**Code:**

/\*

 \* Name: Shahad Eid Albalawi

 \* ID: 438002072

 \* Section: 4C3

 \*/

package shahad\_eid\_albalawi\_438002072;

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

import java.util.Stack;

/\*\*

 \*

 \* @author Dell

 \*/

public class patientInfo {

    private String PatientName;

    private int PatientID;

    public String getPatientName() {

        return PatientName;

    }

    public void setPatientName(String PatientName) {

        this.PatientName = PatientName;

    }

    public int getPatientID() {

        return PatientID;

    }

    public void setPatientID(int PatientID) {

        this.PatientID = PatientID;

    }

    @Override

    public String toString() {

        return "patientInfo{" + "PatientName=" + PatientName + ", PatientID=" + PatientID + '}';

    }

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int size1, size2;

        String ADT1 = "", ADT2 = "";

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        System.out.println("\* 1:Merging two Single Linked List structure type                                        \*");

        System.out.println("\* 2:Merging two Stacks data structure type                                               \*");

        System.out.println("\* 3:Merging two Queses data structure type                                               \*");

        System.out.println("\* 4:Merging Single Linked List with Stack to Linked List                                 \*");

        System.out.println("\* 5:Merging Single Linked List with Queue to Linked List                                 \*");

        System.out.println("\* 6:Merging Singly Linked List with Stack only Patient’s name that start with S to Queue \*");

        System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

        System.out.println("Input Choise >>> ");

        int choise = input.nextInt();

        switch (choise) {

            case 1:

                ADT1 = "Single Linked List";

                ADT2 = "Single Linked List";

                size1 = readSize(ADT1, "First");

                size2 = readSize(ADT2, "Second");

                LinkedList<patientInfo> list1,

                 list2;

                LinkedList<patientInfo> mergedList = new LinkedList<patientInfo>();

                list1 = readLinkedList(size1);

                mergedList.addAll(list1);

                list2 = readLinkedList(size2);

                mergedList.addAll(list2);

                System.out.println(" ----------- Before Merge -----------");

                System.out.println("---- list1 ");

                System.out.println(list1);

                System.out.println("---- list2 ");

                System.out.println(list2);

                System.out.println(" ----------- After Merge -----------");

                System.out.println("---- mergedList ");

                System.out.println(mergedList);

                break;

            case 2:

                ADT1 = "Stack";

                ADT2 = "Stack";

                size1 = readSize(ADT1, "First");

                size2 = readSize(ADT2, "Second");

                Stack<patientInfo> stack1,

                 stack2;

                Stack<patientInfo> mergedStack = new Stack<patientInfo>();

                stack1 = readStack(size1);

                mergedStack.addAll(stack1);

                stack2 = readStack(size2);

                mergedStack.addAll(stack2);

                System.out.println(" ----------- Before Merge -----------");

                System.out.println("---- stack1 ");

                System.out.println(stack1);

                System.out.println("---- stack2 ");

                System.out.println(stack2);

                System.out.println(" ----------- After Merge -----------");

                System.out.println("---- mergedStack ");

                System.out.println(mergedStack);

                break;

            case 3:

                ADT1 = "Queue";

                ADT2 = "Queue";

                size1 = readSize(ADT1, "First");

                size2 = readSize(ADT2, "Second");

                Queue<patientInfo> queue1,

                 queue2;

                Queue<patientInfo> mergedQueue = new LinkedList<patientInfo>();

                queue1 = readQueue(size1);

                mergedQueue.addAll(queue1);

                queue2 = readQueue(size2);

                mergedQueue.addAll(queue2);

                System.out.println(" ----------- Before Merge -----------");

                System.out.println("---- queue1 ");

                System.out.println(queue1);

                System.out.println("---- queue2 ");

                System.out.println(queue2);

                System.out.println(" ----------- After Merge -----------");

                System.out.println("---- mergedQueue ");

                System.out.println(mergedQueue);

                break;

            case 4:

                ADT1 = "Single Linked List";

                ADT2 = "Stack";

                size1 = readSize(ADT1, "First");

                size2 = readSize(ADT2, "Second");

                LinkedList<patientInfo> list1\_1;

                Stack<patientInfo> stack2\_1;

                LinkedList<patientInfo> mergedlist1\_1 = new LinkedList<patientInfo>();

                list1\_1 = readLinkedList(size1);

                mergedlist1\_1.addAll(list1\_1);

                stack2\_1 = readStack(size2);

                mergedlist1\_1.addAll(stack2\_1);

                System.out.println(" ----------- Before Merge -----------");

                System.out.println("---- list1 ");

                System.out.println(list1\_1);

                System.out.println("---- stack2 ");

                System.out.println(stack2\_1);

                System.out.println(" ----------- After Merge -----------");

                System.out.println("---- mergedList ");

                System.out.println(mergedlist1\_1);

                break;

            case 5:

                ADT1 = "Single Linked List";

                ADT2 = "Queue";

                size1 = readSize(ADT1, "First");

                size2 = readSize(ADT2, "Second");

                LinkedList<patientInfo> list1\_2;

                Queue<patientInfo> queue2\_1;

                LinkedList<patientInfo> mergedlist1\_2 = new LinkedList<patientInfo>();

                list1\_2 = readLinkedList(size1);

                mergedlist1\_2.addAll(list1\_2);

                queue2\_1 = readQueue(size2);

                mergedlist1\_2.addAll(queue2\_1);

                System.out.println(" ----------- Before Merge -----------");

                System.out.println("---- list1 ");

                System.out.println(list1\_2);

                System.out.println("---- queue2 ");

                System.out.println(queue2\_1);

                System.out.println(" ----------- After Merge -----------");

                System.out.println("---- mergedList ");

                System.out.println(mergedlist1\_2);

                break;

            case 6:

                ADT1 = "Single Linked List";

                ADT2 = "Stack";

                size1 = readSize(ADT1, "First");

                size2 = readSize(ADT2, "Second");

                LinkedList<patientInfo> list1\_3;

                Stack<patientInfo> stack2\_2;

                Queue<patientInfo> mergedlist1\_3 = new LinkedList<patientInfo>();

                list1\_3 = readLinkedList(size1);

                mergedlist1\_3.addAll(list1\_3);

                stack2\_2 = readStack(size2);

                for (patientInfo info : stack2\_2) {

                    if (info.PatientName.startsWith("S")) {

                        mergedlist1\_3.add(info);

                    }

                }

                System.out.println(" ----------- Before Merge -----------");

                System.out.println("---- list1 ");

                System.out.println(list1\_3);

                System.out.println("---- stack2 ");

                System.out.println(stack2\_2);

                System.out.println(" ----------- After Merge -----------");

                System.out.println("---- mergedList ");

                System.out.println(mergedlist1\_3);

                break;

            default:

                System.err.println("Invalid Input");

                System.exit(0);

        }

    }

    public static int readSize(String ADT, String Order) {

        System.out.println("Enter The Size of The " + Order + " " + ADT + " >>> ");

        return new Scanner(System.in).nextInt();

    }

    public static patientInfo readPatientInfo(int Order) {

        patientInfo info = new patientInfo();

        System.out.println(" ------------------ Reading patientInfo " + Order + " ------------------ ");

        System.out.print("Enter Patient Name  >>> ");

        info.setPatientName(new Scanner(System.in).next());

        System.out.print("Enter Patient ID  >>> ");

        info.setPatientID(new Scanner(System.in).nextInt());

        return info;

    }

    public static LinkedList<patientInfo> readLinkedList(int size) {

        LinkedList<patientInfo> infos = new LinkedList<>();

        for (int i = 0; i < size; i++) {

            infos.add(readPatientInfo(i + 1));

        }

        return infos;

    }

    public static Stack<patientInfo> readStack(int size) {

        Stack<patientInfo> infos = new Stack<>();

        for (int i = 0; i < size; i++) {

            infos.push(readPatientInfo(i + 1));

        }

        return infos;

    }

    public static Queue<patientInfo> readQueue(int size) {

        Queue<patientInfo> infos = new LinkedList<>();

        for (int i = 0; i < size; i++) {

            infos.add(readPatientInfo(i + 1));

        }

        return infos;

    }

}

**Output:**



