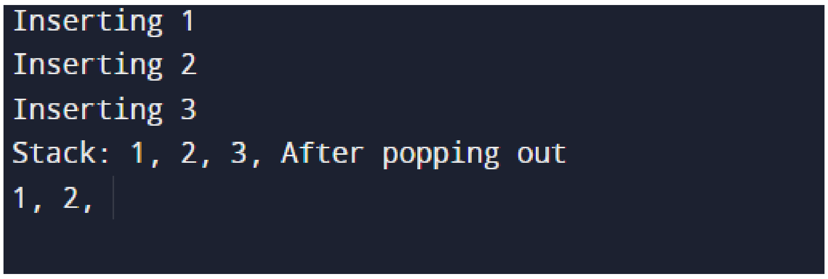
<u>Experiment - 1</u>

Code:-

1. Java program to implement Stack:-

```
class Stack {
  private int arr[];
  private int top;
  private int capacity;
  Stack(int size) {
    arr = new int[size];
    capacity = size;
    top = -1;
  }
  public void push(int x) {
    if (isFull()) {
      System.out.println("Stack OverFlow");

      // terminates the program
      System.exit(1);
    }
    System.out.println("Inserting " + x);
    arr[++top] = x;
  }
  public int pop() {
    if (isEmpty()) {
      System.out.println("STACK EMPTY");
      System.exit(1);
    }

    return arr[top--];
  }
  public int getSize() {
    return top + 1;
  }
  public Boolean isEmpty() {
    return top == -1;
  }
  public Boolean isFull() {
    return top == capacity - 1;
  }
  public void printStack() {
    for (int i = 0; i <= top; i++) {
      System.out.print(arr[i] + ", ");
    }
  }

  public static void main(String[] args) {
    Stack stack = new Stack(5);

    stack.push(1);
    stack.push(2);
    stack.push(3);

    System.out.print("Stack: ");
    stack.printStack();
    stack.pop();
    System.out.println("\nAfter popping out");
    stack.printStack();

  }
}
```

Output:-



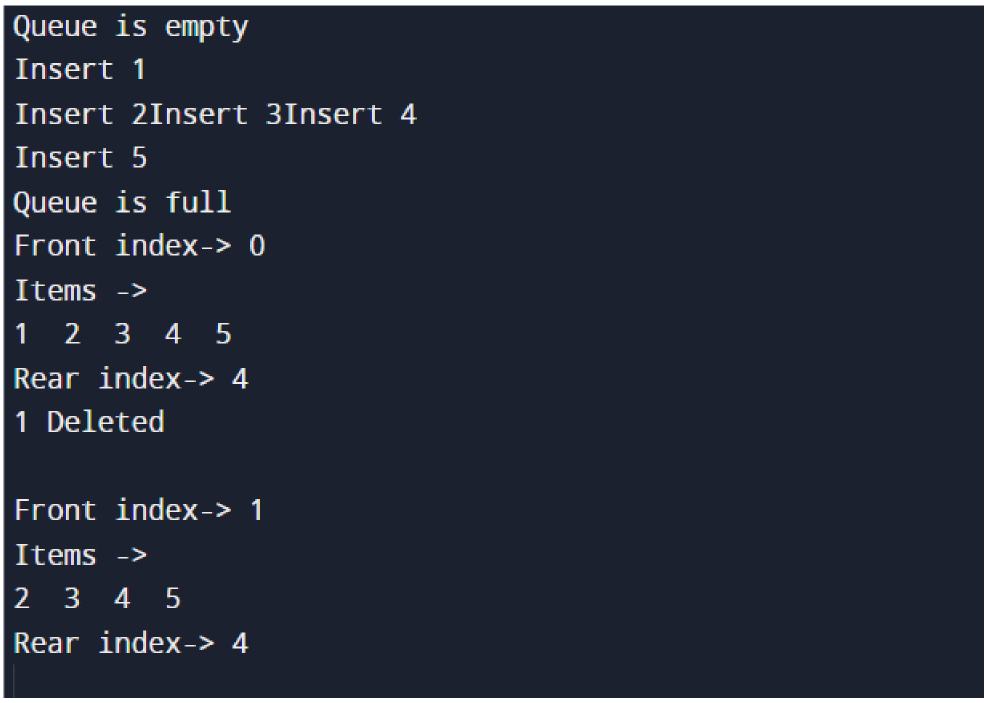
```
Inserting 1
Inserting 2
Inserting 3
Stack: 1, 2, 3, After popping out
1, 2,
```

2. Java program to implement Queue

```
public class Queue {
  int SIZE = 5;
  int items[] = new int[SIZE];
  int front, rear;

  Queue() {
    front = -1;
    rear = -1;
  }

  boolean isFull() {
    if (front == 0 && rear == SIZE - 1) {
      return true;
    }
    return false;
  }

  boolean isEmpty() {
    if (front == -1)
      return true;
    else
    return false;
  }
  void enQueue(int element) {
   if (isFull()) {
    System.out.println("Queue is full");
   }
   else {
    if (front == -1) {
     front = 0;
    }
    rear++;
    items[rear] = element;
    System.out.println("Insert " + element);
   }
  }
  int deQueue() {
   int element;
   if (isEmpty()) {
    System.out.println("Queue is empty");
    return (-1);
   }
   else {
    element = items[front];

    if (front >= rear) {
     front = -1;
     rear = -1;
    }
    else {
     front++;
    }
    System.out.println( element + " Deleted");
    return (element);
  }
}
void display() {
  int i;
  if (isEmpty()) {
    System.out.println("Empty Queue");
  }
  else {
    System.out.println("\nFront index-> " + front);
    System.out.println("Items -> ");
    for (i = front; i <= rear; i++)
      System.out.print(items[i] + "  ");
    System.out.println("\nRear index-> " + rear);
  }
}
public static void main(String[] args) {
  Queue q = new Queue();
  q.deQueue();
  for(int i = 1; i < 6; i ++) {
    q.enQueue(i);
  }
  q.enQueue(6);
  q.display();
  q.deQueue();
  q.display();


}
```

}

Output:-



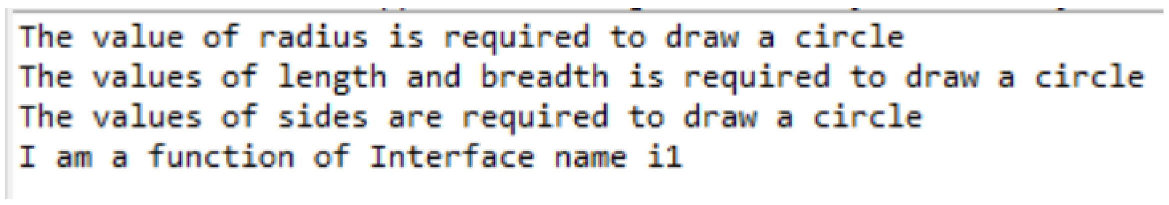
```
Queue is empty
Insert 1
Insert 2Insert 3Insert 4
Insert 5
Queue is full
Front index-> 0
Items ->
1  2  3  4  5
Rear index-> 4
1 Deleted

Front index-> 1
Items ->
2  3  4  5
Rear index-> 4
```

Experiment - 2

Code:-

```
package demo_pack;


interface i1 {
    void demo_func();
}


class circle implements i1 {
    public void demo_func() {
        System.out.println("I am a function of Interface named i1");
    }


    public void draw() {
        System.out.println("The value of radius is required to draw a circle");
    }
}


class rectangle extends circle {
    public void draw() {
        System.out.println("The values of length and breadth is required to draw a circle");
    }
}


class triangle extends circle {
    public void draw() {
        System.out.println("The values of sides are required to draw a circle");
    }
}


public class lab2 {
    public static void main(String[] args) {
        circle obj1 = new circle();
        circle obj2 = new rectangle();
        circle obj3 = new triangle();
        obj1.draw();
        obj2.draw();
        obj3.draw();
        obj1.demo_func();
    }
}
```

Output:-



```
The value of radius is required to draw a circle
The values of length and breadth is required to draw a circle
The values of sides are required to draw a circle
I am a function of Interface name i1
```

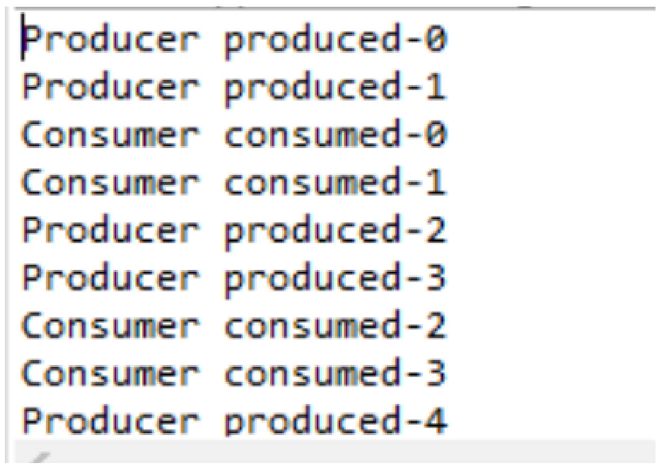
Experiment - 3

Code:-

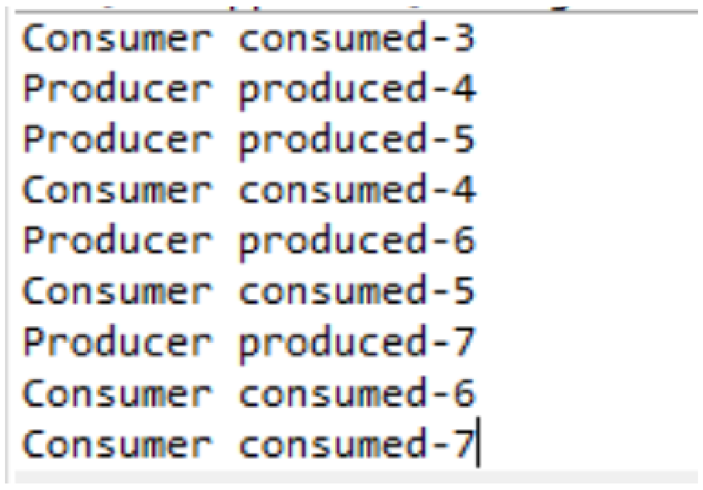
```
package demo_pack;
import java.util.LinkedList;

public class lab3 {
  public static void main(String[] args) throws InterruptedException {
    final PC pc = new PC();
    Thread t1 = new Thread(new Runnable() {
      public void run() {
        try {
          pc.produce();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    });
    Thread t2 = new Thread(new Runnable() {
      public void run() {
        try {
          pc.consume();
        } catch (InterruptedException e) {
          e.printStackTrace();
        }
      }
    });
    t1.start();
    t2.start();
    t1.join();
    t2.join();
  }
```

```
public static class PC {
    LinkedList<Integer> list = new LinkedList<>();
    int capacity = 2;

    public void produce() throws InterruptedException {
        int value = 0;
        while (true) {
            synchronized (this) {
                while (list.size() == capacity)
                    wait();
                System.out.println("Producer produced-" + value);
                list.add(value++);
                notify();
                Thread.sleep(1000);
            }
        }
    }

    public void consume() throws InterruptedException {
        while (true) {
            synchronized (this) {
                while (list.size() == 0)
                    wait();
                int val = list.removeFirst();
                System.out.println("Consumer consumed-" + val);
                notify();
                Thread.sleep(1000);
            }
        }
        }
    }
}
```

Output:-



```
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
Producer produced-3
Consumer consumed-2
Consumer consumed-3
Producer produced-4
```



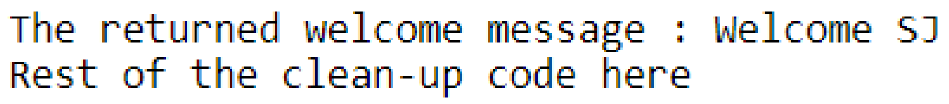
```
Consumer consumed-3
Producer produced-4
Producer produced-5
Consumer consumed-4
Producer produced-6
Consumer consumed-5
Producer produced-7
Consumer consumed-6
Consumer consumed-7
```

<u>Experiment - 4</u>

Code:-

```
package in.bench.resources.exception.handling;

public class DemoOnTryCatchFinallyThrowThrows {

  // main() method - start of JVM execution
  public static void main(String[] args) {

    try {
      // call division() method
      String welcomeMessage = welcomeMessage("SJ");

      // print to console
      System.out.println("The returned welcome message : "
          + welcomeMessage);
    }
    catch (NullPointerException npex){
      System.out.println("Exception handled : "
          + npex.toString());
    }
    finally {
      System.out.println("Rest of the clean-up code here");
    }
  }

  // division method returning quotient
  public static String welcomeMessage(String name)
      throws NullPointerException {

    if(name == null) {
            // explicitly throwing Null Pointer Error
            // using throw keyword
            throw new NullPointerException(
                    "Invoke method with VALID name");
        }

        // performing String concatenation
        String welcomeMsg = "Welcome " + name;

        /// return concatenated string value
        return welcomeMsg;
    }
}
```

Output:-



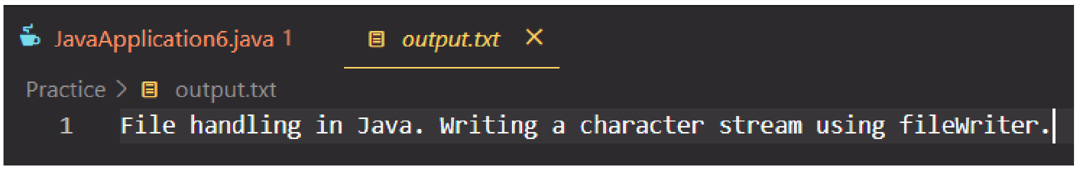
```
The returned welcome message : Welcome SJ
Rest of the clean-up code here
```

Experiment - 5

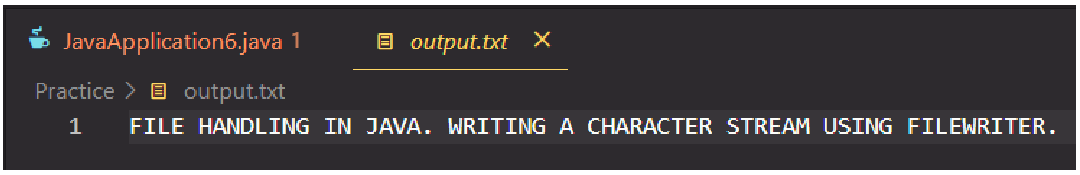
Code:-

```
package javaapplication6;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.FileReader;
import java.io.IOException;

// public class JavaApplication6 {
//    public static void main(String[] args) throws IOException {
//        String str = "File handling in Java. Writing a character stream using fileWriter.";
//        FileWriter fw = new FileWriter("output.txt");
//        for (int i = 0; i < str.length(); i++)
//            fw.write(str.charAt(i));
//        System.out.println("Writing successful");
//        fw.close();
//        int ch;
//        FileReader fr = new FileReader("output.txt");
//        while ((ch = fr.read()) != -1)
//            System.out.print((char) ch);
//        fr.close();
//    } }

public class JavaApplication6 {
    public static void main(String[] args) throws IOException {
        int ch;
        FileWriter fw = new FileWriter("output2.txt");
        FileReader fr = new FileReader("output.txt");
        while ((ch = fr.read()) != -1) {
            if (Character.isLowerCase(ch)) {
                fw.write(Character.toUpperCase(ch));
            } else {
                fw.write(ch);
            }
        }
        fw.close();
        fr.close();
        FileReader r = new FileReader("output2.txt");
        FileWriter w = new FileWriter("output.txt");
        while ((ch = r.read()) != -1) {
            w.write(ch);
        }
        w.close();
        r.close();
        FileReader f = new FileReader("output.txt");
        while ((ch = f.read()) != -1)
            System.out.print((char) ch);
        f.close();
    }}
```

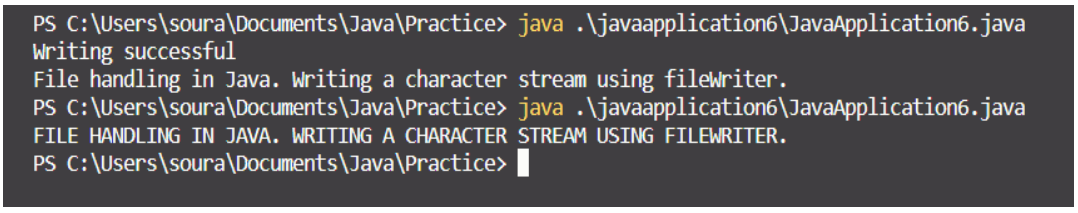
Output:-



```
File handling in Java. Writing a character stream using fileWriter.
```



```
FILE HANDLING IN JAVA. WRITING A CHARACTER STREAM USING FILEWRITER.
```



```
PS C:\Users\soura\Documents\Java\Practice> java .\javaapplication6\JavaApplication6.java
Writing successful
File handling in Java. Writing a character stream using fileWriter.
PS C:\Users\soura\Documents\Java\Practice> java .\javaapplication6\JavaApplication6.java
FILE HANDLING IN JAVA. WRITING A CHARACTER STREAM USING FILEWRITER.
PS C:\Users\soura\Documents\Java\Practice>
```

Experiment - 6

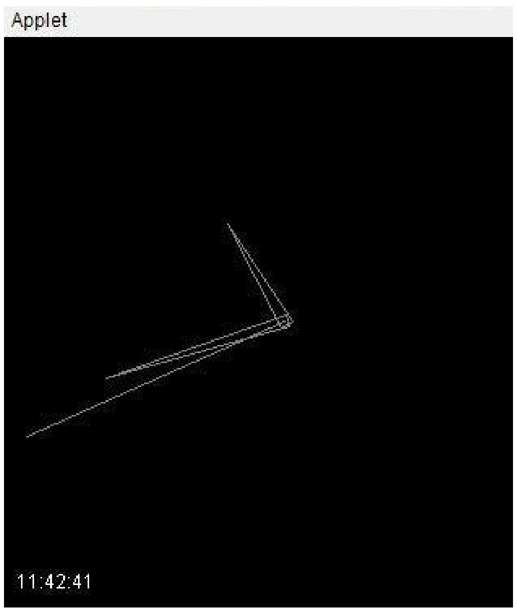
Code:-

```
import java.util.*;

import java.text.*;

import java.applet.*;

import java.awt.*;

public class SimpleAnalogClock extends Applet implements Runnable

{

   int clkhours = 0, clkminutes = 0, clkseconds = 0;

   String clkString = "";

   int clkwidth, clkheight;

   Thread thr = null;

   boolean threadSuspended;

   public void init()

   {

      clkwidth = getSize().width;

      clkheight = getSize().height;

      setBackground(Color.black);

   }

   public void start()

   {

      if (thr == null)

      {

         thr = new Thread(this);

         thr.setPriority(Thread.MIN_PRIORITY);

         threadSuspended = false;

         thr.start();

      } else

      {

         if (threadSuspended)

         {
               threadSuspended = false;
               synchronized(this)
               {
                   notify();
               }
         }
      }
   }
   public void stop()
   {
      threadSuspended = true;
   }
   public void run()
   {
      try
      {
         while (true)
         {
               Calendar clndr = Calendar.getInstance();
               clkhours = clndr.get(Calendar.HOUR_OF_DAY);
               if (clkhours > 12) clkhours -= 12;
               clkminutes = clndr.get(Calendar.MINUTE);
               clkseconds = clndr.get(Calendar.SECOND);
               SimpleDateFormat frmatter = new SimpleDateFormat("hh:mm:ss",
Locale.getDefault());
               Date d = clndr.getTime();
               clkString = frmatter.format(d);
               if (threadSuspended)
               {
                   synchronized(this)
                   {
```

```
                while (threadSuspended)
                {
                    wait();
                }
            }
        }
        repaint();
        thr.sleep(1000);
    }
  } catch (Exception e)
  {}
}
void drawHand(double angle, int radius, Graphics grp)
{
  angle -= 0.5 * Math.PI;
  int a = (int)(radius * Math.cos(angle));
  int b = (int)(radius * Math.sin(angle));
  grp.drawLine(clkwidth / 2, clkheight / 2, clkwidth / 2 + a, clkheight / 2 + b);
}
void drawWedge(double angle, int radius, Graphics grp)
{
  angle -= 0.5 * Math.PI;
  int a = (int)(radius * Math.cos(angle));
  int b = (int)(radius * Math.sin(angle));
  angle += 2 * Math.PI / 3;
  int a2 = (int)(5 * Math.cos(angle));
  int b2 = (int)(5 * Math.sin(angle));
  angle += 2 * Math.PI / 3;
  int a3 = (int)(5 * Math.cos(angle));
  int b3 = (int)(5 * Math.sin(angle));
```

```
        grp.drawLine(clkwidth / 2 + a2, clkheight / 2 + b2, clkwidth / 2 + a, clkheight / 2 + b);
        grp.drawLine(clkwidth / 2 + a3, clkheight / 2 + b3, clkwidth / 2 + a, clkheight / 2 + b);
        grp.drawLine(clkwidth / 2 + a2, clkheight / 2 + b2, clkwidth / 2 + a3, clkheight / 2 + b3);
    }
    public void paint(Graphics grp)
    {
        grp.setColor(Color.gray);
        drawWedge(2 * Math.PI * clkhours / 12, clkwidth / 5, grp);
        drawWedge(2 * Math.PI * clkminutes / 60, clkwidth / 3, grp);
        drawHand(2 * Math.PI * clkseconds / 60, clkwidth / 2, grp);
        grp.setColor(Color.white);
        grp.drawString(clkString, 10, clkheight - 10);
    }
}
/*
<applet code="SimpleAnalogClock.class" width="350" height="350">
</applet>
*/
```

Output:-



```
11:42:41
```

Experiment - 7

Code:-

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.event.*;

public class ScientificCalculator extends JFrame implements ActionListener {
        JTextField tfield;
        double temp, temp1, result, a;
        static double m1, m2;
        int k = 1, x = 0, y = 0, z = 0;
        char ch;
        JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, zero, clr, pow2, pow3, exp,
                        fac, plus, min, div, log, rec, mul, eq, addSub, dot, mr, mc, mp,
                        mm, sqrt, sin, cos, tan;
        Container cont;
        JPanel textPanel, buttonpanel;

        ScientificCalculator() {
                cont = getContentPane();
                cont.setLayout(new BorderLayout());
                JPanel textpanel = new JPanel();
                tfield = new JTextField(25);
                tfield.setHorizontalAlignment(SwingConstants.RIGHT);
                tfield.addKeyListener(new KeyAdapter() {
                        public void keyTyped(KeyEvent keyevent) {
                                char c = keyevent.getKeyChar();
                                if (c >= '0' && c <= '9') {
                                } else {
                                        keyevent.consume();
```

```
                    }
          }
});
textpanel.add(tfield);
buttonpanel = new JPanel();
buttonpanel.setLayout(new GridLayout(8, 4, 2, 2));
boolean t = true;
mr = new JButton("MR");
buttonpanel.add(mr);
mr.addActionListener(this);
mc = new JButton("MC");
buttonpanel.add(mc);
mc.addActionListener(this);

mp = new JButton("M+");
buttonpanel.add(mp);
mp.addActionListener(this);

mm = new JButton("M-");
buttonpanel.add(mm);
mm.addActionListener(this);

b1 = new JButton("1");
buttonpanel.add(b1);
b1.addActionListener(this);
b2 = new JButton("2");
buttonpanel.add(b2);
b2.addActionListener(this);
b3 = new JButton("3");
buttonpanel.add(b3);
```

```
b3.addActionListener(this);

b4 = new JButton("4");
buttonpanel.add(b4);
b4.addActionListener(this);
b5 = new JButton("5");
buttonpanel.add(b5);
b5.addActionListener(this);
b6 = new JButton("6");
buttonpanel.add(b6);
b6.addActionListener(this);

b7 = new JButton("7");
buttonpanel.add(b7);
b7.addActionListener(this);
b8 = new JButton("8");
buttonpanel.add(b8);
b8.addActionListener(this);
b9 = new JButton("9");
buttonpanel.add(b9);
b9.addActionListener(this);

zero = new JButton("0");
buttonpanel.add(zero);
zero.addActionListener(this);

plus = new JButton("+");
buttonpanel.add(plus);
plus.addActionListener(this);
```

```
min = new JButton("-");
buttonpanel.add(min);
min.addActionListener(this);

mul = new JButton("*");
buttonpanel.add(mul);
mul.addActionListener(this);

div = new JButton("/");
div.addActionListener(this);
buttonpanel.add(div);

addSub = new JButton("+/-");
buttonpanel.add(addSub);
addSub.addActionListener(this);

dot = new JButton(".");
buttonpanel.add(dot);
dot.addActionListener(this);

eq = new JButton("=");
buttonpanel.add(eq);
eq.addActionListener(this);

rec = new JButton("1/x");
buttonpanel.add(rec);
rec.addActionListener(this);
sqrt = new JButton("Sqrt");
buttonpanel.add(sqrt);
sqrt.addActionListener(this);
```

```
log = new JButton("log");
buttonpanel.add(log);
log.addActionListener(this);

sin = new JButton("SIN");
buttonpanel.add(sin);
sin.addActionListener(this);
cos = new JButton("COS");
buttonpanel.add(cos);
cos.addActionListener(this);
tan = new JButton("TAN");
buttonpanel.add(tan);
tan.addActionListener(this);
pow2 = new JButton("x^2");
buttonpanel.add(pow2);
pow2.addActionListener(this);
pow3 = new JButton("x^3");
buttonpanel.add(pow3);
pow3.addActionListener(this);
exp = new JButton("Exp");
exp.addActionListener(this);
buttonpanel.add(exp);
fac = new JButton("n!");
fac.addActionListener(this);
buttonpanel.add(fac);

clr = new JButton("AC");
buttonpanel.add(clr);
clr.addActionListener(this);
cont.add("Center", buttonpanel);
```

```
		cont.add("North", textpanel);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}

	public void actionPerformed(ActionEvent e) {
		String s = e.getActionCommand();
		if (s.equals("1")) {
			if (z == 0) {
				tfield.setText(tfield.getText() + "1");
			} else {
				tfield.setText("");
				tfield.setText(tfield.getText() + "1");
				z = 0;
			}
		}
		if (s.equals("2")) {
			if (z == 0) {
				tfield.setText(tfield.getText() + "2");
			} else {
				tfield.setText("");
				tfield.setText(tfield.getText() + "2");
				z = 0;
			}
		}
		if (s.equals("3")) {
			if (z == 0) {
				tfield.setText(tfield.getText() + "3");
			} else {
				tfield.setText("");
				tfield.setText(tfield.getText() + "3");
```

```
                z = 0;
        }
}
if (s.equals("4")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "4");
        } else {
                tfield.setText("");
                tfield.setText(tfield.getText() + "4");
                z = 0;
        }
}
if (s.equals("5")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "5");
        } else {
                tfield.setText("");
                tfield.setText(tfield.getText() + "5");
                z = 0;
        }
}
if (s.equals("6")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "6");
        } else {
                tfield.setText("");
                tfield.setText(tfield.getText() + "6");
                z = 0;
        }
}
```

```
if (s.equals("7")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "7");
        } else {
                tfield.setText("");
                tfield.setText(tfield.getText() + "7");
                z = 0;
        }
}
if (s.equals("8")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "8");
        } else {
                tfield.setText("");
                tfield.setText(tfield.getText() + "8");
                z = 0;
        }
}
if (s.equals("9")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "9");
        } else {
                tfield.setText("");
                tfield.setText(tfield.getText() + "9");
                z = 0;
        }
}
if (s.equals("0")) {
        if (z == 0) {
                tfield.setText(tfield.getText() + "0");
```

```
			} else {
				tfield.setText("");
				tfield.setText(tfield.getText() + "0");
				z = 0;
			}
		}
		if (s.equals("AC")) {
			tfield.setText("");
			x = 0;
			y = 0;
			z = 0;
		}
		if (s.equals("log")) {
			if (tfield.getText().equals("")) {
				tfield.setText("");
			} else {
				a = Math.log(Double.parseDouble(tfield.getText()));
				tfield.setText("");
				tfield.setText(tfield.getText() + a);
			}
		}
		if (s.equals("1/x")) {
			if (tfield.getText().equals("")) {
				tfield.setText("");
			} else {
				a = 1 / Double.parseDouble(tfield.getText());
				tfield.setText("");
				tfield.setText(tfield.getText() + a);
			}
		}
```

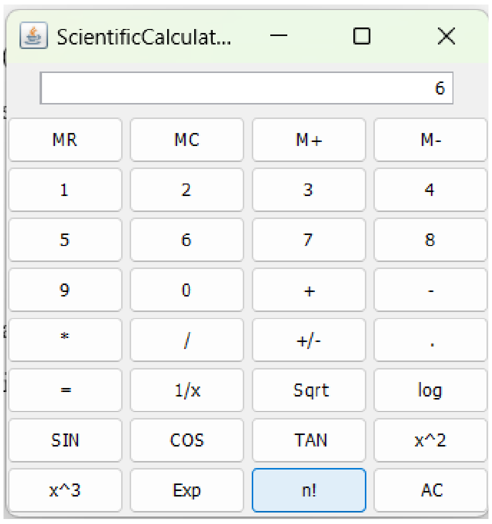
```
if (s.equals("Exp")) {
        if (tfield.getText().equals("")) {
                tfield.setText("");
        } else {
                a = Math.exp(Double.parseDouble(tfield.getText()));
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
        }
}
if (s.equals("x^2")) {
        if (tfield.getText().equals("")) {
                tfield.setText("");
        } else {
                a = Math.pow(Double.parseDouble(tfield.getText()), 2);
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
        }
}
if (s.equals("x^3")) {
        if (tfield.getText().equals("")) {
                tfield.setText("");
        } else {
                a = Math.pow(Double.parseDouble(tfield.getText()), 3);
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
        }
}
if (s.equals("+/-")) {
        if (x == 0) {
                tfield.setText("-" + tfield.getText());
```

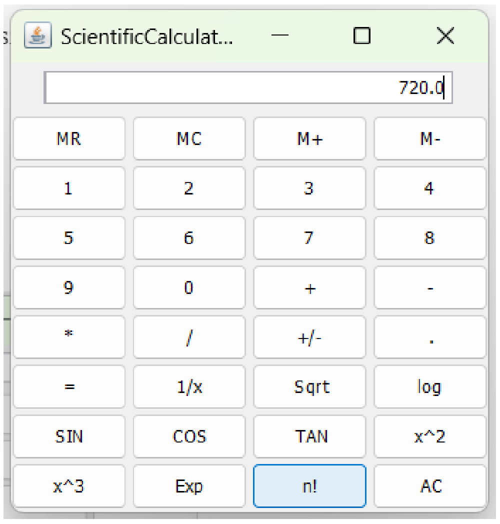
```
                    x = 1;
            } else {
                    tfield.setText(tfield.getText());
            }
    }
    if (s.equals(".")) {
            if (y == 0) {
                    tfield.setText(tfield.getText() + ".");
                    y = 1;
            } else {
                    tfield.setText(tfield.getText());
            }
    }
    if (s.equals("+")) {
            if (tfield.getText().equals("")) {
                    tfield.setText("");
                    temp = 0;
                    ch = '+';
            } else {
                    temp = Double.parseDouble(tfield.getText());
                    tfield.setText("");
                    ch = '+';
                    y = 0;
                    x = 0;
            }
            tfield.requestFocus();
    }
    if (s.equals("-")) {
            if (tfield.getText().equals("")) {
                    tfield.setText("");
```

```
                temp = 0;
                ch = '-';
            } else {
                x = 0;
                y = 0;
                temp = Double.parseDouble(tfield.getText());
                tfield.setText("");
                ch = '-';
            }
            tfield.requestFocus();
        }
        if (s.equals("/")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
                temp = 1;
                ch = '/';
            } else {
                x = 0;
                y = 0;
                temp = Double.parseDouble(tfield.getText());
                ch = '/';
                tfield.setText("");
            }
            tfield.requestFocus();
        }
        if (s.equals("*")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
                temp = 1;
                ch = '*';
            } else {
                x = 0;
                y = 0;
                temp = Double.parseDouble(tfield.getText());
                ch = '*';
                tfield.setText("");
            }
            tfield.requestFocus();
        }
        if (s.equals("MC")) {
            m1 = 0;
            tfield.setText("");
        }
        if (s.equals("MR")) {
            tfield.setText("");
            tfield.setText(tfield.getText() + m1);
        }
        if (s.equals("M+")) {
            if (k == 1) {
                m1 = Double.parseDouble(tfield.getText());
                k++;
            } else {
                m1 += Double.parseDouble(tfield.getText());
                tfield.setText("" + m1);
            }
        }
        if (s.equals("M-")) {
            if (k == 1) {
                m1 = Double.parseDouble(tfield.getText());
                k++;
            } else {
                m1 -= Double.parseDouble(tfield.getText());
                tfield.setText("" + m1);
            }
        }
        if (s.equals("Sqrt")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
            } else {
                a = Math.sqrt(Double.parseDouble(tfield.getText()));
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
            }
        }
        if (s.equals("SIN")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
            } else {
                a = Math.sin(Double.parseDouble(tfield.getText()));
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
            }
        }
        if (s.equals("COS")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
            } else {
                a = Math.cos(Double.parseDouble(tfield.getText()));
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
            }
        }
        if (s.equals("TAN")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
            } else {
                a = Math.tan(Double.parseDouble(tfield.getText()));
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
            }
        }
        if (s.equals("=")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
            } else {
                temp1 = Double.parseDouble(tfield.getText());
                switch (ch) {
                case '+':
                    result = temp + temp1;
                    break;
                case '-':
                    result = temp - temp1;
                    break;
                case '/':
                    result = temp / temp1;
                    break;
                case '*':
                    result = temp * temp1;
                    break;
                }
                tfield.setText("");
                tfield.setText(tfield.getText() + result);
                z = 1;
            }
        }
        if (s.equals("n!")) {
            if (tfield.getText().equals("")) {
                tfield.setText("");
            } else {
                a = fact(Double.parseDouble(tfield.getText()));
                tfield.setText("");
                tfield.setText(tfield.getText() + a);
            }
        }
        tfield.requestFocus();
}

double fact(double x) {
    int er = 0;
    if (x < 0) {
        er = 20;
        return 0;
    }
    double i, s = 1;
    for (i = 2; i <= x; i += 1.0)
        s *= i;
    return s;
}

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

```
                try {
                        UIManager
.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
                } catch (Exception e) {
                }
                ScientificCalculator f = new ScientificCalculator();
                f.setTitle("ScientificCalculator");
                f.pack();
                f.setVisible(true);
        }
}
```

Output:-





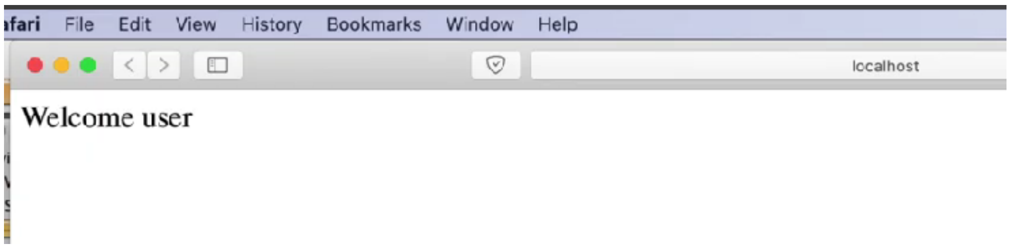
Experiment - 8

Code:-

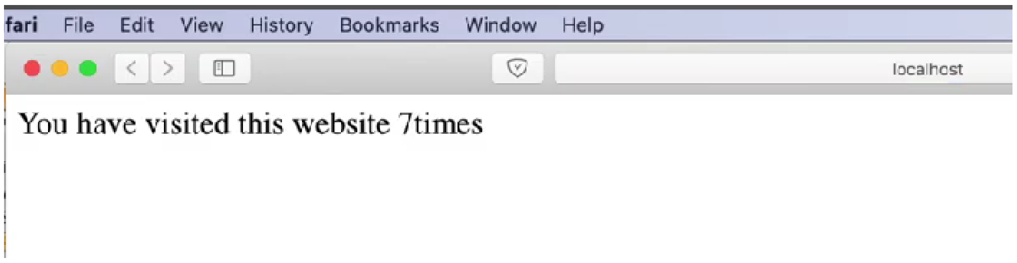
```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;


public class count extends HttpServlet

{

 static int count=0,c2=0;

 public void doGet(HttpServletRequest request, HttpServletResponse response) throws

                 ServletException, IOException

  {

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    String name=request.getParameter("t1");


    Cookie c1=new Cookie("count",String.valueOf(count));

    c2=Integer.parseInt(c1.getValue());


    if(c2==0)

    {

       out.println("Welcome user");

       count++;

    }

   else

    {


       c1=new Cookie("count",String.valueOf(count));

       count++;

       out.println("You have visited the website "+count+"times");

    }

  }}
```

```
<html>
      <body>
        <form action="http://localhost:8080/serv/count" method="get">
      Username:<input type="text" name="t1">
                        <input type="submit" >
        </form>
    </body>
</html>
```

Output:-



```
Welcome user
```



```
You have visited this website 7times
```