**Assignment Questions 6**

💡 **Q1.What is Collection in Java?**

In Java, a collection is a framework that provides a set of interfaces and classes to store and manipulate groups of objects. It allows dynamic resizing, easy insertion, deletion, and retrieval of elements, providing different data structures like lists, sets, and maps to manage data efficiently.

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

1. Collection: It is a framework (java.util.Collection) that provides interfaces like List, Set, and Queue to represent groups of objects. It focuses on data storage and manipulation.

2. collections: It often refers to utility classes (java.util.Collections) providing static methods to work with collections, like sorting, searching, and synchronizing. These classes assist in managing collections effectively.

💡 **Q3. What are the advantages of the Collection framework?**

1. Abstraction: Provides standard interfaces and implementations, promoting code reusability and easy swapping of data structures.

2. Dynamic resizing: Collections automatically resize to accommodate elements, reducing manual memory management.

3. Data manipulation: Offers convenient methods for insertion, deletion, retrieval, and iteration of elements.

4. Improved performance: Efficient algorithms for common operations enhance performance.

5. Stream API: Enables functional-style operations on collections.

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

1. List: Ordered collection with duplicates, accessible by index.

2. Set: Unordered collection with no duplicates, often implementing equals() and hashCode().

3. Queue: Ordered collection for elements processing (FIFO or LIFO) like LinkedList and PriorityQueue.

4. Map: Key-value pairs mapping, no duplicate keys allowed (e.g., HashMap and TreeMap).

5. Collection: The root interface for collections, representing a group of objects.

💡 **Q5.Differentiate between List and Set in Java.**

List:

Ordered: Elements are stored in the order of insertion and can be accessed by index.

Duplicates: Allows duplicate elements.

Implementation: Common implementations are ArrayList and LinkedList.

Example: [1, 2, 3, 2]

Set:

Unordered: No defined order for elements.

Duplicates: Does not allow duplicate elements.

Implementation: Common implementations are HashSet and TreeSet.

Example: {1, 2, 3}

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

Iterator:

1. Supports all collection types.

2. Unidirectional: Traverse elements in one direction (forward).

3. Methods: Has hasNext() and next() methods.

4. Common usage: Used with Lists, Sets, and Queues.

ListIterator:

1. Supports Lists only.

2. Bidirectional: Traverse elements in both directions (forward and backward).

3. Methods: Has hasNext(), next(), hasPrevious(), and previous() methods.

4. Common usage: Used with Lists to traverse and modify elements.

💡 **Q7.What is the Differentiate between Comparable and Comparator**

Comparable:

1. Defined in the object's class.

2. Natural ordering: Specifies the default comparison logic for the object.

3. Method: Contains compareTo() to implement the comparison.

Comparator:

1. Separate class or lambda expression.

2. Custom ordering: Allows multiple comparison logics for objects.

3. Method: Contains compare() to implement the comparison.

💡 **Q8.What is collision in HashMap?**

In HashMap, collision occurs when two or more keys are mapped to the same hash code, causing them to be stored in the same bucket. To handle collisions, HashMap uses a linked list or a balanced tree to store multiple key-value pairs in the same bucket.

💡 **Q9.Distinguish between a hashmap and a Treemap.**

HashMap:

1. No natural order: Elements are not sorted.

2. Faster: Provides constant-time complexity for most operations.

3. No additional memory overhead for ordering.

4. Implementation: Uses hashing for key-value storage.

TreeMap:

1. Natural order: Elements are sorted based on their keys.

2. Slower: Provides logarithmic-time complexity for most operations.

3. Additional memory overhead for ordering (red-black tree).

4. Implementation: Uses balanced red-black tree for key-value storage.

💡 **Q10.Define LinkedHashMap in Java**

LinkedHashMap in Java is a subclass of HashMap that maintains the insertion order of its elements. It combines a hash table and a doubly-linked list to store key-value pairs, providing predictable iteration order based on the order of insertion or access.