List (ArrayList)

1. Create and Display an ArrayList

Write a program to:

- Create an ArrayList of Strings.
- Add 5 fruits to it.
- Display the ArrayList using System.out.println().

```
import java.util.ArrayList;

public class CreateArrayList {

  public static void main(String[] args) {

    ArrayList<String> fruits = new ArrayList<String>();

    fruits.add("Apple");

    fruits.add("Banana");

    fruits.add("Mango");

    fruits.add("Orange");

    fruits.add("Grapes");

    System.out.println(fruits);

  }

}
```

Output:

```
[Apple, Banana, Mango, Orange, Grapes]
```

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.

```
import java.util.ArrayList;

import java.util.Scanner;


public class SearchArrayList {

  public static void main(String[] args) {

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

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

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number to search: ");
        int num = sc.nextInt();

        if(numbers.contains(num)) {
            System.out.println(num + " is found in the list");
        } else {
            System.out.println(num + " is NOT found in the list");
        }
        sc.close();
    }
}
```

Output :

```
Enter number to search: 30
30 is found 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.

```
import java.util.ArrayList;

public class RemoveElement {
    public static void main(String[] args) {
        ArrayList<String> fruits = new ArrayList<String>();
        fruits.add("Apple");
        fruits.add("Banana");
        fruits.add("Mango");
        fruits.add("Orange");
        fruits.add("Grapes");

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

        fruits.remove("Mango");

        System.out.println("After removal: " + fruits);
    }
}
```

Output:

```
Original list: [Apple, Banana, Mango, Orange, Grapes]
After removal: [Apple, Banana, 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.

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

public class SortArrayList {
    public static void main(String[] args) {
        ArrayList<Integer> numbers = new ArrayList<Integer>();
        numbers.add(34);
        numbers.add(12);
        numbers.add(45);
        numbers.add(9);
        numbers.add(1);
        numbers.add(29);
        numbers.add(10);

        System.out.println("Before sort: " + numbers);
        Collections.sort(numbers);
        System.out.println("After sort: " + numbers);
    }
}
```

Output:

```
Before sort: [34, 12, 45, 9, 1, 29, 10]
After sort: [1, 9, 10, 12, 29, 34, 45]
```

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.

```
import java.util.ArrayList;
import java.util.Collections;
public class ReverseArrayList {
    public static void main(String[] args) {
        ArrayList<Character> chars = new ArrayList<Character>();
        chars.add('A');
        chars.add('B');
        chars.add('C');
        chars.add('D');
        chars.add('E');

        System.out.println("Original list: " + chars);
        Collections.reverse(chars);
        System.out.println("Reversed list: " + chars);
    }
}
```

Output:

```
Original list: [A, B, C, D, E]
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.

```
import java.util.ArrayList;

public class UpdateElement {

  public static void main(String[] args) {

    ArrayList<String> subjects = new ArrayList<String>();

    subjects.add("Physics");

    subjects.add("Chemistry");

    subjects.add("Math");

    subjects.add("Biology");

    subjects.add("English");


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


    subjects.set(2, "Statistics"); // Replace Math


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

  }

}
```

Output:

```
Before update: [Physics, Chemistry, Math, Biology, English]
After update: [Physics, Chemistry, Statistics, Biology, English]
```

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.

```
import java.util.ArrayList;

public class RemoveAllElements {

   public static void main(String[] args) {

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

      numbers.add(5);

      numbers.add(10);

      numbers.add(15);

      numbers.add(20);


      System.out.println("Before clear size: " + numbers.size());


      numbers.clear();


      System.out.println("After clear size: " + numbers.size());

   }

}
```

Output:

```
Before clear size: 4
After clear size: 0
```

8. Iterate using Iterator

Write a program to:

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

```
import java.util.ArrayList;

import java.util.Iterator;

public class IterateWithIterator {

   public static void main(String[] args) {

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

      cities.add("Delhi");
```

```
        cities.add("Mumbai");
        cities.add("Chennai");
        cities.add("Kolkata");

        Iterator<String> it = cities.iterator();
        while(it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

Output:

```
Delhi
Mumbai
Chennai
Kolkata
```

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.

```
import java.util.ArrayList;
class Student {
    int id;
    String name;
    int marks;

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


public class StudentArrayList {
    public static void main(String[] args) {
        ArrayList<Student> students = new ArrayList<Student>();
        students.add(new Student(1, "Amit", 85));
        students.add(new Student(2, "Ravi", 90));
        students.add(new Student(3, "Sunil", 75));


        for(Student s : students) {
            System.out.println("ID: " + s.id + ", Name: " + s.name + ", Marks: " + s.marks);
        }
    }
}
```

Output:

```
ID: 1, Name: Amit, Marks: 85
ID: 2, Name: Ravi, Marks: 90
ID: 3, Name: Sunil, Marks: 75
```

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.

```
import java.util.ArrayList;
public class CopyArrayList {
    public static void main(String[] args) {
        ArrayList<String> list1 = new ArrayList<String>();
        list1.add("Apple");
        list1.add("Banana");
        list1.add("Mango");


        ArrayList<String> list2 = new ArrayList<String>();
        list2.addAll(list1);

        System.out.println("List1: " + list1);
        System.out.println("List2: " + list2);
    }
}
```

Output:

```
List1: [Apple, Banana, Mango]
List2: [Apple, Banana, Mango]
```

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.

```
import java.util.LinkedList;
public class CreateLinkedList {
    public static void main(String[] args) {
        LinkedList<String> colors = new LinkedList<String>();
        colors.add("Red");
        colors.add("Green");
        colors.add("Blue");
        colors.add("Yellow");
        colors.add("Black");

        for(String color : colors) {
            System.out.println(color);
        }
    }
}
```

Output:

```
Red
```

```
Green
Blue
Yellow
Black
```

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.

```
import java.util.LinkedList;
public class AddFirstLast {
  public static void main(String[] args) {
    LinkedList<Integer> list = new LinkedList<Integer>();
    list.add(20);
    list.add(30);
    list.add(40);

    list.addFirst(10);
    list.addLast(50);

    System.out.println(list);
  }
}
```

Output:

```
[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.

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

```
public class InsertAtPosition {
    public static void main(String[] args) {
        LinkedList<String> names = new LinkedList<String>();
        names.add("Amit");
        names.add("Ravi");
        names.add("Sunil");

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

        names.add(2, "Kiran");

        System.out.println("After insertion: " + names);
    }
}
```

Output:

```
Before insertion: [Amit, Ravi, Sunil]
After insertion: [Amit, Ravi, Kiran, Sunil]
```

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.

```
import java.util.LinkedList;
public class RemoveElements {
    public static void main(String[] args) {
        LinkedList<String> animals = new LinkedList<String>();
        animals.add("Dog");
        animals.add("Cat");
        animals.add("Horse");
        animals.add("Elephant");
        animals.add("Lion");
        System.out.println("Original list: " + animals);


        animals.removeFirst();

        System.out.println("After removing first: " + animals);


        animals.removeLast();

        System.out.println("After removing last: " + animals);


        animals.remove("Horse");

        System.out.println("After removing Horse: " + animals);

    }

}
```

Output:

```
Original list: [Dog, Cat, Horse, Elephant, Lion]
After removing first: [Cat, Horse, Elephant, Lion]
After removing last: [Cat, Horse, Elephant]
After removing Horse: [Cat, Elephant]
```

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.

```
import java.util.LinkedList;

import java.util.Scanner;

public class SearchLinkedList {

    public static void main(String[] args) {

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

        list.add("Apple");

        list.add("Banana");

        list.add("Mango");

        list.add("Orange");
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter fruit to search: ");
        String fruit = sc.nextLine();

        if(list.contains(fruit)) {
            System.out.println(fruit + " is found in the list");
        } else {
            System.out.println(fruit + " is NOT found in the list");
        }
        sc.close();
    }
}
```

Output :

```
Enter fruit to search: Mango
Mango 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.

```
import java.util.LinkedList;
import java.util.ListIterator;
public class ListIteratorDemo {
    public static void main(String[] args) {
        LinkedList<String> cities = new LinkedList<String>();
        cities.add("Delhi");
        cities.add("Mumbai");
        cities.add("Chennai");
        cities.add("Kolkata");

        ListIterator<String> it = cities.listIterator();
        System.out.println("Forward iteration:");

        while(it.hasNext()) {

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

        }


        System.out.println("Backward iteration:");

        while(it.hasPrevious()) {

            System.out.println(it.previous());

        }

    }

}
```

Output:

```
Forward iteration:
Delhi
Mumbai
Chennai
Kolkata
Backward iteration:
Kolkata
Chennai
Mumbai
Delhi
```

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.

```
import java.util.LinkedList;

import java.util.Collections;

public class SortLinkedList {

    public static void main(String[] args) {

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

        numbers.add(25);
        numbers.add(5);
        numbers.add(15);
        numbers.add(10);
        numbers.add(20);

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

        Collections.sort(numbers);

        System.out.println("After sort: " + numbers);
    }
}
```

Output:

```
Before sort: [25, 5, 15, 10, 20]
After sort: [5, 10, 15, 20, 25]
```

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.

```
import java.util.LinkedList;
import java.util.ArrayList;
public class LinkedListToArrayList {
    public static void main(String[] args) {
        LinkedList<String> linkedList = new LinkedList<String>();
        linkedList.add("Red");
        linkedList.add("Green");
        linkedList.add("Blue");

        ArrayList<String> arrayList = new ArrayList<String>(linkedList);

        System.out.println("LinkedList: " + linkedList);
        System.out.println("ArrayList: " + arrayList);
    }
}
```

Output:

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

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.

```
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 BookLinkedList {
    public static void main(String[] args) {
        LinkedList<Book> books = new LinkedList<Book>();
        books.add(new Book(1, "Java Programming", "John"));
        books.add(new Book(2, "Python Basics", "Anna"));
        books.add(new Book(3, "C++ Guide", "Steve"));

        for(Book b : books) {
            System.out.println("ID: " + b.id + ", Title: " + b.title + ", Author: " + b.author);
        }
    }
}
```

Output:

```
ID: 1, Title: Java Programming, Author: John
ID: 2, Title: Python Basics, Author: Anna
ID: 3, Title: C++ Guide, Author: Steve
```

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.

```
import java.util.LinkedList;

public class CloneLinkedList {
    public static void main(String[] args) {
        LinkedList<Integer> list1 = new LinkedList<Integer>();
        list1.add(10);
        list1.add(20);
        list1.add(30);

        LinkedList<Integer> list2 = (LinkedList<Integer>)list1.clone();

        System.out.println("Original list: " + list1);
        System.out.println("Cloned list: " + list2);
    }
}
```

Output:

```
Original list: [10, 20, 30]
Cloned list: [10, 20, 30]
```

Vector

Vector of Integers - add, insert, remove, Enumeration

```
import java.util.Vector;
import java.util.Enumeration;
public class VectorDemo {
    public static void main(String[] args) {
        Vector<Integer> v = new Vector<Integer>();


        v.add(10);
        v.add(20);
        v.add(30);
        v.add(40);
        v.add(50);


        v.insertElementAt(25, 2);


        System.out.println("Vector elements: " + v);


        v.removeElement(40);


        System.out.println("After removal: " + v);


        System.out.println("Using Enumeration:");
        Enumeration<Integer> en = v.elements();
        while(en.hasMoreElements()) {
            System.out.println(en.nextElement());
        }
    }
}
```

Output:

```
Vector elements: [10, 20, 25, 30, 40, 50]
After removal: [10, 20, 25, 30, 50]
Using Enumeration:
10
20
25
```

```
30
50
```

Vector of Strings - contains, replace, clear

```
import java.util.Vector;

public class VectorStringOps {

    public static void main(String[] args) {

        Vector<String> v = new Vector<String>();

        v.add("Amit");

        v.add("Rahul");

        v.add("Kiran");

        v.add("Sunil");


        System.out.println("Vector: " + v);


        System.out.println("Contains Rahul? " + v.contains("Rahul"));


        v.set(2, "Ravi"); // Replace Kiran


        System.out.println("After replace: " + v);


        v.clear();


        System.out.println("After clear size: " + v.size());

    }

}
```

Output:

```
Vector: [Amit, Rahul, Kiran, Sunil]
Contains Rahul? true
After replace: [Amit, Rahul, Ravi, Sunil]
After clear size: 0
```

Stack

Stack operations and string reverse using Stack

```
import java.util.Stack;
```

```java
import java.util.Scanner;
public class StackDemo {
    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<Integer>();

        stack.push(10);
        stack.push(20);
        stack.push(30);
        stack.push(40);
        stack.push(50);

        System.out.println("Stack: " + stack);

        System.out.println("Pop: " + stack.pop());

        System.out.println("Peek: " + stack.peek());

        System.out.println("Is empty? " + stack.empty());

        // Reverse string
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter string to reverse: ");
        String str = sc.nextLine();

        Stack<Character> s = new Stack<Character>();
        for(char c : str.toCharArray()) {
            s.push(c);
        }

        StringBuilder reversed = new StringBuilder();
        while(!s.empty()) {
            reversed.append(s.pop());
        }

        System.out.println("Reversed string: " + reversed.toString());
        sc.close();
    }
}
```

Output :

```
Stack: [10, 20, 30, 40, 50]
Pop: 50
Peek: 40
Is empty? false
Enter string to reverse: hello
Reversed string: olleh
```

HashSet

HashSet operations with cities

```
import java.util.HashSet;
import java.util.Iterator;
public class HashSetDemo {
    public static void main(String[] args) {
        HashSet<String> set = new HashSet<String>();

        set.add("Delhi");
        set.add("Mumbai");
        set.add("Chennai");
        set.add("Kolkata");
        set.add("Mumbai"); // duplicate

        System.out.println("HashSet: " + set);

        System.out.println("Contains Delhi? " + set.contains("Delhi"));

        set.remove("Chennai");
        System.out.println("After remove: " + set);

        Iterator<String> it = set.iterator();
        while(it.hasNext()) {
            System.out.println(it.next());
        }

        set.clear();
        System.out.println("After clear, is empty? " + set.isEmpty());
    }
}
```

Output:

```
HashSet: [Delhi, Mumbai, Kolkata, Chennai]
Contains Delhi? true
After remove: [Delhi, Mumbai, Kolkata]
Delhi
Mumbai
Kolkata
After clear, is empty? true
```

LinkedHashSet

LinkedHashSet operations (integers with duplicates)

```
import java.util.LinkedHashSet;
public class LinkedHashSetDemo {
    public static void main(String[] args) {
        LinkedHashSet<Integer> set = new LinkedHashSet<Integer>();

        set.add(10);
        set.add(5);
        set.add(20);
        set.add(15);
        set.add(5); // duplicate
        System.out.println(set);
    }
}
```

Output:

```
[10, 5, 20, 15]
```

TreeSet

TreeSet operations with countries

```
import java.util.TreeSet;

public class TreeSetDemo {

    public static void main(String[] args) {

        TreeSet<String> countries = new TreeSet<String>();


        countries.add("India");

        countries.add("USA");

        countries.add("Australia");

        countries.add("Germany");

        countries.add("France");


        System.out.println(countries);


        System.out.println("First: " + countries.first());

        System.out.println("Last: " + countries.last());


        System.out.println("Lower than Germany: " + countries.lower("Germany"));

        System.out.println("Higher than Germany: " + countries.higher("Germany"));

    }

}
```

Output:

```
[Australia, France, Germany, India, USA]
First: Australia
Last: USA
Lower than Germany: France
Higher than Germany: India
```

Queue

Bank Queue simulation using LinkedList

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

public class BankQueue {
  public static void main(String[] args) {
    Queue<String> queue = new LinkedList<String>();

    queue.add("Customer1");
    queue.add("Customer2");
    queue.add("Customer3");
    queue.add("Customer4");
    queue.add("Customer5");

    while(!queue.isEmpty()) {
      System.out.println("Serving: " + queue.poll());
      System.out.println("Queue now: " + queue);
    }
  }
}
```

Output:

```
Serving: Customer1
Queue now: [Customer2, Customer3, Customer4, Customer5]
Serving: Customer2
Queue now: [Customer3, Customer4, Customer5]
Serving: Customer3
Queue now: [Customer4, Customer5]
Serving: Customer4
Queue now: [Customer5]
Serving: Customer5
Queue now: []
```

PriorityQueue

PriorityQueue for hospital emergency with comparator

```
import java.util.PriorityQueue;
import java.util.Comparator;

class Patient {
    String name;
    int severity;

    Patient(String name, int severity) {
        this.name = name;
        this.severity = severity;
    }

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

public class EmergencyQueue {
    public static void main(String[] args) {
        PriorityQueue<Patient> pq = new PriorityQueue<Patient>(new Comparator<Patient>() {
            public int compare(Patient p1, Patient p2) {
                return p2.severity - p1.severity; // high severity first
            }
        });

        pq.add(new Patient("John", 5));
        pq.add(new Patient("Alice", 9));
        pq.add(new Patient("Bob", 3));

        while(!pq.isEmpty()) {
            System.out.println("Serving: " + pq.poll());
        }
    }
}
```

Output:

```
Serving: Alice(9)
Serving: John(5)
Serving: Bob(3)
```

Deque

Palindrome checker using Deque

```
import java.util.ArrayDeque;

import java.util.Deque;

import java.util.Scanner;

public class PalindromeChecker {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

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

        String str = sc.nextLine();


        Deque<Character> deque = new ArrayDeque<Character>();


        for(char c : str.toCharArray()) {

            deque.addLast(c);

        }


        boolean isPalindrome = true;


        while(deque.size() > 1) {

            if(deque.removeFirst() != deque.removeLast()) {

                isPalindrome = false;

                break;

            }

        }
        if(isPalindrome) {

            System.out.println(str + " is palindrome");

        } else {

            System.out.println(str + " is NOT palindrome");

        }


        sc.close();

    }

}
```

Output:

```
Enter string: madam
madam is palindrome
```