# PROJECT REPORT

**OF**

**ADVANCED JAVA PROGRAMMING**

**ON**

**IMPLEMENTING STACK AND QUEUE OPERATIONS ON LINKEDLIST USING GUI**

## PROJECT DESCRIPTION

QUESTION-:

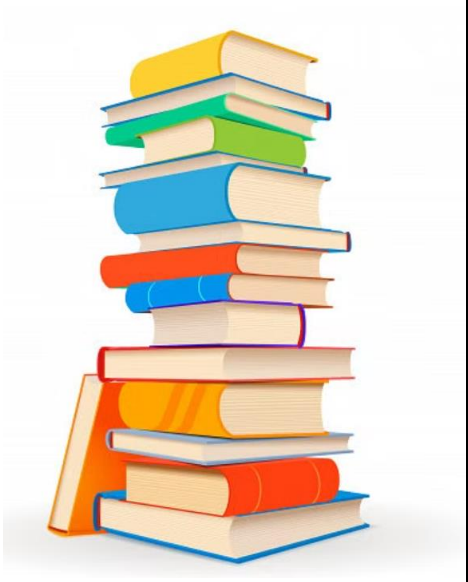
Write a program using JCF (single java file based project with java file name as linklist) to achieve the following:

Write a program that randomly stores 10 integer numbers (to start with) in a LinkedList object. Using this LinkedList object, implement stack and queue operations using GUI as follows: Add 2 Radio Buttons for choosing options (a) Stack and (b) Queue, and 4 JButtons (Push, Pop for Stack, and Add, Delete for Queue).

Once you run the program, it should display a LinkedList of 10 random integers in a text field. You should then be able to choose a radio button of either Stack or Queue, followed by one of its operations (Push or Pop Buttons for Stack ONLY, and Add or Delete Buttons for Queue ONLY (Invalid choices like choosing Stack followed by pressing ADD button should not work). The program should then display the modified list (as per the operation undertaken) in the text field.

## ➔INTRODUCTION

### Stack:



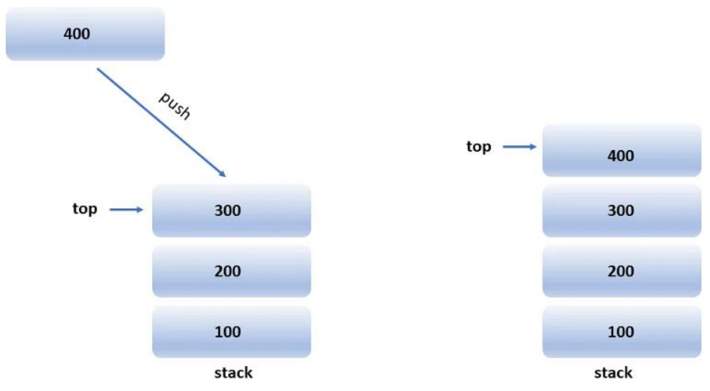
The stack data structure is a linear data structure that accompanies a principle known as LIFO (Last In First Out) or FILO (First In Last Out). Real-life examples of a stack are a deck of cards, piles of books, piles of money, and many more.

It contains only one pointer **top pointer** pointing to the topmost element of the stack. Whenever an element is added in the stack, it is added on the top of the stack, and the element can be deleted only from the stack. In other words, a ***stack can be defined as a container in which insertion and deletion can be done from the one end known as the top of the stack.***

### Basic Operations on Stack

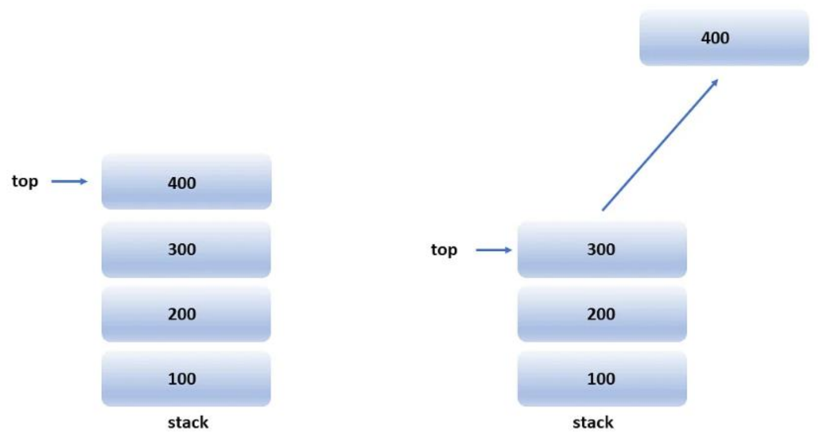
**Push Operation-** Push operation involves inserting new elements in the stack. Since you have only one end to insert a unique element on top of the stack, it inserts the new element at the top of the stack.

- **push():** When we insert an element in a stack then the operation is known as a push. If the stack is full then the overflow condition occurs.



**Pop Operation-** Pop operation refers to removing the element from the stack again since you have only one end to do all top of the stack. So removing an element from the top of the stack is termed pop operation.

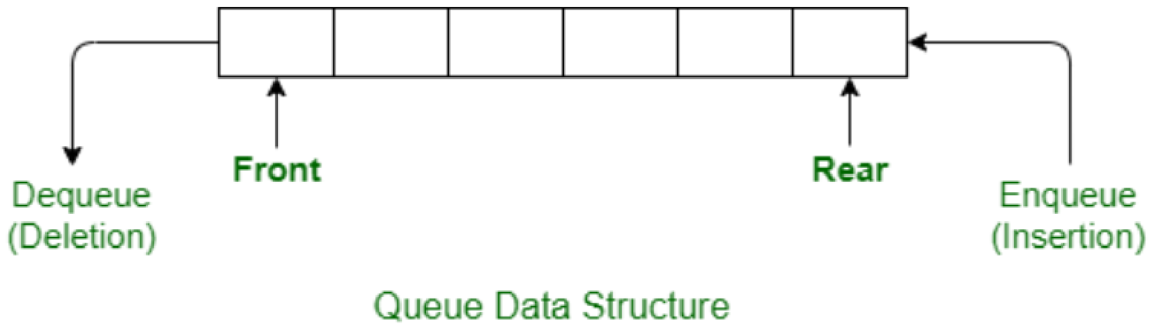
- **pop():** When we delete an element from the stack, the operation is known as a pop. If the stack is empty means that no element exists in the stack, this state is known as an underflow state.



## QUEUE:

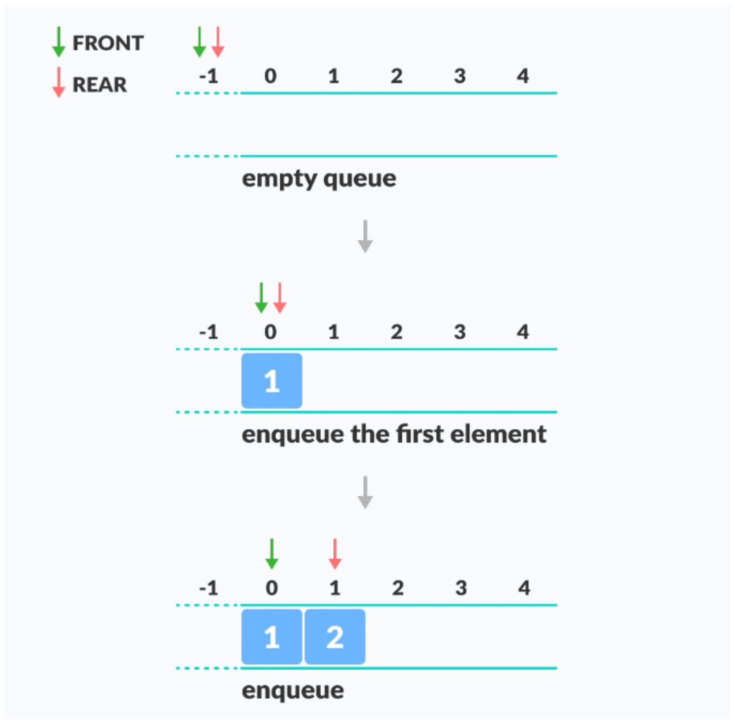
**Queue** is an ordered list in which all insertions at one end called REAR and deletions are made at another end called FRONT. Queues are sometimes referred to as *First In First Out (FIFO)* list.

For example, people waiting in line at the bank queue counter from a queue.
In computer, the jobs waiting in line to use processor for execution, this queue is known as Job Queue.



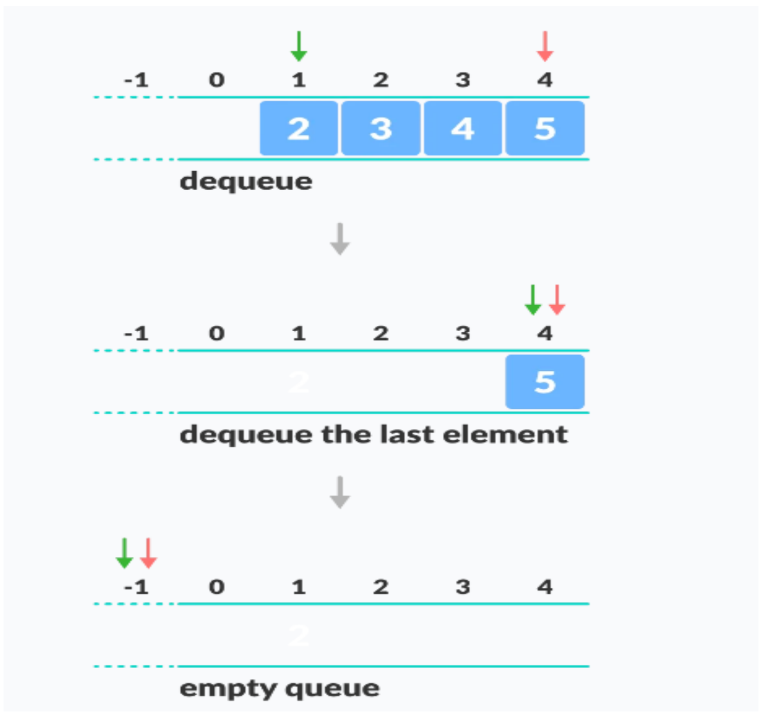
## Basic Operations on Queue

**Enqueue Operation-** It inserts item or element at the rear of the queue. An error



occurs if the queue is full.

**Dequeue Operation-** It removes or deletes an item or element from the front end of



the queue, and returns to the user. An error occurs if the queue is empty.

**CODE:**

```
package project;

import java.awt.Font;

import java.awt.event.ActionEvent;

import javax.swing.*;

import java.util.*;

import java.awt.event.ActionListener;


public class Linklist {

   static JFrame frame;

   static JTextField text;

   static JRadioButton r1,r2;

   static JButton b1,b2,b3,b4;

   static JLabel l1,l2;

   static LinkedList<Integer> list;


//Methods to Implement ADD,DELETE,PUSH,POP Operations in Queue and Stack
respectively

   public static void Add(int i){

      list.offer(i);

   }


   public static void Delete(){

      list.poll();

   }
```

```
public static void Push(int i){
    list.add(i);
}

public static void Pop(){
    list.removeLast();
}

//Methods to get elements of the list in 'String Format'
public static String elements(){
    String input=" ";
    for(int i=0;i<list.size();i++){
        input+=list.get(i)+" ";
    }
    return input;
}

public static void main(String[] args) {

   list =new LinkedList<>();//Linked List creation
   Random random=new Random();//To Generate Random numbers

   for(int i=0;i<10;i++){
       list.add(random.nextInt(200));
   }
```

```
    //To initialise JFrame

    frame=new JFrame("STACK AND QUEUE IMPLEMENTATION USING
GUI");frame.setSize(1000,1000);

    frame.setLayout(null);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


    //To initialise and Add JLabel2

    l1=new JLabel("LIST: ");

    l1.setBounds(40,25,150,30);

    l1.setFont(new Font("Times New Roman", Font.BOLD, 24));

    frame.add(l1);


    //To initialise and Add JLabel2

    l2=new JLabel("SELECT OPERATION: ");

    l2.setBounds(40,150,300,50);

    l2.setFont(new Font("Times New Roman", Font.BOLD, 24));

    frame.add(l2);


    //To initialise and Add JTextField

    text=new JTextField();

    text.setBounds(50,60,300,50);

    text.setText(elements());

    frame.add(text);


    //To initialise and Add JRadioButton 1

    r1=new JRadioButton("STACK");

    r1.setBounds(50,200,75,50);
```

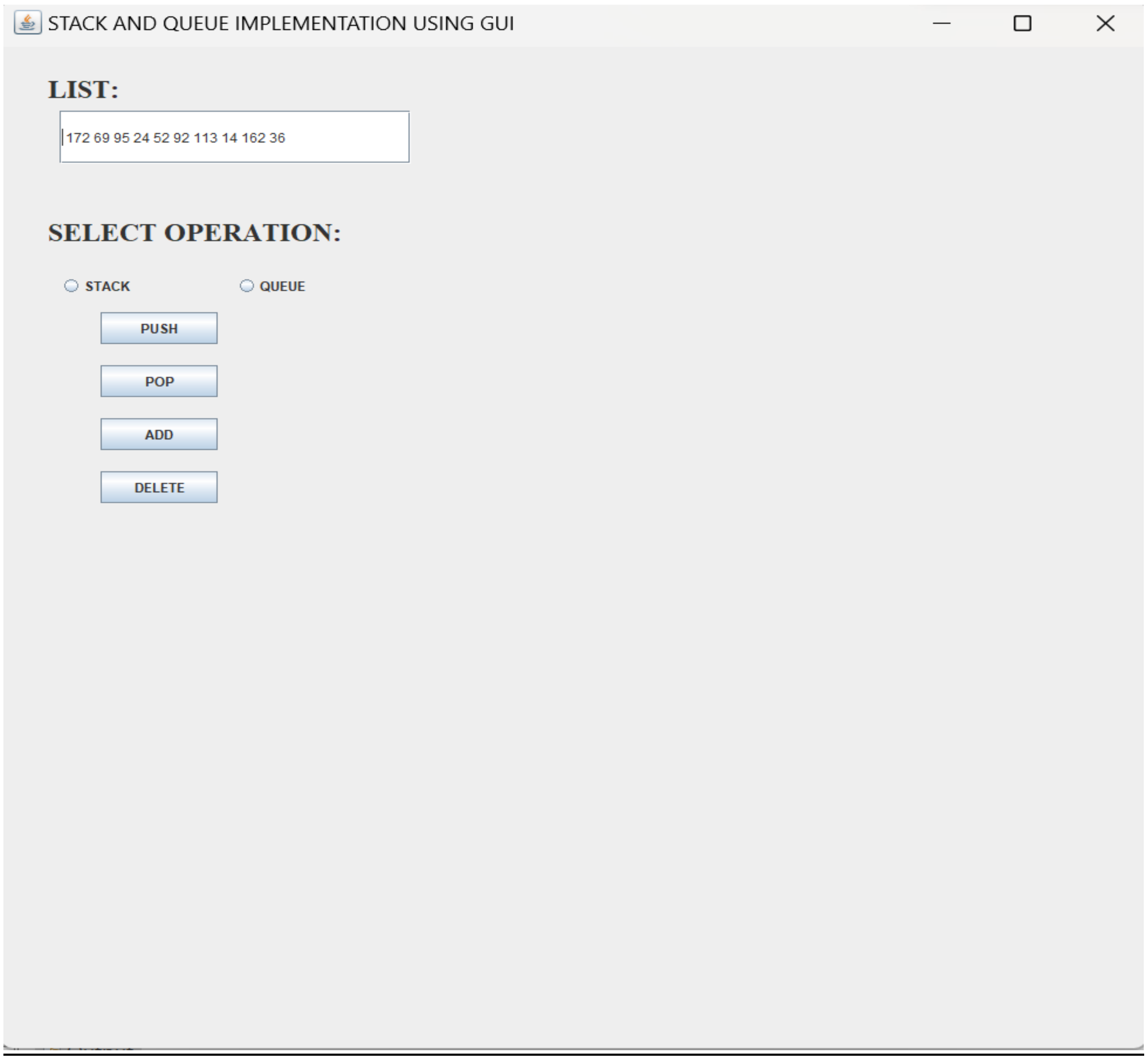
```
frame.add(r1);

//To initialise and Add JRadioButton 2
r2=new JRadioButton("QUEUE");
r2.setBounds(200,200,75,50);
frame.add(r2);

//To initialise and Add JButton 1,2,3,4
b1=new JButton("PUSH");
b1.setBounds(85,250,100,30);
b1.addActionListener(new A1());
frame.add(b1);

b2=new JButton("POP");
b2.setBounds(85,300,100,30);
b2.addActionListener(new A2());
frame.add(b2);

b3=new JButton("ADD");
b3.setBounds(85,350,100,30);
b3.addActionListener(new A3());
frame.add(b3);

b4=new JButton("DELETE");
b4.setBounds(85,400,100,30);
b4.addActionListener(new A4());
```

```
      frame.add(b4);


      frame.setVisible(true);

    }


    //Action Listener for PUSH button

    static class A1 implements ActionListener{


        @Override

        public void actionPerformed(ActionEvent e){

            String t=JOptionPane.showInputDialog(this,"Enter a Number");

            if(r1.isSelected()){

             try{

                int x=Integer.parseInt(t);

                list.addFirst(x);

                JOptionPane.showMessageDialog(frame,"ELEMENT PUSHED INTO THE
STACK");

                text.setText(elements());

            }

            catch(NumberFormatException e1){

                    JOptionPane.showMessageDialog(frame,"INVALID INPUT");

                    text.setText(elements());

                    }

            }

            else if(r2.isSelected()){

                    JOptionPane.showMessageDialog(frame,"PUSH OPERATION CANNOT
BE DONE ON QUEUE");
```

```
                text.setText(elements());
            }
            else{
                JOptionPane.showMessageDialog(frame,"INVALID SELECTION");
            }
        }
    }

    //Action Listener for POP button
    static class A2 implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e){
            if(r1.isSelected()){
            if(list.isEmpty()){
                JOptionPane.showMessageDialog(frame,"CANNOT POP AN ELEMENT
FROM AN EMPTY STACK");}



            else{
                list.removeFirst();
                text.setText(elements());
                JOptionPane.showMessageDialog(frame,"ELEMENT POPPED FROM
STACK");
            }
            }
            else if(r2.isSelected()){
                JOptionPane.showMessageDialog(frame,"POP OPERATION CANNOT BE
DONE ON QUEUE");
```

```
                text.setText(elements());
            }
            else{
                JOptionPane.showMessageDialog(frame,"INVALID SELECTION");
            }
        }
    }

    //Action Listener for ADDbutton
    static class A3 implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e){
            String t=JOptionPane.showInputDialog(this,"Enter a Number");
            if(r2.isSelected()){
            try{
                int x=Integer.parseInt(t);
                list.addLast(x);
                text.setText(elements());
                JOptionPane.showMessageDialog(frame,"ELEMENT ADDED TO QUEUE");
            }
            catch(NumberFormatException e1){
                JOptionPane.showMessageDialog(frame,"INVALID INPUT");
                text.setText(elements());
                }
            }
            else if(r1.isSelected()){
                JOptionPane.showMessageDialog(frame,"ADD OPERATION CANNOT BE
DONE ON STACK");

                text.setText(elements());

            }

            else{

                JOptionPane.showMessageDialog(frame,"INVALID SELECTION");

            }

        }

    }


    //Action Listener for DELETE button

    static class A4 implements ActionListener{

        @Override

        public void actionPerformed(ActionEvent e){

            if(r2.isSelected()){

            if(list.isEmpty()){

              JOptionPane.showMessageDialog(frame,"CANNOT DELETE AN ELEMENT
FROM AN EMPTY QUEUE");

            }



            else{

                list.removeFirst();

                text.setText(elements());

                JOptionPane.showMessageDialog(frame,"ELEMENT DELETED FROM
QUEUE");

            }

            }
            else if(r1.isSelected()){
                    JOptionPane.showMessageDialog(frame,"DELETE OPERATION CANNOT
BE DONE ON STACK");
                    text.setText(elements());
            }
            else{
                    JOptionPane.showMessageDialog(frame,"INVALID SELECTION");
            }
        }
    }
}
```

## OUTPUT

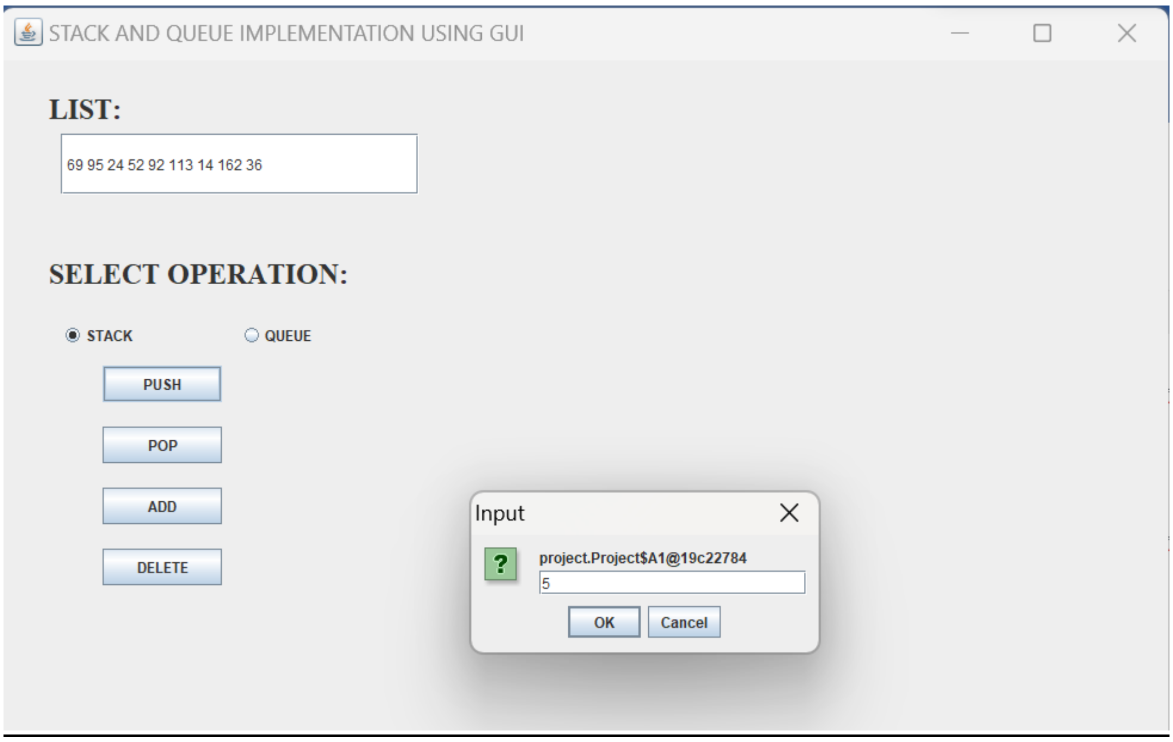

STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

```
172 69 95 24 52 92 113 14 162 36
```

## STACK-:

- **PUSH**



STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36

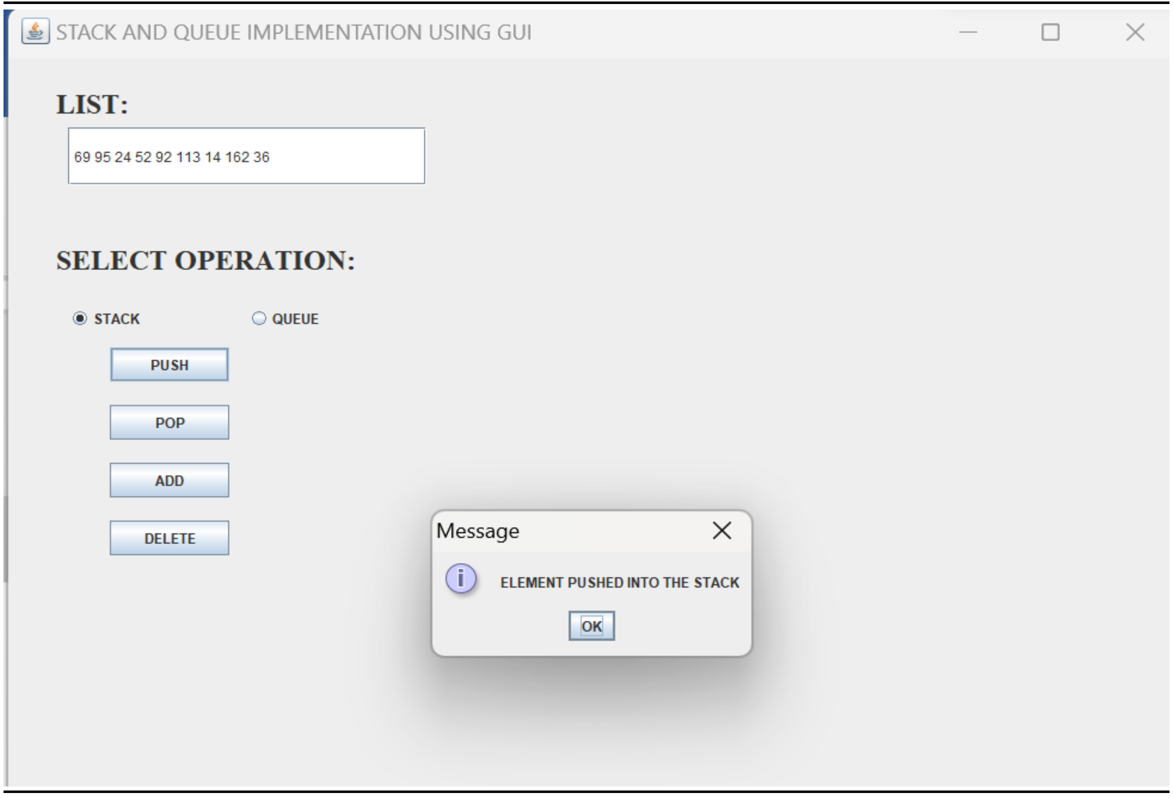
SELECT OPERATION:

[✓ STACK] QUEUE

Input

project.Project$A1@19c22784

5



STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

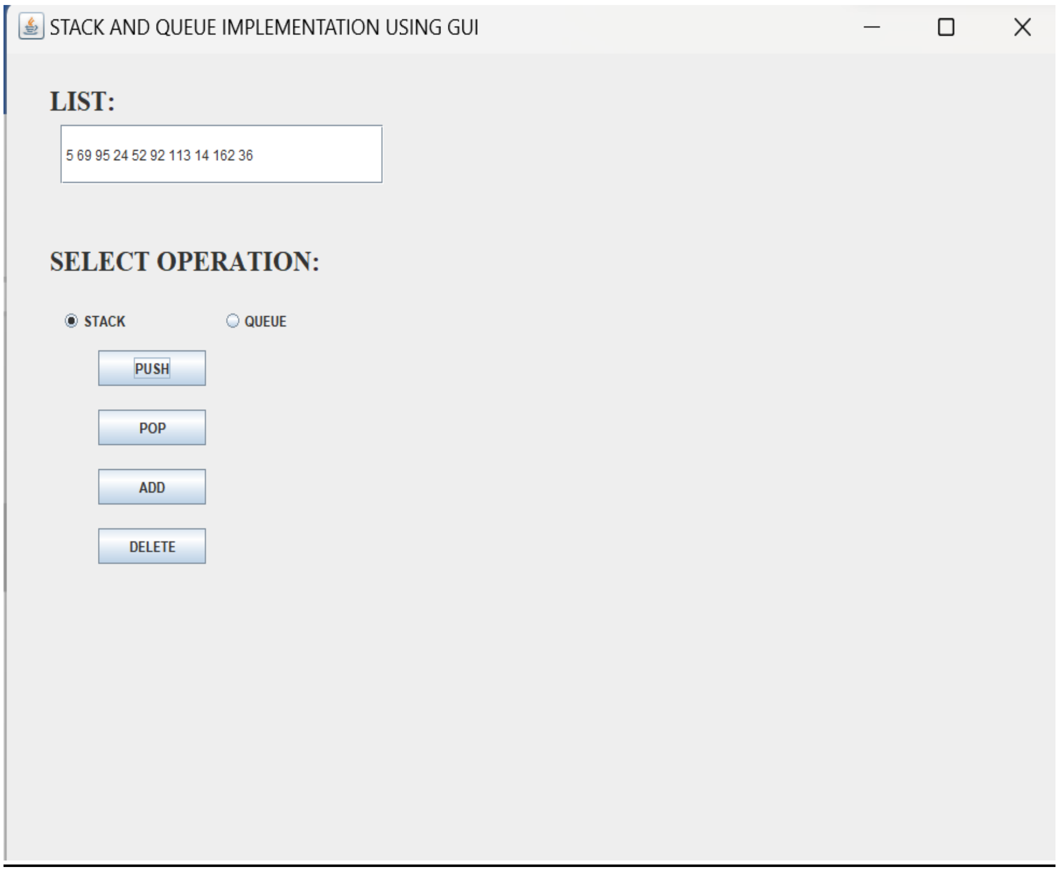
69 95 24 52 92 113 14 162 36

SELECT OPERATION:

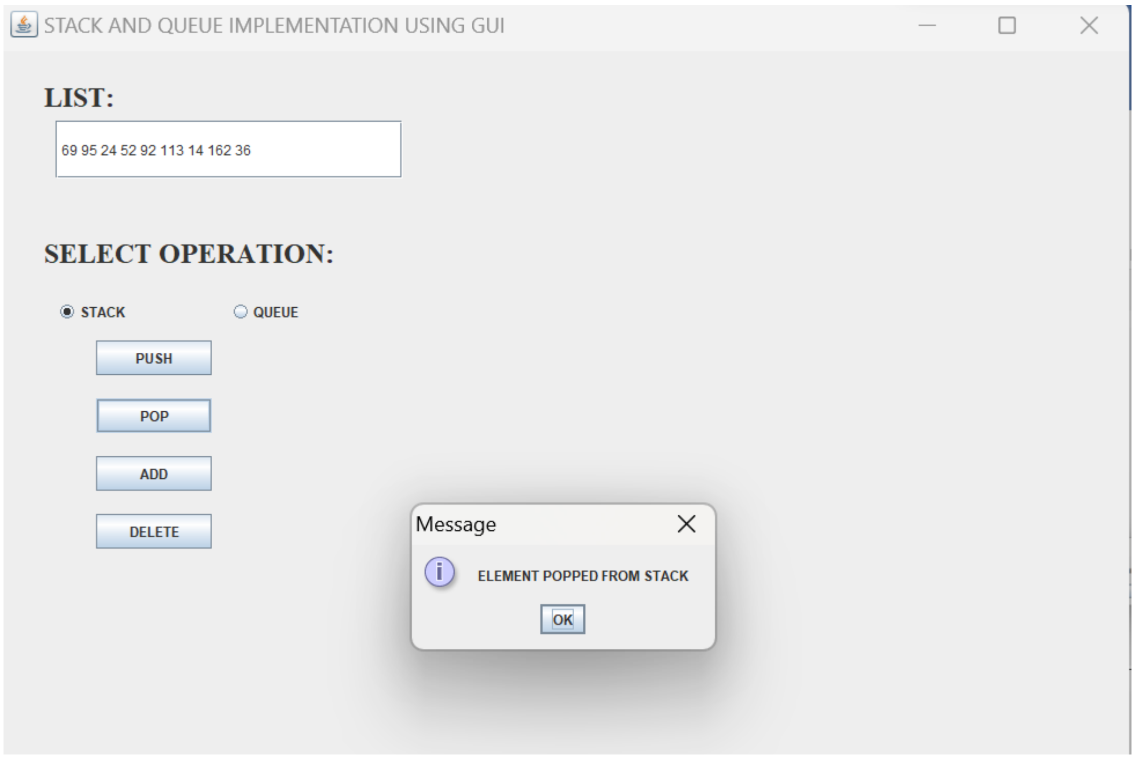
[✓ STACK] QUEUE

Message

ELEMENT PUSHED INTO THE STACK

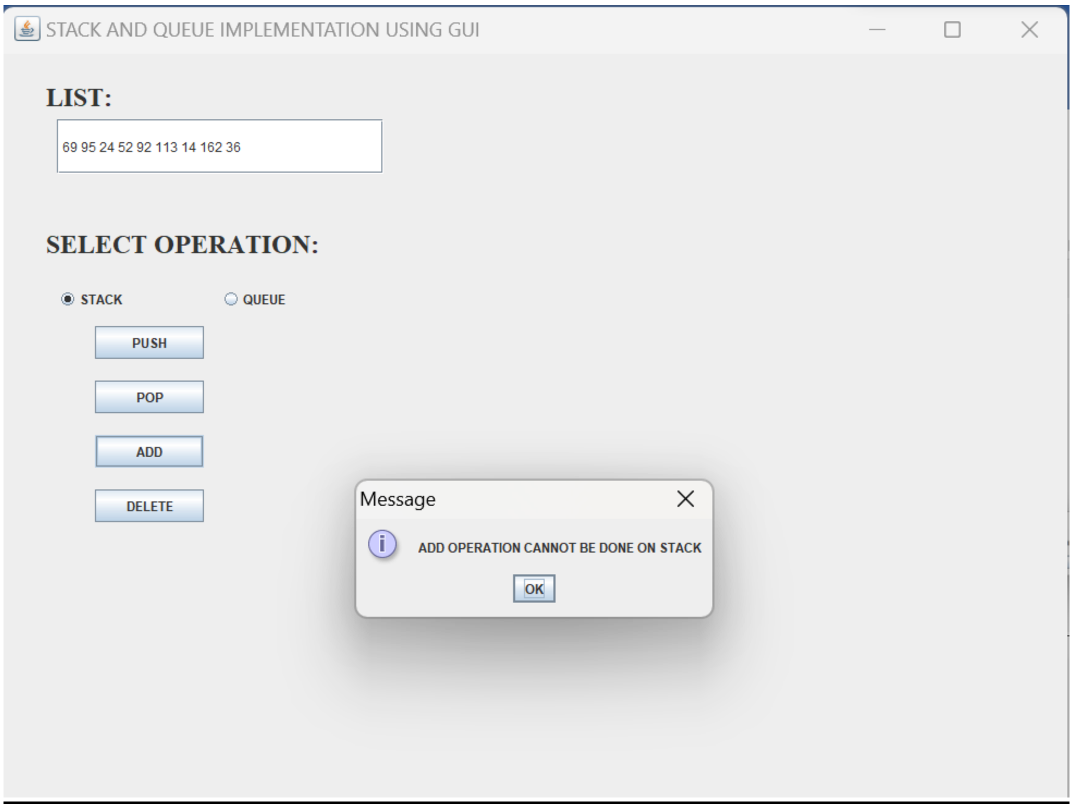


## ➢ POP



ELEMENT POPPED FROM STACK

## ➢ ADD:



STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36

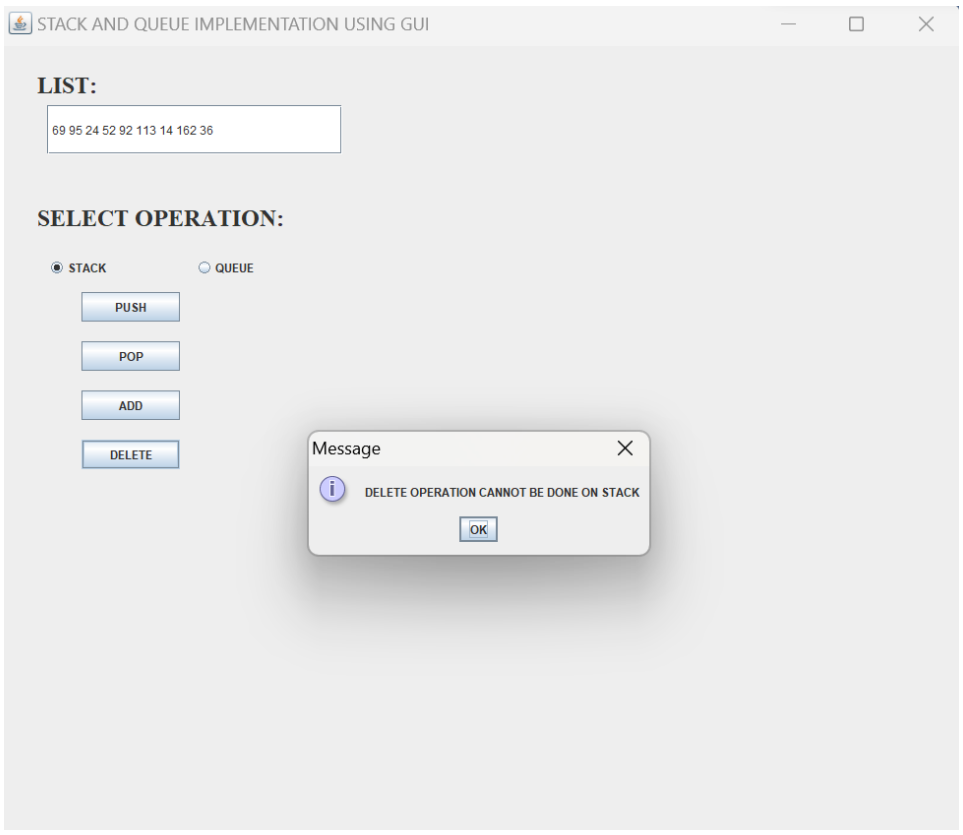
SELECT OPERATION:

◉ STACK ○ QUEUE

Message

ADD OPERATION CANNOT BE DONE ON STACK

## ➢ DELETE:



STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36

SELECT OPERATION:

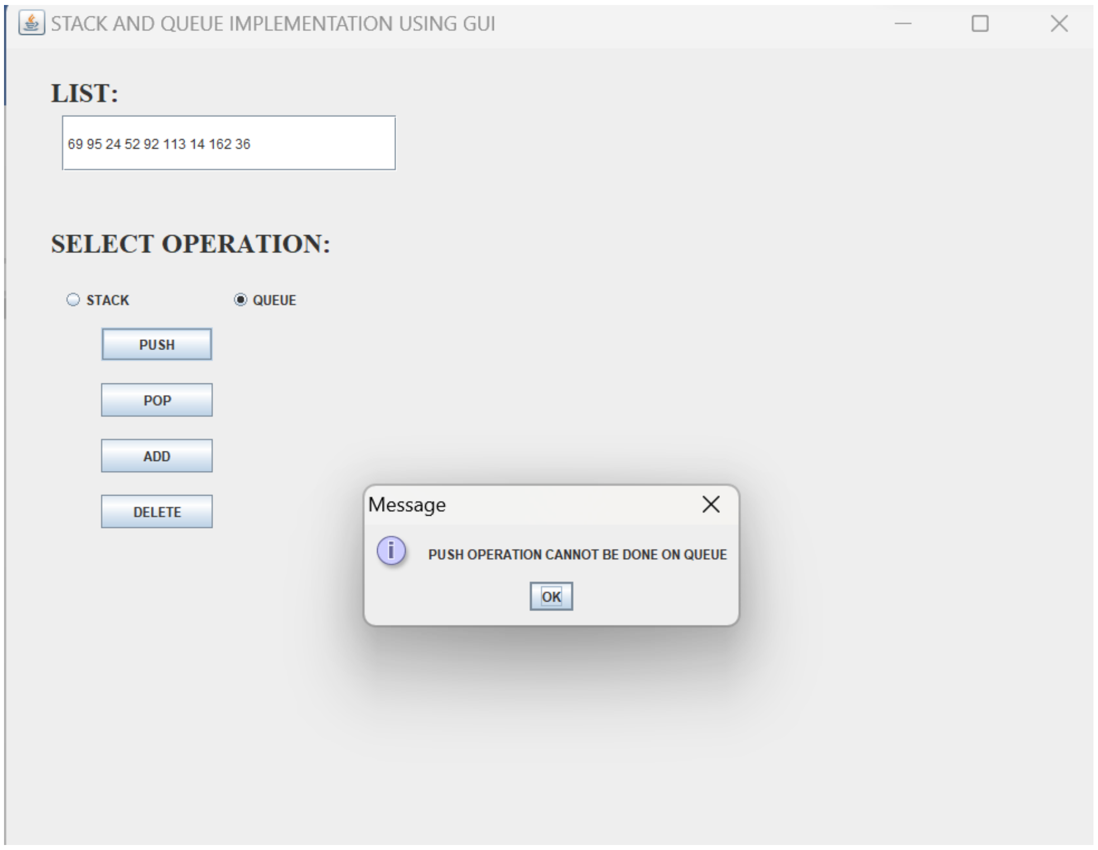
◉ STACK ○ QUEUE

Message

DELETE OPERATION CANNOT BE DONE ON STACK

## QUEUE-:

- **PUSH**



STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36

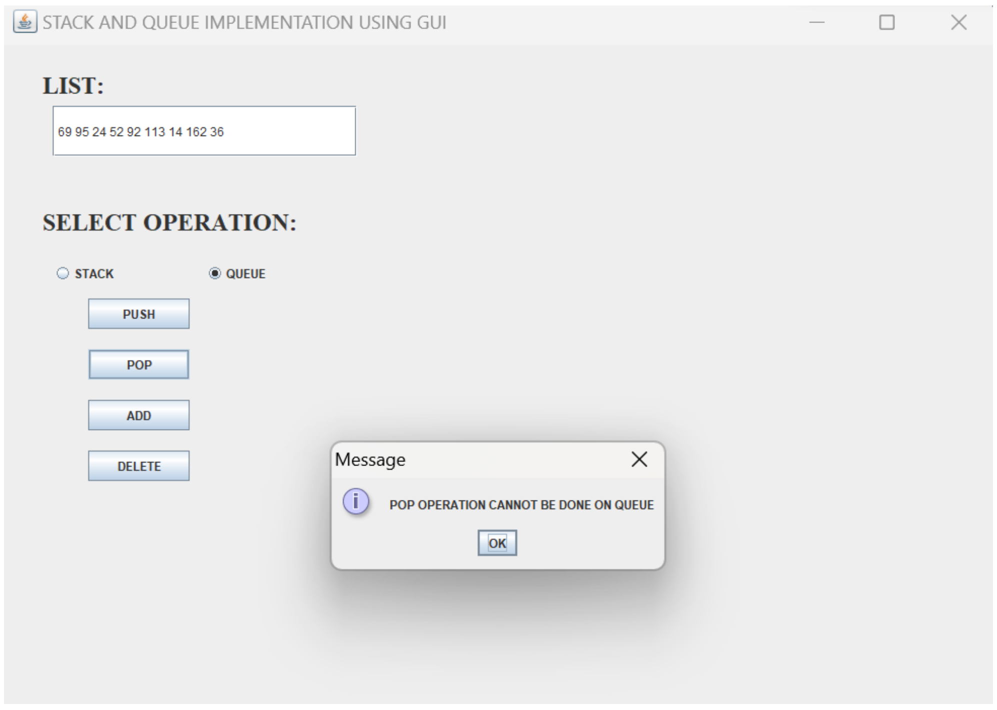
SELECT OPERATION:

STACK [✓ QUEUE]

Message

PUSH OPERATION CANNOT BE DONE ON QUEUE

- **POP**



STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36

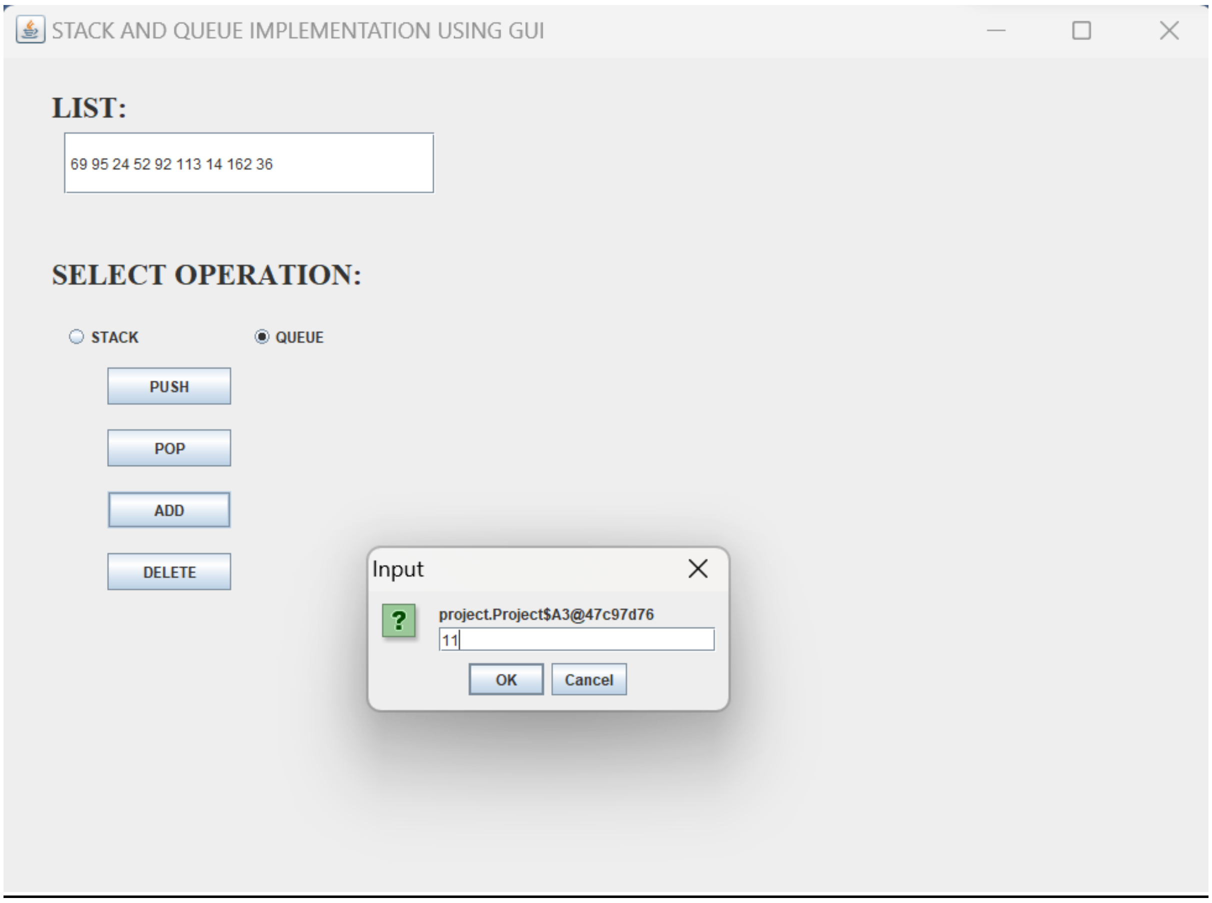
SELECT OPERATION:

STACK [✓ QUEUE]

Message

POP OPERATION CANNOT BE DONE ON QUEUE

## ➢ ADD


STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36

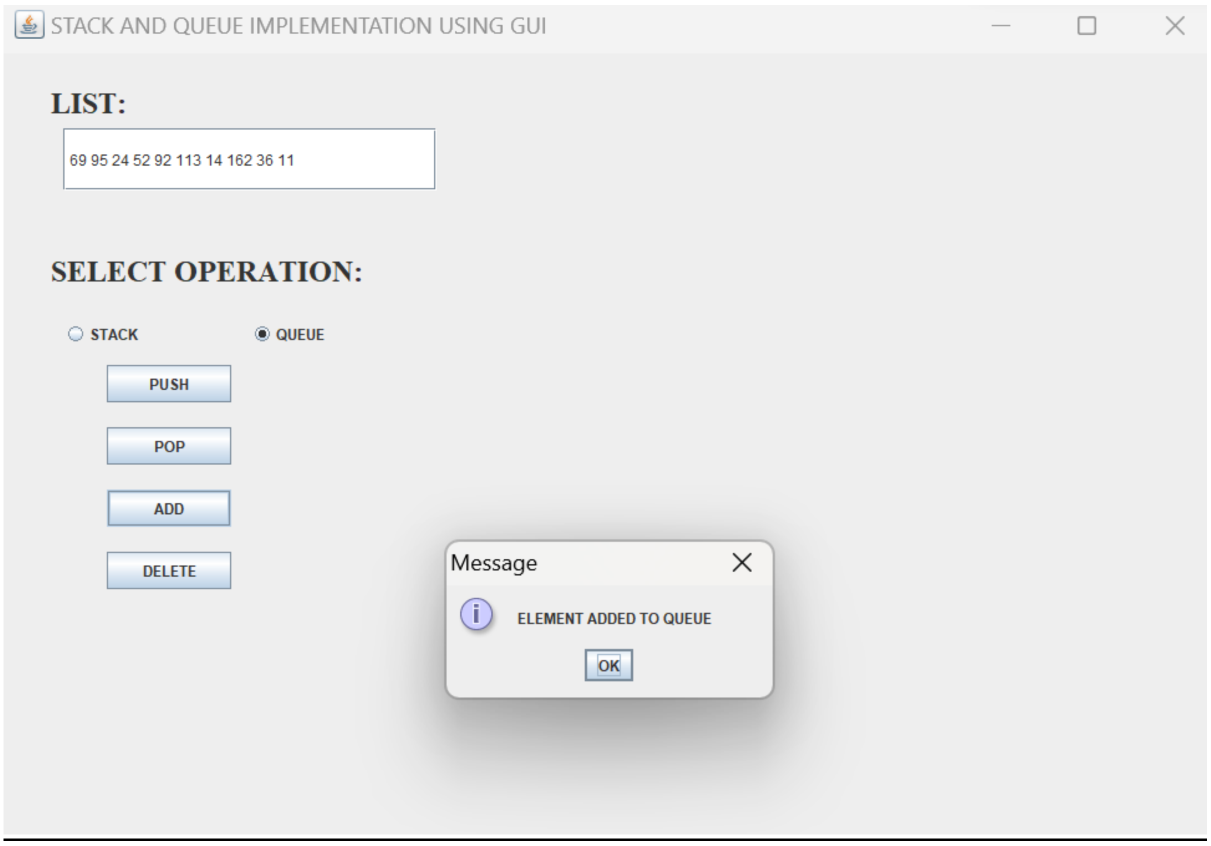
SELECT OPERATION:

STACK [✓ QUEUE]

Input

project.Project$A3@47c97d76

11


STACK AND QUEUE IMPLEMENTATION USING GUI

LIST:

69 95 24 52 92 113 14 162 36 11

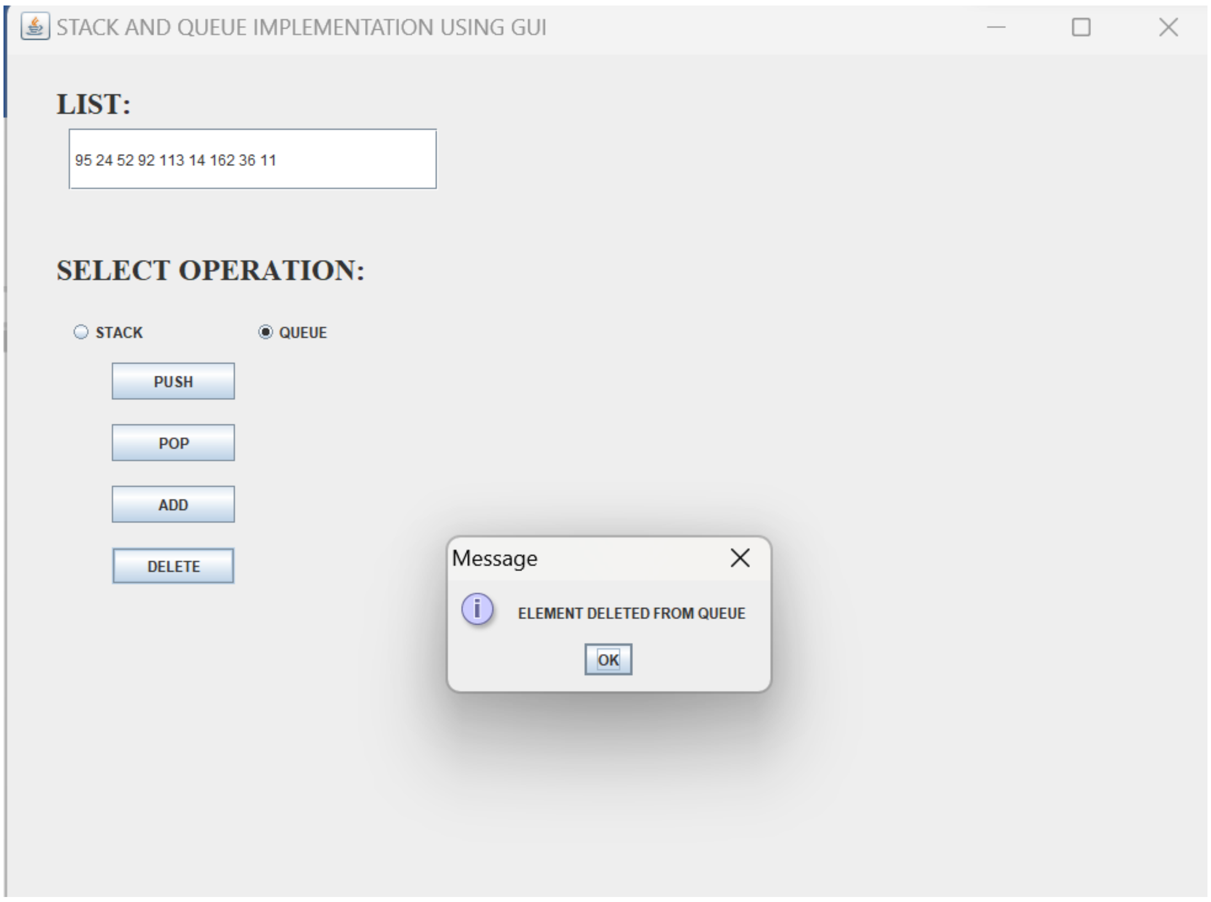
SELECT OPERATION:

STACK [✓ QUEUE]

Message

ELEMENT ADDED TO QUEUE

- **DELETE**



```
95 24 52 92 113 14 162 36 11
```

ELEMENT DELETED FROM QUEUE