Source Code:

Stack Implementation:

```
import java.util.*;
public class Experiment_01_Stack {

    //Stack Class
    public static class Stack {

        static final int MAX = 10;
        int top = -1;
        int st[] = new int[MAX];

        boolean isEmpty(){
            if(top==-1) return true;
            else return false;
        }

        int size(){
            return top+1;
        }

        int top(){
            if(top==-1){
                System.out.println("Stack is Empty!");
                return 0;
            }
            else{
                return st[top];
            }
        }

        void push(int n){
```

```
        if(top>=MAX-1){
            System.out.println("Stack Overflow!");
        }
        else{
            top++;
            st[top] = n;
        }
    }

    void pop(){
        if(top==-1){
            System.out.println("Stack Underflow!");
        }
        else{
            System.out.println(st[top--] + " Popped!");
        }
    }

    void display(){
        if(top==-1){
            System.out.println("Stack is Empty!");
        }
        else{
            System.out.println("Stack is : ");
            for(int i=top;i>=0;i--){
                System.out.println(st[i]);
            }
        }
    }

}
```

public static void main(String[] args){

```
System.out.println("\n***** Stack Implementation *****\n");
System.out.println("Enter your choice from (1 - 7): ");
System.out.println("1. Push");
System.out.println("2. Pop");
System.out.println("3. Top");
System.out.println("4. Display");
System.out.println("5. isEmpty");
System.out.println("6. Size");
System.out.println("7. Exit");

Scanner sc = new Scanner(System.in);
Stack s = new Stack();
int option = 0;

while(option!=7){
    System.out.print("\nEnter your choice: ");
    option = sc.nextInt();

    switch(option){
        case 1:
            System.out.print("Enter the number to be pushed: ");
            int n = sc.nextInt();
            s.push(n);
            break;

        case 2:
            s.pop();
            break;

        case 3:
            System.out.println("Top element is: "+s.top());
            break;
```

```
            case 4:
                s.display();
                break;

            case 5:
                System.out.println("isEmpty: "+s.isEmpty());
                break;

            case 6:
                System.out.println("Size: "+s.size());
                break;

            case 7:
                System.out.println("Exited!");
                break;

            default:
                System.out.println("Invalid Choice!");
        }
    }

    sc.close();

  }
}
```

Queue Implementation:

```
import java.util.*;
public class Experiment_01_Queue {

  public static class Queue{
    static final int MAX = 10;
    int front = -1;
```

```
int rear = -1;
int q[] = new int[MAX];

boolean isEmpty(){
    if(front==-1 && rear==-1) return true;
    else return false;
}

int size(){
    return rear-front+1;
}

int front(){
    if(front==-1){
        System.out.println("Queue is Empty!");
        return 0;
    }
    else{
        return q[front];
    }
}

int rear(){
    if(rear==-1){
        System.out.println("Queue is Empty!");
        return 0;
    }
    else{
        return q[rear];
    }
}
```

```
void enqueue(int n){

    if(rear>=MAX-1){
        System.out.println("Queue Overflow!");
    }
    else{
        if(front==-1) front++;
        rear++;
        q[rear] = n;
    }
}

void dequeue(){
    if(front==-1){
        System.out.println("Queue Underflow!");
    }
    else{
        System.out.println(q[front++] + " Dequeued!");
        if(front>rear){
            front = -1;
            rear = -1;
        }
    }
}
void display(){
    if(front==-1){
        System.out.println("Queue is Empty!");
    }
    else{
        for(int i=front;i<=rear;i++){
            System.out.print(q[i] + " ");
        }
```

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

public static void main(String[] args){
    System.out.println("\n***** Queue Implementation *****\n");
    System.out.println("Enter your choice from (1 - 7): ");
    System.out.println("1. Enqueue: ");
    System.out.println("2. Dequeue: ");
    System.out.println("3. Front: ");
    System.out.println("4. Rear: ");
    System.out.println("5. Display");
    System.out.println("6. isEmpty");
    System.out.println("7. Size");
    System.out.println("8. Exit");

    Queue q = new Queue();
    Scanner sc = new Scanner(System.in);
    int option = 0;

    while(option!=8){

        System.out.print("\nEnter your choice: ");
        option = sc.nextInt();

        switch(option) {
            case 1:
                System.out.print("Enter the element to be enqueued: ");
                int n = sc.nextInt();
                q.enqueue(n);
                break;
```

```
    case 2:
        q.dequeue();
        break;

    case 3:
        System.out.println("Front: " + q.front());
        break;

    case 4:
        System.out.println("Rear: " + q.rear());
        break;

    case 5:
        q.display();
        break;

    case 6:
        System.out.println("isEmpty: " + q.isEmpty());
        break;

    case 7:
        System.out.println("Size: " + q.size());
        break;

    case 8:
        System.out.println("Exited!");
        break;

    default:
        System.out.println("Invalid Choice!");
}
```

```
        }
        sc.close();
    }
}
```

Output:

```
***** Stack Implementation *****

Enter your choice:
1. Push
2. Pop
3. Top
4. Display
5. isEmpty
6. Size
7. Exit

Enter your choice: 1
Enter the number to be pushed: 10

Enter your choice: 1
Enter the number to be pushed: 20

Enter your choice: 1
Enter the number to be pushed: 30

Enter your choice: 1
Enter the number to be pushed: 40

Enter your choice: 4
Stack is :
40
30
20
10

Enter your choice: 6
Size: 4

Enter your choice: 5
isEmpty: false

Enter your choice: 2
40 Popped!

Enter your choice: 2
30 Popped!

Enter your choice: 2
20 Popped!

Enter your choice: 2
10 Popped!

Enter your choice: 2
Stack Underflow!

Enter your choice: 5
isEmpty: true

Enter your choice: 7
Exited!
```

```
***** Queue Implementation *****

Enter your choice from (1 - 7):
1. Enqueue:
2. Dequeue:
3. Front:
4. Rear:
5. Display
6. isEmpty
7. Size
8. Exit

Enter your choice: 1
Enter the element to be enqueued: 10

Enter your choice: 1
Enter the element to be enqueued: 20

Enter your choice: 1
Enter the element to be enqueued: 30

Enter your choice: 1
Enter the element to be enqueued: 40

Enter your choice: 5
10 20 30 40

Enter your choice: 3
Front: 10

Enter your choice: 4
Rear: 40

Enter your choice: 6
isEmpty: false

Enter your choice: 7
Size: 4

Enter your choice: 2
10 Dequeued!

Enter your choice: 2
20 Dequeued!

Enter your choice: 2
30 Dequeued!

Enter your choice: 2
40 Dequeued!

Enter your choice: 2
Queue Underflow!

Enter your choice: 8
Exited!
```

Source Code:

Dynamic Polymorphism in Java:

```
import java.lang.*;

class Animal{
   void eat(){
      System.out.println("Animal eating...! \n");
   }
}

class Dog extends Animal{
   void eat(){
      System.out.println("Dog eating Bread...! \n");
   }
}

class Cat extends Animal{
   void eat(){
      System.out.println("Cat eating Rat...! \n");
   }
}

class Lion extends Animal{
   void eat(){
      System.out.println("Lion eating Meat...! \n");
   }
}
public class Experiment_02_Polymorphism{
```

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

        System.out.println("Demonstrating Concept of Polymorphism in Java!\n");

        Animal a = new Animal();
        Animal b = new Dog();
        Animal c = new Cat();
        Animal d = new Lion();

        a.eat();
        b.eat();
        c.eat();
        d.eat();
    }
}
```

Interfaces in Java:

```
import java.lang.*;
// Interface
interface Animal{
    void sound();
}

// Class Dog
class Dog implements Animal{
    public void sound(){
        System.out.println("Dog: Bark...!\n");
    }
}

// Class Cat
```

```
class Cat implements Animal{
    public void sound(){
        System.out.println("Cat: Meow...!\n");
    }
}

// Class Lion
class Lion implements Animal{
    public void sound(){
        System.out.println("Lion: Roar...!\n");
    }
}

// Main Class
public class Experiment_02_Interfaces{

    public static void main(String[] args){

        System.out.println("Demonstrating Concept of Interfaces in Java!\n");
        Animal a = new Dog();
        Animal b = new Cat();
        Animal c = new Lion();

        a.sound();
        b.sound();
        c.sound();
    }
}
```

Output:

```
Demonstrating Concept of Polymorphism in Java!

Animal eating...!

Dog eating Bread...!

Cat eating Rat...!

Lion eating Meat...!
```

```
Demonstrating Concept of Interfaces in Java!

Dog: Bark...!

Cat: Meow...!

Lion: Roar...!
```

Source Code:

```
import java.util.Scanner;


class Q {
    int n;
    boolean signal = false;


    synchronized int consume() {
        while (!signal)
            try {
                System.out.println("Consumer thread sleeps");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        System.out.println("Consumer thread awakes");
        System.out.println("Consumed: " + n);
        signal = false;
        notify();
        return n;
    }


    synchronized void produce(int n) {
        while (signal)
            try {
                System.out.println("Producer thread sleeps");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        this.n = n;
        System.out.println("Producer thread awakes");
```

```
        signal = true;
        System.out.println("Produced: " + n);
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    int n;

    Producer(Q q, int n) {
        this.n = n;
        this.q = q;
        new Thread(this, "Producer").start();
    }

    public void run() {
        System.out.println("Producer thread created");
        int i = 0;
        while (i < n) {
            q.produce(++i);
        }
        System.out.println("Producer thread sleeps");
    }
}

class Consumer implements Runnable {
    Q q;
    int n;

    Consumer(Q q, int n) {
        this.q = q;
```

```
        this.n = n;
        new Thread(this, "Consumer").start();
    }

    public void run() {
        System.out.println("Consumer thread created");
        while (true) {
            q.consume();
        }
    }
}

public class Experiment_03 {
    public static void main(String[] args) {
        int n;
        System.out.println("Enter the production limit");
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        System.out.println("The production limit is " + n);
        Q q = new Q();
        new Producer(q, n);
        new Consumer(q, n);

        sc.close();
    }
}
```

Output:

```
Enter the production limit
5
The production limit is 5
Producer thread created
Producer thread awakes
Consumer thread created
Produced: 1
Producer thread sleeps
Consumer thread awakes
Consumed: 1
Consumer thread sleeps
Producer thread awakes
Produced: 2
Producer thread sleeps
Consumer thread awakes
Consumed: 2
Consumer thread sleeps
Producer thread awakes
Produced: 3
Producer thread sleeps
Consumer thread awakes
Consumed: 3
Consumer thread sleeps
Producer thread awakes
Produced: 4
Producer thread sleeps
Consumer thread awakes
Consumed: 4
Consumer thread sleeps
Producer thread awakes
Produced: 5
Consumer thread awakes
Producer thread sleeps
Consumed: 5
Consumer thread sleeps
```

Source Code:

```
import java.lang.*;

public class Experiment_04{

    static void divide(int a, int b) throws ArithmeticException{

        if(b = = 0){
            throw new ArithmeticException("Divide by zero error! ");
        }
        else{
            System.out.println("Result: " + a/b);
        }
    }

    public static void main(String[] args){

        //Try and Catch and Finally Keywords -----
        try{
            int a = 10;
            int b = 0;
            int c = a/b;
            System.out.println(c);

        }catch (ArithmeticException e){
            System.out.println(e);
        }finally{

            System.out.println("\nFinally Block Always Executed!\n");
        }

        //Throw Keyword ----------
```

```
        int num = -5;
        if (num < 1) {
            throw new ArithmeticException("\nNumber is negative, cannot calculate square root! \n");
        }
        else {
            System.out.println("\nSquare root of " + num + " is: " + Math.sqrt(num));
        }

        //Throws Keyword----------------
        int n = 10;
        int m = 0;
        divide(n,m);
    }
}
```

Output:

```
java.lang.ArithmeticException: / by zero

Finally Block Always Executed!

Exception in thread "main" java.lang.ArithmeticException:
Number is negative, cannot calculate square root!

        at Experiment_04.main(Experiment_04.java:41)


Exception in thread "main" java.lang.ArithmeticException: Divide by
zero error!
        at Experiment_04.divide(Experiment_04.java:9)
        at Experiment_04.main(Experiment_04.java:52)
```

Source Code:

```
import java.util.*;
import java.io.*;

public class Experiment_05{

    public static void main(String[] args) {

        try{
            File file = new File("file.txt");
            Scanner sc = new Scanner(file);
            String str = "";

            while(sc.hasNextLine()){
                str += sc.nextLine();
                str += '\n';
            }

            System.out.println("\nFile Before Converting: \n\n" + str);

            str = str.toUpperCase();

            System.out.println("\nFile After Converting: \n\n" + str);

            FileWriter fw = new FileWriter(file);
            fw.write(str);
            fw.close();
            sc.close();

        }catch(FileNotFoundException e){

            System.out.println("File not found");
```

```
        }

        catch(IOException e){

            System.out.println("Error");
        }
    }
}
```

Output:

```
File Before Converting:

name : rahul kumar
roll no : 10714802720
batch : 5c6


File After Converting:

NAME : RAHUL KUMAR
ROLL NO : 10714802720
BATCH : 5C6
```

file.txt Experiment_05.java

file.txt

```
1  name : rahul kumar
2  roll no : 10714802720
3  batch : 5c6
4
```

file.txt Experiment_05.java

file.txt

```
1  NAME : RAHUL KUMAR
2  ROLL NO : 10714802720
3  BATCH : 5C6
4
```

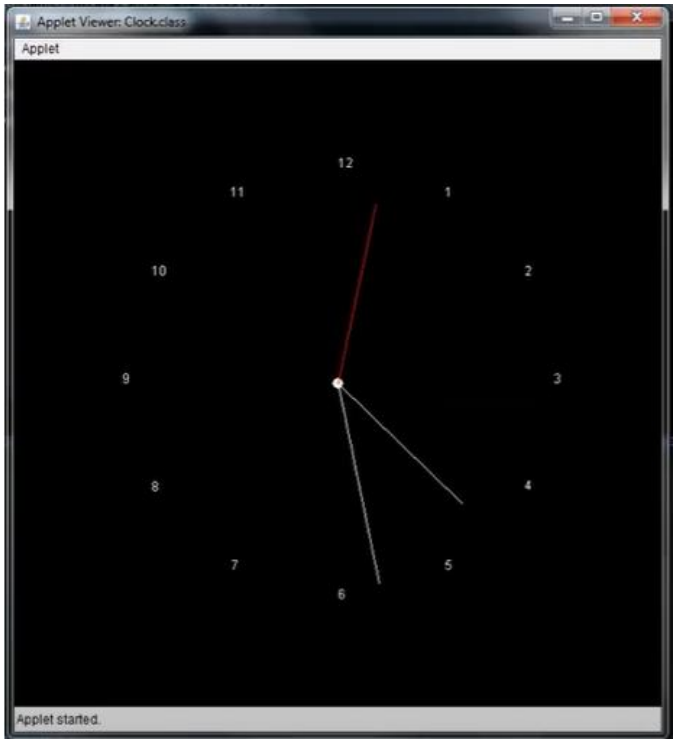
Source Code:

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Color;
import java.util.Calendar;

public class Clock extends Applet {
    //override init

    public void init() {
        //set background
        setBackground(new Color(0, 0, 0));
    }

    public void start() {
        new Thread() {
            public void run() {
                while (true) {
                    try {
                        Thread.sleep(1000);
                    } catch (Exception e) {
                    }
                    repaint();
                }
            }
        }.start();
    }
//override paint
    public void paint(Graphics g) {
        //declare variables
        double angle;
```

```
int x, y;

//draw numbers
g.setColor(new Color(255, 255, 255));
for (int i = 1; i <= 12; i++) {
    angle = i * Math.PI / 6;
    x = (int) (200 * Math.sin(angle));
    y = (int) (200 * Math.cos(angle));
    g.drawString(Integer.toString(i), 300 + x, 300 - y);
}
g.fillOval(295, 295, 10, 10);

//get calendar instance
Calendar cal = Calendar.getInstance();
int hour = cal.get(Calendar.HOUR_OF_DAY);
int min = cal.get(Calendar.MINUTE);
int sec = cal.get(Calendar.SECOND);

hour = hour % 12;
//draw hour hand
angle = (hour * Math.PI / 6)
        + (min * Math.PI / (6 * 60))
        + (sec * Math.PI / (360 * 60));
x = (int) (160 * Math.sin(angle));
y = (int) (160 * Math.cos(angle));
g.drawLine(300, 300, 300 + x, 300 - y);

//draw minute hand
angle = (min * Math.PI / 30)
        + (sec * Math.PI / (30 * 60));
x = (int) (190 * Math.sin(angle));
y = (int) (190 * Math.cos(angle));
```

```
        g.drawLine(300, 300, 300 + x, 300 - y);

        g.setColor(new Color(255, 0, 0));
        angle = (sec * Math.PI / (30));
        x = (int) (190 * Math.sin(angle));
        y = (int) (190 * Math.cos(angle));
        g.drawLine(300, 300, 300 + x, 300 - y);
    }
}

/*
applet code to view applet
<applet code="Clock.class" width=600 height=600>
</applet>
*/
```

Output:



CODE:

```
package javaexperiment;

import java.awt.event.*;

import javax.swing.*;

import java.awt.*;

class calculator extends JFrame implements ActionListener {

	static JFrame f;

	static JTextField l;

	String s0, s1, s2;

	calculator(){

		s0 = s1 = s2 = "";

	}

	public static void main(String args[])

	{

		f = new JFrame("RAHUL 107(C6)");

		try {

		UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

		}

		catch (Exception e) {

			System.err.println(e.getMessage());

		}

		calculator c = new calculator();

		l = new JTextField(16);

		l.setEditable(false);
```

```
JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, ba, bs, bd, bm, be, beq, beq1;
b0 = new JButton("0");
b1 = new JButton("1");
b2 = new JButton("2");
b3 = new JButton("3");
b4 = new JButton("4");
b5 = new JButton("5");
b6 = new JButton("6");
b7 = new JButton("7");
b8 = new JButton("8");
b9 = new JButton("9");
beq1 = new JButton("=");
ba = new JButton("+");
bs = new JButton("-");
bd = new JButton("/");
bm = new JButton("*");
beq = new JButton("C");
be = new JButton(".");
JPanel p = new JPanel();
bm.addActionListener(c);
bd.addActionListener(c);
bs.addActionListener(c);
ba.addActionListener(c);
b9.addActionListener(c);
b8.addActionListener(c);
```

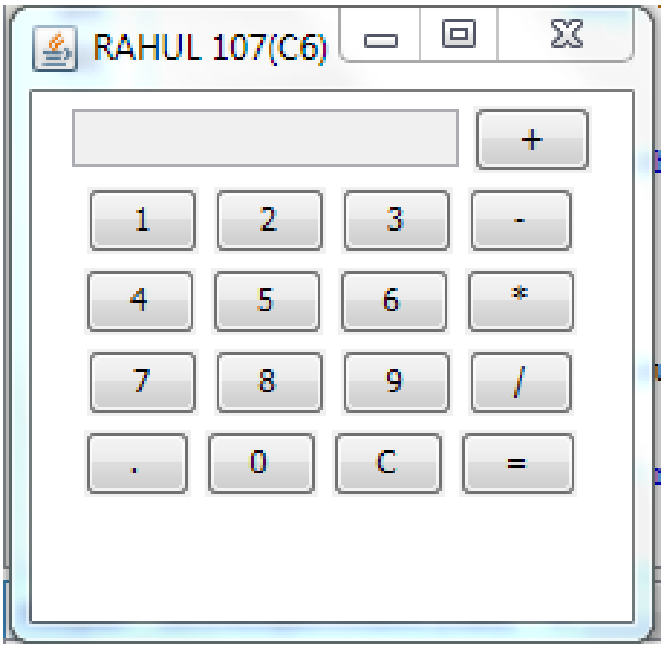
```
b7.addActionListener(c);
b6.addActionListener(c);
b5.addActionListener(c);
b4.addActionListener(c);
b3.addActionListener(c);
b2.addActionListener(c);
b1.addActionListener(c);
b0.addActionListener(c);
be.addActionListener(c);
beq.addActionListener(c);
beq1.addActionListener(c);

p.add(l);
p.add(ba);
p.add(b1);
p.add(b2);
p.add(b3);
p.add(bs);
p.add(b4);
p.add(b5);
p.add(b6);
p.add(bm);
p.add(b7);
p.add(b8);
p.add(b9);
```

```
        p.add(bd);
        p.add(be);
        p.add(b0);
        p.add(beq);
        p.add(beq1);
        p.setBackground(Color.white);
        f.add(p);
        f.setSize(200, 220);
        f.show();
    }
    public void actionPerformed(ActionEvent e){
        String s = e.getActionCommand();
        if ((s.charAt(0) >= '0' && s.charAt(0) <= '9') || s.charAt(0) == '.') {
            if (!s1.equals(""))
                s2 = s2 + s;
            else
                s0 = s0 + s;
            l.setText(s0 + s1 + s2);
        }
        else if (s.charAt(0) == 'C'){
            s0 = s1 = s2 = "";
            l.setText(s0 + s1 + s2);
        }
        else if (s.charAt(0) == '=') {
```

```
            double te;
            if (s1.equals("+"))
                te = (Double.parseDouble(s0) + Double.parseDouble(s2));
            else if (s1.equals("-"))
                te = (Double.parseDouble(s0) - Double.parseDouble(s2));
            else if (s1.equals("/"))
                te = (Double.parseDouble(s0) / Double.parseDouble(s2));
            else
                te = (Double.parseDouble(s0) * Double.parseDouble(s2));
            l.setText(s0 + s1 + s2 + "=" + te);
            s0 = Double.toString(te);
            s1 = s2 = "";
        }
        else {
            if (s1.equals("") || s2.equals(""))
                s1 = s;
            else {
                double te;

                if (s1.equals("+"))
                    te = (Double.parseDouble(s0) + Double.parseDouble(s2));
                else if (s1.equals("-"))
                    te = (Double.parseDouble(s0) - Double.parseDouble(s2));
                else if (s1.equals("/"))
                    te = (Double.parseDouble(s0) / Double.parseDouble(s2));
```

```
                    else
                        te = (Double.parseDouble(s0) * Double.parseDouble(s2));
                    s0 = Double.toString(te);
                    s1 = s;
                    s2 = "";
                }
                l.setText(s0 + s1 + s2);
            }
        }
}
```

OUTPUT:



CODE:

```
package javaexperiment;

import java.awt.*;

import javax.swing.*;

import java.io.*;

import java.awt.event.*;

import javax.swing.plaf.metal.*;

import javax.swing.text.*;


class editor extends JFrame implements ActionListener {

    JTextArea t;

    JFrame f;

    editor() {

        f = new JFrame("editor");

        try {

            UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");

            MetalLookAndFeel.setCurrentTheme(new OceanTheme());

        } catch (Exception e) {

        }

        t = new JTextArea();

        JMenuBar mb = new JMenuBar();

        JMenu m1 = new JMenu("File");

        JMenuItem mi1 = new JMenuItem("New");

        JMenuItem mi2 = new JMenuItem("Open");

        JMenuItem mi3 = new JMenuItem("Save");
```

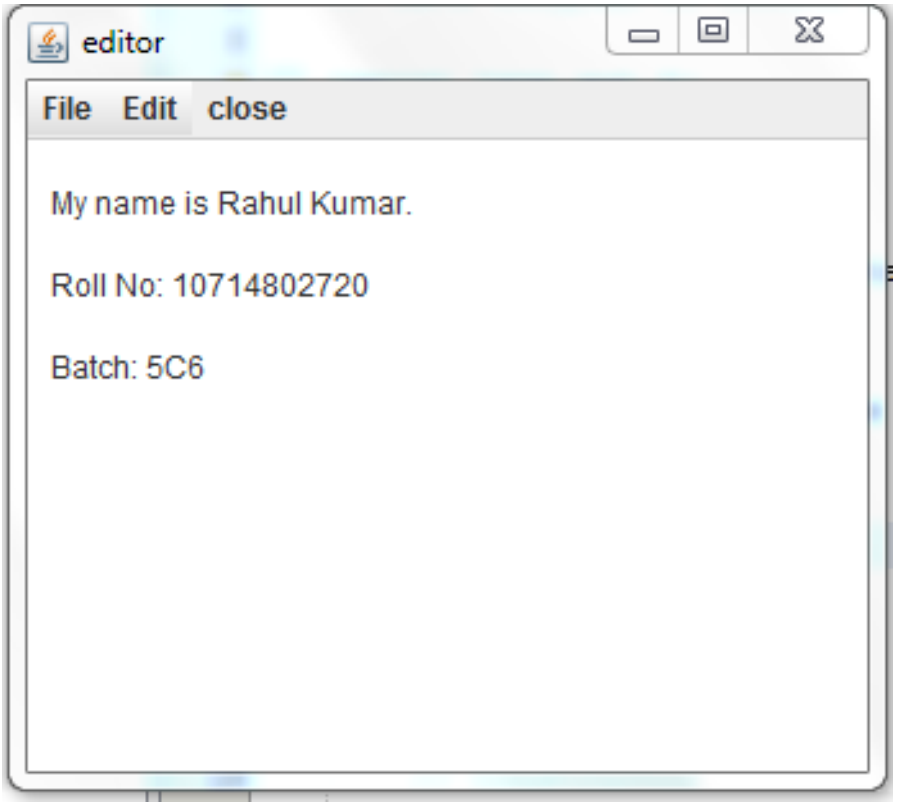
```
JMenuItem mi9 = new JMenuItem("Print");
mi1.addActionListener(this);
mi2.addActionListener(this);
mi3.addActionListener(this);
mi9.addActionListener(this);

m1.add(mi1);
m1.add(mi2);
m1.add(mi3);
m1.add(mi9);
JMenu m2 = new JMenu("Edit");
JMenuItem mi4 = new JMenuItem("cut");
JMenuItem mi5 = new JMenuItem("copy");
JMenuItem mi6 = new JMenuItem("paste");
mi4.addActionListener(this);
mi5.addActionListener(this);
mi6.addActionListener(this);

m2.add(mi4);
m2.add(mi5);
m2.add(mi6);

JMenuItem mc = new JMenuItem("close");
mc.addActionListener(this);
mb.add(m1);
```

```
    mb.add(m2);
    mb.add(mc);

    f.setJMenuBar(mb);
    f.add(t);
    f.setSize(500, 500);
    f.show();
}
public void actionPerformed(ActionEvent e) {
    String s = e.getActionCommand();

    if (s.equals("cut")) {
        t.cut();
    } else if (s.equals("copy")) {
        t.copy();
    } else if (s.equals("paste")) {
        t.paste();
    } else if (s.equals("Save")) {
        // Create an object of JFileChooser class
        JFileChooser j = new JFileChooser("f:");
        int r = j.showSaveDialog(null);

        if (r == JFileChooser.APPROVE_OPTION) {
            File fi = new File(j.getSelectedFile().getAbsolutePath());
            try {
```

```
            FileWriter wr = new FileWriter(fi, false);
            BufferedWriter w = new BufferedWriter(wr);
            w.write(t.getText());
            w.flush();
            w.close();
        } catch (Exception evt) {
            JOptionPane.showMessageDialog(f, evt.getMessage());
        }
    }
    else {
        JOptionPane.showMessageDialog(f, "the user cancelled the operation");
    }
} else if (s.equals("Print")) {
    try {
        t.print();
    } catch (Exception evt) {
        JOptionPane.showMessageDialog(f, evt.getMessage());
    }
} else if (s.equals("Open")) {
    JFileChooser j = new JFileChooser("f:");
    int r = j.showOpenDialog(null);
    if (r == JFileChooser.APPROVE_OPTION) {
        File fi = new File(j.getSelectedFile().getAbsolutePath());
        try {
            String s1 = "", sl = "";
```

```
                FileReader fr = new FileReader(fi);
                BufferedReader br = new BufferedReader(fr);
                sl = br.readLine();
                while ((s1 = br.readLine()) != null) {
                    sl = sl + "\n" + s1;
                }
                t.setText(sl);
            } catch (Exception evt) {
                JOptionPane.showMessageDialog(f, evt.getMessage());
            }
        }
        else {
            JOptionPane.showMessageDialog(f, "the user cancelled the operation");
        }
    } else if (s.equals("New")) {
        t.setText("");
    } else if (s.equals("close")) {
        f.setVisible(false);
    }
  }
  public static void main(String args[]) {
    editor e = new editor();
  }
}
```

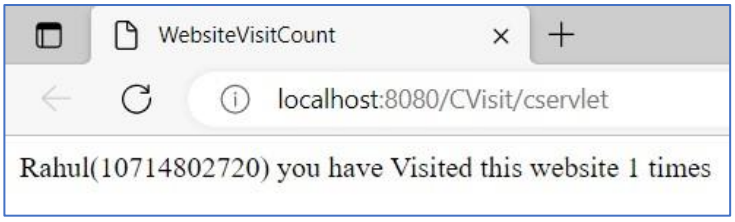
OUTPUT:

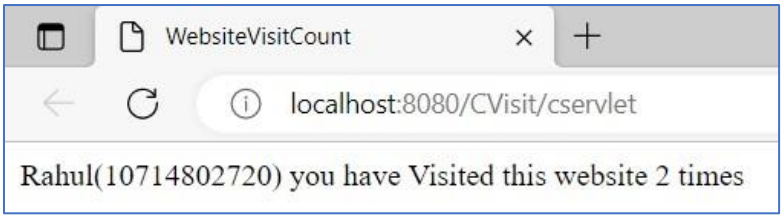

editor
File
Edit
close
My name is Rahul Kumar.
Roll No: 10714802720
Batch: 5C6

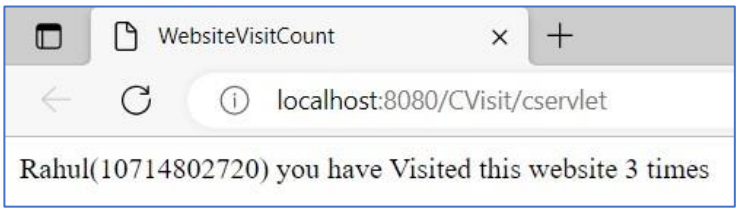
CODE:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Visit extends HttpServlet {
    static int i = 0;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<title>WebsiteVisitCount</title>");
        String name = "Rahul(10714802720)";
        Cookie c = new Cookie("visit", String.valueOf(i));
        response.addCookie(c);

        int j = Integer.parseInt(c.getValue());
        if (j == 0) {
            out.println("Welcome " + name);
        } else {
            out.println(name + " you have Visited this website " + j + " times");
        }
        i++;
    }
}
```

OUTPUT:







CODE:

```
import java.beans.*;

public class BoundBeans {
    private String name;
    private int age;

    private PropertyChangeSupport pcs = new PropertyChangeSupport(this);
    private VetoableChangeSupport vcs = new VetoableChangeSupport(this);

    public BoundBeans() {
    }

    public String getName() {
        return name;
    }

    public void setName(String name) throws PropertyVetoException {
        String oldName = this.name;
        vcs.fireVetoableChange("name", oldName, name);
        this.name = name;
        pcs.firePropertyChange("name", oldName, name);
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) throws PropertyVetoException {
        int oldAge = this.age;
        vcs.fireVetoableChange("age", oldAge, age);
        this.age = age;
        pcs.firePropertyChange("age", oldAge, age);
    }

    public void addPropertyChangeListener(PropertyChangeListener listener) {
        pcs.addPropertyChangeListener(listener);
    }
```

```
    public void removePropertyChangeListener(PropertyChangeListener listener) {
        pcs.removePropertyChangeListener(listener);
    }

    public void addVetoableChangeListener(VetoableChangeListener listener) {
        vcs.addVetoableChangeListener(listener);
    }

    public void removeVetoableChangeListener(VetoableChangeListener listener) {
        vcs.removeVetoableChangeListener(listener);
    }

    public static void main(String[] args) throws PropertyVetoException {
        BoundBeans bean = new BoundBeans();
        bean.addVetoableChangeListener(new VetoableChangeListener() {
            @Override
            public void vetoableChange(PropertyChangeEvent evt) throws PropertyVetoException {
                if (evt.getNewValue().equals("Invalid name")) {
                    throw new PropertyVetoException("Invalid name", evt);
                }
            }
        });
        bean.addPropertyChangeListener(new PropertyChangeListener() {
            @Override
            public void propertyChange(PropertyChangeEvent evt) {
                System.out.println("\n" + evt.getPropertyName() + " Changed From " +
evt.getOldValue() + " To " + evt.getNewValue());
            }
        });
        bean.setName("Rahul Kumar");
        bean.setAge(20);
        try {
            bean.setName("Invalid Name \n");
        } catch (PropertyVetoException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

OUTPUT:

```
name changed from null to Rahul Kumar
age changed from 0 to 20
Invalid name
```

CODE:

```
package JAVA;
import java.util.*;

public class Merge {
    public static void Merge_Sort(int[] Arr, int Begin, int End) {
        if (Begin < End) {
            int Mid = (Begin + End) / 2;
            Merge_Sort(Arr, Begin, Mid);
            Merge_Sort(Arr, Mid + 1, End);
            merge(Arr, Begin, Mid, End);
        }
    }
    public static void merge(int[] Arr, int Begin, int Mid, int End) {
        int n1 = Mid - Begin + 1;
        int n2 = End - Mid;
        int[] L = new int[n1];
        int[] R = new int[n2];

        for (int i = 0; i < n1; i++) {
            L[i] = Arr[Begin + i];
        }
        for (int j = 0; j < n2; j++) {
            R[j] = Arr[Mid + 1 + j];
        }

        int i = 0, j = 0;
        int k = Begin;

        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                Arr[k] = L[i];
                i++;
            } else {
                Arr[k] = R[j];
                j++;
            }
            k++;
        }
```

```
        while (i < n1) {
            Arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            Arr[k] = R[j];
            j++;
            k++;
        }
    }

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);
        System.out.print("\n Enter the size of the array: ");
        int n = sc.nextInt();

        int[] Arr = new int[n];
        System.out.print(" Enter the Elements of the array: ");
        for (int i = 0; i < n; i++) {
            Arr[i] = sc.nextInt();
        }

        Merge_Sort(Arr, 0, n - 1);

        System.out.print("\n The Sorted array is: ");
        for (int i = 0; i < n; i++) {
            System.out.print(Arr[i] + " ");
        }
        sc.close();
    }
}
```

OUTPUT:

```
Enter the size of the array: 10

Enter the elements of the array: 9 10 5 2 3 4 1 6 8 7

The sorted array is: 1 2 3 4 5 6 7 8 9 10
```

CODE:

```
package JAVA;
import java.util.*;
interface Enter_Number{
    int Enter_Num();
}

interface CalculatePrime{
    boolean isPrime(int n);
    void Print_Prime(int n);
}
class Print_Prime_Num implements Enter_Number, CalculatePrime{

    public int Enter_Num()
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("\nEnter the value of n: ");
        int n = sc.nextInt();
        sc.close();
        return n;
    }
    public boolean isPrime(int n)
    {
        for (int i = 2; i <= n / 2; i++) {
            if (n % i == 0) {
                return false;
            }
        }
        return true;
    }
```

```
    public void Print_Prime(int n) {
        int count = 0;
        int i = 2;
        while (count < n) {
            if (isPrime(i)) {
                System.out.println(i);
                count++;
            }
            i++;
        }
    }

}
public class Experiment_12{
    public static void main(String[] args) {
        Print_Prime_Num Obj = new Print_Prime_Num();
        int n = Obj.Enter_Num();
        System.out.println("First " + n + " prime numbers are: ");
        Obj.Print_Prime(n);
    }
}
```

OUTPUT:

```
Enter the value of n: 10
First 10 prime numbers are:
1 : 2
2 : 3
3 : 5
4 : 7
5 : 11
6 : 13
7 : 17
8 : 19
9 : 23
10 : 29
```