

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

Object Oriented Java Programming (23CS3PCOOJ)

Submitted by

Tanay S Huddar (1WA23CS007)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING

COMPUTER SCIENCEⁱⁿ AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2024 to 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 (23CS3PCOOJ)” carried out by **Tanay S Huddar (1WA23CS007)**, who is a 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. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Lab faculty Incharge Syed Akram Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
--	--

Index

Sl. No.	Date	Experiment Title	Page No.
1	9/10/24	Quadratic Equation	4-5
2	16/10/24	Calculate SGPA	6-7
3	23/10/24	N book objects	8-9
4	23/10/24	Area of the given Shape	10-11
5	30/10/24	Bank Account	12-17
6	30/10/24	Packages	18-20
7	13/11/24	Exceptions	21-22
8	20/11/24	Threads	23-24
9	4/12/24	Interface to perform integer division	25-26
10	4/11/24	Inter process Communication and Deadlock	27-31

Program 1

Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c = 0$. Read in a, b, c and use the quadratic formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions.

Code:

```
import java.io.*;
import java.util.Scanner;
import java.lang.Math;
class quadEq{
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the co-efficients:");
        int a=sc.nextInt();
        int
        b=sc.nextInt();
        int c=sc.nextInt();
        if(a==0){
            System.out.println("Invalid input");
        }
        else{
            double
            d,r1,r2;
            d=b*b-4*a*c;
            if(d>0)
            {
                r1=(-b+Math.sqrt(d))/(2*a);
                r2=(-b-Math.sqrt(d))/(2*a);
                System.out.println("The real roots are r1="+r1+"and r2="+r2);
            }
            else if(d==0)
            {
                r1=-b/(2*a);
                System.out.println("The real and equal roots are r1=r2="+r1);
            }
        }
    }
}
```

```

    }
    else
    {
        System.out.println("The roots are imaginary");
        r1=Math.sqrt(Math.abs(d))/(2*a);
        r2=-b/(2*a);
        System.out.println("The roots r1="+r2+"+i"+r1+"and r2="+r2+"-i"+r1);
    }
}
}
}
}

```

output:

```

C:\Users\Dell\Desktop>java Quadratic
Enter the first coefficient:
5
Enter the second coefficient:
1
Enter the third coefficient:
6
Roots are:-2.5+1.0908712114635715i
-2.5-1.0908712114635715i

```

Program 2

Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

Code:

```
import java.util.*;
class Student {
    String usn;
    String name;
    int[] credits;
    int[] marks;

    Student(int numSubjects) {
        credits = new int[numSubjects];
        marks = new int[numSubjects];
    }

    public static void main(String[] args) { Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of students (minimum 2): ");
        int numStudents = sc.nextInt();
        while (numStudents < 2) {
            System.out.print("Please enter at least 2 students: "); numStudents = sc.nextInt();
        }

        System.out.print("Enter the number of subjects: "); int numSubjects = sc.nextInt();
        sc.nextLine(); // Consume newline
        Student[] students = new Student[numStudents];
        for (int i = 0; i < numStudents; i++) {
            System.out.println("\nEnter details for student " + (i + 1) + ":");
            students[i] = new Student(numSubjects);
            System.out.print("Enter USN: ");

            students[i].usn = sc.nextLine();

            System.out.print("Enter Name: "); students[i].name = sc.nextLine();
            for (int j = 0; j < numSubjects; j++) {
                System.out.print("Enter credits for subject " + (j + 1) + ": ");
                students[i].credits[j] = sc.nextInt();
                System.out.print("Enter marks for subject " + (j + 1) + ": ");
```

```
students[i].marks[j] = sc.nextInt();
}
sc.nextLine(); // Consume newline after reading marks
}
```

Output:

```
C:\Users\Dell\Desktop>java Student
Enter the number of students (minimum 2): 2
Enter the number of subjects: 4

Enter details for student 1:
Enter USN: 77
Enter Name: John
Enter credits for subject 1: 3
Enter marks for subject 1: 87
Enter credits for subject 2: 4
Enter marks for subject 2: 90
Enter credits for subject 3: 2
Enter marks for subject 3: 70
Enter credits for subject 4: 3
Enter marks for subject 4: 66

Enter details for student 2:
Enter USN: 80
Enter Name: Austin
Enter credits for subject 1: 3
Enter marks for subject 1: 80
Enter credits for subject 2: 4
Enter marks for subject 2: 92
Enter credits for subject 3: 2
Enter marks for subject 3: 77
Enter credits for subject 4: 3
Enter marks for subject 4: 80

Student 1 Details:
USN: 77
Name: John
Subject 1 - Credits: 3, Marks: 87
Subject 2 - Credits: 4, Marks: 90
Subject 3 - Credits: 2, Marks: 70
Subject 4 - Credits: 3, Marks: 66

SGPA for student 1: 8.67
```

Student 2 Details:

USN: 80

Name: Austin

Subject 1 - Credits: 3, Marks: 80

Subject 2 - Credits: 4, Marks: 92

Subject 3 - Credits: 2, Marks: 77

Subject 4 - Credits: 3, Marks: 80

SGPA for student 2: 9.17

Program 3

Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java program to create n book objects.

code:

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

```
class Book {
```

```
    private String name;  
    private String author;  
    private double price;  
    private int numPages;
```

```
    Book(String name, String author, double price, int numPages) {  
        this.name = name;  
        this.author = author;  
        this.price = price;  
        this.numPages = numPages;  
    }
```

```
    void setName(String name) {  
        this.name = name;  
    }
```

```
    void setAuthor(String author) {  
        this.author = author;  
    }
```

```
    void setPrice(double price) {  
        this.price = price;  
    }
```

```
    void setNumPages(int numPages) {  
        this.numPages = numPages;  
    }
```

```
    String getName() {  
        return name;  
    }
```

```
    String getAuthor() {  
        return author;}  
}
```

```

Double getPrice() { return
price;
}

int getNumPages() {
    return numPages;
}

@Override
public String toString() {
    return "Book [name=" + name + ", author=" + author + ", price=" + price + ", No. of pages=" +
numPages + "]";
}
}

class BookDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of books:");
        int n = scanner.nextInt();
        scanner.nextLine();

        Book[] books = new Book[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter the details for book " + (i + 1) + ":");
            System.out.println("Name:");
            String name = scanner.nextLine();
            System.out.println("Author:");
            String author = scanner.nextLine();
            System.out.println("Price:");
            double price = scanner.nextDouble();
            System.out.println("Number of pages:");
            int numPages = scanner.nextInt();
            scanner.nextLine();

            books[i] = new Book(name, author, price, numPages);
        }

        for (Book book : books) {
            System.out.println(book);
        }

    }
}

```

Output:

```
C:\Users\Dell\Desktop>java BookDemo
Enter the no. of books
2
Enter the details for book1:
Name:
Harry Potter
Author:
J.K Rowling
Price:
500
No. of pages:
800
Enter the details for book2:
Name:
Lord of the Rings
Author:
JRR Tolkien
Price:
1000
No. of pages:
900
Book [name= Harry Potter author= J.K Rowling price= 500.0 No. of pages= 800]
Book [name= Lord of the Rings author= JRR Tolkien price= 1000.0 No. of pages= 900]
```

Program 4

Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide three classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

Code:

```
import java.util.*;

abstract class Shape {
    int dim1, dim2;

    Shape(int dim1, int dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }

    abstract void printArea()
class Rectangle extends Shape {
    Rectangle(int length, int breadth) {
        super(length, breadth);
    }

    @Override
    void printArea() {
        int area = dim1 * dim2;
        System.out.println("Area of Rectangle: " +
            area);
    }
}

class Triangle extends Shape {
    Triangle(int base, int height) {
        super(base, height);
    }

    @Override
    void printArea() {
        double area = 0.5 * dim1 * dim2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {
    Circle(int radius) {
        super(radius, 0);
    }
}
```

```

    }

    @Override
    void printArea() {
        double area = Math.PI * dim1 * dim1;
        System.out.println("Area of Circle: " + area);
    }
}

public class Shapes {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the length of the rectangle:");
        int len = scanner.nextInt();

        System.out.println("Enter the breadth of the rectangle:");
        int breth = scanner.nextInt()

        System.out.println("Enter the base of the triangle:"); int b =
        scanner.nextInt();

        System.out.println("Enter the height of the
        triangle:"); int h = scanner.nextInt();

        System.out.println("Enter the radius of the circle:");
        int r = scanner.nextInt();

        Shape rectangle = new Rectangle(len, breth);
        Shape triangle = new Triangle(b, h);
        Shape circle = new Circle(r);

        rectangle.printArea();
        triangle.printArea();
        circle.printArea();
    }
}

```

```

C:\Users\Dell\Desktop>javac Shapes.java
C:\Users\Dell\Desktop>java Shapes
Enter the length of the rectangle
4
Enter the breadth of the rectangle
5
Enter the base of the triangle
12
Enter the height of the triangle
9
Enter the radius of the circle
7
Area of Rectangle: 20
Area of Triangle: 54.0
Area of Circle: 153.93804002589985
}

```

Program 5

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

- a) Accept deposit from customer and update the balance.
- b) Display the balance.
- c) Compute and deposit interest
- d) Permit withdrawal and update the balance

Check for the minimum balance, impose penalty if necessary and update the balance.

Code:

```
import java.util.*;

class Account {
    String customerName;
    String accountNumber;
    double balance;
    String accountType;

    Account(String customerName, String accountNumber, String accountType) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = 0.0;
    }

    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount);
    }

    double getBalance() {
        return balance;
    }

    void displayBalance() {
        System.out.println("Current balance: " + balance);
    }
}
```

```

class SavingsAccount extends Account {
    private double interestRate;

    SavingsAccount(String customerName, String accountNumber, double interestRate) {
        super(customerName, accountNumber, "Savings");
        this.interestRate = interestRate;
    }

    void computeAndDepositInterest() {
        double interest = balance * interestRate / 100;
        deposit(interest);
        System.out.println("Interest of " + interest + " deposited.");
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance for withdrawal.");
        }
    }
}

```

```

class CurrentAccount extends Account {
    private static final double MINIMUM_BALANCE = 1000.0;
    private static final double SERVICE_CHARGE = 50.0;

    CurrentAccount(String customerName, String accountNumber) {
        super(customerName, accountNumber, "Current");
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance for withdrawal.");
        }
        checkMinimumBalance();
    }

    void checkMinimumBalance() {
        if (balance < MINIMUM_BALANCE) {
            balance -= SERVICE_CHARGE;
            System.out.println("Minimum balance not maintained. Service charge of " +

```

```
}  
    }  
}
```

```
public class Bank {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        System.out.println("Welcome to the Bank!");  
  
        System.out.print("Enter customer name for Savings Account: ");  
        String savCustomerName = scanner.nextLine();  
        System.out.print("Enter account number for Savings Account: ");  
        String savAccountNumber = scanner.nextLine();  
        System.out.print("Enter interest rate for Savings Account: ");  
        double interestRate = scanner.nextDouble();  
  
        SavingsAccount savingsAccount = new SavingsAccount(savCustomerName,  
savAccountNumber, interestRate);  
  
        scanner.nextLine(); // Consume newline  
        System.out.print("Enter customer name for Current Account:  
"); String currCustomerName = scanner.nextLine();  
        System.out.print("Enter account number for Current Account:  
"); String currAccountNumber = scanner.nextLine();  
  
        CurrentAccount currentAccount = new CurrentAccount(currCustomerName,  
currAccountNumber);  
  
        int i = 1;  
        while (i == 1) {  
            System.out.println("\n1. Deposit into Savings account");  
            System.out.println("2. Deposit into Current account");  
            System.out.println("3. Deposit interest");  
            System.out.println("4. Withdraw from Savings account");  
            System.out.println("5. Withdraw from Current account");  
            System.out.println("6. Display balance in Savings account");  
            System.out.println("7. Display balance in Current account");  
            System.out.println("8. Exit");  
            System.out.print("Enter your choice: ");  
            int choice = scanner.nextInt();  
  
            switch (choice) {  
                case 1:  
                    System.out.print("Enter amount to be deposited: ");  
                    double amt = scanner.nextDouble();  
                    savingsAccount.deposit(amt);  
                    break;
```



```
case 2:
    System.out.print("Enter amount to be deposited: ");
    double a = scanner.nextDouble();
    currentAccount.deposit(a);
    break;
case 3:
    savingsAccount.computeAndDepositInterest();
    break;
case 4:
    System.out.print("Enter amount to be withdrawn: ");
    double am = scanner.nextDouble();
    savingsAccount.withdraw(am);
    break;
case 5:
    System.out.print("Enter amount to be withdrawn: ");
    double amot = scanner.nextDouble();
    currentAccount.withdraw(amot);
    break;
case 6:
    savingsAccount.displayBalance();
    break;
case 7:
    currentAccount.displayBalance();
    break;
case 8:
    i = 0;
    break;
default:
    System.out.println("Invalid choice.");
    break;
}
}

}
```

Output:

```
C:\Users\Dell\Desktop>javac Bank.java

C:\Users\Dell\Desktop>java Bank
Welcome to the Bank!
Enter customer name for Savings Account: David
Enter account number for Savings Account: 1
Enter interest rate for Savings Account: 5
Enter customer name for Current Account: Henry
Enter account number for Current Account: 2
1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit
Enter the choice
1
Enter amount to be deposited
500
Deposited: 500.0
1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit
Enter the choice
2
Enter amount to be deposited
3000
Deposited: 3000.0
1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
```

```
Enter amount to be deposited
3000
Deposited: 3000.0
1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit
Enter the choice
3
Deposited: 25.0
Interest of 25.0 deposited.
1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit
Enter the choice
5
Enter amount to be withdrawn
2500
Withdrawn: 2500.0
Minimum balance not maintained. Service charge of 50.0 imposed.
1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit
Enter the choice
6
Current balance: 525.0
```

Enter the choice

6

Current balance: 525.0

1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit

Enter the choice

7

Current balance: 450.0

1. Deposit into Savings account
2. Deposit into Current account
3. Deposit interest
4. Withdraw from Savings account
5. Withdraw from Current account
6. Display balance in Savings account
7. Display balance in Current account
8. Exit

Enter the choice

8

Program 6

Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

Code:

```
//Internals.java
package cie;
public class Internals {
    public int[] internalMarks = new int[5];
    public Internals(int[] marks) {
        for (int i = 0; i < 5; i++) {
            internalMarks[i] = marks[i];
        }
    }
}

//Student.java
package cie;
public class Student {
    public String usn;
    public String name;
    public int sem;
    public Student(String usn, String name, int sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    }
}

//Externals.java
package see;
import cie.Student;
public class External extends Student
{
```

```

public External(String usn, String name, int sem, int seeMarks[])
{
    super(usn, name, sem);
    for (int i = 0; i < 5; i++)
    {
        this.seeMarks[i] = seeMarks[i];
    }
}
}
//Main.java
import cie.Internals;
import see.External;
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of students: ");
        int n = sc.nextInt();

        for (int i=0;i<n;i++)
        {
            System.out.println("\nEnter details for the "+(i + 1)+" Student " + " :");

            sc.nextLine();
            System.out.print("USN: ");
            String usn = sc.nextLine();
            System.out.print("Name: ");
            String name = sc.nextLine();
            System.out.print("Semester: ");
            int sem = sc.nextInt();

            System.out.println("Enter internal marks for 5 courses (out of 50): ");
            int internalMarks[] = new int[5];
            for (int j=0;j<5;j++)
            {
                System.out.print("Course " + (j + 1) + " internal marks: ");
                internalMarks[j] = sc.nextInt();
            }
        }
    }
}

```

```

Internals internal = new Internals(internalMarks);

System.out.println("Enter SEE marks for 5 courses (out of 100): ");
int seeMarks[] = new int[5];
for (int j=0;j<5;j++)
{
    System.out.print("Course " + (j + 1) + " SEE marks: ");
    seeMarks[j] = sc.nextInt();
}
External external = new External(usn, name, sem, seeMarks);

System.out.println("\nFinal Marks for " + external.name + " : " + external.usn + ":for:");
System.out.println("Semester: " + external.sem);
System.out.println("Total Marks:");
for (int j=0;j<5;j++)
{
    System.out.println("Course " + (j + 1) + ": " + (double)(internal.internalMarks[j] +
    (external.seeMarks[j]/2)));
}
}
}
}
}

```

Output:

```

Enter internal marks for 5 courses (out of 50):
Course 1 internal marks: 35
Course 2 internal marks: 37
Course 3 internal marks: 38
Course 4 internal marks: 36
Course 5 internal marks: 39
Enter SEE marks for 5 courses (out of 100):
Course 1 SEE marks: 89
Course 2 SEE marks: 90
Course 3 SEE marks: 91
Course 4 SEE marks: 92
Course 5 SEE marks: 93

```

```
Semester: 3
Total Marks:
Course 1: 79.0
Course 2: 82.0
Course 3: 83.0
Course 4: 82.0
Course 5: 85.0
```


Program 7

Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception Wrong Age() when the input age<0. In Son class, implement a constructor that cases both father and son’s age and throws an exception if son’s age is >=father’s age.

Code:

```
import java.util.*;

class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message);
    }
}

class Father {
    int age;

    public Father(int age) throws WrongAge {
        if (age < 0) {
            throw new WrongAge("Age of the father cannot be negative");
        }
        this.age = age;
        System.out.println("The age of the Father is " + this.age);
    }
}
```

```
}
```

```
class Son extends Father {  
    int sage;  
  
    public Son(int age, int sage) throws WrongAge {  
        super(age);  
        if (sage >= age) {  
            throw new WrongAge("Son cannot be older than the father");  
        }  
        this.sage = sage;  
        System.out.println("The son's age is " + this.sage);  
    }  
}
```

```
public class Check {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        try {  
            System.out.println("Enter the age of the Father: ");  
            int a = sc.nextInt();  
            Father f = new Father(a);  
        } catch (WrongAge e) {  
            System.out.println(e.getMessage());  
            ;  
        }  
        try {  
            System.out.println("Enter the father's age:  
"); int b = sc.nextInt();  
            System.out.println("Enter the son's age: ");  
            int c = sc.nextInt();  
            Son s = new Son(b, c);  
        } catch (WrongAge e) {  
            System.out.println(e.getMessage());  
            ;  
        }  
    }  
}
```

Output:

```
C:\Users\Dell\Desktop>javac Check.java
C:\Users\Dell\Desktop>java Check
Enter the age of the Father:
-5
Age of the father cannot be negative
Enter the father's age:
40
Enter the son's age:
45
The age of the Father is 40
Son cannot be older than the father
```

Program 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.

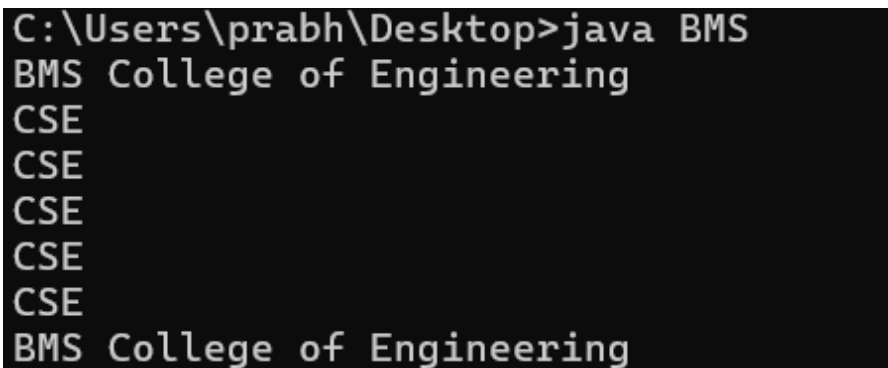
Code:

```
class MyThread extends Thread {
    String s;
    int n;

    public MyThread(String s, int n) {
        this.s = s;
        this.n = n;
    }

    public void run() {
        while (true) {
            System.out.println(s);
            try {
                Thread.sleep(n);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}

public class BMS {
    public static void main(String[] args) {
        MyThread t1 = new MyThread("BMS College of Engineering", 10000);
        MyThread t2 = new MyThread("CSE", 2000);
        t1.start();
        t2.start();
    }
}
```



```
C:\Users\prabh\Desktop>java BMS
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
```

Program 9

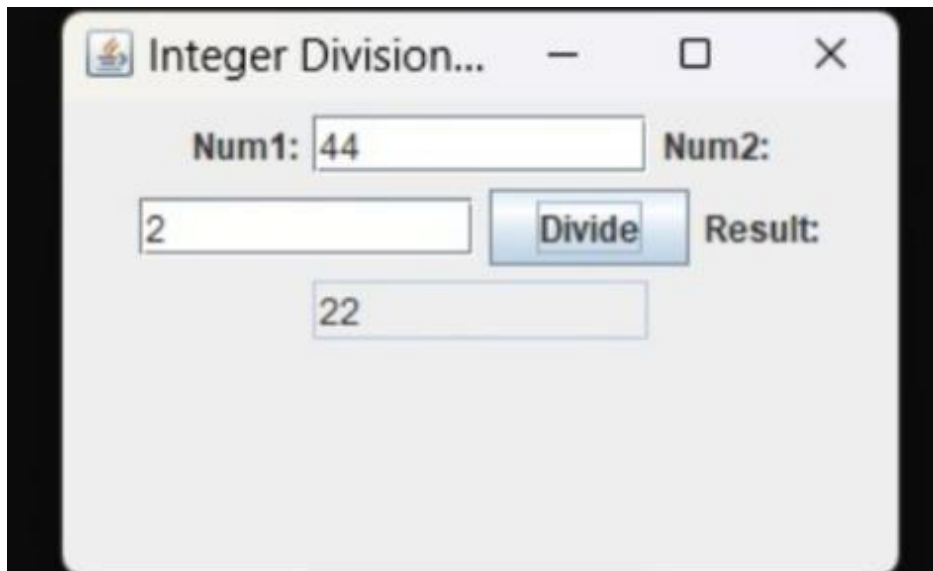
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a Number Format Exception. If Num2 were Zero, the program would throw an Arithmetic Exception Display the exception in a message dialog box.

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
class DivisionCalculator {
public static void main(String[] args) {
    JFrame frame = new JFrame("Integer Division Calculator");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 200);
    frame.setLayout(new GridLayout(4, 2, 10, 10));
    JLabel num1Label = new JLabel("Num1:");
    JTextField num1Field = new JTextField();
    JLabel num2Label = new JLabel("Num2:");
    JTextField num2Field = new JTextField();
    JLabel resultLabel = new JLabel("Result:");
    JTextField resultField = new JTextField();
    resultField.setEditable(false);
    JButton divideButton = new JButton("Divide");
    divideButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                int num1 = Integer.parseInt(num1Field.getText());
                int num2 = Integer.parseInt(num2Field.getText());
                int result = num1 / num2;
                resultField.setText(String.valueOf(result));
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(frame, "Please enter valid integers
                for Num1 and Num2.", "Input Error", JOptionPane.ERROR_MESSAGE);
            } catch (ArithmeticException ex) {
```

```
JOptionPane.showMessageDialog(frame, "Division by zero is not  
allowed.", "Arithmetic Error", JOptionPane.ERROR_MESSAGE);  
}  
}  
});  
frame.add(num1Label);  
frame.add(num1Field);  
frame.add(num2Label);  
frame.add(num2Field);  
frame.add(resultLabel);  
frame.add(resultField);  
frame.add(new JLabel());  
frame.add(divideButton);  
frame.setVisible(true);  
}  
}
```

Output:



Program 10.1

Demonstrate Inter process Communication and deadlock

Code:

```
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;
    }
}
```

```

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:

```
Press Control-C to stop.  
Put: 0  
Got: 0  
Put: 1  
Got: 1  
Put: 2  
Got: 2  
Put: 3  
Got: 3  
Put: 4  
Got: 4  
Put: 5  
Got: 5  
Put: 6  
Got: 6  
Put: 7  
Got: 7  
Put: 8  
Got: 8  
Put: 9  
Got: 9  
Put: 10  
Got: 10
```

Program 10.2

```
class AA {
    synchronized void foo(BB 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 BB {
    synchronized void bar(AA a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered BB.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 {
    AA a = new AA();
    BB    b = new
    BB(); 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();
    }
}

```

Output:

```

RacingThread entered BB.bar
MainThread entered A.foo
MainThread trying to call B.last()
RacingThread trying to call A.last()
^C

```