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.

```
package day8_Assessment;
import java.util.ArrayList;
import java.util.Scanner;
public class ArrayListExample {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(25);
        numbers.add(30);
        numbers.add(40);
        numbers.add(50);
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number to search: ");
        int inputNumber = scanner.nextInt();
        if (numbers.contains(inputNumber)) {
            System.out.println(inputNumber + " exists in the list.");
        } else {
            System.out.println(inputNumber + " does not exist in the list.");
        }
```

```
        scanner.close();
    }
}
```

Output:

Enter a number to search: 30

30 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.

```
package day8_Assessment;
import java.util.ArrayList;
import java.util.Scanner;
public class RemoveElement {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Orange");
        fruits.add("Grapes");
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter the name of the fruit to remove: ");
        String fruitToRemove = scanner.nextLine();
        if (fruits.remove(fruitToRemove)) {
            System.out.println(fruitToRemove + " removed successfully.");
        } else {
            System.out.println(fruitToRemove + " not found in the list.");
        }
        System.out.println("Updated list of fruits: " + fruits);
        scanner.close();
    }
}
```

Output:

Enter the name of the fruit to remove: Banana

Banana removed successfully.

Updated list of fruits: [Apple, Mango, Orange, Grapes]

---

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.

```
package day8_Assessment;
import java.util.ArrayList;
```

```
import java.util.Collections;
public class SortElements {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(34);
        numbers.add(12);
        numbers.add(89);
        numbers.add(5);
        numbers.add(67);
        numbers.add(23);
        numbers.add(78);

        Collections.sort(numbers);

        System.out.println("Sorted list: " + numbers);
    }
}
```

Output:

Sorted list: [5, 12, 23, 34, 67, 78, 89]

---

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.

```
package day8_Assessment;
```

```
import java.util.ArrayList;
import java.util.Collections;
public class ReverseArrayList {
    public static void main(String[] args) {
        ArrayList<Character> chars = new ArrayList<>();
        chars.add('A');
        chars.add('B');
        chars.add('C');
        chars.add('D');
        chars.add('E');
        Collections.reverse(chars);
        System.out.println("Reversed list: " + chars);
    }
}
```

Output:

Reversed list: [E, D, C, B, A]

---

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.

```
package day8_Assessment;
import java.util.ArrayList;

public class UpdateArrayList {
   public static void main(String[] args) {
      ArrayList<String> subjects = new ArrayList<>();
      subjects.add("English");
```

```
        subjects.add("Math");
        subjects.add("Science");
        subjects.add("History");

        System.out.println("Before update: " + subjects);

        int index = subjects.indexOf("Math");
        if (index != -1) {
            subjects.set(index, "Statistics");
        } else {
            System.out.println("\"Math\" not found in the list.");
        }

        System.out.println("After update: " + subjects);
    }
}
```

Output:

```
Before update: [English, Math, Science, History]
After update: [English, Statistics, Science, History]
```

---

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.

```
package day8_Assessment;
import java.util.ArrayList;

public class RemoveAllElementsInArrayList {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(10);
        numbers.add(20);
        numbers.add(30);
```

```
        numbers.add(40);
        numbers.add(50);

        numbers.clear();

        System.out.println("Size of the list after clear: " + numbers.size());
    }
}
```

Output:

Size of the list after clear: 0

---

8. Iterate using Iterator

Write a program to:

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

```
package day8_Assessment;
import java.util.ArrayList;
import java.util.Iterator;
public class IterateUsingIterator {
    public static void main(String[] args) {
        ArrayList<String> cities = new ArrayList<>();
        cities.add("New York");
        cities.add("London");
        cities.add("Tokyo");
        cities.add("Paris");
        cities.add("Sydney");
        Iterator<String> iterator = cities.iterator();
```

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

Output:

New York

London

Tokyo

Paris

Sydney

---

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.

```
package day8_Assessment;
import java.util.ArrayList;
class Student {
    int id;
    String name;
    double marks;
    Student(int id, String name, double marks) {
        this.id = id;
        this.name = name;
        this.marks = marks;
    }
```

```
}
public class StoreCustomObjects {
   public static void main(String[] args) {
      ArrayList<Student> students = new ArrayList<>();
      students.add(new Student(1, "Alice", 85.5));
      students.add(new Student(2, "Bob", 90.0));
      students.add(new Student(3, "Charlie", 78.0));
      for (Student s : students) {
         System.out.println("ID: " + s.id + ", Name: " + s.name + ", Marks: " +
s.marks);
      }
   }
}
```

Output:

```
ID: 1, Name: Alice, Marks: 85.5
ID: 2, Name: Bob, Marks: 90.0
ID: 3, Name: Charlie, Marks: 78.0
```

---

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.

```
package day8_Assessment;
import java.util.ArrayList;
public class CopyArrayList {
   public static void main(String[] args) {
      ArrayList<String> firstList = new ArrayList<>();
      firstList.add("Apple");
      firstList.add("Banana");
      firstList.add("Cherry");
      ArrayList<String> secondList = new ArrayList<>();
      secondList.addAll(firstList);
      System.out.println("First List: " + firstList);
      System.out.println("Second List: " + secondList);
   }
```

}

Output:

First List: [Apple, Banana, Cherry]
Second List: [Apple, Banana, Cherry]

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.

```
package day8_Assessment;
import java.util.LinkedList;

public class CreateDisplayLinkedList {
   public static void main(String[] args) {
      LinkedList<String> colors = new LinkedList<>();
      colors.add("Red");
      colors.add("Blue");
      colors.add("Green");
      colors.add("Yellow");
      colors.add("Orange");
      for (String color : colors) {
         System.out.println(color);
      }
   }
}
```

Output:

Red
Blue
Green
Yellow

Orange

---

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.

```
package day8_Assessment;
import java.util.LinkedList;
public class AddFirstLast {
   public static void main(String[] args) {
      LinkedList<Integer> numbers = new LinkedList<>();
      numbers.add(20);
      numbers.add(30);
      numbers.add(40);
      numbers.addFirst(10);
      numbers.addLast(50);
      System.out.println("Updated LinkedList: " + numbers);
   }
}
```

Output:

Updated LinkedList: [10, 20, 30, 40, 50]

---

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.

```
package day8_Assessment;
```

```java
import java.util.LinkedList;
public class InsertElementAtPosition {
    public static void main(String[] args) {
        LinkedList<String> names = new LinkedList<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        names.add("David");
        System.out.println("Before insertion: " + names);
        names.add(2, "Eve");  // Insert "Eve" at index 2
        System.out.println("After insertion: " + names);
    }
}
```

Output:

```
Before insertion: [Alice, Bob, Charlie, David]
After insertion: [Alice, Bob, Eve, Charlie, David]
```

---

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.

```java
package day8_Assessment;
import java.util.LinkedList;
public class RemoveElementFromLinkedList {
    public static void main(String[] args) {
        LinkedList<String> animals = new LinkedList<>();
        animals.add("Lion");
        animals.add("Tiger");
        animals.add("Elephant");
        animals.add("Giraffe");
        animals.add("Zebra");
        System.out.println("Original list: " + animals);
        animals.removeFirst();
        System.out.println("After removing first element: " + animals);
```

```
        animals.removeLast();
        System.out.println("After removing last element: " + animals);
        animals.remove("Elephant");
        System.out.println("After removing 'Elephant': " + animals);
    }
}
```

Output:

```
Original list: [Lion, Tiger, Elephant, Giraffe, Zebra]
After removing first element: [Tiger, Elephant, Giraffe, Zebra]
After removing last element: [Tiger, Elephant, Giraffe]
After removing 'Elephant': [Tiger, Giraffe]
```

---

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.

```
package day8_Assessment;

import java.util.LinkedList;

import java.util.Scanner;

public class SearchElementInLinkedList {

    public static void main(String[] args) {

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

        list.add("Apple");

        list.add("Banana");

        list.add("Cherry");

        list.add("Date");

        list.add("Elderberry");

        Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter a string to search: ");
        String searchStr = scanner.nextLine();
        if (list.contains(searchStr)) {
            System.out.println(searchStr + " is found in the list.");
        } else {
            System.out.println(searchStr + " is NOT found in the list.");
        }
        scanner.close();
    }
}
```

Output:

Enter a string to search: Apple

Apple is found in the list.

---

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.

```
package day8_Assessment;
import java.util.LinkedList;
import java.util.ListIterator;
public class IterateListIterator {
    public static void main(String[] args) {
        LinkedList<String> cities = new LinkedList<>();
        cities.add("New York");
```

```
        cities.add("London");
        cities.add("Tokyo");
        cities.add("Paris");
        cities.add("Sydney");
        ListIterator<String> listIterator = cities.listIterator();
        System.out.println("Forward iteration:");
        while (listIterator.hasNext()) {
            System.out.println(listIterator.next());
        }
        System.out.println("\nReverse iteration:");
        while (listIterator.hasPrevious()) {
            System.out.println(listIterator.previous());
        }
    }
}
```

Output:

```
Forward iteration:
New York
London
Tokyo
Paris
Sydney

Reverse iteration:
Sydney
Paris
Tokyo
London
New York
```

---

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.

```
package day8_Assessment;
import java.util.Collections;
import java.util.LinkedList;
public class SortLinkedList {
    public static void main(String[] args) {
        LinkedList<Integer> numbers = new LinkedList<>();
        numbers.add(42);
        numbers.add(7);
        numbers.add(19);
        numbers.add(3);
        numbers.add(25);
        Collections.sort(numbers);
        System.out.println("Sorted LinkedList: " + numbers);
    }
}
```

Output:

```
Sorted LinkedList: [3, 7, 19, 25, 42]
```

---

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.

```
package day8_Assessment;
import java.util.ArrayList;
import java.util.LinkedList;
public class ConvertLinkedListToArrayList {
   public static void main(String[] args) {
      LinkedList<String> linkedList = new LinkedList<>();
      linkedList.add("Red");
      linkedList.add("Blue");
      linkedList.add("Green");
      linkedList.add("Yellow");
      ArrayList<String> arrayList = new ArrayList<>(linkedList);
      System.out.println("LinkedList: " + linkedList);
      System.out.println("ArrayList: " + arrayList);
   }
}
```

Output:

```
LinkedList: [Red, Blue, Green, Yellow]
ArrayList: [Red, Blue, Green, Yellow]
```

---

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.

```
package day8_Assessment;
import java.util.LinkedList;
class Book {
   int id;
   String title;
   String author;
   Book(int id, String title, String author) {
      this.id = id;
```

```
        this.title = title;
        this.author = author;
    }
}
public class StoreBooksInLinkedList {
    public static void main(String[] args) {
        LinkedList<Book> books = new LinkedList<>();
        books.add(new Book(1, "The Alchemist", "Paulo Coelho"));
        books.add(new Book(2, "1984", "George Orwell"));
        books.add(new Book(3, "To Kill a Mockingbird", "Harper Lee"));
        for (Book book : books) {
            System.out.println("ID: " + book.id + ", Title: " + book.title + ", Author:
" + book.author);
        }
    }
}
```

Output:

```
ID: 1, Title: The Alchemist, Author: Paulo Coelho
ID: 2, Title: 1984, Author: George Orwell
ID: 3, Title: To Kill a Mockingbird, Author: Harper Lee
```

---

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.

```
package day8_Assessment;
import java.util.LinkedList;
public class CloneLinkedList {
    public static void main(String[] args) {
        LinkedList<Integer> originalList = new LinkedList<>();
        originalList.add(5);
        originalList.add(10);
        originalList.add(15);
        originalList.add(20);
```

```
        LinkedList<Integer> clonedList = (LinkedList<Integer>)
originalList.clone();
        System.out.println("Original LinkedList: " + originalList);
        System.out.println("Cloned LinkedList: " + clonedList);
    }
}
```

Output:

```
Original LinkedList: [5, 10, 15, 20]
Cloned LinkedList: [5, 10, 15, 20]
```

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.

```
package day8_Assessment;

import java.util.Vector;

import java.util.Enumeration;

public class VectorOperations {

    public static void main(String[] args) {

        Vector<Integer> vector = new Vector<>();

        vector.add(10);

        vector.add(20);

        vector.add(30);

        vector.add(40);

        vector.add(50);

        vector.insertElementAt(25, 2);

        vector.remove(1);

        Enumeration<Integer> elements = vector.elements();
```

```
            System.out.println("Elements in Vector:");
            while (elements.hasMoreElements()) {
                System.out.println(elements.nextElement());
            }
        }
    }
```

Output:

Elements in Vector:

10

25

30

40

50

- **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.

```
package day8_Assessment;
import java.util.Vector;
public class VectorStringOperations {
    public static void main(String[] args) {
        Vector<String> names = new Vector<>();
        names.add("Alice");
        names.add("Bob");
        names.add("Charlie");
        names.add("Diana");
        if (names.contains("Charlie")) {
            System.out.println("Charlie is in the vector.");
        } else {
            System.out.println("Charlie is not in the vector.");
        }
```

```
        int index = names.indexOf("Bob");
        if (index != -1) {
            names.set(index, "Brian");
        }
        System.out.println("After replacement: " + names);
        names.clear();
        System.out.println("After clearing, vector size: " + names.size());
    }
}
```

Output:

```
Charlie is in the vector.
After replacement: [Alice, Brian, Charlie, Diana]
After clearing, vector size: 0
```

- **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**.

```
package day8_Assessment;
import java.util.Vector;

public class CopyVector {
    public static void main(String[] args) {
        Vector<Integer> vector1 = new Vector<>();
        vector1.add(10);
        vector1.add(20);
        vector1.add(30);
        vector1.add(40);
        Vector<Integer> vector2 = new Vector<>();
        vector2.addAll(vector1);
        System.out.println("Vector1: " + vector1);
        System.out.println("Vector2: " + vector2);
        if (vector1.equals(vector2))
        {
            System.out.println("Both vectors are equal.");
        } else {
            System.out.println("Vectors are not equal.");
```

```
        }
        int sum = sumVectorElements(vector1);
        System.out.println("Sum of elements in vector1: " + sum);
    }
    public static int sumVectorElements(Vector<Integer> vector) {
        int sum = 0;
        for (int num : vector) {
            sum += num;
        }
        return sum;
    }
}
```

Output:

```
Vector1: [10, 20, 30, 40]
Vector2: [10, 20, 30, 40]
Both vectors are equal.
Sum of elements in vector1: 100
```

- ---
-

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.

```
package day8_Assessment;
import java.util.Stack;
public class StackOperations {
    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        stack.push(10);
```

```
        stack.push(20);
        stack.push(30);
        stack.push(40);
        stack.push(50);
        System.out.println("Stack after pushes: " + stack);
        int popped = stack.pop();
        System.out.println("Popped element: " + popped);
        int top = stack.peek();
        System.out.println("Current top element: " + top);
        System.out.println("Is stack empty? " + stack.isEmpty());
    }
}
```

Output:

```
Stack after pushes: [10, 20, 30, 40, 50]
Popped element: 50
Current top element: 40
Is stack empty? false
```

- **Reverse a string using Stack**:
- Input a string from the user.
- Use a stack to reverse and print the string.

```
package day8_Assessment;

import java.util.Scanner;

import java.util.Stack;

public class ReverseStringUsingStack {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");

        String input = scanner.nextLine();

        Stack<Character> stack = new Stack<>();

        for (char ch : input.toCharArray()) {

            stack.push(ch);
```

```
        }
        StringBuilder reversed = new StringBuilder();
        while (!stack.isEmpty()) {
            reversed.append(stack.pop());
        }
        System.out.println("Reversed string: " + reversed.toString());
        scanner.close();
    }
}
```

Output:

Enter a string: manasa

Reversed string: asanam

- **Use Stack to check for balanced parentheses** in an expression.
- Input: (a+b) * (c-d)
- Output: Valid or Invalid expression

```
package day8_Assessment;
import java.util.Scanner;
import java.util.Stack;
public class BalancedParentheses {
    public static boolean isBalanced(String expression) {
        Stack<Character> stack = new Stack<>();
        for (char ch : expression.toCharArray()) {
            if (ch == '(') {
                stack.push(ch);
            } else if (ch == ')') {
```

```
                if (stack.isEmpty()) {
                    return false;
                }
                stack.pop();
            }
        }
        return stack.isEmpty();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an expression: ");
        String expr = scanner.nextLine();
        if (isBalanced(expr)) {
            System.out.println("Valid expression");
        } else {
            System.out.println("Invalid expression");
        }
        scanner.close();
    }
}
```

Output: Enter an expression: (a+b)*(a-b)

Valid expression

- **Convert a decimal number to binary using Stack**.

```
package day8_Assessment;
import java.util.Scanner;
import java.util.Stack;
public class DecimalToBinaryStack {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a decimal number: ");
        int decimal = scanner.nextInt();
        Stack<Integer> stack = new Stack<>();
        if (decimal == 0) {
            System.out.println("Binary: 0");
            scanner.close();
            return;
        }
        while (decimal > 0) {
            stack.push(decimal % 2);
            decimal = decimal / 2;
        }
        System.out.print("Binary: ");
        while (!stack.isEmpty()) {
            System.out.print(stack.pop());
        }
        scanner.close();
    }
}
```

Output:

Enter a decimal number: 23

Binary: 10111

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.

```
package day8_Assessment;

import java.util.HashSet;

import java.util.Iterator;

public class HashSetOperations {

   public static void main(String[] args) {

      HashSet<String> cities = new HashSet<>();

      cities.add("New York");

      cities.add("London");

      cities.add("Tokyo");

      cities.add("Paris");

      cities.add("Sydney");

      boolean added = cities.add("Tokyo");

      System.out.println("Trying to add duplicate 'Tokyo': " + (added ? "Added" : "Not Added"));

      System.out.println("Cities in the HashSet:");

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

      while (iterator.hasNext()) {

         System.out.println(iterator.next());

      }
```

```
        cities.remove("Paris");
        System.out.println("\nAfter removing 'Paris': " + cities);
        String cityToCheck = "London";
        System.out.println("Does '" + cityToCheck + "' exist? " +
cities.contains(cityToCheck));
        cities.clear();
        System.out.println("After clearing, size of HashSet: " + cities.size());
        HashSet<Integer> numbers = new HashSet<>();
        numbers.add(10);
        numbers.add(35);
        numbers.add(20);
        numbers.add(50);
        numbers.add(5);
        System.out.println("\nNumbers in HashSet: " + numbers);
        int max = getMax(numbers);
        System.out.println("Maximum element in numbers HashSet: " +
max);
    }
    public static int getMax(HashSet<Integer> set) {
        int max = Integer.MIN_VALUE;
        for (int num : set) {
            if (num > max) {
                max = num;
            }
        }
        return max;
    }
}
```

Output:

```
Trying to add duplicate 'Tokyo': Not Added
Cities in the HashSet:
New York
Tokyo
London
Paris
Sydney

After removing 'Paris': [New York, Tokyo, London, Sydney]
Does 'London' exist? true
After clearing, size of HashSet: 0

Numbers in HashSet: [50, 35, 20, 5, 10]
Maximum element in numbers HashSet: 50
```

LinkedHashSet

1.Create a LinkedHashSet of Integers:

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

```
package day8_Assessment;
import java.util.LinkedHashSet;
public class LinkedHashSetExample {
   public static void main(String[] args) {
      LinkedHashSet<Integer> numbers = new LinkedHashSet<>();
      numbers.add(10);
      numbers.add(5);
      numbers.add(20);
      numbers.add(15);
      numbers.add(5);
      System.out.println("Elements in LinkedHashSet:");
      for (int num : numbers) {
         System.out.println(num);
      }
   }
}
```

Output:

Elements in LinkedHashSet:
10
5
20
15

2. **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.

```
package day8_Assessment;
import java.util.LinkedHashSet;
import java.util.Objects;
public class LinkedHashSetCustomObjects {
    static class Student {
        int id;
        String name;
        Student(int id, String name) {
            this.id = id;
            this.name = name;
        }
        public boolean equals(Object obj) {
            if (this == obj) return true;
            if (obj == null || getClass() != obj.getClass()) return false;
            Student other = (Student) obj;
            return id == other.id && Objects.equals(name, other.name);
        }
```

```
    public int hashCode() {
        return Objects.hash(id, name);
    }
    public String toString() {
        return "Student{id=" + id + ", name='" + name + "'}";
    }
}
public static void main(String[] args) {
    LinkedHashSet<Student> students = new
LinkedHashSet<>();
    Student s1 = new Student(1, "Alice");
    Student s2 = new Student(2, "Bob");
    Student s3 = new Student(3, "Charlie");
    Student sDuplicate = new Student(2, "Bob");
    students.add(s1);
    students.add(s2);
    students.add(s3);
    boolean added = students.add(sDuplicate);
    System.out.println("Trying to add duplicate student: " +
(added ? "Added" : "Not Added"));
    System.out.println("Students in LinkedHashSet:");
    for (Student s : students) {
        System.out.println(s);
    }
}
}
```

Output:

```
Trying to add duplicate student: Not Added
```

Students in LinkedHashSet:
Student{id=1, name='Alice'}
Student{id=2, name='Bob'}
Student{id=3, name='Charlie'}

3. **Write a program** to:
    - Merge two LinkedHashSets and print the result.

```
package day8_Assessment;
import java.util.LinkedHashSet;
public class MergeLinkedHashSets {
   public static void main(String[] args) {
      LinkedHashSet<String> set1 = new LinkedHashSet<>();
      set1.add("Apple");
      set1.add("Banana");
      set1.add("Cherry");
      LinkedHashSet<String> set2 = new LinkedHashSet<>();
      set2.add("Date");
      set2.add("Banana");
      set2.add("Elderberry");
      set1.addAll(set2);
      System.out.println("Merged LinkedHashSet: " + set1);
   }
}
```

Output:

Merged LinkedHashSet: [Apple, Banana, Cherry, Date, Elderberry]

TreeSet

1. Create a TreeSet of Strings:
- Add 5 country names in random order.
- Print the sorted list of countries using TreeSet.

```
package day8_Assessment;
import java.util.TreeSet;
public class TreeSetStrings {
```

```
    public static void main(String[] args) {
        TreeSet<String> countries = new TreeSet<>();
        countries.add("India");
        countries.add("Germany");
        countries.add("Brazil");
        countries.add("Australia");
        countries.add("Canada");
        System.out.println("Sorted countries:");
        for (String country : countries) {
            System.out.println(country);
        }
    }
}
```

Output:

```
Sorted countries:
Australia
Brazil
Canada
Germany
India
```

2. **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.

```
package day8_Assessment;
import java.util.TreeSet;
public class TreeSetIntegers {
    public static void main(String[] args) {
        TreeSet<Integer> numbers = new TreeSet<>();

        numbers.add(10);
        numbers.add(20);
        numbers.add(30);
        numbers.add(40);
        numbers.add(50);
```

```
        System.out.println("First element: " + numbers.first());
        System.out.println("Last element: " + numbers.last());

        int givenNumber = 30;
        System.out.println("Element lower than " + givenNumber + ": " +
numbers.lower(givenNumber));
        System.out.println("Element higher than " + givenNumber + ": " +
numbers.higher(givenNumber));
    }
}
```

Output:

```
First element: 10
Last element: 50
Element lower than 30: 20
Element higher than 30: 40
```

3. **Create a TreeSet with a custom comparator**:
    - Sort strings in **reverse alphabetical order** using Comparator.

```
package day8_Assessment;

import java.util.TreeSet;

import java.util.Comparator;

public class TreeSetComparator {

    public static void main(String[] args) {

        Comparator<String> reverseComparator = (s1, s2) ->
s2.compareTo(s1);

        TreeSet<String> names = new
TreeSet<>(reverseComparator);

        names.add("Alice");

        names.add("Bob");

        names.add("Charlie");

        names.add("David");

        names.add("Eve");
```

```
        System.out.println("Strings in reverse alphabetical order:");
        for (String name : names) {
            System.out.println(name);
        }
    }
}
```

Output:

```
Strings in reverse alphabetical order:
Eve
David
Charlie
Bob
Alice
```