VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Vijeth M D(1WA23CS041)

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 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 **Vijeth M D(1WA23CS041),** 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. 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

Program 1

Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2-4ac 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 quad
{
public static void main(String args[])
{
System.out.println(" Enter coefficients of the quadratic Equation : ");
Scanner x = new Scanner(System.in);
int a = x.nextInt();
int b = x.nextInt();
int c = x.nextInt();
double d = b*b - 4*a*c;
double r1,r2;
if(d>0)
{
System.out.println(" Roots are real and distinct ");
r1 = (-b + Math.sqrt(d))/(2*a);
r2 = (-b - Math.sqrt(d))/(2*a);
System.out.println(r1 + " " + r2);
}
else if(d == 0)
{
System.out.println(" Roots are real and equal ");
r1 = -b/(2.0*a);
r2 = -b/(2.0*a);
System.out.println(r1 + " " + r2);
}
else
{
System.out.println(" Roots are Imaginary ");

double p = Math.abs(d);
System.out.println(-b/(2.0*a) + " + " + "i" + Math.sqrt(p)/(2*a));
System.out.println(-b/(2.0*a) + " - " + "i" + Math.sqrt(p)/(2*a));
}
}
}
```

output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac quad.java

C:\Users\Lenovo\Desktop\Java_Github>java quad
 Enter coefficients of the quadratic Equation :
2 3 6
 Roots are Imaginary
-0.75 + i1.5612494995995996
-0.75 - i1.5612494995995996

C:\Users\Lenovo\Desktop\Java_Github>javac quad.java

C:\Users\Lenovo\Desktop\Java_Github>java quad
 Enter coefficients of the quadratic Equation :
1 -9 10
 Roots are real and distinct
7.701562118716424 1.2984378812835757

C:\Users\Lenovo\Desktop\Java_Github>java quad
 Enter coefficients of the quadratic Equation :
1 -4 4
 Roots are real and equal
2.0 2.0
```

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.Scanner;

class Student {
    private String usn;
    private String name;
    private int[] credits;
    private int[] marks;

    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = scanner.nextLine();
        System.out.print("Enter Name: ");
        name = scanner.nextLine();
        System.out.print("Enter number of subjects: ");
        int n = scanner.nextInt();
        credits = new int[n];
        marks = new int[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            credits[i] = scanner.nextInt();
            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            marks[i] = scanner.nextInt();
        }
    }

    public void displayDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Subject-wise details:");
        for (int i = 0; i < credits.length; i++) {
            System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i] + ", Marks: " + marks[i]);
        }
    }

    public double calculateSGPA() {
        int totalCredits = 0;
        int weightedSum = 0;
        for (int i = 0; i < credits.length; i++) {
            int gradePoint = getGradePoint(marks[i]);
            weightedSum += gradePoint * credits[i];
            totalCredits += credits[i];
```

```
        }
        return (double) weightedSum / totalCredits;
    }

    private int getGradePoint(int marks) {
        if (marks >= 90) return 10;
        if (marks >= 80) return 9;
        if (marks >= 70) return 8;
        if (marks >= 60) return 7;
        if (marks >= 50) return 6;
        if (marks >= 40) return 5;
        return 0;
    }
}

public class Cal {
    public static void main(String[] args) {
        Student student = new Student();
        student.acceptDetails();
        student.displayDetails();
        System.out.printf("SGPA: %.2f\n", student.calculateSGPA());
    }
}
```

output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac Cal.java

C:\Users\Lenovo\Desktop\Java_Github>java Cal
Enter USN: 1WA23CS001
Enter Name: Vineeth
Enter number of subjects: 5
Enter credits for subject 1: 4
Enter marks for subject 1: 96
Enter credits for subject 2: 4
Enter marks for subject 2: 84
Enter credits for subject 3: 3
Enter marks for subject 3: 90
Enter credits for subject 4: 2
Enter marks for subject 4: 76
Enter credits for subject 5: 1
Enter marks for subject 5: 94
USN: 1WA23CS001
Name: Vineeth
Subject-wise details:
Subject 1 - Credits: 4, Marks: 96
Subject 2 - Credits: 4, Marks: 84
Subject 3 - Credits: 3, Marks: 90
Subject 4 - Credits: 2, Marks: 76
Subject 5 - Credits: 1, Marks: 94
SGPA: 9.43
```

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:

```
class Book {
    private String name;
    private String author;
    private double price;
    private int num_pages;

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

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


    @Override
    public String toString() {
        return "Book{name='" + name + '\'' + ", author='" + author + '\'' + ", price=" + price + ",
num_pages=" + num_pages + '}';
    }
}

public class BookManagement {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("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 details for book " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = scanner.nextLine();
            System.out.print("Author: ");
            String author = scanner.nextLine();
```

```
            System.out.print("Price: ");
            double price = scanner.nextDouble();
            System.out.print("Number of pages: ");
            int num_pages = scanner.nextInt();
            scanner.nextLine();
            books[i] = new Book(name, author, price, num_pages);
        }

        System.out.println("Details of books:");
        for (Book book : books) {
            System.out.println(book);
        }
    }
}
```

Output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac BookManagement.java

C:\Users\Lenovo\Desktop\Java_Github>java BookManagement
Enter the number of books: 3
Enter details for book 1:
Name: Wings Of Fire
Author: APJ Abdul Kalam
Price: 500
Number of pages: 900
Enter details for book 2:
Name: Attack on Titan
Author: Isayama
Price: 300
Number of pages: 1000
Enter details for book 3:
Name: Thor
Author: Subhash
Price: 250
Number of pages: 400
Details of books:
Book{name='Wings Of Fire', author='APJ Abdul Kalam', price=500.0, num_pages=900}
Book{name='Attack on Titan', author='Isayama', price=300.0, num_pages=1000}
Book{name='Thor', author='Subhash', price=250.0, num_pages=400}
```

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:

```
abstract class Shape {
    protected int dimension1;
    protected int dimension2;

    public Shape(int dimension1, int dimension2) {
        this.dimension1 = dimension1;
        this.dimension2 = dimension2;
    }

    public abstract void printArea();
}

class Rectangle extends Shape {

    public Rectangle(int length, int breadth) {
        super(length, breadth);
    }

    @Override
    public void printArea() {
        System.out.println("Area of Rectangle: " + (dimension1 * dimension2));
    }
}

class Triangle extends Shape {

    public Triangle(int base, int height) {
        super(base, height);
    }

    @Override
    public void printArea() {
        System.out.println("Area of Triangle: " + (0.5 * dimension1 * dimension2));
    }
}

class Circle extends Shape {
    private static final double PI = 3.14159;
```

```
    public Circle(int radius) {
        super(radius, 0);
    }

    @Override
    public void printArea() {
        System.out.println("Area of Circle: " + (PI * dimension1 * dimension1));
    }
}

public class ShapeExecute {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle(10, 5);
        rectangle.printArea();

        Shape triangle = new Triangle(10, 5);
        triangle.printArea();

        Shape circle = new Circle(7);
        circle.printArea();
    }
}
```

Output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac ShapeExecute.java

C:\Users\Lenovo\Desktop\Java_Github>java ShapeExecute
Area of Rectangle: 50
Area of Triangle: 25.0
Area of Circle: 153.93791

C:\Users\Lenovo\Desktop\Java_Github>
```

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.Scanner;

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

    public Account(String customerName, String accountNumber, String accountType, double
initialDeposit) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = initialDeposit;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposit successful. Updated balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

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

```
class SavAcct extends Account {
    private static final double INTEREST_RATE = 0.04; // 4% annual interest

    public SavAcct(String customerName, String accountNumber, double initialDeposit) {
        super(customerName, accountNumber, "Savings", initialDeposit);
    }

    public void computeAndDepositInterest() {
        double interest = balance * INTEREST_RATE;
        balance += interest;
        System.out.println("Interest of " + interest + " has been added. Updated balance: " + balance);
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient funds. Withdrawal denied.");
        } else {
            balance -= amount;
            System.out.println("Withdrawal successful. Updated balance: " + balance);
        }
    }
}

class CurAcct extends Account {
    private static final double MIN_BALANCE = 1000.0;
    private static final double SERVICE_CHARGE = 100.0;

    public CurAcct(String customerName, String accountNumber, double initialDeposit) {
        super(customerName, accountNumber, "Current", initialDeposit);
    }

    public void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient funds. Withdrawal denied.");
        } else {
            balance -= amount;
            if (balance < MIN_BALANCE) {
                balance -= SERVICE_CHARGE;
                System.out.println("Minimum balance not maintained. Service charge of " +
SERVICE_CHARGE + " imposed.");
            }
            System.out.println("Withdrawal successful. Updated balance: " + balance);
        }
    }
}
```

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

        System.out.println("Enter customer name: ");
        String name = scanner.nextLine();

        System.out.println("Enter account number: ");
        String accNumber = scanner.nextLine();

        System.out.println("Enter account type (Savings/Current): ");
        String accType = scanner.nextLine();

        System.out.println("Enter initial deposit amount: ");
        double initialDeposit = scanner.nextDouble();

        Account account;
        if (accType.equalsIgnoreCase("Savings")) {
            account = new SavAcct(name, accNumber, initialDeposit);
        } else if (accType.equalsIgnoreCase("Current")) {
            account = new CurAcct(name, accNumber, initialDeposit);
        } else {
            System.out.println("Invalid account type.");
            scanner.close();
            return;
        }

        while (true) {
            System.out.println("\nChoose an option:");
            System.out.println("1. Deposit");
            System.out.println("2. Display Balance");
            System.out.println("3. Withdraw");
            if (account instanceof SavAcct) {
                System.out.println("4. Compute and Deposit Interest");
            }
            System.out.println("5. Exit");

            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.println("Enter amount to deposit: ");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;
                case 2:
                    account.displayBalance();
                    break;
```

```
                case 3:
                    System.out.println("Enter amount to withdraw: ");
                    double withdrawAmount = scanner.nextDouble();
                    if (account instanceof SavAcct) {
                        ((SavAcct) account).withdraw(withdrawAmount);
                    } else if (account instanceof CurAcct) {
                        ((CurAcct) account).withdraw(withdrawAmount);
                    }
                    break;
                case 4:
                    if (account instanceof SavAcct) {
                        ((SavAcct) account).computeAndDepositInterest();
                    } else {
                        System.out.println("Invalid option for Current account.");
                    }
                    break;
                case 5:
                    System.out.println("Exiting... Thank you for banking with us.");
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    }
}
```

Output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac Bank.java

C:\Users\Lenovo\Desktop\Java_Github>java Bank
Enter customer name:
Naveen
Enter account number:
12345
Enter account type (Savings/Current):
Savings
Enter initial deposit amount:
4000

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
5. Exit
1
Enter amount to deposit:
7000
Deposit successful. Updated balance: 11000.0

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
5. Exit
3
Enter amount to withdraw:
40000
Insufficient funds. Withdrawal denied.

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
5. Exit
```

```
Insufficient funds. Withdrawal denied.

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
5. Exit
3
Enter amount to withdraw:
10500
Withdrawal successful. Updated balance: 500.0

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
5. Exit
Exception in thread "main" ^C
C:\Users\Lenovo\Desktop\Java_Github>javac Bank.java

C:\Users\Lenovo\Desktop\Java_Github>java Bank
Enter customer name:
Karthik
Enter account number:
23456
Enter account type (Savings/Current):
Current
Enter initial deposit amount:
5000

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
5. Exit
3
Enter amount to withdraw:
4500
Minimum balance not maintained. Service charge of 100.0 imposed.
```

```
5. Exit
Exception in thread "main" ^C
C:\Users\Lenovo\Desktop\Java_Github>javac Bank.java

C:\Users\Lenovo\Desktop\Java_Github>java Bank
Enter customer name:
Karthik
Enter account number:
23456
Enter account type (Savings/Current):
Current
Enter initial deposit amount:
5000

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
5. Exit
3
Enter amount to withdraw:
4500
Minimum balance not maintained. Service charge of 100.0 imposed.
Withdrawal successful. Updated balance: 400.0

Choose an option:
1. Deposit
2. Display Balance
3. Withdraw
5. Exit
```

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 int[] seeMarks = new int[5];
   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:

```
C:\Users\Lenovo\Desktop\PackageFolder>javac CIE/Internals.java

C:\Users\Lenovo\Desktop\PackageFolder>javac CIE/Student.java

C:\Users\Lenovo\Desktop\PackageFolder>javac SEE/External.java

C:\Users\Lenovo\Desktop\PackageFolder>javac Main.java
```

```
C:\Users\Lenovo\Desktop\PackageFolder>java Main
Enter the number of students: 2

Enter details for the 1 Student :
USN: 1WA23CS001
Name: Vineeth
Semester: 3
Enter internal marks for 5 courses (out of 50):
Course 1 internal marks: 48
Course 2 internal marks: 46
Course 3 internal marks: 38
Course 4 internal marks: 43
Course 5 internal marks: 35
Enter SEE marks for 5 courses (out of 100):
Course 1 SEE marks: 90
Course 2 SEE marks: 87
Course 3 SEE marks: 76
Course 4 SEE marks: 90
Course 5 SEE marks: 99

Final Marks for Vineeth :1WA23CS001:for:
Semester: 3
Total Marks:
Course 1: 93.0
Course 2: 89.0
Course 3: 76.0
Course 4: 88.0
Course 5: 84.0

Enter details for the 2 Student :
USN: 1WA23CS002
Name: Varun
Semester: 3
Enter internal marks for 5 courses (out of 50):
Course 1 internal marks: 49
Course 2 internal marks: 43
Course 3 internal marks: 38
Course 4 internal marks: 36
Course 5 internal marks: 33
Enter SEE marks for 5 courses (out of 100):
```

```
Final Marks for Vineeth :1WA23CS001:for:
Semester: 3
Total Marks:
Course 1: 93.0
Course 2: 89.0
Course 3: 76.0
Course 4: 88.0
Course 5: 84.0

Enter details for the 2 Student :
USN: 1WA23CS002
Name: Varun
Semester: 3
Enter internal marks for 5 courses (out of 50):
Course 1 internal marks: 49
Course 2 internal marks: 43
Course 3 internal marks: 38
Course 4 internal marks: 36
Course 5 internal marks: 33
Enter SEE marks for 5 courses (out of 100):
Course 1 SEE marks: 90
Course 2 SEE marks: 78
Course 3 SEE marks: 86
Course 4 SEE marks: 68
Course 5 SEE marks: 99

Final Marks for Varun :1WA23CS002:for:
Semester: 3
Total Marks:
Course 1: 94.0
Course 2: 82.0
Course 3: 81.0
Course 4: 70.0
Course 5: 82.0
```

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.io.*;
class WrongAge extends Exception {
   public WrongAge(String txt) {
      super(txt);
   }
}
class Father {
   private int age;
   public Father(int age) throws WrongAge {
      if (age < 0) {
         throw new WrongAge("Age cannot be negative.");
      }
      this.age = age;
   }
   public int getAge() {
      return age;
   }
}
class Son extends Father {
   private int sonAge;
   public Son(int fatherAge, int sonAge) throws WrongAge {
      super(fatherAge);
      if (sonAge < 0) {
         throw new WrongAge("Son's age cannot be negative.");
      }
      if (sonAge >= fatherAge) {
         throw new WrongAge("Son's age must be less than father's age.");
      }
      this.sonAge = sonAge;
   }
   public int getSonAge() {
      return sonAge;
   }
```

```
}
class Main {
   public static void main(String[] args) {
      try {
         Father father = new Father(40);
         Son s1 = new Son(40, 20);
         System.out.println("Father's Age: " + father.getAge());
         System.out.println("Son's Age: " + s1.getSonAge());
      } catch (WrongAge e) {
         System.out.println(e);
      }
      try {
         Son s2 = new Son(-5, 10);
      } catch (WrongAge e) {
         System.out.println(e);
      }
      try {
         Son s3 = new Son(12, -5);
      } catch (WrongAge e) {
         System.out.println(e);
      }
   }
}
```

Output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac Lab7.java

C:\Users\Lenovo\Desktop\Java_Github>java Lab7
Father's Age: 40
Son's Age: 20
WrongAge: Age cannot be negative.
WrongAge: Son's age cannot be negative.
```

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:

```
@SuppressWarnings("ALL")
class MyThread extends Thread {
final String message;
final int interval;
MyThread(String message, int interval) {
this.message = message;
this.interval = interval;
}

@Override
public void run() {
while (true) {
try {
System.out.println(message);
Thread.sleep(interval * 1000);
} catch (InterruptedException e) {
System.out.println(e.getMessage());
}
}
}
}

class Main {
public static void main(String[] args) {
MyThread thread1 = new MyThread("BMS College of Engineering",
10);
MyThread thread2 = new MyThread("CSE", 2);
thread1.start();
thread2.start();
}
}
```

Output:

```
C:\Users\Lenovo\Desktop\Java_Github>javac Main.java

C:\Users\Lenovo\Desktop\Java_Github>java Main
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
```

Program 9

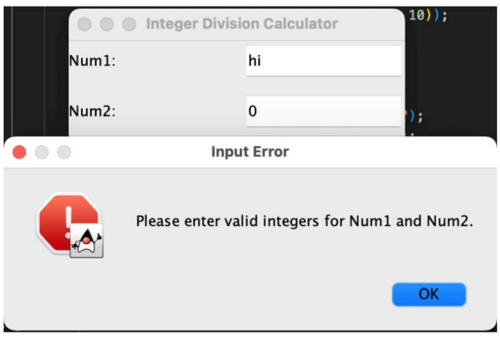
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a Number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

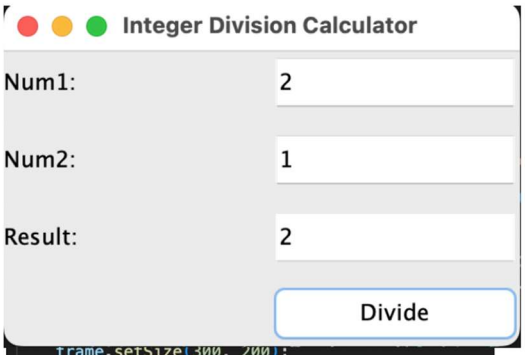
Code:

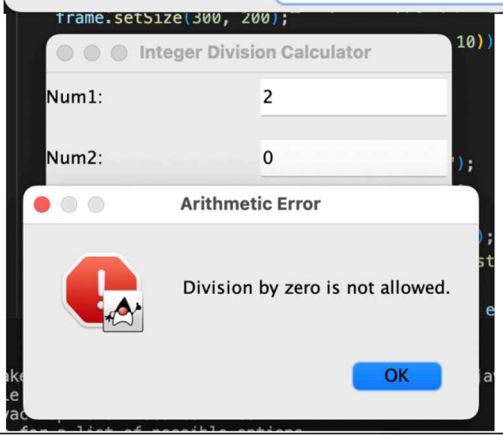
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
class DivisionCalculator {
public static void main(String[] args) {
JFrame frame = new JFrame("Integer Division Calculator");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(300, 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

Demonstrate Inter process Communication and deadlock

Code:

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

	synchronized void put(int n) {
		while(valueSet)
			try {
				wait();
			} catch(InterruptedException e) {
				System.out.println("InterruptedException caught");
		}
		this.n = n;
		valueSet = true;
```

```
            System.out.println("Put: " + 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(true) {
                q.put(i++);
            }
        }
}

class Consumer implements Runnable {
        Q q;
        Consumer(Q q) {
            this.q = q;
            new Thread(this, "Consumer").start();
        }
        public void run() {
            while(true) {
                q.get();
            }
```

```
        }
}

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:

```
C:\Users\Lenovo\Desktop\Java_Github>javac PCFixed.java

C:\Users\Lenovo\Desktop\Java_Github>java PCFixed
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
Put: 11
Got: 11
Put: 12
Got: 12
Put: 13
Got: 13
Put: 14
Got: 14
Put: 15
Got: 15
Put: 16
Got: 16
Put: 17
Got: 17
Put: 18
```

```
//Deadlock.java

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:

```
C:\Users\Lenovo\Desktop\Java_Github>javac Deadlock.java

C:\Users\Lenovo\Desktop\Java_Github>java Deadlock
RacingThread entered BB.bar
MainThread entered A.foo
MainThread trying to call B.last()
RacingThread trying to call A.last()
```