Collection Framework

Assignment Questions

1. What is the Collection framework in Java?

Ans: Collection Framework is a combination of classes and interface, which is used to store and manipulate the  data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and  interfaces such as List, Queue, Set, etc. for this purpose.

2. What is the difference between ArrayList and LinkedList?

Ans:

| Aspect                 | ArrayList                                           | LinkedList                                        |

|------------------------|-----------------------------------------------------|---------------------------------------------------|

| Implementation         | Backed by a resizable array.                         | Implemented as a doubly-linked list.               |

| Access time             | Faster for random access (get() by index).            | Slower for random access, faster for add/remove operations in the middle. |

| Insertions/Deletions    | Slower for add/remove operations in the middle.       | Faster for add/remove operations in the middle.    |

| Memory                 | Consumes more memory due to unused array slots.      | Consumes less memory as nodes only hold references. |

| Iteration              | Faster for iteration due to array-based indexing.    | Slower for iteration due to sequential traversal.  |

| Usage                   | Suitable for scenarios requiring random access and less frequent modifications. | Suitable for frequent add/remove operations, especially in the middle of the list.

3. What is the difference between Iterator and ListIterator?

Ans:

| Aspect                | Iterator                                           | ListIterator                                     |

|-----------------------|----------------------------------------------------|--------------------------------------------------|

| Interface             | Part of the Collection framework.                   | Extends Iterator, specific to List collections.   |

| Direction             | Unidirectional, moves forward only.                 | Bidirectional, moves both forward and backward.  |

| Supported Collections| Works with various collections - Set, List, Queue.   | Specifically for Lists (like ArrayList, LinkedList). |

| Methods               | hasNext(), next(), remove()                        | Additionally supports previous(), hasPrevious(), nextIndex(), previousIndex(), add(). |

| Modifications         | Cannot add or modify elements during iteration.      | Allows adding, removing, and modifying elements during iteration. |

| Implementation        | Implemented by all collection classes via iterator() method. | Implemented by List interface classes via listIterator() method. |

4. What is the difference between Iterator and Enumeration?

Ans:

| Aspect                 | Iterator                                           | Enumeration                                       |

|------------------------|----------------------------------------------------|---------------------------------------------------|

| Introduced in          | Introduced in Java 1.2                              | Introduced in Java 1.0                             |

| Interface              | Part of the Collection framework.                   | Part of the legacy Collections framework.          |

| Direction              | Unidirectional, moves forward only.                 | Unidirectional, moves forward only.                |

| Modifications          | Supports safe removal of elements using remove() method. | Does not support removal of elements during iteration. |

| Legacy Support         | Does not provide legacy support.                    | Part of the legacy Collections framework.          |

| Methods                | hasNext(), next(), remove()                         | hasMoreElements(), nextElement()                   |

| Additional Operations  | Provides safe iteration with fail-fast behavior.     | Lacks fail-fast behavior and safe iteration.       |

5. What is the difference between List and Set?

Ans:

"The List and Set both extend the Collection interface. However, there are some differences between the two which are listed below:

- The List can contain duplicate elements, whereas Set includes unique items.

- The List is an ordered collection that maintains the insertion order, whereas Set is an unordered collection that does not preserve the insertion order.

- The List interface contains a single legacy class, which is the Vector class, whereas the Set interface does not have any legacy class.

- The List interface can allow multiple null values, whereas the Set interface only allows a single null value."

6. What is the difference between HashSet and TreeSet?

Ans:

Ordering:

- HashSet: Unordered collection of elements.

- TreeSet: Sorted set based on natural order or a custom comparator.

Duplication:

- Both HashSet and TreeSet do not allow duplicate elements.

Implementation:

- HashSet: Implemented using a hash table.

- TreeSet: Implemented using a self-balancing binary search tree (Red-Black tree).

Performance:

- HashSet: Offers constant-time complexity (O(1)) for adding, removing, and testing the existence of an element.

- TreeSet: Has logarithmic-time complexity (O(log n)) for these operations due to its self-balancing property.

Memory usage:

- HashSet uses less memory as it primarily stores elements.

- TreeSet consumes additional memory to maintain order information.

Iteration:

- HashSet doesn't guarantee the order of iteration.

- TreeSet guarantees that elements are iterated in sorted order.

Usage:

- HashSet is preferable when ordering isn't crucial, and fast access and membership tests are required.

- TreeSet is suitable when elements need to be sorted or accessed in a specific order.

7. What is the difference between Array and ArrayList?

Ans:

1. Type:

   - Arrays can hold both primitive data types and objects.

   - ArrayLists can only hold objects.

2. Size:

   - Arrays have a fixed size once created.

   - ArrayLists can dynamically increase or decrease in size by adding or removing elements.

3. Mutability:

   - Arrays and ArrayLists are both mutable.

   - Arrays allow direct modification of elements, while ArrayLists primarily modify by adding, removing, or altering elements.

4. Performance\*:

   - Arrays often have better performance for certain operations like accessing elements by index due to contiguous memory allocation.

   - ArrayLists use dynamic memory allocation and might result in more memory overhead and slower performance for certain operations.

5. Methods:

   - Arrays have a limited set of methods available compared to ArrayLists.

   - ArrayLists offer a wider range of methods for manipulating the collection, such as adding, removing, and sorting elements.

6. Initialization:

   - Arrays can be initialized with values at the time of creation using literals.

   - ArrayLists require methods to add elements to the collection as they cannot be directly initialized with values.

7. Compatibility:

   - Arrays work well with traditional for-loops and can be easily passed to methods.

   - ArrayLists require the use of a specialized for-each loop and might require more complex handling to pass to methods.