a group of objects. It is a hierarchy of interfaces and classes that provides easy management of a group of objects.

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

Ans : Collection vs Collections in Java

* Collection is called interface in java whereas Collections is called a utility class in java and both of them can be found in java.util.package.
* Collection is used to represent a single unit with a group of individual objects whereas collections is used to operate on collection with several utility methods.
* Since [java 8](https://www.simplilearn.com/java-8-interview-questions-and-answers-article), collection is an interface with static as well as abstract and default methods whereas collections operate only with static methods.

Q3. What are the advantages of the Collection framework?

Ans : Main benefits of Collections Framework in Java

I. Reusability: Java Collections Framework provides common classes and utility methods than can be used with different types of collections. This promotes the reusability of the code. A developer does not have to re-invent the wheel by writing the same method again.

II. Quality: Using Java Collection Framework improves the program quality, since the code is already tested and used by thousands of developers.

III. Speed: Most of programmers report that their development speed increased since they can focus on core logic and use the generic collections provided by Java framework.

IV. Maintenance: Since most of the Java Collections framework code is open source and API documents is widely available, it is easy to maintain the code written with the help of Java Collections framework. One developer can easily pick the code of previous developer.

V. Reduces effort to design new APIs: This is the flip side of the previous advantage. Designers and implementers don't have to reinvent the wheel each time they create an API that relies on collections; instead, they can use standard collection interfaces.

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

Ans : 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.

Q5. Difference between List and Set in Java.

Ans :

|  |  |  |
| --- | --- | --- |
| S.No | List | Set |
| 1. | The list implementation allows us to add the same or duplicate elements. | The set implementation doesn't allow us to add the same or duplicate elements. |
| 2. | The insertion order is maintained by the List. | It doesn't maintain the insertion order of elements. |
| 3. | List allows us to add any number of null values. | Set allows us to add at least one null value in it. |
| 4. | The List implementation classes are LinkedList and ArrayList. | The Set implementation classes are TreeSet, HashSet and 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 based on the index because it doesn't provide any get method(). |
| 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 collection of distinct elements. |
| 7. | The method of List interface listiterator() is used to iterate the List elements. | The iterator is used when we need to iterate the Set elements. |

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

Ans :

| BASIS FOR COMPARISON | ITERATOR | LISTITERATOR |
| --- | --- | --- |
| Basic | Iterator can traverse the elements in a collection only in forward direction. | ListIterator can traverse the elements in a collection in forward as well as the backwards direction. |
| Add | Iterator is unable to add elements to a collection. | ListIteror can add elements to a collection. |
| Modify | Iterator can not modify the elements in a collection. | ListIterator can modify the elements in a collection using set(). |
| Traverse | Iterator can traverse Map, List and Set. | ListIterator can traverse List objects only. |
| Index | Iterator has no method to obtain an index of the element in a collection. | Using ListIterator, you can obtain an index of the element in a collection. |

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 : 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. Difference between hashmap and treemap in java

Ans : HashMap :-

HashMap implements Map<K, V>, Cloneable and Serializable interface. It extends AbstractMap<K, V> class. It belongs to java.util package.

* HashMap contains value based on the key.
* It may have a single null key and multiple null values.
* HashMap does not maintain order while iterating.
* It contains unique elements.
* It works on the principle of hashing.

TreeMap :-

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.

* TreeMap also contains value based on the key.
* TreeMap is sorted by keys.
* It contains unique elements.
* It cannot have a null key but have multiple null values.
* Keys are in ascending order.
* It stores the object in the tree structure.

Q10.Define LinkedHashMap in Java.

Ans : The LinkedHashMap class of the Java collections framework provides the hash table and linked list implementation of the Map interface. The LinkedHashMap interface extends the HashMap class to store its entries in a hash table. It internally maintains a doubly-linked list among all of its entries to order its entries.