Collections

List(ArrayList)

**2. Search an Element**

Write a program to:

* Create an ArrayList of integers.
* Ask the user to enter a number.
* Check if the number exists in the list.

**3. Remove Specific Element**

Write a program to:

* Create an ArrayList of Strings.
* Add 5 fruits.
* Remove a specific fruit by name.
* Display the updated list.

**4. Sort Elements**

Write a program to:

* Create an ArrayList of integers.
* Add at least 7 random numbers.
* Sort the list in ascending order.
* Display the sorted list.

**5. Reverse the ArrayList**

Write a program to:

* Create an ArrayList of characters.
* Add 5 characters.
* Reverse the list using Collections.reverse() and display it.

**6. Update an Element**

Write a program to:

* Create an ArrayList of subjects.
* Replace one of the subjects (e.g., “Math” to “Statistics”).
* Print the list before and after the update.

**7. Remove All Elements**

Write a program to:

* Create an ArrayList of integers.
* Add multiple elements.
* Remove all elements using clear() method.
* Display the size of the list.

**8. Iterate using Iterator**

Write a program to:

* Create an ArrayList of cities.
* Use Iterator to display each city.

**9. Store Custom Objects**

Write a program to:

* Create a class Student with fields: id, name, and marks.
* Create an ArrayList of Student objects.
* Add at least 3 students.
* Display the details using a loop.

**10. Copy One ArrayList to Another**

Write a program to:

* Create an ArrayList with some elements.
* Create a second ArrayList.
* Copy all elements from the first to the second using addAll() method.

List(LinkedList)

**1. Create and Display a LinkedList**

Write a program to:

* Create a LinkedList of Strings.
* Add five colors to it.
* Display the list using a for-each loop.

**2. Add Elements at First and Last Position**

Write a program to:

* Create a LinkedList of integers.
* Add elements at the beginning and at the end.
* Display the updated list.

**3. Insert Element at Specific Position**

Write a program to:

* Create a LinkedList of names.
* Insert a name at index 2.
* Display the list before and after insertion.

**4. Remove Elements**

Write a program to:

* Create a LinkedList of animal names.
* Remove the first and last elements.
* Remove a specific element by value.
* Display the list after each removal.

**5. Search for an Element**

Write a program to:

* Create a LinkedList of Strings.
* Ask the user for a string to search.
* Display if the string is found or not.

**6. Iterate using ListIterator**

Write a program to:

* Create a LinkedList of cities.
* Use ListIterator to display the list in both forward and reverse directions.

**7. Sort a LinkedList**

Write a program to:

* Create a LinkedList of integers.
* Add unsorted numbers.
* Sort the list using Collections.sort().
* Display the sorted list.

**8. Convert LinkedList to ArrayList**

Write a program to:

* Create a LinkedList of Strings.
* Convert it into an ArrayList.
* Display both the LinkedList and ArrayList.

**9. Store Custom Objects in LinkedList**

Write a program to:

* Create a class Book with fields: id, title, and author.
* Create a LinkedList of Book objects.
* Add 3 books and display their details using a loop.

**10. Clone a LinkedList**

Write a program to:

* Create a LinkedList of numbers.
* Clone it using the clone() method.
* Display both original and cloned lists.

Vector

* **Create a Vector of integers** and perform the following operations:
* Add 5 integers to the Vector.
* Insert an element at the 3rd position.
* Remove the 2nd element.
* Display the elements using Enumeration.
* **Create a Vector of Strings** and:
* Add at least 4 names.
* Check if a specific name exists in the vector.
* Replace one name with another.
* Clear all elements from the vector.
* **Write a program** to:
* Copy all elements from one Vector to another Vector.
* Compare both vectors for equality.
* **Write a method** that takes a Vector<Integer> and returns the **sum of all elements**.

**Stack**

* Understand how to use the Stack class for LIFO (Last In, First Out) operations.
* **Create a Stack of integers** and:
* Push 5 elements.
* Pop the top element.
* Peek the current top.
* Check if the stack is empty.
* **Reverse a string using Stack**:
* Input a string from the user.
* Use a stack to reverse and print the string.
* **Use Stack to check for balanced parentheses** in an expression.
* Input: (a+b) \* (c-d)
* Output: Valid or Invalid expression
* **Convert a decimal number to binary using Stack**.

HashSet

1. **Create a HashSet of Strings**:
   * Add 5 different city names.
   * Try adding a duplicate city and observe the output.
   * Iterate using an Iterator and print each city.
2. **Perform operations**:
   * Remove an element.
   * Check if a city exists.
   * Clear the entire HashSet.
3. **Write a method** that takes a HashSet<Integer> and returns the maximum element.

**LinkedHashSet**

**1.Create a LinkedHashSet of Integers**:

* + Add numbers: 10, 5, 20, 15, 5.
  + Print the elements and observe the order.

1. **Create a LinkedHashSet of custom objects (e.g., Student with id and name)**:
   * Override hashCode() and equals() properly.
   * Add at least 3 Student objects.
   * Try adding a duplicate student and check if it gets added.
2. **Write a program** to:
   * Merge two LinkedHashSets and print the result.

**TreeSet**

**1. Create a TreeSet of Strings**:

* + Add 5 country names in random order.
  + Print the sorted list of countries using TreeSet.

1. **Create a TreeSet of Integers**:
   * Add some numbers and print the first and last elements.
   * Find the elements lower than and higher than a given number using lower() and higher() methods.
2. **Create a TreeSet with a custom comparator**:
   * Sort strings in **reverse alphabetical order** using Comparator.

Queue

1. **Bank Queue Simulation**:
   * Create a queue of customer names using Queue<String>.
   * Add 5 customers to the queue.
   * Serve (remove) customers one by one and print the queue after each removal.
2. **Task Manager**:
   * Queue of tasks (String values).
   * Add tasks, peek at the next task, and poll completed tasks.
3. **Write a method**:
   * That takes a queue of integers and returns a list of even numbers.

**PriorityQueue**

1. **Hospital Emergency Queue**:
   * Create a class Patient with fields: name and severityLevel (int).
   * Use PriorityQueue<Patient> with a comparator to serve the most critical patients first (highest severityLevel).
2. **Print Jobs Priority**:
   * Add different print jobs (String) with priority levels.
   * Use PriorityQueue to simulate serving high-priority jobs before others.
3. **Write a method**:
   * To merge two PriorityQueue<Integer> and return a sorted merged queue.

**Deque**

1. **Palindrome Checker**:
   * Input a string and check if it is a palindrome using a Deque<Character>.
2. **Double-ended Order System**:
   * Add items from front and rear.
   * Remove items from both ends.
   * Display contents of the deque after each operation.
3. **Browser History Simulation**:
   * Implement browser back and forward navigation using two deques.