Introduction to Collections Framework

1. Write a program to demonstrate adding and printing elements from an ArrayList.

```
import java.util.ArrayList;

public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();

        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Orange");

        System.out.println("Fruits in the list:");
        for (String fruit : fruits) {
            System.out.println(fruit);
        }
    }
}
```

2. Show how to use Collections.max() and Collections.min() on a list of integers

```
import java.util.ArrayList;
import java.util.Collections;

public class MaxMinExample {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();

        numbers.add(34);
        numbers.add(15);
        numbers.add(76);
        numbers.add(89);

        int max = Collections.max(numbers);
        int min = Collections.min(numbers);

        System.out.println("List: " + numbers);
        System.out.println("Maximum value: " + max);
        System.out.println("Minimum value: " + min);
    }
}
```

3. Demonstrate the use of Collections.sort() on a list of strings.

```
import java.util.ArrayList;
    import java.util.Collections;

    public class SortExample {
        public static void main(String[] args) {
            ArrayList<String> names = new ArrayList<>();

            names.add("Reena");
            names.add("Anu");
            names.add("James");
            names.add("Bella");

            System.out.println("Before sorting: " + names);

            Collections.sort(names);

            // Display list after sorting
            System.out.println("After sorting: " + names);
        }
    }
```

4. You need to store a dynamic list of student names and display them in alphabetical order. Implement this using a suitable collection.

```
    import java.util.ArrayList;
    import java.util.Collections;
    import java.util.Scanner;

    public class Student {
        public static void main(String[] args) {
            // Create a dynamic list to store student names
            ArrayList<String> studentNames = new ArrayList<>();

            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter the number of students: ");
            int n = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            // Take input from the user
            for (int i = 1; i <= n; i++) {
                System.out.print("Enter name of student " + i + ": ");
                String name = scanner.nextLine();
                studentNames.add(name);
            }


            Collections.sort(studentNames);
```

```
        System.out.println("\nStudent names in alphabetical order:");
        for (String name : studentNames) {
            System.out.println(name);
        }
                    scanner.close();
    }
}
```

5.A user can input any number of integers. Your program should store them and display the sum of all elements using the Collection Framework

```
import java.util.ArrayList;
import java.util.Scanner;

public class SumofallElements{
public static void main(String[] args) {

        ArrayList<Integer> numbers = new ArrayList<>();

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter integers (type -1 to finish):");

        while (true) {
            int num = scanner.nextInt();
            if (num == -1) {
                break; // Stop input on -1
            }
            numbers.add(num); // Add to list
        }

        // Calculate sum
        int sum = 0;
        for (int n : numbers) {
            sum += n;
        }

        // Display result
        System.out.println("Numbers entered: " + numbers);
        System.out.println("Sum of all elements: " + sum);

        scanner.close();
    }
}
```

List Interface

1. Write a Java program to add, remove, and access elements in an ArrayList.

```
import java.util.ArrayList;

public class ArrayListOperations {
   public static void main(String[] args) {
      // Step 1: Create an ArrayList of Strings
      ArrayList<String> fruits = new ArrayList<>();


      fruits.add("Apple");
      fruits.add("Banana");
      fruits.add("Cherry");


      System.out.println("Fruits list after adding elements: " + fruits);


      System.out.println("First fruit: " + fruits.get(0));
      System.out.println("Third fruit: " + fruits.get(2));


      fruits.remove("Banana");
      fruits.remove(2);

      System.out.println("Fruits list after removing elements: " + fruits);

      fruits.add("Orange");
       fruits.add("blueberry");

      System.out.println("Final fruits list: " + fruits);
   }
}
```

2. Implement a LinkedList that stores and prints employee names.

```
import java.util.LinkedList;

public class EmployeeList {
   public static void main(String[] args) {

      LinkedList<String> employees = new LinkedList<>();

      employees.add("Alice");
      employees.add("Bob");
      employees.add("Charlie");
      employees.add("Diana");

      System.out.println("Employee Names:");
```

```
        for (String name : employees) {
            System.out.println(name);
        }
    }
}
```

3. Demonstrate inserting an element at a specific position in a List.

```
import java.util.ArrayList;
import java.util.List;

public class InsertElement {
    public static void main(String[] args) {

        List<String> colors = new ArrayList<>();


        colors.add("Pink");
        colors.add("Orange");
        colors.add("Blue");


        System.out.println("Original list: " + colors);


        colors.add(1, "Yellow");


        System.out.println("List after inserting 'Yellow' at index 1: " + colors);
    }
}
```

4. You're building a to-do list manager. Use ArrayList to add tasks, remove completed ones, and display pending tasks

```
import java.util.ArrayList;
import java.util.Scanner;

public class Manager {
    public static void main(String[] args) {
        ArrayList<String> tasks = new ArrayList<>();
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println("\n--- To-Do List Menu ---");
            System.out.println("1. Add Task");
            System.out.println("2. Remove Task");
             System.out.println("3. View Pending Tasks");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");

            int choice = sc.nextInt();
            sc.nextLine();
```

```
            switch {
             case 1:
                System.out.print("Enter task to add: ");
                String task = sc.nextLine();
                tasks.add(task);
                System.out.println("Task added.");
                break;

             case 2:
                System.out.print("Enter task number to remove: ");
                int taskNum = sc.nextInt();
                sc.nextLine();
                if (taskNum >= 1 && taskNum <= tasks.size()) {
                   tasks.remove(taskNum - 1);
                   System.out.println("Task removed.");
                } else {
                   System.out.println("Invalid task number.");
                }
                break;

             case 3:
                System.out.println("\nPending Tasks:");
                if (tasks.isEmpty()) {
                   System.out.println("No tasks in the list.");
                } else {
                   for (int i = 0; i < tasks.size(); i++) {
                      System.out.println((i + 1) + ". " + tasks.get(i));
                   }
                }
                break;

             case 4:
                System.out.println("Exiting To-Do List Manager.");
                sc.close();
                return;

             default:
                System.out.println("Invalid choice. Try again.");
          }
       }
    }
}
```

5. Create a simple shopping cart system where users can add/remove products using a List

```
import java.util.ArrayList;
import java.util.Scanner;

public class ShoppingCart {
   public static void main(String[] args) {
      ArrayList<String> cart = new ArrayList<>();
      Scanner sc = new Scanner(System.in);
      int choice;

      while {
         System.out.println("1. Add Product");
         System.out.println("2. Remove Product");
         System.out.println("3. View Cart");
         System.out.println("4. Exit");
         System.out.print("Enter your choice: ");
         choice = sc.nextInt();
         sc.nextLine();

         switch {
          case 1:
               System.out.print("Enter product name to add: ");
               String product = sc.nextLine();
               cart.add(product);
               System.out.println(product + " added to the cart.");
               break;

            case 2:
               if (cart.isEmpty()) {
                  System.out.println("Cart is empty!");
               } else {
                  System.out.print("Enter product number to remove: ");
                  int index = sc.nextInt();
                  sc.nextLine(); // consume newline
                  if (index >= 1 && index <= cart.size()) {
                     String removed = cart.remove(index - 1);
                     System.out.println(removed + " removed from the cart.");
                  } else {
                     System.out.println("Invalid product number.");
                  }
               }
               break;

            case 3:
               System.out.println("Your Shopping Cart:");
               if (cart.isEmpty()) {
                  System.out.println("Cart is empty.");
               } else {
```

```
                    for (int i = 0; i < cart.size(); i++) {
                        System.out.println((i + 1) + ". " + cart.get(i));
                    }
                }
                break;

            case 4:
                System.out.println("Thank you for shopping. Exiting...");
                sc.close();
                return;

            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
}
```

Set Interface.

1. Write a program using HashSet to store unique student roll numbers.

```
import java.util.HashSet;
import java.util.Scanner;

public class UniqueRollNumbers {
    public static void main(String[] args) {
        HashSet<Integer> rollNumbers = new HashSet<>();
        Scanner sc = new Scanner(System.in);

        System.out.print("How many roll numbers do you want to enter? ");
        int count = sc.nextInt();

        for (int i = 1; i <= count; i++) {
            System.out.print("Enter roll number " + i + ": ");
            int roll = sc.nextInt();

            // Add to set, duplicates will be automatically ignored
            if (rollNumbers.add(roll)) {
                System.out.println("Added successfully.");
            } else {
                System.out.println("Duplicate roll number! Not added.");
            }
        }

        System.out.println("\nUnique Roll Numbers:");
        for (int num : rollNumbers) {
            System.out.println(num);
        }

        sc.close();
```

```
    }
}
```

2. Demonstrate how to use TreeSet to automatically sort elements. import java.util.Scanner;

```
import java.util.TreeSet;

public class SortedNames {
 public static void main(String[] args) {
      TreeSet<String> names = new TreeSet<>();
      Scanner sc = new Scanner(System.in);

      System.out.print("How many names do you want to enter? ");
      int count = sc.nextInt();
      sc.nextLine(); // consume newline

      for (int i = 1; i <= count; i++) {
         System.out.print("Enter name " + i + ": ");
         String name = sc.nextLine();
         names.add(name);
      }

      System.out.println("\nNames in Sorted Order:");
      for (String name : names) {
         System.out.println(name);
      }

      sc.close();
   }
}
```

3. Use LinkedHashSet to maintain insertion order and prevent duplicates.

```
import  java.util.LinkedHashSet;
import java.util.Scanner;

public class LinkedHashSet {
public static void main(String[] args) {
      LinkedHashSet<String> cities = new LinkedHashSet<>();
      Scanner sc = new Scanner(System.in);

      System.out.print("How many cities do you want to enter? ");
      int count = sc.nextInt();
      sc.nextLine(); // consume the leftover newline

      for (int i = 1; i <= count; i++) {
         System.out.print("Enter city " + i + ": ");
         String city = sc.nextLine();
         if (cities.add(city)) {
            System.out.println("Added: " + city);
         } else {
            System.out.println("Duplicate city! Not added.");
         }
```

```
        }

        System.out.println("\nCities in the order entered (no duplicates):");
        for (String city : cities) {
            System.out.println(city);
        }

        sc.close();
    }
}
```

4. Design a program to store registered email IDs of users such that no duplicates are allowed.

```
import java.util.HashSet;
import java.util.Scanner;

public class EmailRegistry {
    public static void main(String[] args) {
        HashSet<String> emailSet = new HashSet<>();
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of email IDs to register: ");
        int count = sc.nextInt();
        sc.nextLine();  // consume leftover newline

        for (int i = 1; i <= count; i++) {
            System.out.print("Enter email ID " + i + ": ");
            String email = sc.nextLine().toLowerCase(); // to treat emails case-
insensitively
            if (emailSet.add(email)) {
                System.out.println("Registered: " + email);
            } else {
                System.out.println("Duplicate email! Already registered.");
            }
        }

        System.out.println("\nRegistered Email IDs:");
        for (String email : emailSet) {
            System.out.println(email);
        }

        sc.close();
    }
}
```

5. Create a program where a Set is used to eliminate duplicate entries from a list of city names entered by users.

```
import java.util.HashSet;
import  java.util.Scanner;
import java.util.Set;

public class UniqueCities {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      Set<String> cities = new HashSet<>();

      System.out.print("Enter the number of city names: ");
      int n = sc.nextInt();
      sc.nextLine(); // consume leftover newline

      for (int i = 1; i <= n; i++) {
         System.out.print("Enter city name " + i + ": ");
         String city = sc.nextLine().trim().toLowerCase(); // normalize input
         if (cities.add(city)) {
            System.out.println("Added: " + city);
         } else {
            System.out.println("Duplicate! City already exists.");
         }
      }

      System.out.println("\nUnique city names:");
      for (String city : cities) {
         System.out.println(city);
      }

      sc.close();
   }
}
```

Map Interface

1. Write a program using HashMap to store student names and their marks.

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Marks {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      HashMap<String, Integer> studentMap = new HashMap<>();

      System.out.print("Enter the number of students: ");
      int n = sc.nextInt();
      sc.nextLine(); // consume leftover newline

      for (int i = 1; i <= n; i++) {
         System.out.print("Enter name of student " + i + ": ");
         String name = sc.nextLine();
```

```
            System.out.print("Enter marks of " + name + ": ");
            int marks = sc.nextInt();
            sc.nextLine(); // consume leftover newline

            studentMap.put(name, marks);
        }

        System.out.println("\nStudent Marks:");
        for (Map.Entry<String, Integer> entry : studentMap.entrySet()) {
            System.out.println("Name: " + entry.getKey() + ", Marks: " +
    entry.getValue());
        }

        sc.close();
    }
}
```

2. Demonstrate how to iterate over a Map using entrySet().

```
import java.util.HashMap;
import java.util.Map;

public class IterateMap {
    public static void main(String[] args) {

        Map<String, String> countryCapitalMap = new HashMap<>();


        countryCapitalMap.put("India", "New Delhi");
        countryCapitalMap.put("USA", "Washington D.C.");
        countryCapitalMap.put("France", "Paris");
        countryCapitalMap.put("Japan", "Tokyo");


        System.out.println("Country - Capital List:");
        for (Map.Entry<String, String> entry : countryCapitalMap.entrySet()) {
            String country = entry.getKey();
            String capital = entry.getValue();
            System.out.println(country + " → " + capital);
        }
    }
}
```

3. Show how to update the value associated with a key in a Map.

```
import java.util.HashMap;
import java.util.Map;

public class UpdateMapValue {
    public static void main(String[] args) {

        Map<String, Integer> studentMarks = new HashMap<>();
```

```
        studentMarks.put("Alice", 75);
        studentMarks.put("Bob", 82);
        studentMarks.put("Charlie", 68);

        System.out.println("Before Update: " + studentMarks);


        if (studentMarks.containsKey("Bob")) {
            studentMarks.put("Bob", 90);  // Overwrites old value
            System.out.println("Updated Bob's marks to 90");
        }


        if (studentMarks.containsKey("David")) {
            studentMarks.put("David", 70);
        } else {
            System.out.println("David not found in the map.");
        }

        System.out.println("After Update: " + studentMarks);
    }
}
```

4. Build a phone directory where names are keys and phone numbers are values.

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Phone {
    public static void main(String[] args) {

        Map<String, String> phoneDirectory = new HashMap<>();

        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            System.out.println("\n Phone Directory Menu:");
            System.out.println("1. Add Contact");
            System.out.println("2. View Contact");
            System.out.println("3. Display All Contacts");
            System.out.println("4. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch {
            case 1:
                System.out.print("Enter Name: ");
                String name = scanner.nextLine();
```

```
                System.out.print("Enter Phone Number: ");
                String phone = scanner.nextLine();
                phoneDirectory.put(name, phone);
                System.out.println("Contact added.");
                break;

            case 2:
                System.out.print("Enter Name to Search: ");
                String searchName = scanner.nextLine();
                if (phoneDirectory.containsKey(searchName)) {
                    System.out.println(searchName + "'s Phone Number: " +
phoneDirectory.get(searchName));
                } else {
                    System.out.println("Contact not found.");
                }
                break;

            case 3:
                System.out.println("\nAll Contacts:");
                for (Map.Entry<String, String> entry : phoneDirectory.entrySet()) {
                    System.out.println(entry.getKey() + " → " + entry.getValue());
                }
                break;

            case 4:
                System.out.println("Exiting Phone Directory.");
                break;

            default:
                System.out.println("Invalid choice. Try again.");
        }

    } while (choice != 4);

    scanner.close();
  }
}
```

5. Create a frequency counter for words in a sentence using a Map.

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class WordFrequency{
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);


    System.out.print("Enter a sentence: ");
    String sentence = scanner.nextLine();
```

```
        String[] words = sentence.toLowerCase().split("\\s+"); // Split by whitespace


        Map<String, Integer> wordCountMap = new HashMap<>();


        for (String word : words) {
            if (wordCountMap.containsKey(word)) {
                // Word already exists, increment count
                wordCountMap.put(word, wordCountMap.get(word) + 1);
            } else {
                // Word is new, add with count 1
                wordCountMap.put(word, 1);
            }
        }


        System.out.println("\nWord Frequencies:");
        for (Map.Entry<String, Integer> entry : wordCountMap.entrySet()) {
            System.out.println(entry.getKey() + " → " + entry.getValue());
        }

        scanner.close();
    }
}
```

Queue Interface

1. Implement a simple task queue using LinkedList as a Queue.

```
import java.util.LinkedList;
import java.util.Queue;

public class Task {
    public static void main(String[] args) {

        Queue<String> taskQueue = new LinkedList<>();


        taskQueue.add("Task 1: Email the client");
        taskQueue.add("Task 2: Review code");
        taskQueue.add("Task 3: Deploy application");


        System.out.println("Processing Tasks in Queue Order:\n");
        while (!taskQueue.isEmpty()) {
            String task = taskQueue.poll(); // Retrieves and removes the head of the
queue
            System.out.println("Processing: " + task);
        }
```

```
        System.out.println("\nAll tasks processed.");
    }
}
```

2. Demonstrate how to add and remove elements using offer() and poll().

```
import java.util.LinkedList;
import java.util.Queue;

public class QueueExample {
    public static void main(String[] args) {

        Queue<String> queue = new LinkedList<>();


        queue.offer("Task A");
        queue.offer("Task B");
        queue.offer("Task C");

        System.out.println("Queue after offers: " + queue);


        String firstTask = queue.poll();  // removes "Task A"
        System.out.println("Polled: " + firstTask);
        System.out.println("Queue after first poll: " + queue);

        String secondTask = queue.poll(); // removes "Task B"
        System.out.println("Polled: " + secondTask);
        System.out.println("Queue after second poll: " + queue);
    }
}
```

1. Use a PriorityQueue to order tasks by priority (integers).

```
import java.util.PriorityQueue;

public class TaskPriorityQueue {
    public static void main(String[] args) {

        PriorityQueue<Task> taskQueue = new PriorityQueue<>();


        askQueue.add(new Task("Complete Java Assignment", 3));
        taskQueue.add(new Task("Pay Electricity Bill", 1));
        taskQueue.add(new Task("Grocery Shopping", 4));
        taskQueue.add(new Task("Call Mom", 2));


        System.out.println("Tasks in priority order:");
        while (!taskQueue.isEmpty()) {
            Task task = taskQueue.poll();
            System.out.println(task.name + " (Priority: " + task.priority + ")");
        }
```

```
    }
}


class Task implements Comparable<Task> {
    String name;
    int priority;


    public Task(String name, int priority) {
        this.name = name;
        this.priority = priority;
    }


    public int compareTo(Task other) {
        return Integer.compare(this.priority, other.priority);
    }
}
```

3. Simulate a print queue system where print jobs are processed in order.

```
import java.util.LinkedList;
import java.util.Queue;

class Job {
    private String documentName;

    public PrintJob(String documentName) {
        this.documentName = documentName;
    }

    public String getDocumentName() {
        return documentName;
    }
}

public class PrintQueueSimulation {
    public static void main(String[] args) {

        Queue<PrintJob> printQueue = new LinkedList<>();


        printQueue.offer(new PrintJob("Document1.pdf"));
        printQueue.offer(new PrintJob("Invoice_March.docx"));
        printQueue.offer(new PrintJob("Resume.pdf"));
        printQueue.offer(new PrintJob("Poster_Design.ppt"));

        System.out.println(" Print Queue Simulation Started...\n");


        while (!printQueue.isEmpty()) {
```

```
            PrintJob currentJob = printQueue.poll(); // remove and get head of queue
            System.out.println("Printing: " + currentJob.getDocumentName());
        }

        System.out.println("\nAll documents printed.");
    }
}
```

4. Create a ticket booking system where customer names are added to a queue and served in order.

```
import java.util.LinkedList;
import java.util.Queue;

public class TicketBookingSystem {
    public static void main(String[] args) {

        Queue<String> customerQueue = new LinkedList<>();


        customerQueue.offer("Alice");
        customerQueue.offer("Bob");
        customerQueue.offer("Charlie");
        customerQueue.offer("Diana");

        System.out.println(" Ticket Booking System Started...\n");


        while (!customerQueue.isEmpty()) {
            String customer = customerQueue.poll();  // Retrieves and removes the head of the queue
            System.out.println("Ticket issued to: " + customer);
        }

        System.out.println("\n All customers have been served.");
    }
}
```

Iterator Interface

1. Write a program to iterate through a list using Iterator.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class IteratorExample {
    public static void main(String[] args) {

        List<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Orange");
```

```
        Iterator<String> iterator = fruits.iterator();


        System.out.println("Fruits in the list:");
        while (iterator.hasNext()) {
            String fruit = iterator.next();
            System.out.println(fruit);
        }
    }
}
```

2. Demonstrate removing an element from a list while iterating using Iterator.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Remove {
public static void main(String[] args) {
        // Create a List of names
        List<String> names = new ArrayList<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        names.add("David");


        Iterator<String> iterator = names.iterator();


        while (iterator.hasNext()) {
            String name = iterator.next();
            if (name.equals("Charlie")) {
                iterator.remove();  // Safe removal
            }
        }


        System.out.println("Updated list after removal: " + names);
    }
}
```

3. Show how to use ListIterator to iterate in both directions.

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorExample {
    public static void main(String[] args) {

        List<String> fruits = new ArrayList<>();
        fruits.add("Apple");
```

```
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Orange");


        ListIterator<String> listIterator = fruits.listIterator();

        System.out.println("Forward  Direction:");

        while (listIterator.hasNext()) {
            String fruit = listIterator.next();
            System.out.println(fruit);
        }

        System.out.println("\nBackward  Direction:");

        while (listIterator.hasPrevious()) {
            String fruit = listIterator.previous();
            System.out.println(fruit);
        }
    }
}
```

4. Design a program that reads a list of book titles and removes those starting with a specific letter using an iterator.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class RemoveBooks{
public static void main(String[] args) {

        List<String> books = new ArrayList<>();
        books.add("The Hobbit");
        books.add("To Kill a Mockingbird");
        books.add("Moby Dick");
        books.add("Twilight");
        books.add("Harry Potter");


        char targetLetter = 'T';


        Iterator<String> iterator = books.iterator();
        while (iterator.hasNext()) {
            String book = iterator.next();
            if (book.startsWith(String.valueOf(targetLetter))) {
                iterator.remove();  // Safe removal during iteration
            }
        }
```

```
            System.out.println("Books after removing those starting with '" +
        targetLetter + "':");
            for (String title : books) {
                System.out.println(title);
            }
        }
    }
```

5. Create a program that reverses the elements in a list using ListIterator.

```
    import java.util.ArrayList;
    import java.util.List;
    import java.util.ListIterator;

    public class ReverseListWithListIterator {
        public static void main(String[] args) {

            List<String> fruits = new ArrayList<>();
            fruits.add("Apple");
            fruits.add("Banana");
            fruits.add("Cherry");
            fruits.add("Date");


            ListIterator<String> listIterator = fruits.listIterator(fruits.size());


            System.out.println("Fruits in reverse order:");
            while (listIterator.hasPrevious()) {
                System.out.println(listIterator.previous());
            }
        }
    }
```

Sorting and Searching Collections

1. Sort an ArrayList of integers in ascending and descending order.

```
    import java.util.*;

    public class SortIntegers {
    public static void main(String[] args) {
            List<Integer> numbers = Arrays.asList(5, 2, 9, 1, 3);

              Collections.sort(numbers);
            System.out.println("Ascending: " + numbers);

          Collections.sort(numbers, Collections.reverseOrder());
         System.out.println("Descending: " + numbers);
    }
}
```

2. Use Collections.binarySearch() to find an element in a sorted list.

```
import java.util.*;

public class BinarySearchExample {
public static void main(String[] args) {
        List<String> list = Arrays.asList("Apple", "Banana", "Cherry", "Date");
        Collections.sort(list); // Ensure sorted before searching

       int index = Collections.binarySearch(list, "Cherry");
       System.out.println("Index of 'Cherry': " + index);
    }
}
```

3. Sort a list of custom objects like Employees by name using Comparator.

```
import java.util.*;

class Employee {
String name;
int id;

Employee(String name, int id) {
 this.name = name;
 this.id = id;
}

 public String toString() {
 return name + " (" + id + ")";
}
}

 public class SortEmployees {
  public static void main(String[] args) {
        List<Employee> list = new ArrayList<>();
          list.add(new Employee("Alice", 103));
          list.add(new Employee("Bob", 101));
         list.add(new Employee("Charlie", 102));

        list.sort(Comparator.comparing(emp -> emp.name));

        System.out.println("Sorted by name:");
       for (Employee e : list) {
       System.out.println(e);
    }
  }
}
```

4. You have a list of products with prices. Sort them by price and then search for a product within a specific price range.

```
import java.util.*;

class Product {
    String name;
    double price;

    Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    public String toString() {
        return name + ": $" + price;
    }
}

public class ProductSortSearch {
    public static void main(String[] args) {
        List<Product> products = new ArrayList<>();
        products.add(new Product("Book", 20.0));
        products.add(new Product("Pen", 5.0));
        products.add(new Product("Laptop", 1200.0));
        products.add(new Product("Mouse", 25.0));

        products.sort(Comparator.comparingDouble(p -> p.price));

        System.out.println("Sorted by price:");
        for (Product p : products) {
            System.out.println(p);
        }

        System.out.println("\nProducts between $10 and $30:");
        for (Product p : products) {
            if (p.price >= 10 && p.price <= 30) {
                System.out.println(p);
            }
        }
    }
}
```

5. Build a leaderboard system that keeps players sorted by scores (highest first). Allow searching for a specific player's rank

```
import java.util.*;

class Player {
    String name;
    int score;

    Player(String name, int score) {
        this.name = name;
        this.score = score;
    }

    public String toString() {
        return name + ": " + score;
    }
}

public class Leaderboard {
    public static void main(String[] args) {
        List<Player> players = new ArrayList<>();
        players.add(new Player("Alice", 250));
        players.add(new Player("Bob", 300));
        players.add(new Player("Charlie", 275));

        players.sort((a, b) -> b.score - a.score);

        System.out.println("Leaderboard:");
        for (int i = 0; i < players.size(); i++) {
            System.out.println((i + 1) + ". " + players.get(i));
        }

        // Search for a specific player
        String searchName = "Charlie";
        for (int i = 0; i < players.size(); i++) {
            if (players.get(i).name.equals(searchName)) {
                System.out.println(searchName + "'s rank: " + (i + 1));
                break;
            }
        }
    }
```