DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

Programming Paradigm

CS433P

B. Tech – Computer Science and Engineering
(Artificial Intelligence and Machine Learning)

School of Engineering and Technology,

CHRIST (Deemed to be University),

Kumbalagodu, Bengaluru-560 074

April 2023

CHRIST
(DEEMED TO BE UNIVERSITY)
BANGALORE · INDIA

Certificate

This is to certify that Ashvath Suresh Babu Piriya has successfully completed the record / ~~Mini Project~~ work for (CS433P - Programming Paradigm) in partial fulfillment for the award of Bachelor of Technology in Computer Science and Engineering (Artificial Intelligence & Machine Learning) during the year 2022-2023.

HEAD OF THE DEPARTMENT **FACULTY- IN CHARGE**

EXAMINER 1:

EXAMINER 2:

Name : Ashvath S.P

Register No. : 2162014

Examination Center : SoET, CHRIST (Deemed to be University)

Date of Examination :

INDEX

Exp. No	Date	Experiment Name	Page No	Marks	Signature

CS433P Programming Paradigm Lab

DATE: 13-01-2023 **EXPERIMENT NO** 1 **REGISTER NO**: 2162014

DEMONSTRATE FUNDAMENTALS OF OOP

AIM: Write a java program to make a simple calculator.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.Scanner;

public class Calculator {

   public double Operations(char o, double r, double n1, double n2) {
      // conditions to perform arithmetic operations
      switch (o) {
         case '+' -> {
            System.out.println("\nAddition: " + n1 + " + " + n2);
            r = n1 + n2;
            break;
         }
         case '-' -> {
            System.out.println("\nSubtraction: " + n1 + " - " + n2);
            r = n1 - n2;
            break;
         }
         case '*' -> {
            System.out.println("\nMultiplication: " + n1 + " * " + n2);
            r = n1 * n2;
            break;
         }
         case '/' -> {
            System.out.println("\nDivision: " + n1 + " / " + n2);
            r = n1 / n2;
            break;
         }
         default -> {
            System.out.println("\nInvalid input!\n");
            break;
         }
      }
      return r;
   }

            public static void main(String[] args) {
```

```
        Scanner scn = new Scanner(System.in);
        Calculator obj = new Calculator();
        System.out.print("\n-+-+-+-+Calculator-+-+-+-+\n");
        // Menu
        System.out.print("\nOperations:\n1. Addition\n2. Subtraction\n3. Multiplication\n4.
Division\n");
        // get numbers from user
        System.out.print("\nEnter first number: ");
        double num1 = scn.nextDouble();
        System.out.print("\nEnter second number: ");
        double num2 = scn.nextDouble();
        // get operation from user
        System.out.print("\nChoose operations(+,-,*,/): ");
        char op = scn.next().charAt(0);
        double result = 0, ans = obj.Operations(op, result, num1, num2);
        // display output to user
        System.out.printf("\nResult: %.2f \n", ans);
        System.out.print("\n");
    }
}
```

OUTPUTS:

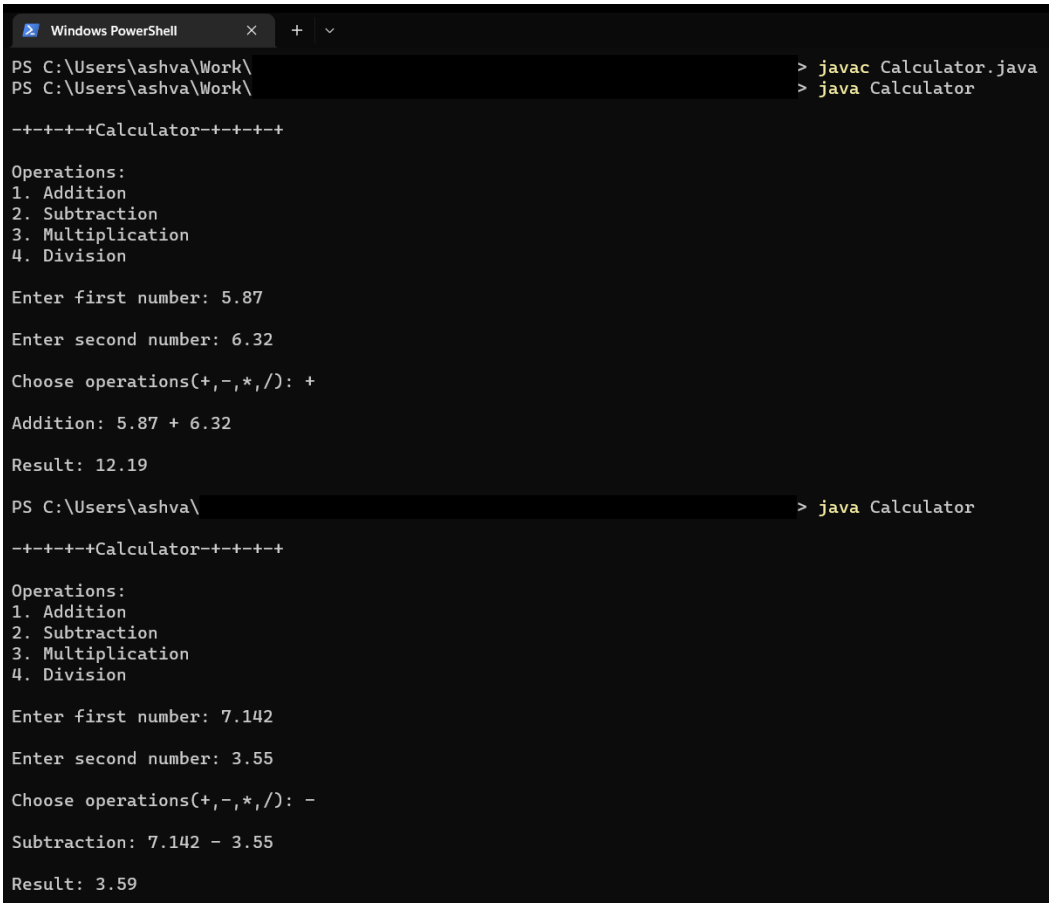


Fig. 1 – Output for addition and subtraction

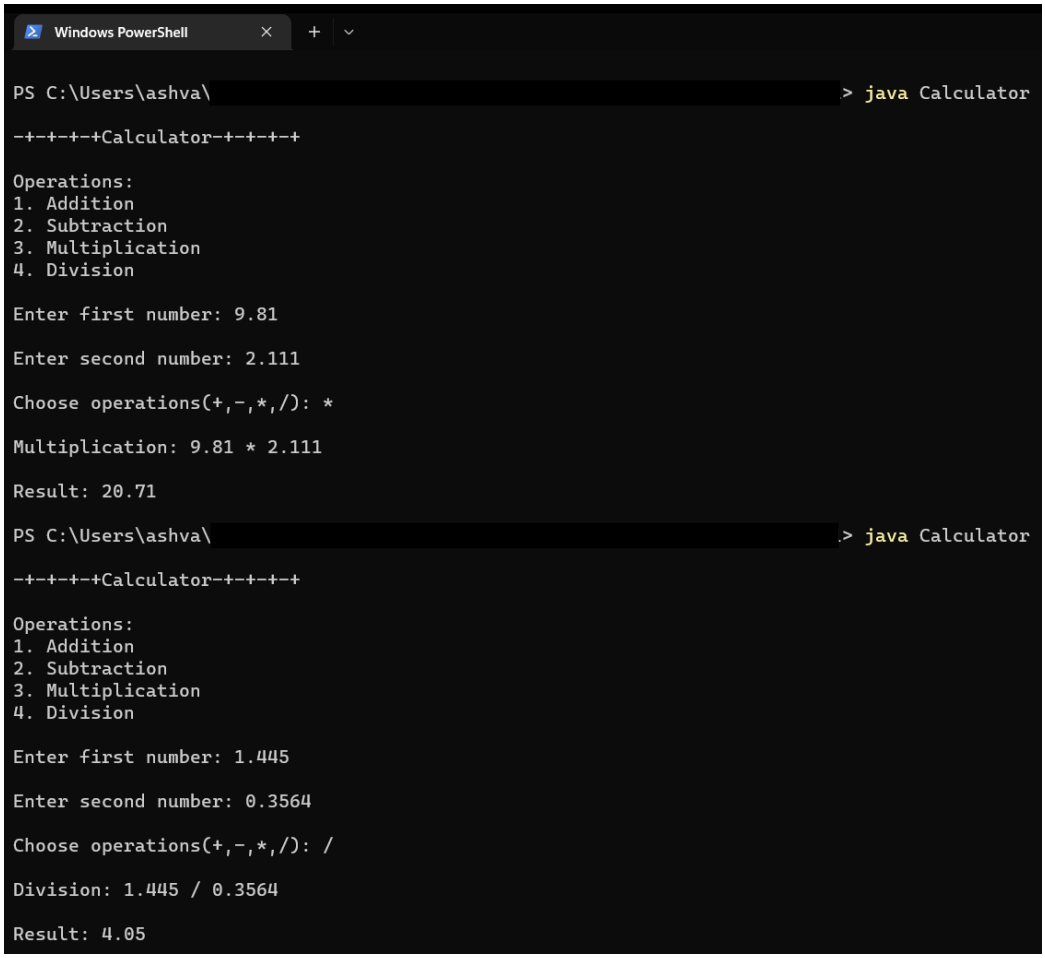


Fig. 2 – Output for multiplication and division

RESULTS:

The simple calculator was created successfully in java using basic arithmetic operations and switch cases functionality.

DATE: 20-01-2023 **EXPERIMENT NO** 2 **REGISTER NO**: 2162014

IMPLEMENTATION OF CLASSES AND OBJECTS

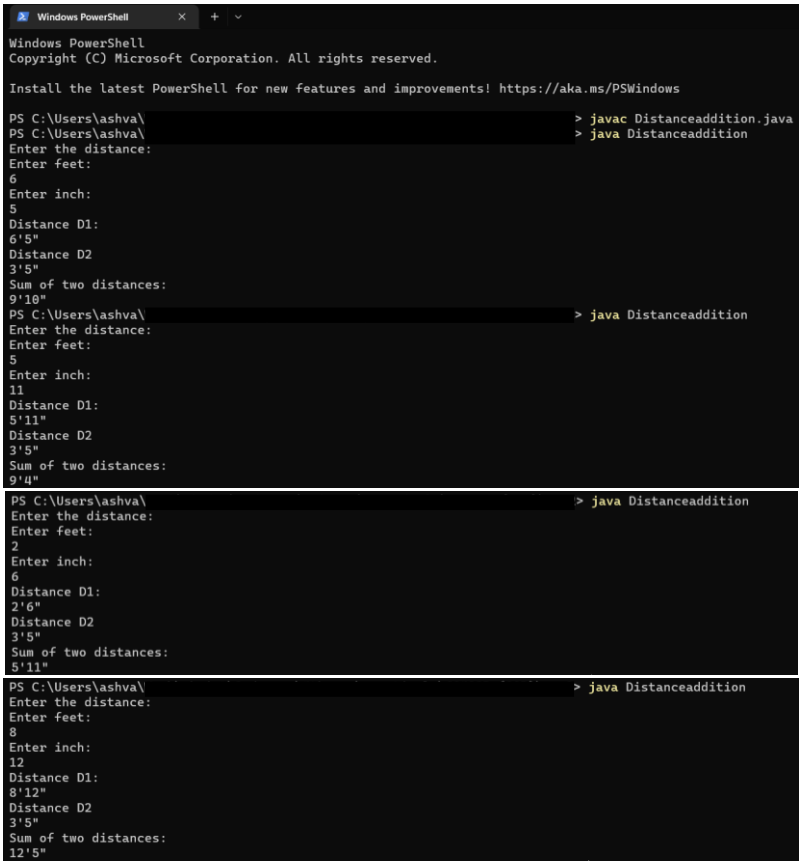
AIM: Write a java program to create a class Distance with data members' feet and inches.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.Scanner;

class Distance {

   int feet, inch;

   Distance() {
      feet = 0;
      inch = 0;
   }

   Distance(int ft, int in) {
      feet = ft;
      inch = in;
   }

   void readDistance() {
      Scanner in = new Scanner(System.in);
      System.out.println("Enter the distance: ");
      System.out.println("Enter feet: ");
      feet = in.nextInt();
      System.out.println("Enter inch: ");
      inch = in.nextInt();
   }

   void printDistance() {
      System.out.println(feet + "\'" + inch + "\"");
   }

   void addDistance(Distance d) {
      feet = this.feet + d.feet;
      inch = this.inch + d.inch;
      if (inch > 11) {
         ++feet;
         inch -= 12;
      }
```

```
    }

}

public class Distanceaddition {

    public static void main(String[] args) {
        Distance d1 = new Distance();
        Distance d2 = new Distance(3, 5);
        d1.readDistance();
        System.out.println("Distance D1:");
        d1.printDistance();
        System.out.println("Distance D2");
        d2.printDistance();
        System.out.println("Sum of two distances: ");
        d1.addDistance(d2);
        d1.printDistance();
    }
}
```

OUTPUTS:



RESULTS:

The java program was created successfully to demonstrate classes and objects.

DATE: 27-01-2023 **EXPERIMENT NO** 3 **REGISTER NO**: 2162014

IMPLEMENTATION OF BUBBLE SORT

AIM: Write a java program to implement a single-dimensional array and sort using bubble sort.

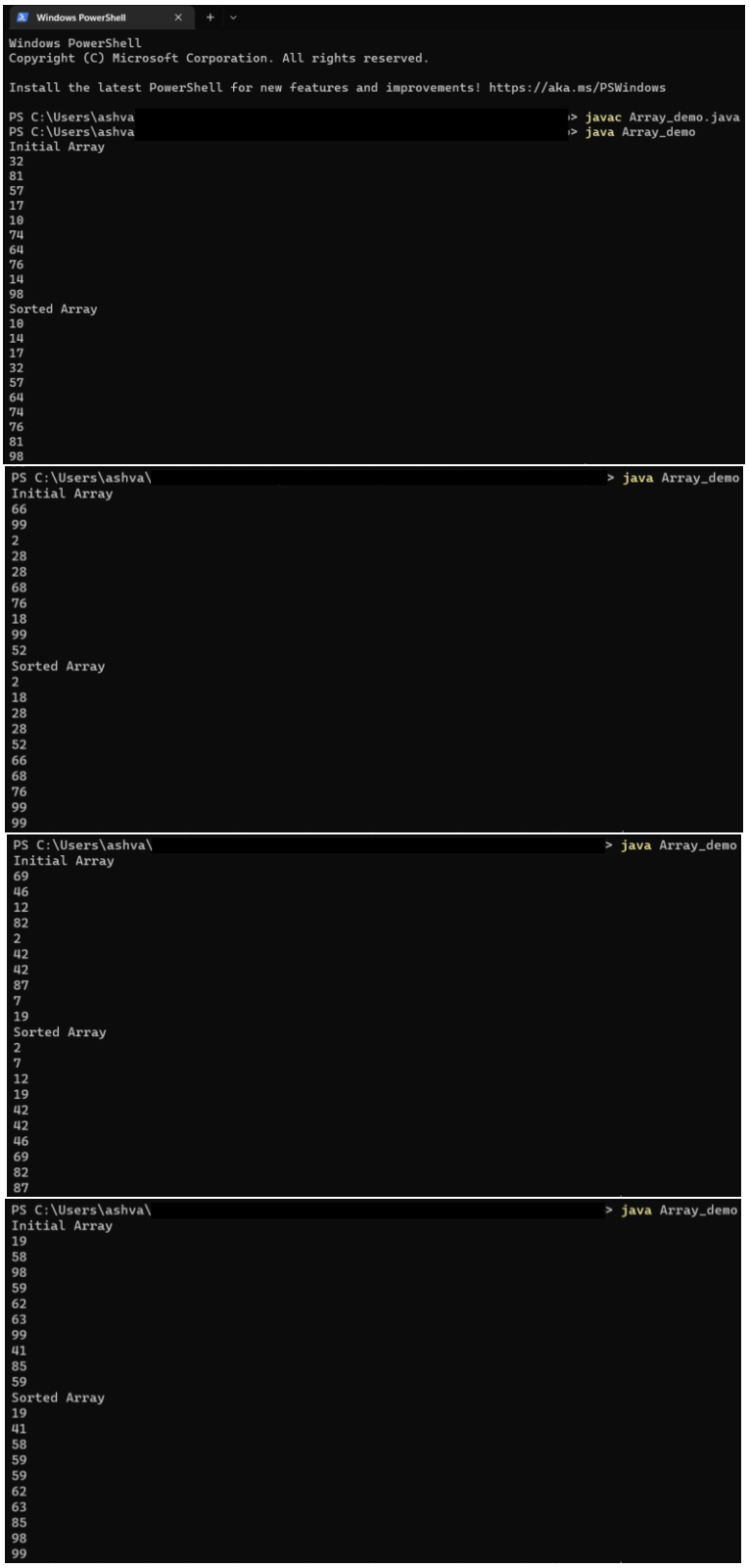
PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.Random;

public class Array_demo {

   public static void main(String[] args) {
      int arr[] = new int[10];
      Random rn = new Random();
      for (int i = 0; i < 10; i++) {
         arr[i] = rn.nextInt(100);
      }
      System.out.println("Initial Array");
      for (int i : arr) //for each loop
      {
         System.out.println(i);
      }
      //sort the array
      int n = arr.length;
      int temp;
      for (int i = 0; i < n - 1; i++) {
         for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
               temp = arr[j];
               arr[j] = arr[j + 1];
               arr[j + 1] = temp;
            }
         }
      }
      System.out.println("Sorted Array");
      for (int i : arr) {
         System.out.println(i);
      }
   }
}
```

CS433P Programming Paradigm Lab

DATE: 27-01-2023 **EXPERIMENT NO** 3 **REGISTER NO**: 2162014

OUTPUTS:



RESULTS:

The java program was successfully created to implement a single-dimensional array and sort it using bubble sort.

DATE: 10-02-2023 **EXPERIMENT NO** 4 **REGISTER NO**: 2162014

IMPLEMENTATION OF INHERITENCE

AIM:

Create an interface called “Bank” and declare a method to get customer details: customer name, customer id , number of years, and customer balance. Three classes: Axes, ICIC, and SBI, should be derived from Bank. The customer details and interest rate should be overridden in the third class. The interest rate for Axes is 5%, ICIC is 7%, and SBI is 8%.
Display the Menu:
1. AXES
2. ICIC
3. SBI
According to the selection, the total amount after a number of years should be calculated for the given balance_amount, and all the details should be displayed as follows:

CUSTOMER NAME	ID	NO OF YEARS	BALANCE	BANK	RATE OF INTEREST	TOTAL AMOUNT
AAAA	111	5	5000	AXIS	5%	5500

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.Scanner;

interface Bank {

   void get_details();
}

class Axes implements Bank {

   String Cname;
   int Cid, nay;
   double balance, tbalance, roi;
   String ROI;

   @Override
   public void get_details() {
      Scanner sc = new Scanner(System.in);
      System.out.println("ENTER CUSTOMER NAME: ");
      Cname = sc.nextLine();
      System.out.println("ENTER CID: ");
      Cid = sc.nextInt();
      System.out.println("ENTER NUMBER OF YEARS: ");
      nay = sc.nextInt();
      System.out.println("ENTER CURRENT BALANCE: ");
      balance = sc.nextDouble();
      roi = 5;
```

```
        ROI = "5%";
        tbalance = balance + ((roi * balance * nay) / 100);
    }

    void show_details() {
        System.out.println("CUSTOMER NAME \t\t ID \t\t NO. OF YEARS \t\t BALANCE \t\t
BANK \t\t RATE OF INTEREST \t\t TOTAL AMOUNT");
        System.out.println(Cname + "\t\t\t" + Cid + "\t\t\t" + nay + "\t\t\t" + balance + "\t\t\t" +
"AXES" + "\t\t\t" + ROI + "\t\t\t" + tbalance);
    }
}

class SBI implements Bank {

    String Cname;
    int Cid, nay;
    double balance, tbalance, roi;
    String ROI;

    @Override
    public void get_details() {
        Scanner sc = new Scanner(System.in);
        System.out.println("ENTER CUSTOMER NAME: ");
        Cname = sc.nextLine();
        System.out.println("ENTER CID: ");
        Cid = sc.nextInt();
        System.out.println("ENTER NUMBER OF YEARS: ");
        nay = sc.nextInt();
        System.out.println("ENTER CURRENT BALANCE: ");
        balance = sc.nextDouble();
        roi = 8;
        ROI = "8%";
        tbalance = balance + ((roi * balance * nay) / 100);
    }

    void show_details() {
        System.out.println("CUSTOMER NAME \t\t ID \t\t NO. OF YEARS \t\t BALANCE \t\t
BANK \t\t RATE OF INTEREST \t\t TOTAL AMOUNT");
        System.out.println(Cname + "\t\t\t" + Cid + "\t\t\t" + nay + "\t\t\t" + balance + "\t\t\t" +
"SBI" + "\t\t\t" + ROI + "\t\t\t" + tbalance);
    }
}

class ICIC implements Bank {
```

```
    String Cname;
    int Cid, nay;
    double balance, tbalance, roi;
    String ROI;

    @Override
    public void get_details() {
        Scanner sc = new Scanner(System.in);
        System.out.println("ENTER CUSTOMER NAME: ");
        Cname = sc.nextLine();
        System.out.println("ENTER CID: ");
        Cid = sc.nextInt();
        System.out.println("ENTER NUMBER OF YEARS: ");
        nay = sc.nextInt();
        System.out.println("ENTER CURRENT BALANCE: ");
        balance = sc.nextDouble();
        roi = 7;
        ROI = "7%";
        tbalance = balance + ((roi * balance * nay) / 100);
    }

    void show_details() {
        System.out.println("CUSTOMER NAME \t\t ID \t\t NO. OF YEARS \t\t BALANCE \t\t
BANK \t\t RATE OF INTEREST \t\t TOTAL AMOUNT");
        System.out.println(Cname + "\t\t\t" + Cid + "\t\t\t" + nay + "\t\t\t" + balance + "\t\t\t" +
"ICIC" + "\t\t\t" + ROI + "\t\t\t" + tbalance);
    }
}

public class interBank {

    public static void main(String[] args) {
        int ch;
        Scanner sc = new Scanner(System.in);
        System.out.println("1. AXES");
        System.out.println("2. ICIC");
        System.out.println("3. SBI");
        ch = sc.nextInt();
        switch (ch) {
            case 1 -> {
                Axes a = new Axes();
                a.get_details();
                a.show_details();
            }
            case 2 -> {
```

```
            ICIC i = new ICIC();
            i.get_details();
            i.show_details();
        }
        case 3 -> {
            SBI s = new SBI();
            s.get_details();
            s.show_details();
        }
        default -> System.out.println("Invalid choice!");
      }
   }
}
```

OUTPUTS:

```
C:\Users\2162014\OOP\exp3\src\main\java>java interBank
1. AXES
2. ICIC
3. SBI
1
ENTER CUSTOMER NAME:
ASHVATH
ENTER CID:
454
ENTER NUMBER OF YEARS:
5
ENTER CURRENT BALANCE:
600
CUSTOMER NAME            ID             NO. OF YEARS           BALANCE               BANK           RATE OF INTEREST              TOTAL AMOUNT
ASHVATH                 454                    5                   600.0               AXES                     5%                     750.0

C:\Users\2162014\OOP\exp3\src\main\java>java interBank
1. AXES
2. ICIC
3. SBI
2
ENTER CUSTOMER NAME:
BARATH
ENTER CID:
655
ENTER NUMBER OF YEARS:
4
ENTER CURRENT BALANCE:
5000
CUSTOMER NAME            ID             NO. OF YEARS           BALANCE               BANK           RATE OF INTEREST              TOTAL AMOUNT
BARATH                  655                    4                  5000.0               ICIC                     7%                    6400.0
```

```
C:\Users\2162014\OOP\exp3\src\main\java>java interBank
1. AXES
2. ICIC
3. SBI
3
ENTER CUSTOMER NAME:
ASHVATH
ENTER CID:
665
ENTER NUMBER OF YEARS:
9
ENTER CURRENT BALANCE:
2100
CUSTOMER NAME            ID             NO. OF YEARS           BALANCE               BANK           RATE OF INTEREST              TOTAL AMOUNT
ASHVATH                 665                    9                  2100.0               SBI                      8%                    3612.0
```

RESULTS:

The java program was successfully created to implement the concept of Inheritance.

DATE: 10-02-2023 **EXPERIMENT NO** 5 **REGISTER NO**: 2162014

IMPLEMENTATION OF NESTED CLASS

AIM: Implementation of Inner Class

PROGRAM:

```
/**
 *
 * @author 2162014
 */
public class OuterClass {

    int a = 7, b = 7, c = 0;

    class inner {

        public void output() {
            System.out.println("Inner Class");
            c = a + b;
        }
    }

    public void input() {
        inner i = new inner();
        i.output();
        System.out.println("Outer Class");
        System.out.println("a+b=" + c);
    }

    public static void main(String[] args) {
        OuterClass o = new OuterClass();
        o.input();
    }
}
```

OUTPUTS:

```
C:\Users\2162014\                          >java OuterClass
Inner Class
Outer Class
a+b=14
```

RESULTS:

The java program is created successfully to implement the concept of Nested Class.

DATE: 24-02-2023 **EXPERIMENT NO** 6 **REGISTER NO**: 2162014

IMPLEMENTATION OF EVENT HANDLING

AIM: Write a java program to demonstrate the use of textfields, radiobuttons, and button.

PROGRAM:

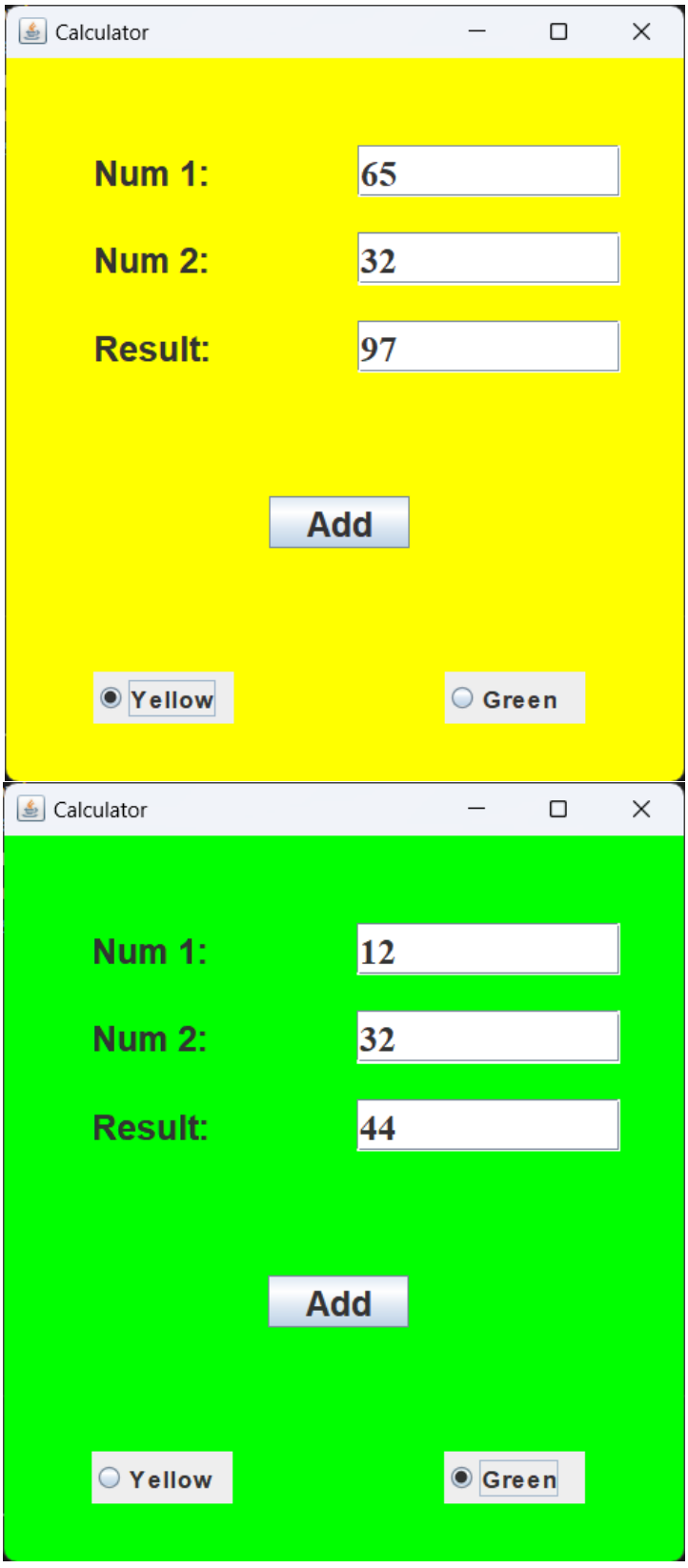
```
/**
 *
 * @author 2162014
 */
import java.awt.Color;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JTextField;

class Calculator implements ActionListener {
    // Create a JFrame to hold the calculator components
    JFrame frame = new JFrame("Calculator");

    // Create labels for input and output fields
    JLabel input1Label = new JLabel("Num 1:");
    JLabel input2Label = new JLabel("Num 2:");
    JLabel resultLabel = new JLabel("Result:");

    // Create input and output fields
    JTextField input1Field = new JTextField();
    JTextField input2Field = new JTextField();
    JTextField resultField = new JTextField();

    // Create buttons
    JButton addButton = new JButton("Add");

    // Create panel to hold the components
    JPanel panel = new JPanel();

    // Create radio buttons for background color
    JRadioButton yellowButton = new JRadioButton("Yellow");
    JRadioButton greenButton = new JRadioButton("Green");
```

```
// Create fonts for labels, input and output fields, and buttons
Font sansSerif = new Font("SansSerif", Font.BOLD, 20);
Font serif = new Font("Serif", Font.BOLD, 20);
Font bgFont = new Font("SansSerif", Font.BOLD, 14);

Calculator() {
   // Set the layout for the panel
   panel.setLayout(null);

   // Set the size of the JFrame
   frame.setSize(400, 450);

   // Make the JFrame exit on close
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   // Make the JFrame visible
   frame.setVisible(true);

   // Set the position of the labels and input and output fields
   input1Label.setBounds(50, 50, 150, 30);
   input2Label.setBounds(50, 100, 150, 30);
   resultLabel.setBounds(50, 150, 150, 30);
   input1Field.setBounds(200, 50, 150, 30);
   input2Field.setBounds(200, 100, 150, 30);
   resultField.setBounds(200, 150, 150, 30);

   // Set the position of the buttons
   addButton.setBounds(150, 250, 80, 30);

   // Set the position of the radio buttons
   yellowButton.setBounds(50, 350, 80, 30);
   greenButton.setBounds(250, 350, 80, 30);

   // Set the font of the labels, input and output fields, and buttons
   input1Label.setFont(sansSerif);
   input2Label.setFont(sansSerif);
   resultLabel.setFont(sansSerif);
   input1Field.setFont(serif);
   input2Field.setFont(serif);
   resultField.setFont(serif);
   addButton.setFont(sansSerif);
   yellowButton.setFont(bgFont);
   greenButton.setFont(bgFont);

   // Add components to the panel
```

```
        panel.add(input1Label);
        panel.add(input2Label);
        panel.add(resultLabel);
        panel.add(input1Field);
        panel.add(input2Field);
        panel.add(resultField);
        panel.add(addButton);
        panel.add(yellowButton);
        panel.add(greenButton);

        // Add the panel to the JFrame
        frame.add(panel);

        // Register listeners for buttons and radio buttons
        addButton.addActionListener(this);
        yellowButton.addActionListener(this);
        greenButton.addActionListener(this);
    }

    // Handle button clicks
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == addButton) {
            int x = Integer.parseInt(input1Field.getText());
            int y = Integer.parseInt(input2Field.getText());
            int sum = x + y;
            resultField.setText(Integer.toString(sum));
        } else if (e.getSource() == yellowButton) {
            panel.setBackground(Color.yellow);
            greenButton.setSelected(false);
        } else if (e.getSource() == greenButton) {
            panel.setBackground(Color.green);
            yellowButton.setSelected(false);
        }
    }
}

public class UI_demo {
    public static void main(String[] args) {
        new Calculator();
    }
}
```

DATE: 24-02-2023 **EXPERIMENT NO** 6 **REGISTER NO**: 2162014

OUTPUTS:



RESULTS:

The java program was created successfully to demonstrate the use of textfields, radiobuttons, and button.

DATE: 31-03-2023 **EXPERIMENT NO** 7 **REGISTER NO**: 2162014

IMPLEMENTATION OF EXCEPTIONS

AIM:

Write a java program that uses the try-catch-finally block to handle exceptions during the bank transfer. An exception should be thrown when the transfer amount exceeds the available balance in the customer's account.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
public class excep {

    private double balance;

    public excep(double initialBalance) {
        balance = initialBalance;
    }

    public void transfer(double amount, excep recipient) throws
            InsufficientFundsException {
        try {
            if (amount > balance) {
                throw new InsufficientFundsException("Transfer amount exceeds available
balance");
            } else {
                balance -= amount;
                recipient.balance += amount;
                System.out.println("Transfer successful!");
            }
        } catch (InsufficientFundsException e) {
            System.out.println("Transfer failed: " + e.getMessage());
            throw e;
        }
    }

    public static void main(String[] args) {
        excep account1 = new excep(1000.0);
        excep account2 = new excep(500.0);
        try {
            System.out.println("Acc1 balance before transfer: " + account1.balance);
            System.out.println("Acc2 balance before transfer: " + account2.balance);
            account1.transfer(600.0, account2);
            // account2.transfer(400.0, account1);
```

```
        } catch (InsufficientFundsException e) {
            // Handle the exception here
        } finally {
            System.out.println("Acc1 balance after transfer: " + account1.balance);
            System.out.println("Acc2 balance after transfer: " + account2.balance);
        }
    }
}

class InsufficientFundsException extends Exception {

    public InsufficientFundsException(String message) {
        super(message);
    }
}
```

OUTPUTS:

```
PS C:\Users\ashva\                                                    \exp7> javac excep.java
PS C:\Users\ashva\                                                    \exp7> java excep
Acc1 balance before transfer: 1000.0
Acc2 balance before transfer: 500.0
Transfer failed: Transfer amount exceeds available balance
Acc1 balance after transfer: 1000.0
Acc2 balance after transfer: 500.0
PS C:\Users\ashva\                                                    \exp7> javac excep.java
PS C:\Users\ashva\                                                    \exp7> java excep
Acc1 balance before transfer: 1000.0
Acc2 balance before transfer: 500.0
Transfer successful!
Acc1 balance after transfer: 400.0
Acc2 balance after transfer: 1100.0
```

RESULTS:

The java program was created successfully to demonstrate the use of the try-catch-finally block to handle exceptions during the bank transfer

DATE: 31-03-2023 **EXPERIMENT NO** 8 **REGISTER NO**: 2162014

IMPLEMENTATION OF GENERIC PROGRAMMING

AIM:
Write a java program to implement Generic Programming.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.ArrayList;
import java.util.NoSuchElementException;

public class Stack<T> {

    private ArrayList<T> items;

    public Stack() {
        items = new ArrayList<>();
    }

    public void push(T item) {
        items.add(item);
    }

    public T pop() {
        if (isEmpty()) {
            throw new NoSuchElementException("Stack is empty");
        }
        return items.remove(items.size() - 1);
    }

    public T peek() {
        if (isEmpty()) {
            throw new NoSuchElementException("Stack is empty");
        }
        return items.get(items.size() - 1);
    }

    public boolean isEmpty() {
        return items.isEmpty();
    }

    public int size() {
        return items.size();
```

```
    }

    public static void main(String[] args) {
        Stack<Integer> intStack = new Stack<>();
        intStack.push(61);
        intStack.push(87);
        intStack.push(32);
        System.out.println("Top element: " + intStack.peek());
        System.out.println("Size of stack: " + intStack.size());
        while (!intStack.isEmpty()) {
            System.out.println(intStack.pop());
        }
    }
}
```

OUTPUTS:

```
PS C:\Users\ashva\                                          \exp8> javac Stack.java
PS C:\Users\ashva\                                          \exp8> java Stack
Top element: 32
Size of stack: 3
32
87
61
```

RESULTS:

The java program was created successfully implements Generic Programming.

DATE: 31-03-2023 **EXPERIMENT NO** 9 **REGISTER NO**: 2162014

IMPLEMENTATION OF MULTITHREADED PROGRAMS

AIM:
Write a java program to implement Multithreaded programs.

PROGRAM:

```
/**
 *
 * @author 2162014
 */
import java.util.Random;

public class th {

    public static void main(String[] args) {
        Random random = new Random();
        NumberGenerator numberGenerator = new NumberGenerator(random);
        SquareCalculator squareCalculator = new SquareCalculator();
        CubeCalculator cubeCalculator = new CubeCalculator();
        Thread generatorThread = new Thread(numberGenerator);
        Thread squareThread = new Thread(squareCalculator);
        Thread cubeThread = new Thread(cubeCalculator);
        generatorThread.start();
        squareThread.start();
        cubeThread.start();
    }
}

class NumberGenerator implements Runnable {

    private final Random random;

    public NumberGenerator(Random random) {
        this.random = random;
    }

    @Override
    public void run() {
        while (true) {
            int number = random.nextInt(10);
            if (number % 2 == 0) {
                SquareCalculator.handleNumber(number);
            } else {
                CubeCalculator.handleNumber(number);
            }
```

```
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

class SquareCalculator implements Runnable {

    public static synchronized void handleNumber(int number) {
        System.out.println("Received an even number: " + number);
        int square = number * number;
        System.out.println("Square of the number: " + square);
    }

    @Override
    public void run() {
        // This thread doesn't need to do anything, as the handleNumber() method
        // is static and synchronized, so it can be called from any thread.
    }
}

class CubeCalculator implements Runnable {

    public static synchronized void handleNumber(int number) {
        System.out.println("Received an odd number: " + number);
        int cube = number * number * number;
        System.out.println("Cube of the number: " + cube);
    }

    @Override
    public void run() {
        // This thread doesn't need to do anything, as the handleNumber() method
        // is static and synchronized, so it can be called from any thread.
    }
}
```

OUTPUTS:

DATE: 31-03-2023 **EXPERIMENT NO** 9 **REGISTER NO**: 2162014

```
PS C:\Users\ashva\                                                \exp9> javac th.java
PS C:\Users\ashva\                                                \exp9> java th
Received an even number: 6
Square of the number: 36
Received an odd number: 1
Cube of the number: 1
Received an odd number: 3
Cube of the number: 27
Received an even number: 4
Square of the number: 16
Received an even number: 2
Square of the number: 4
Received an odd number: 5
Cube of the number: 125
Received an even number: 4
Square of the number: 16
Received an even number: 8
Square of the number: 64
Received an even number: 4
Square of the number: 16
Received an odd number: 7
Cube of the number: 343
Received an odd number: 1
Cube of the number: 1
Received an even number: 8
Square of the number: 64
Received an even number: 4
Square of the number: 16
Received an odd number: 5
Cube of the number: 125
Received an even number: 2
Square of the number: 4
```

RESULTS:

The java program was created successfully implements Multithreaded programs.