(1) How Multi-threading works in Java?

In Java, multithreading allows multiple threads (independent paths of execution) to run concurrently within a single program.

(2) Is it possible to call run() method instead of start() on a thread in Java?

Yes, it is possible to call the **run()** method directly on a Java thread. When you call the **run()** method directly, it runs in the context of the current thread, not in a new thread of execution. This is different from calling the **start()** method, which creates a new thread and then invokes the **run()** method in that new thread

(3) What are the differences between the two data structures: a Vector and an Arraylist?

Vector is synchronized, but ArrayList is not synchronized. ArrayList is generally faster.

(4) What are the differences between Collection and Collections in Java?

Collection is an interface in the Java Collections Framework, representing a group of objects. Classes like **List**, **Set**, and **Queue** implement this interface. **Collections** (with an 's') is a utility class in **java.util** providing static methods for operations on collections, such as sorting or creating synchronized collections.

(5) In which scenario, LinkedList is better than ArrayList in Java?

LinkedList is better than **ArrayList** in scenarios where frequent insertions or deletions occur in the middle of the list, as it involves updating references, leading to faster operations. Additionally, **LinkedList** may be more memory-efficient when dealing with a large number of insertions and deletions due to its non-contiguous memory allocation. If the code heavily relies on iterator-based operations, **LinkedList** is preferable, as it allows efficient element addition or removal during iteration.

(6) What are the differences between a List and Set collection in Java?

List is an ordered collection that allows duplicates, and elements can be accessed by their index. Examples include **ArrayList** and **LinkedList**. A **Set** is an unordered collection that does not allow duplicates, ensuring each element is unique. Examples include **HashSet** and **TreeSet**. In a **List**, elements can be accessed by index using methods like **get(index)**. In a **Set**, there is no direct access by index, as it is an unordered collection.

(7) What are the differences between a HashSet and TreeSet collection in Java?

HashSet is an unordered collection that does not maintain any specific order of elements. **TreeSet** is a sorted set, storing elements in sorted order based on their natural order or a custom comparator.

(8) In Java, how will you decide when to use a List, Set or a Map

List: You need an ordered collection that allows duplicates, and you want to access elements by index.

Set: need a collection with unique elements, and the order of elements is not important.

Map: You need to associate keys with values and require fast lookup based on keys.

(9) What are the differences between a HashMap and a Hashtable in Java?

HashMap is not synchronized, but Hashtable is synchronized. HashMap allows null values, but Hashtable does not allow null values. HashMap is generally faster.

(10)What are the differences between Comparable and Comparator?

Comparable is implemented by the class whose instances are being compared, and the comparison logic is defined in the class itself using the **compareTo** method. **Comparator** is a separate class or object that provides comparison logic, allowing for external control over the ordering of objects

(11)What is Join in database? Explain different types of join in SQL.

In a relational database, a "join" is an operation that combines rows from two or more tables based on a related column between them.

INNER JOIN: Returns rows when there is a match in both tables based on the specified condition.

LEFT (OUTER) JOIN: Returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.

RIGHT (OUTER) JOIN: Returns all rows from the right table and the matched rows from the left table. If there is no match, NULL values are returned for columns from the left table.