classmate
Date
Page

Lab-9 (Week-10)

```
import java.io.*;

class B extends Thread{
    public void run(){
        try{
            for(int i=0;i<3;i++){
                System.out.println("BMS");
                Thread.sleep(10000);
            }
            catch(InterruptedException e){
                System.out.println(e);
            }
        }
    }
class C extends Thread{
    public void run(){
        try{
            for(int i=0;i<3;i++){
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        }catch(InterruptedException e){
            System.out.println(e);
        }
    }
}
```

```
class ThreadDemo {
    public static void main(String args[])
    {
        B b = new B();
        C c = new C();
        b.start();
        c.start();
    }
}
```

Deadlock.java

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

```
void last(){
    S.O.p ("inside A.last");
  }
class B{
    Synchronized void bar (A a)
    String name = Thread.currentThread().getName
    System.out.println (name+"extend B.bar");
    try{
        Thread.sleep (1000);
    } catch (Exception e)
        System.out.println ("B interrupted");
    }
    System.out.println (name + " trying to call A.
        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("Main
        Thread");
        Thread t = new Thread (this, "Rac
        t.start();
        a.foo(b);
```

```
        S.O.println("Back in Thread");
    }
    new Deadlock;
}
```

Output:

```
Main Thread entered A.foo
Racing Thread entered B.bar
Main Thread trying to call B.last()
Inside A.last
Back in main Thread
Racing Thread trying to call A.last()
Inside A.last
Back in other Thread
```

Pro.con Java

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while(!valueSet)
        try {
            System.out.println("Consumed waiting");
            wait();
        } catch (InterruptedException e) {
            S.O.ph(e);
        }
        S.O.Ph(" " + n);
        valueSet = false;
        S.O.ph("Test produces");
        notify();
    }
    return n;
}
}

Class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "producer").start();
    }
}
```

```
public void (1) {
        int i=0;
        while (i<3) {
            q.put (i++);
        }
    }
}
class consumer implements Runnable {
        Q q;
        Consumer (Q q) {
            this.q = q;
        new Thread (this, "Consumer").start();
    }
    public void in {
        int i=0;
        while (i<3) {
            int n = q.get();
            S.O.Println ("Consumed :" + n);
            i++;
        }
    }
}
```

Output

Put:1
got: 1
Put :2
got :2
Put :3
got :3

Lab-9

```
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.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel jlab = new JLabel("Enter the divider
          and dividant");

        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(8);

        JButton = 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 ev){
        S.O.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("A = "+a);
            blab.setText("B = "+b);
            ansLab.setText("Ans = "+ans);
```

```
catch (NumberFormatException e){
    alab.setText("");
    blab.setText("");
    anslab.setText("");
    err.setText("Enter only Integers");
}
catch (ArithmeticException e){
    alab.setText("");
    blab.setText("");
    anslab.setText("");
    err.setText("B should be non zero");
}
}
jfrom.setVisible(true);
}
public static void main(String args[]){
SwingUtilities.invokeLater(new Runnable(){
public void run(){
new String Demo();
}
});
}
}
```

OUTPUT:

Enter the dividend and divident:

25 5

Calculate A=25 B=5 Ans=5

classmate
Date
Page

'JFrame' is a class in Java Swing that represents a top level container for a GUI application.

setSize():

The setSize(int width, int height) method is used to set the size of a JFrame in pixels.

setLayout():

The setLayout(LayoutManager manager) method is used to set the layout manager for the frame

setDefaultCloseOperation();

setDefaultCloseOperation(int operation) is used to the set the default close operation for the JFrame.

JLabel:

'JLabel' is a Swing component used to display a non-editable text or image.

JTextfield:

'JTextfield' is a Swing component that allows the user to enter and edit a single line of text.