Assignment 6

Q1.What is Collection in Java?

ans) In Java, a Collection is a framework provided by the Java Collections Framework (JCF) that represents a group of objects. It is used to store, manipulate, and retrieve a collection of elements.

The Collection framework provides several interfaces such as List, Set, and Queue, along with their respective implementation classes such as ArrayList, HashSet, and LinkedList. These interfaces and classes provide different ways to store and organise objects based on specific requirements.

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

ans)

Collectin	Collections
Collection is a Java interface that represents a group of objects and provides a set of operations to work with them.	Collections is a class in Java that provides utility methods for working with collections
It is part of the Java Collections Framework and defines common behavior for classes that implement it.	It contains static methods that operate on or return collections, such as sorting, searching, and manipulating collections.

It provides methods for adding, removing, querying, and manipulating elements in a collection.	It provides algorithms and algorithms-based methods to perform common operations on collections, such as binarySearch, sort, reverse, etc.

Q3. What are the advantages of the Collection?

ans) The advantages of using Collection in Java are as follows:

1. Abstraction: Collections provide a high-level abstraction for working with groups of objects, allowing developers to focus on the functionality rather than low-level implementation details.

2. Flexibility: Collections offer a wide range of data structures and algorithms to choose from, such as lists, sets, queues, and maps.

3. Efficiency: Collections provide optimised implementations of data structures and algorithms, resulting in efficient storage, retrieval, and manipulation of elements.

4. Standardisation: The Java Collections Framework provides a standard set of interfaces, classes, and methods, ensuring consistency and interoperability across different implementations.

5. Code Reusability: Collections promote code reusability by offering a consistent API and allowing developers to write generic algorithms that can work with different collection types.

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

ans)List of interfaces used in collection framework

1. Collection: The root interface of the Collection framework hierarchy. It represents a group of objects and defines basic operations such as adding, removing, and querying elements. Subinterfaces of Collection include List, Set, and Queue.

2. List: An ordered collection that allows duplicate elements. It maintains the insertion order of elements and provides positional access and manipulation methods. Common implementations of List are ArrayList, LinkedList, and Vector.

3. Set: A collection that does not allow duplicate elements. It models the mathematical concept of a set, where each element is unique. Implementations of Set include HashSet, LinkedHashSet, and TreeSet.

4. Queue: A collection that represents a waiting list or a queue data structure. It follows the FIFO (First-In-First-Out) principle, where elements are added at the end and removed from the beginning. Common implementations of Queue are LinkedList and PriorityQueue.

5. Map: An object that maps keys to values. It does not extend the Collection interface directly but is an important part of the Collection framework. It provides methods to add, remove, and retrieve values based on keys. Common implementations of Map include HashMap, TreeMap, and LinkedHashMap.

6. Iterator: An interface that provides methods to iterate over elements in a collection. It allows sequential access to elements and supports removal of elements during iteration. The Iterator interface is commonly used in conjunction with the Collection interfaces.

7. Iterable: An interface that represents a collection of elements that can be iterated. It provides a single method, iterator(), which returns an Iterator to traverse the elements.

Q5.Differentiate between List and Set in Java?

ans)

List	Set
Allows duplicate elements	Does not allow duplicate elements
Maintains the insertion order of elements	Does not guarantee the order of elements
Allows multiple occurrences of the same element	Elements cannot be accessed by their index
Implementations include ArrayList, LinkedList	Contains only unique elements

Commonly used for ordered data or sequences	Implementations include HashSet, TreeSet, etc.
Example: [1, 2, 3, 2]	Example: [1, 2, 3]

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

ans)

Iterator	ListIterator
Can be used to iterate over elements in a collection	Can be used to iterate over elements in a List
Supports forward-only traversal	Supports both forward and backward traversal
Provides methods like hasNext(), next(), and remove()	Provides additional methods like hasPrevious(), previous()
Cannot add or modify elements during iteration	Allows adding, modifying, and removing elements
Can be used with any collection type	Specifically designed for List collections
Example: Iterator<Integer> it = list.iterator();	Example: ListIterator<Integer> it = list.listIterator();

Q7.What is the Differentiate between Comparable and Comparator?

ans)

Comparable	Comparator
Interface implemented by the class	Separate class or interface implementation
Defines natural ordering	Defines custom ordering
Single sorting sequence	Multiple sorting sequences
Example: Comparable<Student>	Example: Comparator<Student>

Q8.What is collision in HashMap?

ans) Collision in HashMap refers to a situation where two or more keys in a hash map are mapped to the same bucket or index. In other words, different keys have the same hash code or produce the same index after applying the hash function. This can occur because the number of possible hash codes is usually larger than the number of buckets in the hash map.

Q9.Distinguish between a hashmap and a Treemap.

ans)

Feature	Hashmap	Treemap
Implementation	Hash table	Allows null values (at most one key)
Ordering	No specific ordering	Sorted according to the natural order or a

		custom Comparator
Keys	Unordered keys	ordered keys
Null value	Allows null values (at most one key)	Allows null values (at most one key)
Performance	Generally faster for most operations	Slower for insertion and removal, but faster for ordered iteration
Memory	Less memory overhead	More memory overhead
Use Cases	General-purpose HashMap	When key ordering is required

Q10.Define LinkedHashMap in Java.

ans) LinkedHashMap is a class in Java that extends the HashMap class and provides a predictable iteration order, which is the order in which the entries were added or the order specified by an optional access order. It maintains a doubly linked list of the entries, allowing efficient insertion, deletion, and iteration.