VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Anjali Patel (1BM23CS037)

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 **ANJALI PATEL (1BM23CS037),** 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.

Sheetal V A Assistant Professor, Mtech Department of CSE, BMSCE	Dr. Jyoti S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/1BM23CS037/JAVA--PROGRAMS

PROGRAM 1

Implement Quadratic Equation

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:

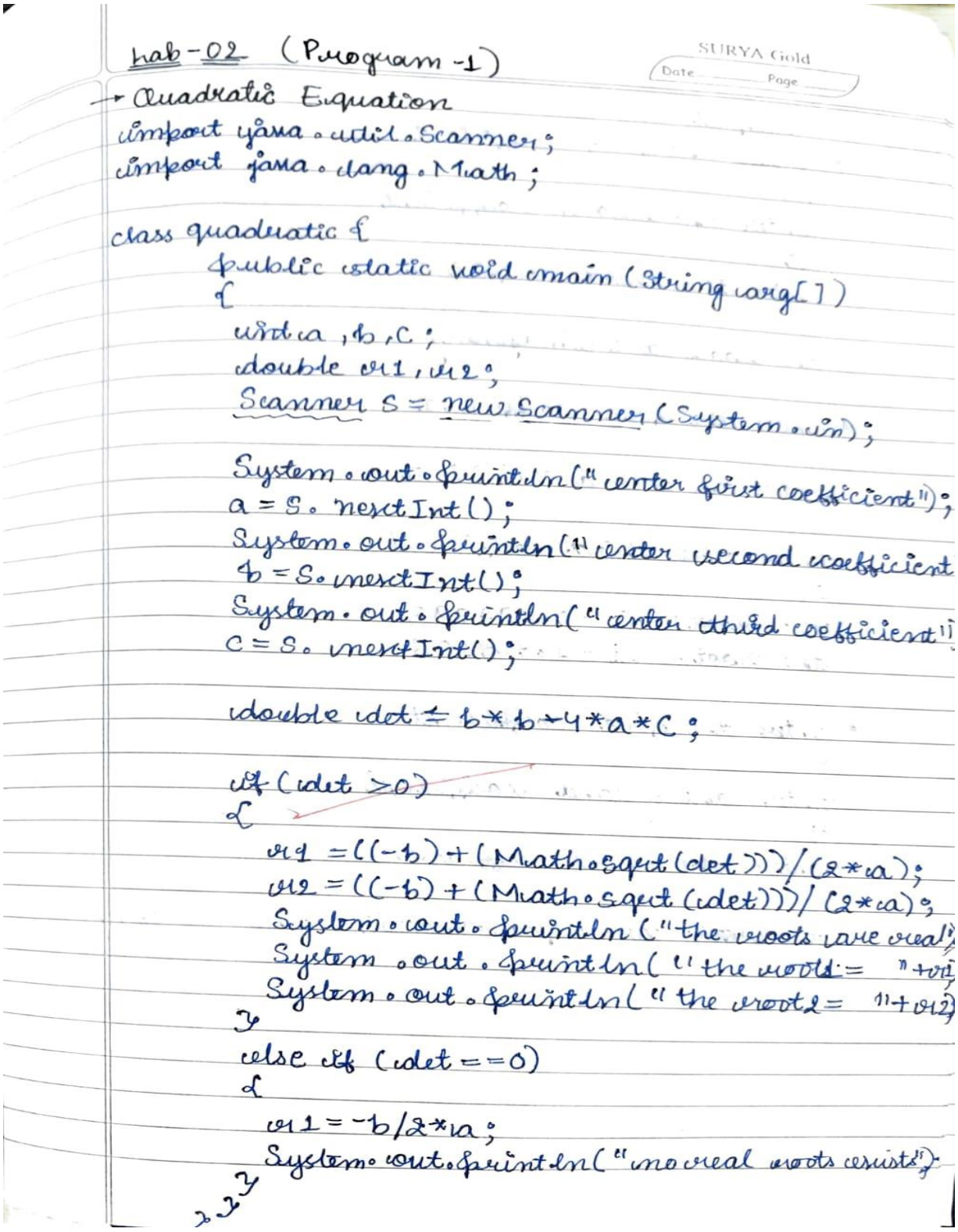
```
//ANJALI PATEL 1BM23CS037

import java.util.Scanner;
import java.lang.Math;

class quadratic
{
public static void main(String arg[])
{
int a ,b, c;
double r1 , r2;
Scanner s = new Scanner(System.in);

System.out.println("enter the first coefficient");
a= s.nextInt();

System.out.println("enter the second coefficient");
b=s.nextInt();

System.out.println("enter the third coefficient");
c= s.nextInt();

double det=b*b-4*a*c;

if(det>0)
{
r1= (-b +(Math.sqrt(det)))/(2*a);
r2= (-b -(Math.sqrt(det)))/(2*a);
System.out.println("the root1 is" +r1);
System.out.println("the root2 is" +r2);
}
else if(det==0)
{
r1 =-b/(2*a);
System.out.println("the roots are equal and its value is" +r1);
```

```
}
else
{
System.out.println("the roots are imaginary");
}
}
}
```

OUTPUT:

```
C:\Users\anjal\OneDrive\Desktop\codes>javac quadratic.java

C:\Users\anjal\OneDrive\Desktop\codes>java quadratic
enter the first coefficient
2
enter the second coefficient
4
enter the third coefficient
1
the root1 is-0.2928932188134524
the root2 is-1.7071067811865475

C:\Users\anjal\OneDrive\Desktop\codes>java quadratic
enter the first coefficient
2
enter the second coefficient
4
enter the third coefficient
2
the roots are equal and its value is-1.0

C:\Users\anjal\OneDrive\Desktop\codes>java quadratic
enter the first coefficient
10
enter the second coefficient
20
enter the third coefficient
30
the roots are imaginary
```

ALGORITHM:



Lab-02 (Program-1)

→ Quadratic Equation

```
import java.util.Scanner;
import java.lang.Math;

class quadratic {
    public static void main (String arg[])
    {
      int a, b, c;
      double r1, r2;
      Scanner s = new Scanner (System.in);

      System.out.println("enter first coefficient");
      a = s.nextInt();
      System.out.println("enter second coefficient
      b = s.nextInt();
      System.out.println("enter third coefficient
      c = s.nextInt();

      double det = b*b-4*a*c;

      if (det >0)
      {
        r1 =((-b)+(Math.sqrt(det)))/(2*a);
        r2 =((-b)+(Math.sqrt(det)))/(2*a);
        System.out.println("the roots are real");
        System.out.println("the root1 = "+r1);
        System.out.println("the root2 = "+r2);
      }
      else if (det ==0)
      {
        r1 = -b/2*a;
        System.out.println("no real roots exists");
      }
    }
}
```

→ Output

- enter the first coefficient
30
enter the second coefficient
40
enter the third coefficient
50
the roots are imaginary

- enter the first coefficient
2
enter the second coefficient
4
enter the third coefficient
1
the root1 is -0.292893
the root2 is -1.707106

- enter the first coefficient
2
enter the second coefficient
4
enter the third coefficient
2
the roots are equal and its value is -1.0.

PROGRAM 2:
Develop a Java program to create a class Student with members usn, name, an array credits and array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

CODE:

```
import java.util.*;

class Stud_det {
    int m[] = new int[8];
    int c[] = new int[8];
    int p[] = new int[8];
    int g, sum;
    String name, usn;
    double sgpa;
    Scanner s = new Scanner(System.in);

    void getdetails() {
        System.out.println("my name is ANJALI PATEL");
        System.out.println("USN: 1BM23CS037");
        System.out.println("Enter name:");
        name = s.next();
        System.out.println("Enter USN:");
        usn = s.next();
        for (int i = 0; i < 8; i++) {
            System.out.println("Enter marks of subject " + (i + 1) + ":");
            m[i] = s.nextInt();
            System.out.println("Enter credits for subject " + (i + 1) + ":");
            c[i] = s.nextInt();
        }
    }

    void gradepoint() {
        for (int i = 0; i < 8; i++) {
            if (m[i] >= 90 && m[i] <= 100)
                p[i] = 10;
            else if (m[i] >= 80 && m[i] < 90)
                p[i] = 9;
            else if (m[i] >= 70 && m[i] < 80)
                p[i] = 8;
            else if (m[i] >= 60 && m[i] < 70)
                p[i] = 7;
            else if (m[i] >= 50 && m[i] < 60)
                p[i] = 6;
            else if (m[i] >= 40 && m[i] < 50)
                p[i] = 5;
            else
```

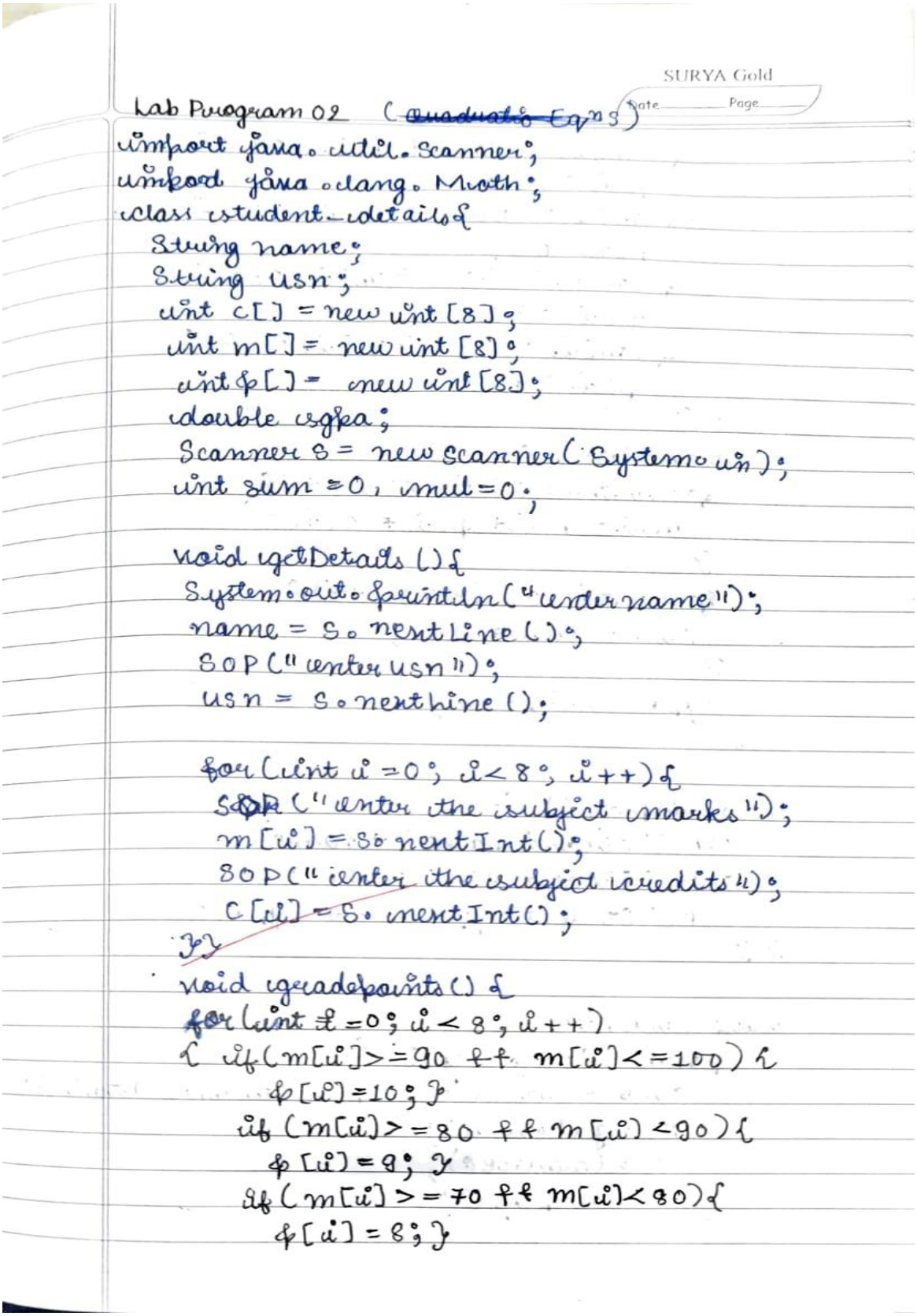
```
            p[i] = 0;
        }
    }

    void calculate() {
        g = 0; // Reset g for each student
        sum = 0; // Reset sum for each student
        for (int i = 0; i < 8; i++) {
            g += c[i] * p[i];
            sum += c[i];
        }
        sgpa = (double) g / sum; // Ensure floating-point division
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("USN: " + usn);
        System.out.println("SGPA: " + sgpa);
    }
}

class student_det {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the number of students:");
        int n = s.nextInt(); // Input the number of students

        Stud_det s1[] = new Stud_det[n];
        for (int i = 0; i < n; i++) {
            s1[i] = new Stud_det();
        }

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details of student " + (i + 1) + ":");
            s1[i].getdetails();
        }

        for (int i = 0; i < n; i++) {
            s1[i].gradepoint();
            s1[i].calculate();
        }

        for (int i = 0; i < n; i++) {
            System.out.println("Details of Student " + (i + 1) + ":");
            s1[i].display();
        }
    }
```

}

OUTPUT

```
C:\Users\anjal\OneDrive\Desktop\codes>java student_det
Enter the number of students:
1
Enter details of student 1:
my name is ANJALI PATEL
USN: 1BM23CS037
Enter name:
altamash
Enter USN:
1bm23cs037
Enter marks of subject 1:
90
Enter credits for subject 1:
4
Enter marks of subject 2:
80
Enter credits for subject 2:
4
Enter marks of subject 3:
90
Enter credits for subject 3:
3
Enter marks of subject 4:
95
Enter credits for subject 4:
3
Enter marks of subject 5:
70
Enter credits for subject 5:
3
Enter marks of subject 6:
90
Enter credits for subject 6:
1
Enter marks of subject 7:
92
Enter credits for subject 7:
1
Enter marks of subject 8:
93
Enter credits for subject 8:
1
Details of Student 1:
Name: altamash
USN: 1bm23cs037
SGPA: 9.5
```

ALGORITHM



Lab Program 02 (~~Quadratic Eqn~~ SGPA)

```
import java.util.Scanner;
import java.lang.Math;
class student_details{
  String name;
  String usn;
  int c[] = new int[8];
  int m[] = new int[8];
  int gp[] = new int[8];
  double sgpa;
  Scanner s = new Scanner(System.in);
  int sum = 0, mul = 0;

  void getDetails(){
   System.out.println("enter name");
   name = s.nextLine();
   SOP("enter usn");
   usn = s.nextline();

   for(int i = 0; i < 8; i++){
    SOP("enter the subject marks");
    m[i] = s.nextInt();
    SOP("enter the subject credits");
    c[i] = s.nextInt();
   }
  void getgradepoints(){
  for(int i = 0; i < 8; i++)
  { if(m[i] >= 90 && m[i] <= 100){
      gp[i] = 10; }
     if(m[i] >= 80 && m[i] < 90){
      gp[i] = 9; }
     if(m[i] >= 70 && m[i] < 80){
      gp[i] = 8; }
```

```
if (m[i]>=60 && m[i]<70){
   gp[i]=7;}
if (m[i]>=50 && m[i]<60){
   gp[i]=6;}
if (m[i]>=40 && m[i]<50){
   gp[i]=5;}
if (m[i]>=30 && m[i]<40){
   gp[i]=4;}
}}}
void calculate(){
   for(int i=0; i<8; i++){
     mul=mul+gp[i]*c[i];
   }
   for(int i=0; i<8; i++){
    sum=sum+c[i];
   }
   sgpa=mul/sum;
}

void display(){
   SOP("student name"+name);
   SOP("student usn"+usn);
   SOP("Student sgpa is"+sgpa);
}}

class student{
   psvm(String arg[]){
      student_details S1=new student_details();
      S1.getDetails();
      S1.Calculate();
      S1.display();
   }
}
```

SURYA Gold

Date Page

Output

enter name
Anshika
enter USN
1BM23CS0100
enter the subject marks
90
enter the subject credit
4
enter the subject mark
80
enter the subject credit
4
—"— ~~Sub~~ marks
90
—"— credits
3
—"— marks
95
—"— credits
3
—"— marks
70
—"— credits
3
—"— marks
90
—"— credits
1
—"— marks
92
—"— credits
1

SURYA Gold

Date _____ Page _____

enter the subject marks
93
enter the subject credits
1
Student name adrshika
Student usn 1bm23cs 700
student sgpa 9.0

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.Scanner;

class Book {
    int id;
    String name;
    String author;
    int numPages;

    // Constructor to initialize book details
    Book(int id, String name, String author, int numPages) {
        this.id = id;
        this.name = name;
        this.author = author;
        this.numPages = numPages;
    }

    // Display book details
    void display() {
        System.out.println("Book ID: " + id);
        System.out.println("Book Name: " + name);
        System.out.println("Author: " + author);
        System.out.println("Number of Pages: " + numPages);
    }
}

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

        System.out.println("Enter number of books: ");
        int n = sc.nextInt();
        Book[] books = new Book[n];

        // Taking input for each book
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details of book " + (i + 1) + ": ");
            System.out.print("Enter Book ID: ");
            int id = sc.nextInt();
            sc.nextLine(); // Consume the newline character

            System.out.print("Enter Book Name: ");
```

```
            String name = sc.nextLine();

            System.out.print("Enter Author Name: ");
            String author = sc.nextLine();

            System.out.print("Enter Number of Pages: ");
            int numPages = sc.nextInt();

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

        // Displaying all book details
        System.out.println("\nDetails of Books:");
        for (int i = 0; i < n; i++) {
            System.out.println("Book " + (i + 1) + ":");
            books[i].display();
            System.out.println();
        }

        sc.close();
    }
}
```

OUTPUT:

```
C:\Users\anjal\OneDrive\Desktop\codes>java Library
Enter number of books:
2
Enter details of book 1:
Enter Book ID: 032
Enter Book Name: thegirlonthetrain
Enter Author Name: BAparis
Enter Number of Pages: 400
Enter details of book 2:
Enter Book ID: 056
Enter Book Name: themurderontheorientexpress
Enter Author Name: AgathaCristie
Enter Number of Pages: 350

Details of Books:
Book 1:
Book ID: 32
Book Name: thegirlonthetrain
Author: BAparis
Number of Pages: 400

Book 2:
Book ID: 56
Book Name: themurderontheorientexpress
Author: AgathaCristie
Number of Pages: 350
```

ALGORITHM:

Lab 02

Lab Program - 3

```
import java.util.Scanner;
class Books{
    String name;
    String author;
    int price;
    int numPages;

    Books (String name, String author, int price,
           int numPages){
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }
    public String toString (){
        String name, author, price, numPages;
        name = "Book name: " + this.name +
               "\n";
        price = "Price: " + this.price + "\n";
        numPages = "Number of pages: " + this.numPages +
                   "\n";
        return name + author + price + numPages;
    }
}
class bookdetails{
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        int n;
        String name;
        String author;
        int price;
        int numPages;
```

```
System.out.println("enter no. of book");
n = S.nextInt();

Books b[] = new Books[n];
for(int i=0; i<n; i++){
    System.out.println("enter book name");
    name = S.nextLine();
    SOP("enter the author name");
    author = S.nextLine();
    SOP("enter the book price");
    price = S.nextInt();
    SOP("enter the numPages");
    numPages = S.nextInt();
    b[i] = new Books(name, author,
            price, numPages);
}
for(int i=0; i<n; i++){
    SOP("details are");
    SOP(b[i]);
}
}
}
```

→ Output

```
enter book name of books
1
enter the book name
The girl on the train
enter the author name
BAParis
enter the book price
400
enter the numPages
305
```

SURYA Gold

Date_____ Page_____

the details are
Book name : Thegirlonthetrain
Author name : BAParis
Price : 400
numPages : 305

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.Scanner;
class PrintInfo {
   static void print() {
        System.out.println("Name: ANJALI PATEL");
        System.out.println("USN: 1BM23CS037");
    }
}

abstract class shape{
      int dim1;
      int dim2;
      abstract void printarea();
}

class rectangle extends shape{
      public rectangle(){
      this.dim1=dim1;
      this.dim2=dim2;
      }
      public void printarea(){
        Scanner s = new Scanner(System.in);
        System.out.println("enter the l and b");
        dim1=s.nextInt();
        dim2=s.nextInt();

        int area=dim1*dim2;
        System.out.println("area of rectangle: "+area);
      }
}
class triangle extends shape{
      public triangle(){
      this.dim1=dim1;
      this.dim2=dim2;
      }
       public void printarea(){
        Scanner s = new Scanner(System.in);

         System.out.println("enter the l and b");
        dim1=s.nextInt();
        dim2=s.nextInt();
```

```
        double area=(dim1*dim2)/2;
        System.out.println("area of triangle: "+area);
      }

}
class circle extends shape{
      final double Pi=3.14;
      public circle(){
      this.dim1=dim1;
      }
      public void printarea(){
      Scanner s = new Scanner(System.in);

       System.out.println("enter the radius");
        dim1=s.nextInt();

        double area=Pi*dim1*dim1;
        System.out.println("area of circle: "+area);
      }
}
public class maindemo{
  public static void main (String [] args){
  PrintInfo.print();




     rectangle R =new rectangle();

     R.printarea();

     triangle T = new triangle();
     T.printarea();

     circle C = new circle();
    C.printarea();
    }
}
```

OUTPUT:

```
C:\Users\anjal\OneDrive\Desktop\codes>java maindemo
Name: ANJALI PATEL
USN: 1BM23CS037
enter the l and b
3 2
area of rectangle: 6
enter the l and b
5 6
area of triangle: 15.0
enter the radius
2
area of circle: 12.56
```

ALGORITHM:

Lab 08

Lab Prog-4

```
import java.util.Scanner;
abstract class Shape {
    int dim1;
    int dim2;
    abstract void printarea();
}
class rectangle extends Shape {
    public rectangle() {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }
    public void printarea() {
        Scanner s = new Scanner(System.in);
        S.OP("enter l and b");
        dim1 = s.nextInt();
        dim2 = s.nextInt();
        int area = dim1 * dim2;
        SOP("area of rectangle:" + area);
    }
}
class triangle extends Shape {
    triangle() {
        this.dim1 = dim1;
        —"— 2 = —"— 2;
    }
    public void printarea() {
        Scanner s = new Scanner(System.in);
        SOP("enter h and b");
        dim1 = s.nextInt();
        dim2 = ——"——;
        double area = (dim1 * dim2)/2;
        SOP("area of triangle:" + area);
```

```
System.out.println("enter no. of book,");
n = So.nextInt();

Books b[] = new Books[n];
for(int i=0; i<n; i++){
    System.out.println("enter book name");
    name = So.nextLine();
    SOP("enter the author name");
    author = So.nextLine();
    SOP("enter the book price");
    price = So.nextInt();
    SOP("enter the numPages");
    numPages = So.nextInt();
    b[i] = new Books(name, author,
        price, numPages);
}
for(int i=0; i<n; i++){
    SOP("details are");
    SOP(b[i]);
}
}
}
```

→ Output

enter book name of books
1
enter the book name
The girl on the train
enter the author name
BAParis
enter the book price
400
enter the numPages
805

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 PrintInfo {
  static void print() {
      System.out.println("Name: ANJALI PATEL");
      System.out.println("USN: 1BM23CS037");
   }
}


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

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

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

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

```
    }

    public abstract void withdraw(double amount);
}

class SavAcct extends Account {
    double interestRate;

    public SavAcct(String customerName, String accountNumber, double interestRate) {
        super(customerName, "savings", accountNumber);
        this.interestRate = interestRate;
    }

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

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

class CurAcct extends Account {
    private static final double minBalance = 1000.0;
    private static final double serviceCharge = 50.0;

    public CurAcct(String customerName, String accountNumber) {
        super(customerName, "current", accountNumber);
    }

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

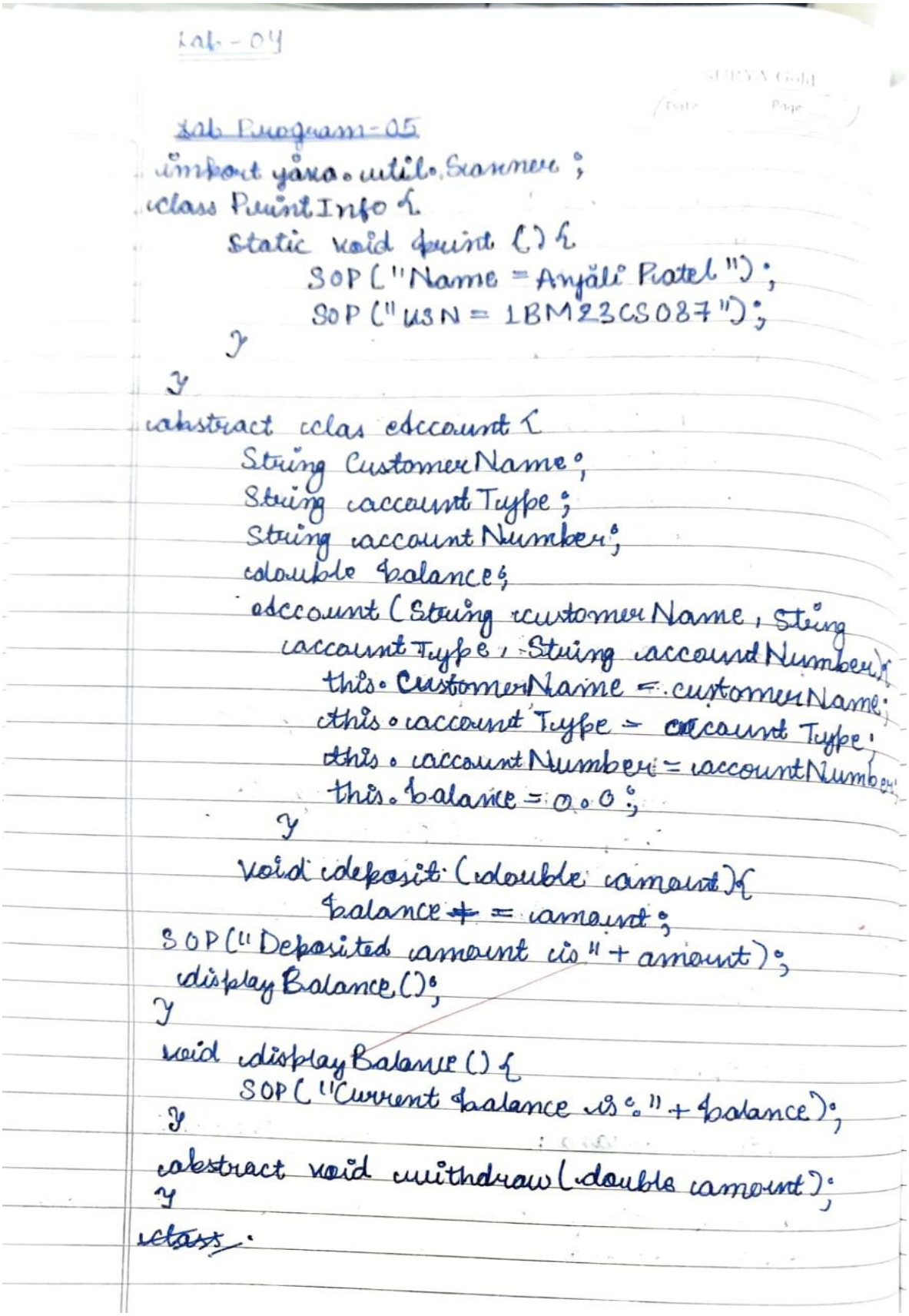
```
        if (balance < minBalance) {
            balance -= serviceCharge;
            System.out.println("Minimum balance not maintained");
            System.out.println("Service charge of: " + serviceCharge + " included");
        }
        displayBalance();
    }
}

public class bank {
    public static void main(String[] args) {
        PrintInfo.print();

        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter your account type (savings/current):");
        String accountType = scanner.nextLine();
        System.out.println("Enter account number:");
        String accountNumber = scanner.nextLine();
        System.out.println("Enter your name:");
        String customerName = scanner.nextLine();

        Account account;
        if (accountType.equals("savings")) {
            System.out.println("Enter the interest rate:");
            double interestRate = scanner.nextDouble();
            account = new SavAcct(customerName, accountNumber, interestRate);
        } else {
            account = new CurAcct(customerName, accountNumber);
        }

        while (true) {
            System.out.println("1. Deposit\n2. Withdraw\n3. Display Balance\n4. 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:
                    System.out.println("Enter amount to withdraw:");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;
                case 3:
                    account.displayBalance();
```

```
                break;
            case 4:
                System.out.println("Exit");
                scanner.close();
                return;
            default:
                System.out.println("Try again");
        }
    }
  }
}
```

OUTPUT:

```
C:\Users\anjal\OneDrive\Desktop\codes>javac bank.java

C:\Users\anjal\OneDrive\Desktop\codes>java bank
Name: ANJALI PATEL
USN: 1BM23CS037
Enter your account type (savings/current):
savings\
Enter account number:
123456789
Enter your name:
Anshika
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
1
Enter amount to deposit:
4000
Deposited amount is: 4000.0
Current balance is: 4000.0
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
2
Enter amount to withdraw:
1500
Withdrawn amount is: 1500.0
Current balance is: 2500.0
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
3
Current balance is: 2500.0
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
4
Exit
```

ALGORITHM:



Lab - 04

Lab Program - 05

```
import java.util.Scanner;
class PrintInfo {
    static void print() {
        SOP("Name = Anjali Patel");
        SOP("USN = 1BM23CS087");
    }
}
abstract class Account {
    String CustomerName;
    String caccountType;
    String caccountNumber;
    double balance;
    Account(String customerName, String
      caccountType, String accountNumber) {
        this.CustomerName = customerName;
        this.caccountType = caccountType;
        this.caccountNumber = accountNumber;
        this.balance = 0.0;
    }
    void deposit(double camount) {
        balance += camount;
SOP("Deposited amount is" + amount);
displayBalance();
}
void displayBalance() {
    SOP("Current balance is" + balance);
}
abstract void withdraw(double camount);
}
class
```

SURYA Gold
Date Page

```
class SavAcct extends eAccount {
    double interestDate;

  public SavAcct (String customerName,
      String accountNumber, double Interest-
    - Rate) {
      super (customerName, "Savings",
      accountNumber);
        this.interestRate =
          interestRate;
    }

public void compoundDeposit(){
  double interest = balance * (interestRate / 100);
  deposit (interest);
    SOP ("Interest of " + interest + "deposited");
}
public void withdraw (double amount){
  if (amount <= balance){
        balance = balance - amount;
SOP ("withdrawn amount is:" + amount);
}
  else {
  SOP("Insufficient amount for withdrawal");
    return;
  }
  displayBalance();
}
}
```

```
class CurAcct extends Account {
    private static final double
        minbalance = 1000.0;
    private static final double
        serviceCharge = 50.0;

    public CurAcct (String customerName,
        String AccountNumber) {
            super (customerName, "current",
        accountNumber);
    }
    public void withdraw (double amount) {
        if (amount <= balance) {
            balance -= amount;
SOP ("withdrawn amount is " + amount);
    }
    else {
        SOP ("Insufficient amount for withdrawal");
            return;
    }
    if (balance < minBalance) {
        balance -= serviceCharg;
SOP ("Minimum balance not maintained);

SOP ("Service charge of :" + serviceCharge +
    "included");
    }
    displayBalance();
}
}
```

```
public class Bank {
    psvm (String arg[]) {
        print Info.print();
        Scanner s = new Scanner(System.in);
        SOP("enter your account type (
         Savings / current): ");
        String accountType = s.nextLine();
      SOP("account Number");
      String accountNumber = s.nextLine();
      SOP("enter your name");
      String customerName = s.nextLine();
      Account account;
      if (accountType.equals("Savings")) {
        SOP("enter interest rate%");
        double interestRate = s.nextDouble();
        account = new SavAcct(
      customerName, accountNumber,
      interestRate); }
  else {
      account = new CurAcct (customerName,
          accountNumber); }
  }
  while(true) {
      SOP("1.Deposit\n 2.withdraw\n
    3.Display Balance\n 4.Exit");
      int choice = s.nextInt();
      switch(choice) {
          case 1: SOP("enter amount to
              deposit");
                  double depositAmount =
                   s.nextDouble();
                  account.deposit(depositAmount);
                  break;
```

```
case 2 :
        SOP("enter amount to withdraw");
        double withdrawAmount = S.nextDo
        account.withdraw(withdrawAmou
        break;
case 3 : account.displayBalance();
        break;
case 4 : SOP("Exit");
        scanner.close();
        return;
default : SOP("Try again");
}
}
}
```

→ Output

```
Name : ANJALI PATEL
USN : 1BM23CS037
Enter your account type (savings/current):
Savings
Enter account number :
1234567
enter your name
anjali
enter the interest rate :
5
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
1
```

SURYA Gold

Date Page

enter the amount to deposit :
4000
Deposited amount is : 4000.0
Current balance is : 4000.0
1. Deposit
2. Withdraw
3. Display Balance
4. Exit

2
Enter amount to withdraw :
2000.0
withdrawn amount is : 2000.0
current balance is : 2000.0
1. Deposit
2. Withdraw
3. Display Balance
4. Exit

4
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:

```
//ANJALI PATEL , 1BM23CS037
//package cie , class student
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;
    }
}

//package cie , class internals
package CIE; internals.java

public class Internals extends Student {
    public int[] internalMarks = new int[5];

    public Internals(String usn, String name, int sem, int[] marks) {
        super(usn, name, sem);
        if (marks.length == 5) {
            System.arraycopy(marks, 0, internalMarks, 0, 5);
        }
    }
}

//package see , class externals
package SEE; Externals.java
import CIE.Student;

public class External extends Student {
    public int[] externalMarks = new int[5];

    public External(String usn, String name, int sem, int[] marks) {
```

```
        super(usn, name, sem);
        if (marks.length == 5) {
            System.arraycopy(marks, 0, externalMarks, 0, 5);
        }
    }
}
//main class

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();

        Internals[] internalStudents = new Internals[n];
        External[] externalStudents = new External[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for student " + (i + 1));
            System.out.print("USN: ");
            String usn = sc.next();
            System.out.print("Name: ");
            String name = sc.next();
            System.out.print("Semester: ");
            int sem = sc.nextInt();

            System.out.println("Enter internal marks for 5 subjects:");
            int[] internalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = sc.nextInt();
            }
            internalStudents[i] = new Internals(usn, name, sem, internalMarks);

            System.out.println("Enter SEE marks for 5 subjects:");
            int[] externalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                externalMarks[j] = sc.nextInt();
            }
            externalStudents[i] = new External(usn, name, sem, externalMarks);
        }

        System.out.println("\nFinal Marks of Students:");
```

```
        for (int i = 0; i < n; i++) {
            System.out.println("\nStudent " + (i + 1) + " - USN: " + internalStudents[i].usn);
            for (int j = 0; j < 5; j++) {
                int finalMarks = internalStudents[i].internalMarks[j] +
(externalStudents[i].externalMarks[j] / 2);
                System.out.println("Subject " + (j + 1) + ": " + finalMarks);
            }
        }

        sc.close();
    }
}
```

OUTPUT:

```
C:\Users\Admin\Documents\student>javac -d . CIE/Internals.java

C:\Users\Admin\Documents\student>javac -d . CIE/Student.java

C:\Users\Admin\Documents\student>javac -d . SEE/Externals.java

C:\Users\Admin\Documents\student>javac Main.java

C:\Users\Admin\Documents\student>java Main
Enter number of students: 1
Enter USN: 1286759
Enter Name: SIRI
Enter Semester: 3
Enter Internal Marks for 5 courses:
Course 1: 34
Course 2: 35
Course 3: 32
Course 4: 37
Course 5: 40
Enter External Marks for 5 courses:
Course 1: 38
Course 2: 39
Course 3: 40
Course 4: 40
Course 5: 30
USN: 1286759
Name: SIRI
Semester: 3
Internal Marks:
Course 1: 34
Course 2: 35
Course 3: 32
Course 4: 37
Course 5: 40
External Marks:
Course 1: 38
Course 2: 39
Course 3: 40
Course 4: 40
Course 5: 30
Final Marks:
Course 1: 72
Course 2: 74
Course 3: 72
Course 4: 77
Course 5: 70
```

ALGORITHM:

SURYA Gold
Date Page

Lab - 6

Q) Create a package CIE which has 2 classes — Student and Internals. The class student has members like usn, name, sem. The class internals derived from student has an array that stores the internal marks scored in 5 courses of the current semester of the student. Create another package SEE which has the class External which is derived class of Student. The class has an array that stores the SEE marks scored in 5 courses of the current semester of the student. Import the 2 packages in a file that declares the final marks of n students in all 5 courses.

```
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;
    }
}

package CIE
public class Internals extends Student {
    public int[] marks = new int[5];
    public Internals (String usn, String name,
                      int sem, int[] marks) {
```

```
        super(usn, name, sem);
        for(int i=0; i>5; i++){
            internal.marks[i] = marks[i];
        }
    }
}
package SEE;
import CIE.Student;
public class External extends Student{
    public int[] external Marks = new int[5];
    public External(String usn, String name,
            int sem, int [] marks){
        super(usn, name, sem);
        if(marks.length == 5){
            System.arraycopy(marks, 0,
                externalMarks, 0, 5);
        }
    }
}

import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;
public static vm (String[] args){
    Scanner sc = new Scanner(System.in);
    SOP("enter no. of students:");
    int n = sc.nextInt();
    Internals[] internal Student = new Internal[n];
    External [] external Student = new External[n];
    for(i=0; i<n; i++){
        SOP("enter details" + (i+1));
        SOP("USN:");
        String usn = sc.next();
```

```
SOP ("Name : ");
String name = sc.next();
SOP ("Semester : ");
int sem = sc.nextInt();

SOP ("Enter internal marks for 5 subjects : ");
int [] internalMarks = new int [5];
for(int j = 0; j < 5; j++){
        internalMarks [j] = sc.nextInt();
}
external Students[i] = new Internals
        (usn, name, sem, internalMarks);
}
SOP ("\n Final marks of students : ");
for(int i = 0; i < n ; i++){
        SOP ("Student " + (i+2) + " - USN : "
                + internalStudents[i].usn);
        for(int j = 0 ; j < 5; j++){
                int finalMarks = internal Student
                        [i].internalMark[j]
    + (external Students[i].external Marks(j);
    SOP ("Subjects " + (j+1) + " " + finalmark);
  }
}
Sc.close();
}
}
```

SURYA Gold

Date Page

→ Output lab06

Enter the number of students : 2
Enter details for student 1
USN : 1BM21CS001
Name : Alice
Semester : 5
Enter internals marks for 5 subjects :
18 20 19 17 20
Enter SEE marks for 5 subjects :
70 80 90 60 75
Enter details for student 2
USN : 1BM21CS002
Name : Bob
Semester : 5
Enter internal marks for 5 subjects :
15 18 20 14 19
Enter SEE marks for 5 subjects :
65 75 80 55 70
Final marks of Students :
Student 1 - USN : 1BM21CS001
Subject 1 : 53
Subject 2 : 60
Subject 3 : 64
Subject 4 : 47
Subject 5 : 57

Student 2 - USN : 1BM21CS002
Subject 1 : 47
Subject 2 : 55
Subject 3 : 60
Subject 4 : 41
subjid 5 : 54

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 WrongAge( ) when the input age<0. In Son class, implement a constructor that uses both father and son’s age and throws an exception if son’s age is >=father’s age.

CODE:

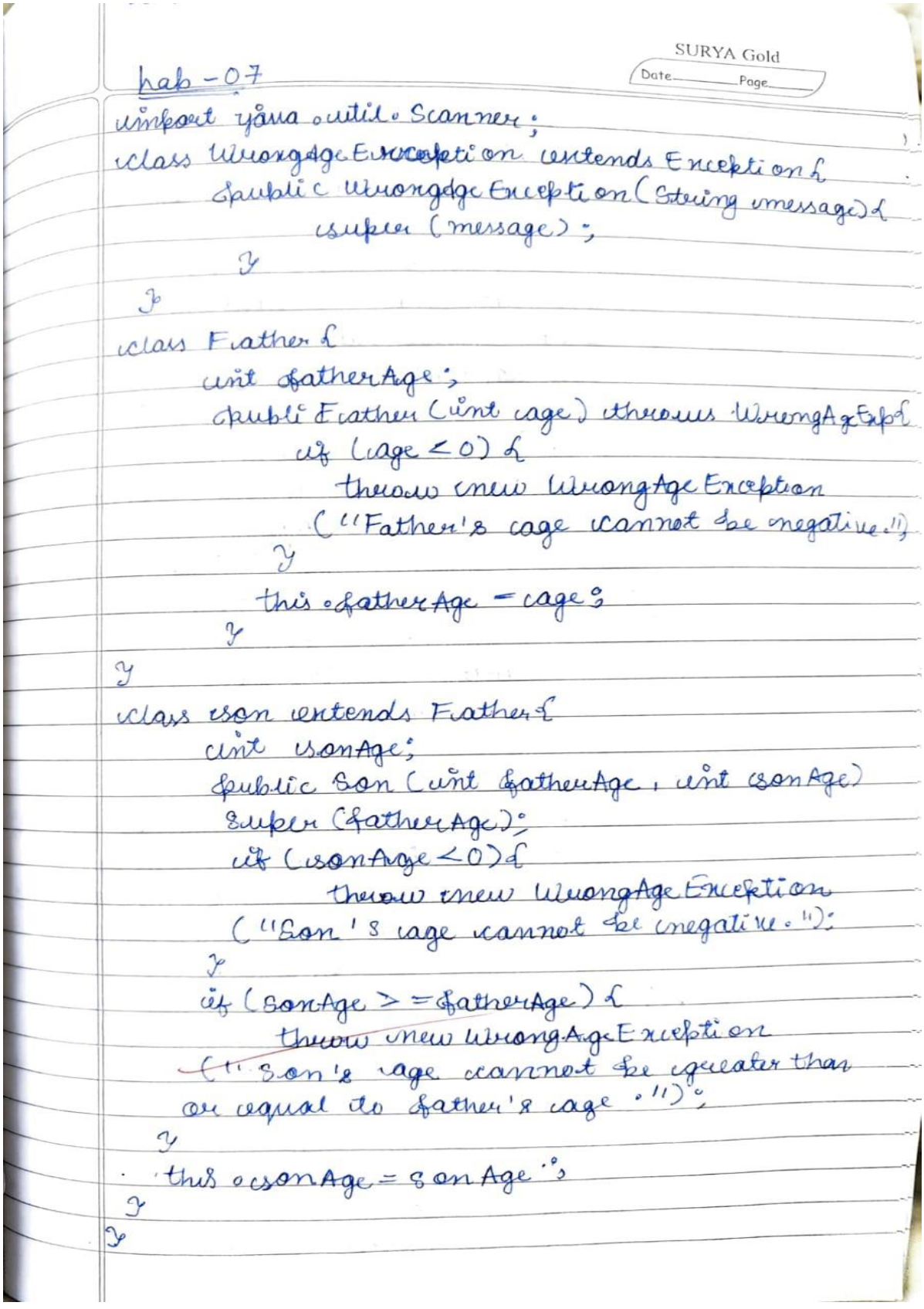
```
//ANJALI PATEL , 1BM23CS037
import java.util.Scanner;

class NegativeAgeError extends Exception {
    int age;

    public NegativeAgeError(int age) {
        this.age = age;
    }

    public String toString() {
        return "Negative Age: " + age;
    }
}

class InvalidAgeError extends Exception {
    int sonAge, fatherAge;

    public InvalidAgeError(int sonAge, int fatherAge) {
        this.sonAge = sonAge;
        this.fatherAge = fatherAge;
    }

    public String toString() {
        return "Invalid Age: Son's age (" + sonAge + ") is greater than or equal to Father's age (" +
fatherAge + ")";
    }
}

class Father {
    int age;

    Father(int age) {
        try {
            if (age < 0) {
                throw new NegativeAgeError(age);
            }
            this.age = age;
        } catch (NegativeAgeError e) {
```

```
            this.age = 20;
            System.out.println(e);
        }
    }
}

class Son extends Father {
    int sonAge;

    Son(int sonAge, int fatherAge) {
        super(fatherAge);
        try {
            if (sonAge < 0) {
                throw new NegativeAgeError(sonAge);
            }
            if (sonAge >= this.age) {
                throw new InvalidAgeError(sonAge, this.age);
            }
            this.sonAge = sonAge;
        } catch (NegativeAgeError e) {
            this.sonAge = 20;
            System.out.println(e);
        } catch (InvalidAgeError e) {
            this.sonAge = 10;
            System.out.println(e);
        }
    }
}

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

        System.out.println("Enter Father's age: ");
        int fatherAge = sc.nextInt();

        System.out.println("Enter Son's age: ");
        int sonAge = sc.nextInt();

        Son son = new Son(sonAge, fatherAge);

        System.out.println("Father's Age: " + son.age);
        System.out.println("Son's Age: " + son.sonAge);

        sc.close();
    }
}
```

OUTPUT:

```
C:\Users\anjal\OneDrive\Desktop\codes>java Exceptions
Enter Father's age:
45
Enter Son's age:
18
Father's Age: 45
Son's Age: 18

C:\Users\anjal\OneDrive\Desktop\codes>java Exceptions
Enter Father's age:
80
Enter Son's age:
90
Invalid Age: Son's age (90) is greater than or equal to Father's age (80)
Father's Age: 80
Son's Age: 10

C:\Users\anjal\OneDrive\Desktop\codes>java Exceptions
Enter Father's age:
50
Enter Son's age:
-67
Negative Age: -67
Father's Age: 50
Son's Age: 20
```

ALGORITHM:



lab - 07

```
import java.util.Scanner;
class WrongAgeException extends Exception {
    public WrongAgeException(String message) {
        super(message);
    }
}
class Father {
    int fatherAge;
    public Father(int age) throws WrongAgeExp{
        if (age < 0) {
            throw new WrongAgeException
            ("Father's age cannot be negative.");
        }
        this.fatherAge = age;
    }
}
class son extends Father{
    int sonAge;
    public Son(int fatherAge, int sonAge)
     super(fatherAge);
     if (sonAge < 0){
            throw new WrongAgeException
     ("Son's age cannot be negative.");
     }
     if (SonAge >= fatherAge) {
          throw new WrongAgeException
    ("Son's age cannot be greater than
    or equal to father's age.");
    }
    this.sonAge = sonAge;
}
}
```

```
public class ExceptionInheritanceDemo {
    psvm(String[] args) {
        Scanner s = new Scanner(System.in);
        try {
            SOP("Enter father's age:");
            int fatherAge = s.nextInt();
            SOP("Enter son's age:");
            int sonAge = s.nextInt();

            SOP("creating valid ages");
            Son validSon = new Son(fatherAge,
                                   sonAge);
            SOP("father's age:" + validSon.fatherAge);
            SOP("son's age:" + validSon.sonAge);
        } catch(WrongAgeException e) {
            System.err.println("Exception
                caught:" + e.getMessage());
        }
        try {
            SOP("Father's age (negative for error):");
            int invalidFatherAge = s.nextInt();
            SOP("Creating father with negative age");
            Father invalidFather = new Father
                (invalidFatherAge);
        } catch(WrongAgeException e) {
            SOP("Exception:" + e.getMessage());
        }
        try {
            SOP("Enter Father's age:");
            int fatherAge = s.nextInt();
            SOP("Enter son's age:");
            int invalidSonAge = s.nextInt();
```

```
SOP("Creating son with age greater than father:!")
Son invalid Son = new Son(fatherAge, invalidSonAg);
}catch(WrongAgeException e){
        SOP("Exception :" e.getMessage());
    }
    S.close();
}
}
```

* Output

→ Enter father's age : -10
Enter son's age : 4
creating father and son valid age
Exception caught : father's age can't be negative.

Enter father's age (negative for error) : 60
creating father with negative age.
Enter father's age : 60.
Enter Son's age (greater than father's age for error) : 80
Creating son with age greater than father's
Exception caught : Son's age cannot be greater than or equal to father's age.

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.

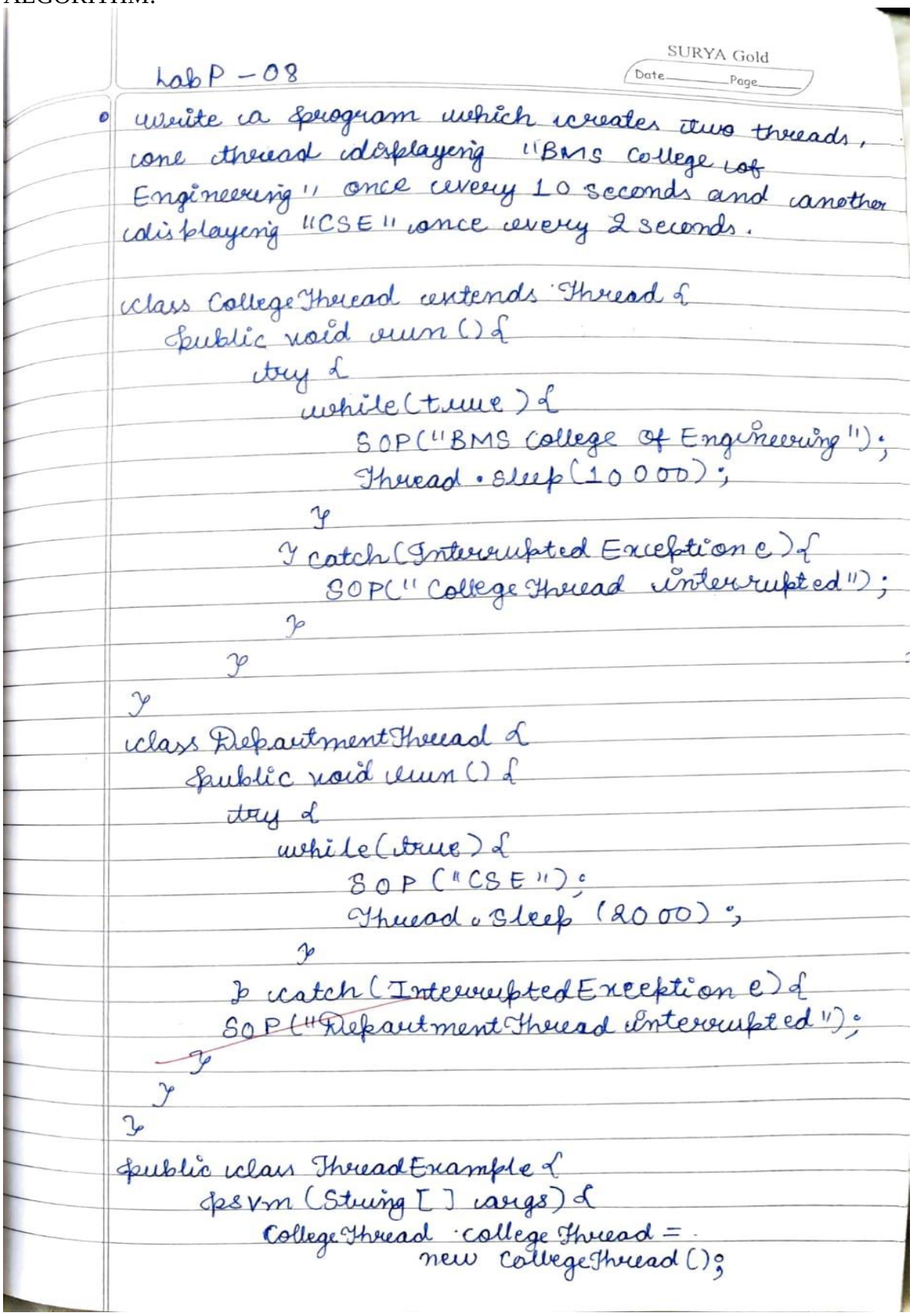
CODE:

```
//ANJALI PATEL , 1BM23CS037
class CollegeThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000); // Sleep for 10 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("CollegeThread interrupted");
        }
    }
}
class DepartmentThread extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000); // Sleep for 2 seconds
            }
        } catch (InterruptedException e) {
            System.out.println("DepartmentThread interrupted");
        }
    }
}
class threadexample {
    public static void main(String[] args) {
        // Creating instances of both threads
        CollegeThread collegeThread = new CollegeThread();
        DepartmentThread departmentThread = new DepartmentThread();

        // Starting both threads
        collegeThread.start();
        departmentThread.start();
    }
}
```

OUTPUT:

```
C:\Users\anjal\OneDrive\Desktop\codes>javac threadexample.java

C:\Users\anjal\OneDrive\Desktop\codes>java threadexample
CSE
BMS College of Engineering
CSE
CSE
```

ALGORITHM:



Lab P – 08

SURYA Gold
Date_____ Page_____

- Write a program which creates two threads, one thread displaying "BMS College of Engineering" once every 10 seconds and another displaying "CSE" once every 2 seconds.

```
class CollegeThread extends Thread {
    public void run() {
        try {
            while(true) {
                SOP("BMS College of Engineering");
                Thread.sleep(10000);
            }
        } catch(InterruptedException e) {
            SOP("CollegeThread interrupted");
        }
    }
}
class DepartmentThread {
    public void run() {
        try {
            while(true) {
                SOP("CSE");
                Thread.sleep(2000);
            }
        } catch(InterruptedException e) {
            SOP("DepartmentThread interrupted");
        }
    }
}
public class ThreadExample {
    psvm(String[] args) {
        CollegeThread collegeThread =
                new CollegeThread();
```

```
DepartmentThread departmentThread
        = new DepartmentThread();
collegeThread.start();
departmentThread.start();
```

→ Output

```
BMS College Of Engineering
CSE
CSE
^C
```

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.

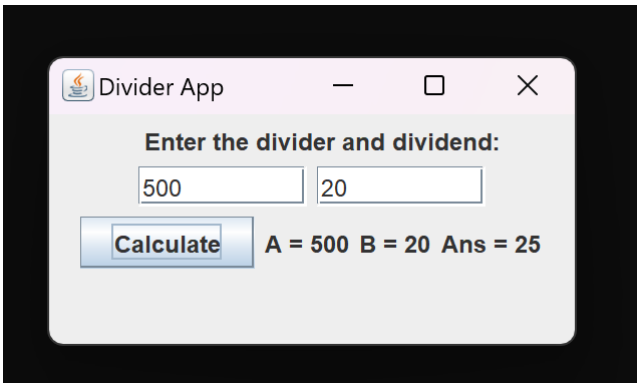
CODE

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

class SwingDemo {
   SwingDemo() {
      // Create JFrame container
      JFrame jfrm = new JFrame("Divider App");
      jfrm.setSize(275, 150);
      jfrm.setLayout(new FlowLayout());

      // To terminate on close
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // Text label
      JLabel jlab = new JLabel("Enter the divider and dividend:");

      // Add text field for both numbers
      JTextField ajtf = new JTextField(8);
      JTextField bjtf = new JTextField(8);

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

      // Labels for displaying values and results
      JLabel err = new JLabel();
      JLabel alab = new JLabel();
      JLabel blab = new JLabel();
      JLabel anslab = new JLabel();

      // Add components in order
      jfrm.add(err);   // Error label
      jfrm.add(jlab);  // Instruction label
      jfrm.add(ajtf);  // Text field for first number
      jfrm.add(bjtf);  // Text field for second number
      jfrm.add(button); // Calculate button
      jfrm.add(alab);   // Label for A value
      jfrm.add(blab);   // Label for B value
```

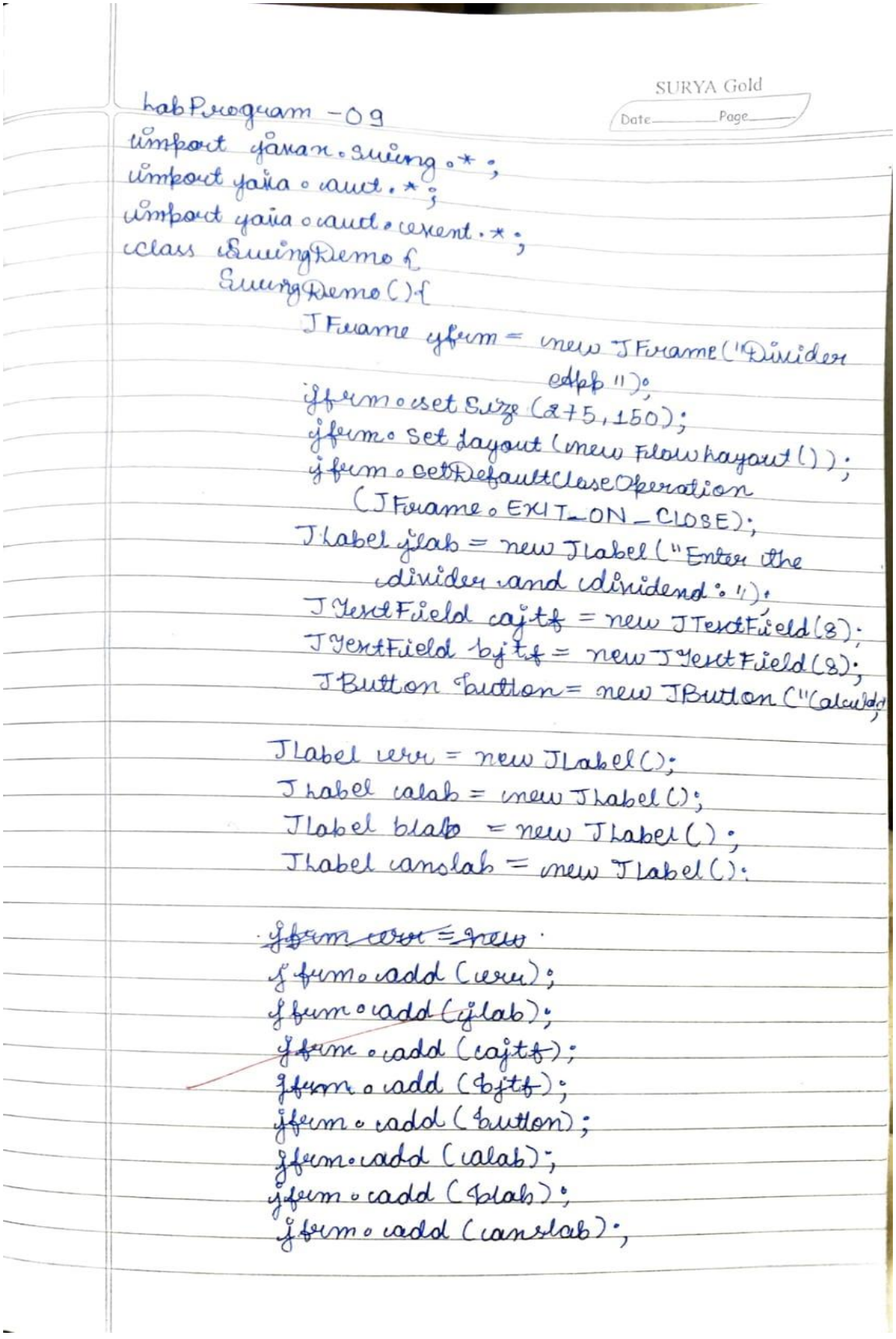
```
    jfrm.add(anslab); // Label for result

    // Action listener for the button
    button.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            try {
                // Clear previous error message
                err.setText("");

                // Get values from text fields
                int a = Integer.parseInt(ajtf.getText());
                int b = Integer.parseInt(bjtf.getText());

                // Perform division
                int ans = a / b;

                // Display values and result
                alab.setText("A = " + a);
                blab.setText("B = " + b);
                anslab.setText("Ans = " + ans);
            } catch (NumberFormatException e) {
                // Handle non-integer input
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("Enter Only Integers!");
            } catch (ArithmeticException e) {
                // Handle division by zero
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("B should be NON-zero!");
            }
        }
    });

    // Display the frame
    jfrm.setVisible(true);
}

public static void main(String args[]) {
    // Create frame on event dispatching thread
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new SwingDemo();
        }
    });
```

```
    }
}
```

OUTPUT


Divider App
Enter the divider and dividend:
500
20
Calculate
A = 500 B = 20 Ans = 25

ALGORITHM :



```
SURYA Gold
Date____ Page____

Lab Program - 09
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo {
    SwingDemo(){
        JFrame jfrm = new JFrame("Divider
                                  edpb");
        jfrm.setSize(275,150);
        jfrm.Set layout(new FlowLayout());
        jfrm.setDefaultCloseOperation
            (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 err = new
    jfrm.add(err);
    jfrm.add(jlab);
    jfrm.add(ajtf);
    jfrm.add(bjtf);
    jfrm.add(button);
    jfrm.add(alab);
    jfrm.add(blab);
    jfrm.add(anslab);
```

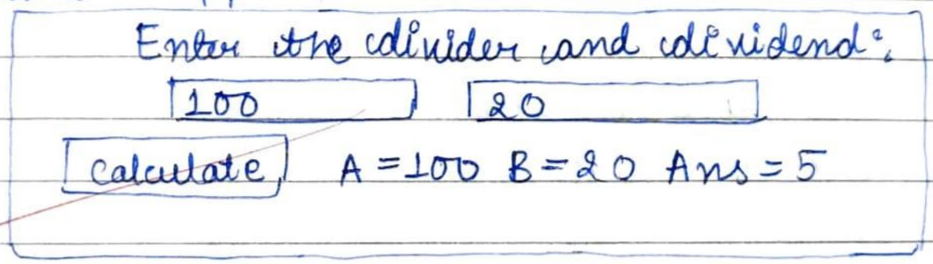
```
button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent evt){
        try{
            err.setText("");
            int a = Integer.parseInt(objtf.getText());
            int b = Integer.parseInt(objtf.getText());
            int ans = a/b;
            alab.setText("A = "+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");
        }
    }
});
jfrm.setVisible(true);
}
public static void main(String args[]){
    SwingUtilities.invokeLater(new Runnable(){
        public void run(){
            new SwingDemo();
        }
    });
}
}
```

SURYA Gold

Date _____ Page _____

→ Output

Divider.cpp



Seen

27/11/24

PROGRAM 10: Demonstrate Inter process Communication and deadlock

INTER PROCESS COMMUNICATION
CODE:

```
class Q {

   int n; // Shared resource
   boolean valueSet = false; // Flag to check if a value is available

   synchronized int get() {
      while (!valueSet) {
         try {
            System.out.println("\nConsumer waiting\n");
            wait(); // Consumer waits if value is not set
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      }
      System.out.println("Got: " + n);
      valueSet = false; // Reset the flag as value is consumed
      System.out.println("\nInforming Producer\n");
      notify(); // Notify the producer that the value is consumed
      return n;
   }

   synchronized void put(int n) {
      while (valueSet) {
         try {
            System.out.println("\nProducer waiting\n");
            wait(); // Producer waits if value is already set
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      }
      this.n = n; // Set the value
      valueSet = true; // Update the flag
      System.out.println("Put: " + n);
      System.out.println("\nInforming Consumer\n");
      notify(); // Notify the consumer that the value is available
   }
}

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++); // Produce values
    }
  }
}

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(); // Consume values
      System.out.println("Consumed: " + r);
      i++;
    }
  }
}

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

DEADLOCK

CODE:

```
class A {

  synchronized void foo(B b) {
    String name = Thread.currentThread().getName();
```

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

        try {
            Thread.sleep(1000); // Simulate some work
        } catch (Exception e) {
            System.out.println("A Interrupted");
        }

        System.out.println(name + " trying to call B.last()");
        b.last(); // Call 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); // Simulate some work
        } catch (Exception e) {
            System.out.println("B Interrupted");
        }

        System.out.println(name + " trying to call A.last()");
        a.last(); // Call 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"); // Set main thread's name

        Thread t = new Thread(this, "RacingThread"); // Create another thread
```

```
        t.start();

        a.foo(b); // Main thread calls A.foo
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // RacingThread calls B.bar
        System.out.println("Back in other thread");
    }

    public static void main(String[] args) {
        new Deadlock(); // Start the program
    }
}
```

OUTPUT:
IPC

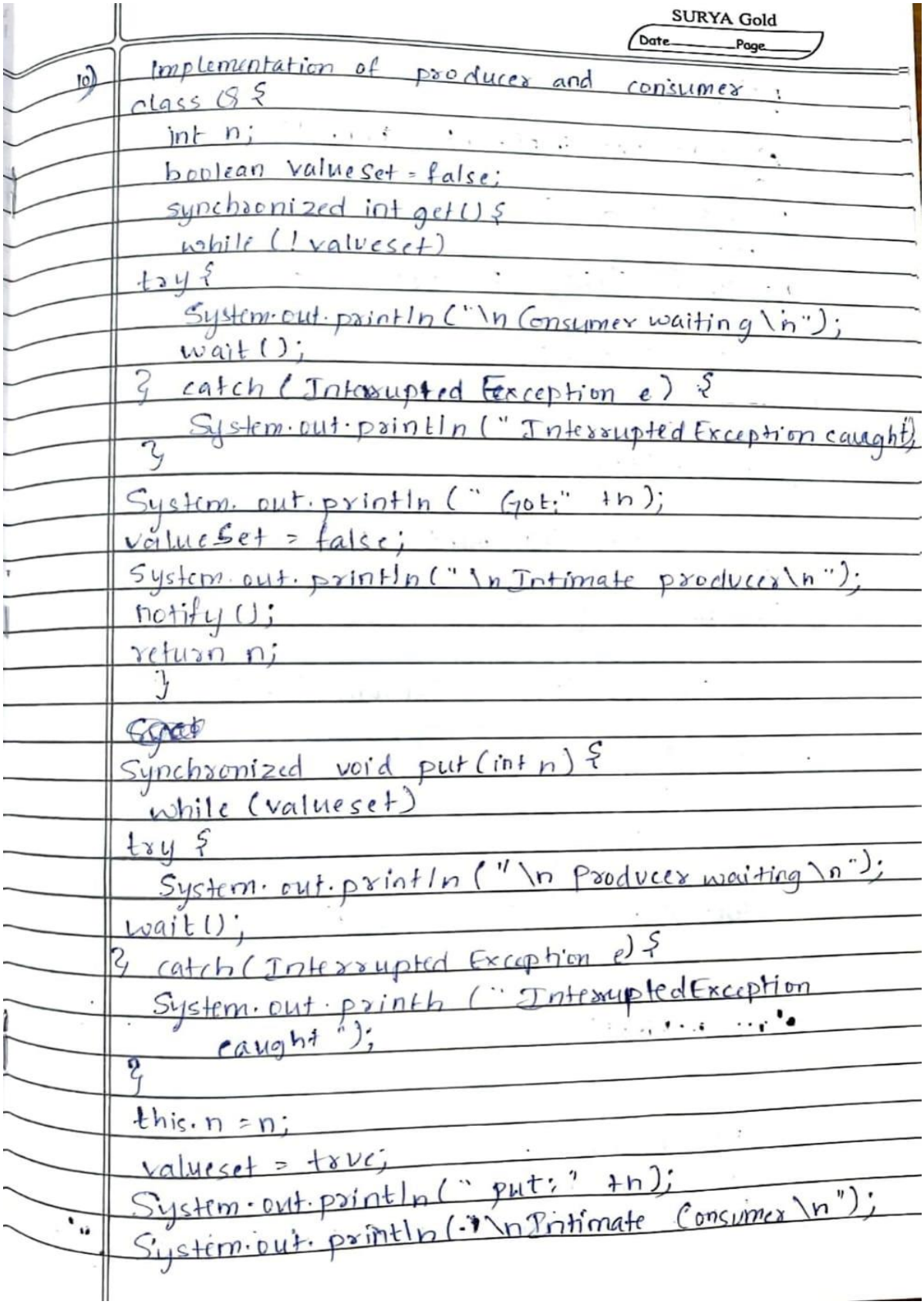
```
C:\Users\anjal\OneDrive\Desktop\codes>javac PCFixed.java

C:\Users\anjal\OneDrive\Desktop\codes>java PCFixed
Press Control-C to stop.
Put: 0

Informing Consumer


Producer waiting

Got: 0

Informing Producer

Put: 1

Informing Consumer


Producer waiting

Consumed: 0
Got: 1

Informing Producer

Consumed: 1
Put: 2

Informing Consumer


Producer waiting

Got: 2
```

```
Informing Producer

Consumed: 2
Put: 3

Informing Consumer


Producer waiting

Got: 3

Informing Producer

Consumed: 3
Put: 4

Informing Consumer


Producer waiting

Got: 4
```

DEADLOCK

```
C:\Users\anjal\OneDrive\Desktop\codes>javac Deadlock.java

C:\Users\anjal\OneDrive\Desktop\codes>java DeadlockMainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
Back in main thread
RacingThread trying to call A.last()
Inside A.last
Back in other thread
```

ALGORITHM:



10) Implementation of producer and consumer :

```
class CS {
  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(" InterruptedException 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(" InterruptedException
        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.");
}}
```

SURYA Gold
Date Page

Dead lock

```
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();
}
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();
}
}
```

Output:

```
MainThread entered A.foo
Racing Thread entered B.bar
MainThread 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.
```

SURYA Gold
Date Page

Lab Program 10
Interprocess Communication
Output
Press Control - C to stop
put : 0
Intimate consumer
producer waiting
got : 0
Intimate producer
put = 1
Intimate consumer
producer waiting
consumed : 0
got : 1
Intimate producer
consumed : 1
put : 2
Intimate consumer
producer waiting
got : 2
Intimate producer
consumed : 2
put : 3
Intimate consumer
producer waiting
got : 3
Intimate producer
Consumed : 3
put : 4
Intimate consumer
got : 4
Intimate producer
consumed : 4

4-11-24