1. Finding the Number of Elements in an ArrayList

In Java, ArrayList is a part of the Collection Framework and is used to store dynamic data. To find the number of elements present in an ArrayList, the size() method is used. The size() method returns an integer value representing the total number of elements currently stored in the list.

Unlike arrays, which have a fixed length, ArrayList can grow or shrink dynamically. Whenever an element is added or removed, the size of the ArrayList changes automatically.

Example: ArrayList list = new ArrayList<>(); list.add(10); list.add(20); list.add(30); System.out.println(list.size());

In this example, size() returns 3 because three elements are present. Thus, the size() method is essential for managing and controlling dynamic data stored in an ArrayList.

2. Constructors and Methods of TreeSet Class

TreeSet is a class in Java that implements the Set interface and stores elements in sorted order. It uses a Red-Black Tree data structure internally.

Constructors of TreeSet: 1. TreeSet() – Creates an empty TreeSet. 2. TreeSet(Collection c) – Creates a TreeSet containing elements of the specified collection. 3. TreeSet(Comparator comp) – Creates a TreeSet sorted using a custom comparator.

Important Methods: add(), remove(), contains(), first(), last(), higher(), lower(), ceiling(), floor(), pollFirst(), pollLast()

TreeSet does not allow duplicate elements and does not allow null values. It is mainly used when sorted and unique data is required.

3. Use of ListIterator Class

ListIterator is an interface in Java that allows traversal of a list in both forward and backward directions. It is available for List implementations such as ArrayList and LinkedList.

Key Uses: • Bidirectional traversal • Modification of elements • Adding elements during iteration

Important Methods: hasNext(), next(), hasPrevious(), previous(), add(), remove(), set()

ListIterator provides more flexibility than Iterator and is commonly used in complex list manipulation operations.

4. Basic Operations Using ArrayList

ArrayList supports several basic operations such as insertion, deletion, retrieval, and updating of elements.

Example: ArrayList names = new ArrayList<>(); names.add("Java"); names.add("Python"); names.add("C++"); names.remove("Python"); names.set(1, "JavaScript"); names.get(0);

Operations Explanation: • add() – Inserts elements • remove() – Deletes elements • set() – Updates elements • get() – Retrieves elements

ArrayList is widely used due to its flexibility and ease of use.

5. Priority Queues in Java

A Priority Queue is a special type of queue where elements are processed based on priority rather than insertion order. Java provides PriorityQueue class which is part of the Queue interface.

Characteristics: • Elements are ordered based on natural ordering or comparator • Does not allow null values • Used in scheduling and resource allocation

Important Methods: add(), offer(), poll(), peek(), remove()

Priority queues are commonly used in operating systems, algorithms, and real-time systems.

6. Methods: offer(), poll(), peek()

offer(): Adds an element to the queue. Returns true if successful.

poll(): Removes and returns the head element. Returns null if queue is empty.

peek(): Returns the head element without removing it. Returns null if empty.

These methods are safer alternatives to add(), remove(), and element() as they do not throw exceptions.

7. Program to Implement ListIterator

import java.util.*;

public class ListIteratorDemo { public static void main(String[] args) { ArrayList list = new ArrayList<>(); list.add("Java"); list.add("Python"); list.add("C++");

ListIterator it = list.listIterator();

while(it.hasNext()) { System.out.println(it.next()); }

while(it.hasPrevious()) { System.out.println(it.previous()); } } }

This program demonstrates forward and backward traversal using ListIterator. It highlights the flexibility and power of ListIterator in Java.