**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

***Submitted by***

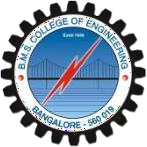
SUBRAMANYA J (1BM23CS343)

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

**(Autonomous Institution under VTU)**

BENGALURU-560019 Sep 2024-Jan 2025

B. M. S. College of Engineering,

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “**OBJECT ORIENTED JAVA PROGRAMMING**” carried out by **SUBRAMANYA J (1BM23CS343),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

**Dr. Nandhini Vineeth Dr. Kavitha Sooda**

Associate Professor, Professor and Head,

Department of CSE, Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 |  | Quadratic Equation |  |
| 2 |  | Student Class |  |
| 3 |  | Book Class with toString() |  |
| 4 |  | Abstract Class Shape |  |
| 5 |  | Bank Class with Inheritance |  |
| 6 |  | CIE Package |  |
| 7 |  | Exception Handling in Father and Son Class |  |
| 8 |  | Multithreaded Programming |  |
| 9 |  | Division with AWT |  |
| 10 |  | Interprocess Communication and Deadlock |  |

Include all the 8 programs as instructed in the classroom.

The order to be maintain for every program is

Question

Observation writeup images (complete)

Soft copy of the program

Screenshot of the output

Editable copies of 9th and 10th program s are attached here. Analyze and execute and include both in the lab record pdf giving the same order as above. Explanation for both topics are included in textbook.

Include page numbers from this page onwards

**LABORATORY PROGRAM - 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. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box.

import java.awt.\*; import java.awt.event.\*;

public class DivisionMain1 extends Frame implements ActionListener

{

TextField num1,num2; Button dResult;

Label outResult; String out=""; double resultNum; int flag=0;

public DivisionMain1()

{

setLayout(new FlowLayout());

dResult = new Button("RESULT");

Label number1 = new Label("Number 1:",Label.RIGHT); Label number2 = new Label("Number 2:",Label.RIGHT); num1=new TextField(5);

num2=new TextField(5);

outResult = new Label("Result:",Label.RIGHT);

add(number1); add(num1); add(number2); add(num2); add(dResult); add(outResult);

num1.addActionListener(this); num2.addActionListener(this); dResult.addActionListener(this); addWindowListener(new WindowAdapter()

{

public void windowClosing(WindowEvent we)

{

System.exit(0);

}

});

}

public void actionPerformed(ActionEvent ae)

{

int n1,n2; try

{

if (ae.getSource() == dResult)

{

n1=Integer.parseInt(num1.getText()); n2=Integer.parseInt(num2.getText());

/\*if(n2==0)

throw new ArithmeticException();\*/ out=n1+" "+n2+" ";

resultNum=n1/n2; out+=String.valueOf(resultNum); repaint();

}

}

catch(NumberFormatException e1)

{

flag=1;

out="Number Format Exception! "+e1; repaint();

}

catch(ArithmeticException e2)

{

flag=1;

out="Divide by 0 Exception! "+e2; repaint();

}

}

public void paint(Graphics g)

{

if(flag==0) g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+outResult. getHeight()-8);

else g.drawString(out,100,200); flag=0;

}

Demonstrate Interprocess communication and deadlock

class Q {

int n;

boolean valueSet = false;

synchronized int get() {

while(!valueSet)

try {

System.out.println("\nConsumer waiting\n");

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException caught");

}

System.out.println("Got: " + n);

valueSet = false;

System.out.println("\nIntimate Producer\n");

notify();

return n;

}

synchronized void put(int n) {

while(valueSet)

try {

System.out.println("\nProducer waiting\n");

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException caught");

}

this.n = n;

valueSet = true;

System.out.println("Put: " + n);

System.out.println("\nIntimate 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);

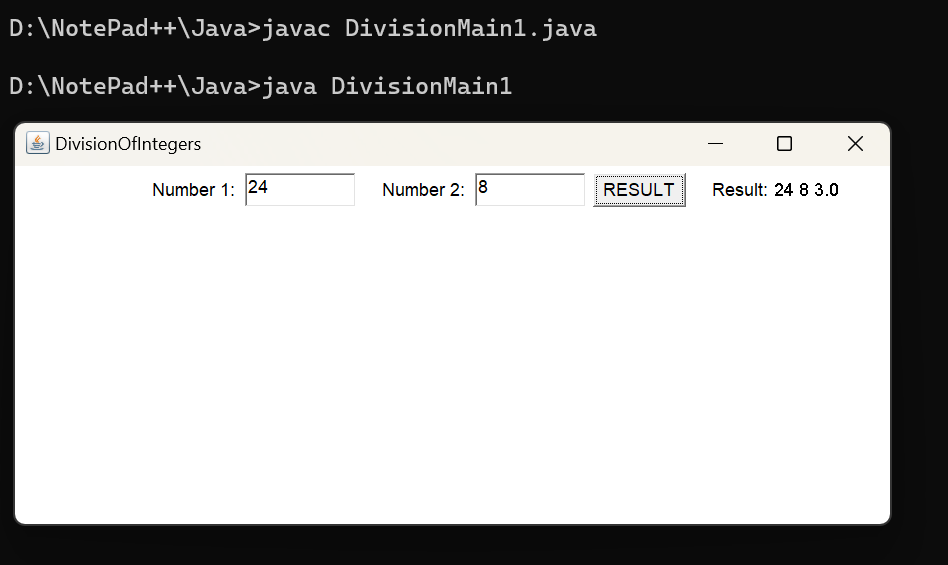
new Consumer(q);

System.out.println("Press Control-C to stop.");

}

}

**OUTPUT**



ii. Demonstration of deadlock

class A

{

synchronized void foo(B b)

{ String name = Thread.currentThread().getName();

System.out.println(name + " entered A.foo");

try { Thread.sleep(1000); }

catch(Exception e) { System.out.println("A Interrupted"); }

System.out.println(name + " trying to call B.last()"); b.last(); }

synchronized 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(); }

synchronized 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); // get lock on a in this thread.

System.out.println("Back in main thread");

}

public void run() { b.bar(a); // get lock on b in other thread.

System.out.println("Back in other thread");

}

public static void main(String args[]) { new Deadlock(); }

}

public static void main(String[] args)

{

DivisionMain1 dm=new DivisionMain1(); dm.setSize(new Dimension(800,400)); dm.setTitle("DivisionOfIntegers"); dm.setVisible(true);

}

}