LAB - 8

8. Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

→
```
class Multithreading1 extends Thread{
    public void run(){
        for(int i=0; i<5; i++){
            System.out.println("BMS
                College of Engineering");
            try{
                Thread.sleep(10000);
            }
        catch(InterruptedException e){
        }
    }}
}
```

```
class Multithreading2 extends Thread{
    public void run(){
        for(int i=0; i<5; i++){
            System.out.println("CSE");
            try{
                Thread.sleep(2000);
            }
        catch(Interrupted Exception e){
        }
    }}
}
```

```
class Mo main{
    public static void main (string
                              arg.[]){

        Multithreading1 m1 = new
                    Multithreading1();
        m1.start();
        Multithreading2 m2 = new
                    Multithreading2();
        m2. start();
    }
}
```

O/P:

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
BMS College of Engineering
BMS College of Engineering
BMS College of Engineering
```

LAB-9

9. Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked.

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

class SwingDemo {
    SwingDemo() {
        JFrame jfrm = new
            JFrame
            ("Divider App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new
            FlowLayout());
        jfrm.setDefaultClose
            Operation(JFrame
            EXIT_ON_CLOSE);

        JLabel jlab = new
            JLabel
            ("Enter the
            divider and
            dividend: ");
        JTextField ajtf = new
            JTextfield(8);
        JTextField bjtf = new
            JTextField(8);
```

```
JButton button = new JButton
                 ("Calculate");

JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();

JLabel ansLab = new JLabel();
jfrm.add(err);
jfrm.add(jlab);
 jfrm.add(ajtf);
 jfrm.add(bjtf);
  jfrm.add(button);
  jfrm.add(alab);
  jfrm.add(blab);
  jfrm.add(ansLab);

ActionListener l = new
                   ActionListener(){

        public void actionPerformed
               (ActionEvent evt){

            System.out.println
                ("Action event
                   from a text
                      field");
                }
    };

ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new
                   ActionListener() {
```

```
public void actionPerformed
        (ActionEvent evt){

    try{
        int a = Integer.parseInt
                (ajtf.getText());
        int b = Integer.parseInt
                (bjtf.getText());
        int ans = a/b;

        alab.setText("\nA = "+a);
        blab.setText("\nB = "+b);
        anslab.setText("\nAns = "+
                ans);
    }

    catch(NumberFormatExceptio
            e){

        alab.setText(" ");
        blab.setText(" ");
        anslab.setText(" ");
        err.setText("Enter Only
                Integers!");
    }
    catch(ArithmeticException e){

        alab.setText(" ");
        blab.setText(" ");
        err.setText("Enter
                Only
                Integers!");
    }
}};
```

```
        jfrm.setVisible(true);
    }

    public static void main
                (String args[])
            {
        SwingUtilities.
            invokeLater
                (new Runnable(){

        public void run(){

            new SwingDemo();
        }
    });
}
}
```

O/P :

Enter the divider and dividend:

125 5

calculate

A = 125
B = 5
Ans = 25

> Code for IPC:

```
class Q {
  int n;
  boolean valueSet = false;
  synchronized int get() {
        while(!valueSet)
          try {
              System.out.println
                    ("\n Consumer
                      waiting\n");
            wait();
          }
        catch(InterruptedException e) {
          System.out.println
              ("Interrupted Execution
              caught");
        }
      System.out.println("Got: " + n);
      valueSet = false;
      System.out.println("\n Intimate
                  producer\n");
      notify();
      return n;
    }

synchronized void put(int n) {
      while(valueSet)
        try {
          System.out.println
              ("\n Producer waiting\n");
          wait();
        }
      catch(InterruptedException e) {
          System.out.println("Interrupted
```

```
            Exception caught");
        }
    this.n = n;
    valueSet = true;
    System.out.println("Put: " + n);
    System.out.println("\n Intimate
                Consumer\n");
    notify();
  }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").
                start();
    }
    public void run() {
        int i = 0;
        while (i < 15) {
```

```
            int r = q.get();
            System.out.println("Consumed: " +
                                    r);
            i++;
        }
    }
}

class PCFixed {
    public static void main(String
                            args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press
                Control-C to stop.");
    }
}
```

> Code for Deadlock:

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getNam
                                    ();
        System.out.println(name +
                    " entered A.foo");
        try {
            Thread.sleep(1000);
        }
        catch(Exception e) {
```

```
        System.out.println("A Interrupte
    }
    System.out.println(name + "trying
                    to call B.last()");
    b.last();
    }
    void last(){
        System.out.println("Inside
                    A.last");
    }
}

class B{
    synchronized void bar(A a){
        String name = Thread.
                    currentThread
                    .getName();
        System.out.println(name +
                    "entered B.bar");
        try{
            Thread.sleep(1000);
        }
        catch(Exception e){
            System.out.println("B
                    interrupted");
        }
        System.out.println(name +
                    "trying to call
                    A.last()");
        a.last();
    }
    void last(){
```

```
        System.out.println("Inside A.last");
    }
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName
                ("MainThread");
        Thread t = new Thread(this,
                "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in
                main thread");
    }

public void run() {
        b.bar(a);
        System.out.println("Back in
                other thread");
    }
    public static void main(String
                args[]){

        new Deadlock();
    }
}
```

LAB - 10

10. Demonstrate IPC and Deadlock

→ Press control-C to stop
Put : 0
Intimate Consumer
Producer waiting
Got : 0
Intimate Producer
Put : 1
Intimate Consumer
Producer Waiting
Consumed : 0
Got : 0
~~Producer~~
Consumed : 1
Put : 2
~~Init~~ Intimate Producer
Consumed : 2
Put : 3
Intimate Consumer
Producer waiting
Got : 3

Deadlock :
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last
MainThread trying to call B.last

24/11/24