1. **What is the collection framework in Java?**

Ans. The Collection Framework in Java is a unified architecture that provides a set of interfaces, classes, and algorithms to handle and manipulate groups of objects, commonly referred to as collections. It offers a standard way to store, retrieve, manipulate, and process data in various data structures, such as lists, sets, queues, and maps. The Collection Framework is part of the Java Collections API, which is built into the Java Standard Library.

2. **What is the difference between ArrayList and LinkedList?**

Ans.

	ArrayList	LinkedList
Underlying Data Structure	Dynamic Array	Doubly Linked List
Random Access	O(1)	O(n)
Insertion/Deletion (End)	O(1)	O(1)
Insertion/Deletion (Mid/Begin)	O(n)	O(1)
Search	O(n)	O(n)
Memory Overhead	Lower	Higher
Suitable Usage Scenarios	Random Access, Iteration	Insertion/Deletion, Sequential Access, Modification

3. **What is the difference between Iterator and Listiterator?**

Ans.

	Iterator	**ListIterator**
Supports Bidirectional Iteration	No	Yes
Traverse Forward	Yes	Yes
Traverse Backward	No	Yes
Modifying Collection	No	Yes (Add, remove, and set elements during iteration)
Accessing Index	No	Yes (get current index, previous index, and next index)

Applicable Collections	All implementations of Collection and Map interfaces	Only List implementations
Methods	**hasNext()**, **next()**, **remove()**	**hasNext()**, **next()**, **remove()**, **hasPrevious()**, **previous()**, etc.

4. **What is the difference between Iterator and Enumeration?**

Ans.

	Iterator	**Enumeration**
Introduced in	Java 1.2	Java 1.0
Bidirectional Iteration	No	No
Traverse Forward	Yes	Yes
Traverse Backward	No	No
Modifying Collection	Yes (through **remove()** method)	No
Fail-Fast	Yes	No
Additional Methods	**hasNext()**, **next()**, **remove()**	**hasMoreElements()**, **nextElement()**
Applicable Collections	All implementations of Collection and Map interfaces	Legacy collections like **Vector**, **Hashtable**, **Stack**

5. **What is the difference between List and Set?**

Ans.

	List	**Set**
Duplicates	Allows duplicates	Does not allow duplicates
Order	Ordered (elements have a specific position/index)	Unordered (no defined order)
Access by Index	Yes (get elements by index)	No (no direct access by index)
Implementation	Examples: ArrayList, LinkedList, etc.	Examples: HashSet, TreeSet, LinkedHashSet, etc.
Performance	Slower performance for large lists	Faster performance for lookup and uniqueness checks

Use Case	When order and duplicates matter	When uniqueness and efficient membership tests are crucial

6. **What is the difference between HashSet and TreeSet?**

Ans.

	HashSet	**TreeSet**
Ordering of Elements	Unordered (no defined order)	Sorted (elements are sorted in natural order or custom order)
Internal Data Structure	Hash table	Red-Black Tree
Null Elements	Allows null elements	Allows a single null element
Performance	Faster for insertion, deletion, and retrieval	Slower for insertion, deletion, and retrieval
Iteration Order	Unpredictable iteration order	Elements are iterated in sorted order
Use of Comparator	Not applicable	Can use a Comparator for custom sorting
Applicable Scenarios	When order is not important and uniqueness is crucial	When elements need to be sorted and uniqueness is crucial

7. **What is the difference between Array and ArrayList?**

Ans.

	Array	**ArrayList**
Size	Fixed size (determined at the time of creation)	Dynamic size (can grow or shrink as elements are added/removed)
Type	Can hold elements of any type (including primitives)	Can hold elements of any reference type (not primitives)
Resizing	Manual resizing required if size changes	Automatically resized as elements are added/removed
Methods for Manipulation	Limited methods for manipulation	Extensive methods for manipulation (add, remove, etc.)
Performance	Slightly faster performance	Slightly slower performance
Memory Overhead	Less memory overhead	More memory overhead

	Compatibility with Legacy Code	More compatible with legacy code	Less compatible with legacy code