VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SANCHIT KASHYAP (1BM23CS298)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep 2024-Jan 2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by SANCHIT KASHYAP **(1BM23CS298),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

LABORATORY 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.

```
import  java.util.Scanner;  public
class LabProgram1 {
    public  static  void  main(String[]  args)  {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the coefficients of quadratic equation:");
        int a = sc.nextInt();
        int  b  =  sc.nextInt(); int  c  =
        sc.nextInt(); double r1, r2;

        if(a == 0){
            System.out.println("Please enter valid value"); return;
        }
        else{
            int d = b * b - 4 * a * c; if(d > 0){
                System.out.println("Real and Distinct roots."); r1 =
                (-b + Math.sqrt(d)) / (2
                * a);
                r2 = (-b - Math.sqrt(d)) / (2 * a); System.out.println("Roots
                are: " + r1 + " and " + r2);
            }
            else if(d < 0){
```

```
                System.out.println("Imaginary roots."); d
                = Math.abs(d);
                r1 = (-b + Math.pow(d, 0.5)) / (2 * a);
                r2 = (-b - Math.pow(d, 0.5)) / (2 * a); System.out.println("Roots
                are: " + r1 + " and "
                + r2);
            }
            else{
                System.out.println("Real and Equal roots."); r1 = r2
                = (-b) / (2 * a);
                System.out.println("Roots are: " + r1 + " and "
                + r2);
            }
        }
    }
}
```

```
Enter the coefficients of a,b,c
1 2 3
Roots are imaginary
Root1 = -1.0 + i1.4142135623730951
Root1 = -1.0 - i1.4142135623730951

Process finished with exit code 0
```

```
"C:\Program Files\Java\jdk-23\bin\
Enter the coefficients of a,b,c
2 3 1
Roots are real and distinct
Roo1 = -0.5 Root2 = -1.0

Process finished with exit code 0
```

DATE: PAGE:

LAB PROGRAM - 1

(1) Develop a java program that prints all real sol" to a quadratic eq" $ax^2+bx+c=0$. Read a, b, c and use quadratic formula. If discriminant b^2-4ac is negative display there are no real sol".

```
import java.util.*;
class Quad {
        public static void main() {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter a, b, c");
        a = in.nextInt();
        b = in.nextInt();
        c = in.nextInt();

    if (a==0) {
        System.out.println(" Enter valid value");
        return;
    }

double d = b*b - 4*a*c;
  if (d>0) {
      System.out.println(" Real and distinct roots");
      double r1 = (-b + Math.sqrt(d))/(2*a);
      double r2 = (-b - Math.sqrt(d))/2*a;
  System.out.println("Roots are " + r1 + "and" + r2);
  }
```

DATE:

PAGE:

```
else if (d==0) {
    System.out.println("Roots are real and
    Double s = -b/(2*a); equal");
    System.out.println("Root="+s);
}
else
    System.out.println("No real solution");
    }
}
}
```

O/P

```
Enter coefficients of quadratic equation:
2
3
6
Imaginary roots.
Roots are: 0.81124 and -2.31124

Enter coefficients of quadratic equation
1
5
2
Real and distinct roots.
Roots are: -0.43844 and -4.56155

Enter the coefficients of quadratic equation
1
2
1
Real and Equal roots
Roots are: -1.0 and -1.0
```

26/9/24

LABORATORY 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.

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

    void get(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the usn of student: ");
        usn = sc.next(); System.out.println("Enter the
        name of student: "); name = sc.next();

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

        for(int i=0; i<marks.length; i++){ System.out.println("Enter the
            credits of 3 subjects: "); credits[i] = sc.nextInt();
            System.out.println("Enter the marks of 3 subjects: ");
            marks[i] = sc.nextInt();
        }
    }

    void display(){ System.out.println("Display details
        of Student: "); System.out.println("USN: " +
        usn); System.out.println("Name: "
        + name);
        System.out.println("Marks scored in various
```

```
subjects along with credits are: "); for(int i=0;
        i<marks.length; i++){
            System.out.println(marks[i] + " " + credits[i]);
        }
    }

    double calcSGPA(){
        double sgpa = 0,
        totalCredits = 0; for(int i=0;
        i<marks.length; i++){
            sgpa += (marks[i] / 10) * credits[i];
            totalCredits += credits[i];
        }
        return sgpa / totalCredits;
    }
}
public class Lab { public
    static void
        main(String[] args) { Student s1 =
        new Student(); s1.get();
         s1.display();

        System.out.println(s1.calcS
        GPA());
    }
    }
```

```
Enter the usn of student:
1BM23CS304
Enter the name of student:
Sarim
Enter number of subjects :
3
Enter the credits:
2
Enter the marks:
70
Enter the credits:
3
Enter the marks:
80
Enter the credits:
4
Enter the marks:
90
Display details of Student:
USN: 1BM23CS304
Name: Sarim
Marks scored in various subjects along with credits are:
70 2
80 3
90 4
8.222222222222221
```

DATE:
PAGE:

(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.

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

void get() {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter name, usn and
                        no. of subjects");
    usn = in.nextInt();
    name = in.nextLine();
    num = in.nextInt();
    credits = new int [num];
    marks = new int [num];

    for(int i = 0; i<num; i++) {
    System.out.println("Enter credits for subject" +
                        (i+1) + ":");
    credits[i] = in.nextInt();
    System.out.println("Enter marks for subject" + (i+1)
                        + ":");
    marks[i] = in.nextInt();
    }
}
```

```
void display()
{
System.out.println("USN: " + usn);
System.out.println("name: " + name);

for (int i=0; i< credits.length; i++) {
    System.out.println("Subject" + (i+1) +
    "-credits:" + credits[i] + "marks" +
    marks[i]);
}
}

double calcSGPA() {
    double sgpa = 0;
    int totalCredits = 0;
    for(int i=0; i< credits.length; i++) {
    sgpa += (marks[i] / 10) * credits[i];
    totalcredits += credits[i];
    }
    return sgpa / totalCredits;
}
}

class StudentSGPA {
    public static void main() {
    Student s = new Student();
    s.get();
    s.display();
    System.out.println("SGPA: " + s.calcSGPA());
}
}
```

DATE:
PAGE:

* PROGRAM 2 O/P :-

Enter the usn of student
1BM23CS298
Enter the name of student
Sanchit
Enter the number of subjects
4
Enter credits
4
Enter marks
78
Enter credits
3
Enter the marks
91
Enter the credits
2
Enter marks :
81
Enter credits
4
Enter marks
90
USN : 1BM23CS298
Name : Sanchit
CGPA : 8.230769

3/10/24

LABORATORY 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.

```
import java.util.*;
class Book{ String
    name, author;
    float price;
    int num_pages;

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

    public void setDetails(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the name of the book: ");
        name = sc.next(); System.out.println("Enter the
        name of the author: "); author = sc.next();
        System.out.println("Enter the price of the book: ");
        price = sc.nextFloat(); System.out.println("Enter the
        num of pages: "); numOfPages = sc.nextInt();
    }

    public void getDetails(){
        System.out.println("Name of the book: " + name);
        System.out.println("Name of the author: " + author);
        System.out.println("Price of the book: "
        + price); System.out.println("Number of pages: "
```

```
            + num_pages);
        }

        public String toString(){
            return "Book Name: " + name + " Author Name: " + author + "
  Price: " + price + " Number Of Pages: " + num_pages;
        }
  }
  public class Lab { public
        static void
            main(String[] args) { Scanner sc
            = new Scanner(System.in);
            System.out.println("Enter the number of
            books: "); int n = sc.nextInt(); Book[] books =
            new Book[n];

            for(int i=0; i<n; i++){ books[i].setDetails();
                books[i].getDetails();
            }

for(int i=0; i<n; i++){ System.out.println(books[i].toString());
}
}
```

```
Enter the number of books:
2
Enter the name of the book:
1984
Enter the name of the author:
Anthony
Enter the price of the book:
400
Enter the num of pages:
200
Enter the name of the book:
MockingBird
Enter the name of the author:
Harper
Enter the price of the book:
500
Enter the num of pages:
150
Name of the book: 1984
Name of the author: Anthony
Price of the book: 400.0
Number of pages: 200
Name of the book: MockingBird
Name of the author: Harper
Price of the book: 500.0
Number of pages: 150
```

DATE: PAGE:

(3) Create a class Book which contains four members = name, author, price, num_Pages. Include a constructor to set the values for mem. Include methods to set and get details of the object. Include a toString() method that could display the complete details of the book.

```
Class Book {
    String name, author;
    double price;
    int num_Pages;

    Book (String name; String author, float price, int num_
    Pages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.num_Pages = num_Pages;
    }
public void setDetails() {
        Scanner in = new Scanner (System.in);
        System.out.println("Enter all details");
        author = in.next();
        name = in.nextInt();
        price = in.nextFloat();
        numofPages = in.nextInt();
    }
public void getDetails() {
    System.out.println("Name of book" + name);
    System.out.println("Name of author" + author
    System.out.println("Name of book" + price);
```

```
System.out.println("Number of Pages" + num_pages
}

public String toString() {
    return "Book name: " + name + "Author name:"
+ author + "Price" + price + "Number of Pages:
+ num_pages);
}
}
public class Lab {
        public static void main( ) {
    Scanner in = new Scanner(System.in);
    System.out.println("Enter no of books");
    int n = in.nextInt();
    Book[] books = new Book[n];
    for(int i=0; i<n; i++) {
        book[i].setDetails();
    }
    for(int i=0; i<n; i++) {
        book[i].getDetails();
}
}
```

LABORATORY 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

```
abstract class
  Shape{ int x,
  y;
  Shape(int x,
    int y){
    this.x
    = x;
    this.y =
    y;
  }
  void printArea(){}
}

class          Rectangle
  extends        Shape{
  Rectangle(int  length,
  int breadth){
    super(length, breadth);
  }
  void
    printArea(){
    int area = x
    * y;
    System.out.println("Area of Rectangele = " + area);
```

}

}

```
class Triangle extends Shape{
  Triangle(int base, int
  height){
    super(base, height);
  }
  void printArea(){
    double area = x * y * 0.5;
    System.out.println("Area of
    Triangle = " + area);
  }
}

class          Circle
  extends    Shape{
  Circle(int
  radius){
```

```
    super(radius, 0);
  }
  void printArea(){
    double area = Math.PI * x * x;
    System.out.println("Area of Circle = "
    + area);
  }
}

public class LP4 {
  public static void
    main(String[] args) {
    Rectangle r = new
    Rectangle(10,
    5); r.printArea();
    Triangle t = new
    Triangle(6, 8);
    t.printArea();
    Circle c = new Circle(7);
    c.printArea();
  }
}
```

```
Area of Rectangele = 50
Area of Triangle = 24.0
Area of Circle = 153.93804002589985
```

DATE: PAGE:

(4) Develop a Java program to create an abstract class named shape that contains 2 integers and empty method named printArea(). Provide three classes named rectangle, triangle and circle such that each of these extends the class shape. Calculate area for each shape

```
abstract class shape {

    int x, y;
  void printArea() { }
  }

class Rectangle extends shape {

      Rectangle (int length, int breadth) {
        this.x = length;
        this.y = breadth;
    }

  void printArea() {
      int area = x*y
      System.out.println("Area of Rectangle: " + area
    }
  }
class Triangle extends Shape {
      Triangle (int base, int height) {
        this.x = base;
        this.y = height;
    }
```

```
void printArea() {
    double area = 0.5 * x * y;
    System.out.println("Area of triangle:"+area);
  }
}

class Circle extends Shape {
    Circle(int radius){
        this.x = radius;
    }
  void printArea() {
      double area = Math.PI * radius * radius;
      System.out.println("Area of circle:"+area);
  }
}

Class Main {
    public static void main(String[] args) {
  Shape r = new Rectangle(10,5);
  Shape t = new Triangle(6,8);
  Shape c = new Circle(7);
  r.printArea();
  t.printArea();
  c.printArea();
  }
}
```

Output: Area of rectangle: 50
Area of triangle: 24.00
Area of circle: 153.9380

LABORATORY 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.

```
import java.util.Scanner;
class Account {

  String customerName, accountNumber,

  accountType; double balance;


  public Account(String name, String accNo, String

    type, double bal) { customerName = name;

    accountNumber =

    accNo; accountType

    = type; balance =

    bal;

  }


  public void deposit(double

    amt){ if(amt < 0){

      System.out.println("Deposit amount must be positive.");

    }

    balance += amt;

    System.out.println("Deposit successful. Updated balance: " + balance);

  }
```

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

class SavAcct extends Account {
  static final double interestRate = 0.05;

  public SavAcct(String name, String
    accNo, double bal) { super(name,
    accNo, "Savings", bal);
```

```
  }

  public void calcInterest(int years) {
    double interest = balance * Math.pow(1 +
    interestRate, years) - balance; balance += interest;
    System.out.println("Interest of " + interest + " deposited. Updated balance: " +
    balance);
  }

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

class CurAcct extends Account {
  static final double minBalance =
  10000; static final double
  penaltyCharge = 50;

  public CurAcct(String name, String
    accNo, double bal) { super(name,
    accNo, "Current", bal);
  }
```

```
public void
  withdraw(double amount)
  { if(amount <= balance) {
    balance -= amount;
    System.out.println("Withdrawal successful. Updated balance: " + balance);
```

```
        if(balance <
          minBalance) { balance
          -= penaltyCharge;
          System.out.println("Balance below minimum. Service charge of
" + penaltyCharge + " imposed");
          System.out.println("Updated balance: " + balance);
        }
      }
      else {
        System.out.println("Insufficient balance.");
      }
    }
}

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

    SavAcct sA = new
    SavAcct("ABC", "SA123", 1000);
    CurAcct cA = new
    CurAcct("XYZ", "CA456", 600);

    sA.deposit(200);
    sA.calcInterest(2
    );
    sA.withdraw(500
    );
    sA.displayBalan
```

```
        ce();

        cA.deposit(300)
        ;
        cA.withdraw(7
        0 0);
        cA.displayBalan
        ce();
    }
}
```

```
Deposit successful. Updated balance: 1200.0
Interest of 123.0 deposited. Updated balance: 1323.0
Withdrawal successful. Updated balance: 823.0
Current Account Balance: 823.0
Deposit successful. Updated balance: 900.0
Withdrawal successful. Updated balance: 200.0
Balance below minimum. Service charge of 50.0 imposed
Updated balance: 150.0
Current Account Balance: 150.0
```

(5) WAP to create class Bank that maintains two kinds of account for its customers, one called savings account and other called current account. The current account have checkbook facility but no interest. Current account holders should also maintain min. balance. Do the following tasks.

(a) Accept deposit from customer and update balance.
(b) display balance
(c) compute and deposit interest
(d) Permit withdrawal and update the balance

```
import java.util.*;
class Account {
    String customerName, accountNumber;
    accountType; double balance;
```

```
Public Account (String name, String acNo,
String type, double bal) {
        customerName = name;
        accountNumber = acNo;
        accountType = type;
        balance = bal;
    }
public void deposit(double amt) {
        if (amt<0) {
        System.out.println("Dep. amt must be positive");
        return; }
        balance+=amt;
        System.out.println("Dep successful. Updated balance:"+balance);
    }
public void displayBalance () {
    System.out.println("Current account bal:"+balance);
}}

Class SavAcct extends Account {
    static final double interestRate=0.05;
public SavAcct (String name, String AccNo, double bal){
    super(name, acNo, "Savings", bal);
    }
public void calcInterest (int years) {
    double interest = balance * Math.pow(1+
interestRate, years) - balance;
    balance += interest;
    System.out.println("Interest of "+ ~~interest~~ interest +"
deposited. Updated balance: "+ balance);}

public void withdraw (double amt) {
    if (amt <= balance) {
```

DATE: PAGE:

```
balance -= amount;
System.out.println("Withdrawal successful. Updated
balance: " + balance); }
else {
    System.out.println("Insufficient balance");
}
}

class CurAcct extends Account {
    static final double minBalance = 10000;
    static final double penaltyCharge = 50;

    public CurAcct (String name, String acc No, double bal) {
        super(name, acc No, "current", bal); }

    public void withdraw (double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.(balance);
            if (balance < minBalance) {
                balance -= penaltyCharge;
                System.(penaltyCharge);
            }
        }
        else {
            System.out.println("Insufficient balance");
        }
    }
}

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

```
SavAcct SA = new SavAcct("ABC", "SA123",
1000);
~~SavAcct SA = new SavAcct~~
CurAcct CA = new CurAcct("XYZ", "CA456",
600);
SA.deposit(200);
SA.calcInterest(2);
SA.withdraw(500);
SA.displayBalance();
CA.deposit(300);
CA.withdraw(700);
CA.displayBalance();
}
}
```

Output

~~Enter number of students: 3~~
~~Enter USN:~~
~~1BM23CS240~~
~~Enter name:~~
~~Sonchit~~

Deposit successful. Updated balance: 1200.0
Interest of 123.0 deposited. Updated balance: 1323.0
withdraw successful. updated balance: 823.0
Current Acc balance: 823.0

Deposit successful. Updated balance: 900
withdrawal successful. Updated balance: 827.0
Balance below minimum service charge of 50.0
current balance: 150

LABORATORY 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.

```
// Internals.java

package CIE;


public class Internals{

  public int[] marksInternal = new

  int[5]; public void set(int[]

  marks){

    for(int i=0;

      i<marks.length; i++){

      marksInternal[i] =

      marks[i];

    }

  }

  public int[] get(){

    return

    marksInternal;

  }

}


// Suudent.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 Externals
  extends Student{ public
  int[] marksSEE = new
  int[5];
  public Externals(String usn, String
    name, int sem){ super(usn, name,
    sem);
  }
  public void set(int[]
    marks){ for(int i=0;
    i<marks.length; i++){
      marksSEE[i] = marks[i];
    }
  }
  public int[]
    get(){ return
    marksSEE;
  }
}
```

```
//
Main.ja
```

a

```
import
CIE.*;
import
SEE.*;
import java.util.Scanner;

public class LP6 {
  public static void main(String[]
    args) { Scanner sc = new
    Scanner(System.in); int n;
    System.out.println("Enter the number
    of students: "); n = sc.nextInt();
```

```
Student[] studs = new
Student[n]; Internals[]
internals = new Internals[n];
Externals[] externals = new
Internals[n];

for(int i=0; i<n; i++){
  System.out.println("Enter
  the USN: "); String usn =
  sc.next();
  System.out.println("Enter
  name: "); String name =
  sc.next();
  System.out.println("Enter
  the sem: "); int sem =
  sc.nextInt();

  studs[i] = new Student(usn, name,
  sem); internals[i] = new Internals();
  externals[i] = new Externals(usn, name, sem);

  System.out.println("Enter marks of
  internals: "); int[] marksInt = new
  int[5];
  for(int j=0; j<5; j++){
    marksInt[j] =
    sc.nextInt();
```

```
    }
    internals[i].set(marksInt);

    System.out.println("Enter marks
    of see: "); int[] marksSee = new
    int[5];
    for(int j=0; j<5; j++){
      marksSee[j] =
      sc.nextInt();
    }
    externals[i].set(marksSee);
  }
```

```
    for(int i=0; i<n; i++){
      System.out.println("USN: "
      + studs[i].usn);
      System.out.println("Name: " + studs[i].name);

      int[] totalMarks = new
      int[5];  int[]  marksInt
      = internals[i].get();
      int[] marksSee = externals[i].get();

      for(int j=0; j<5; j++){
        totalMarks[j] = marksInt[j] + marksSee[j]/2;
        System.out.println(totalMarks[j]);
      }
    }
  }
}
```

DATE: PAGE:

(6) Create a package CIE which has 2 classes - Student & 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 ~~author~~ another package SEE which the class External which is derived from Student. This 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.

CIE / Students.java

```
package CIE;
public class Student {
    Public String name;
    Public String usn;
    Public int sem;
  Public Student (String usn, String name, int
   sem) {
      this.usn = usn;
      this.name = name;
      this.sem = sem;
  }
}
```

DATE

PAGE

CIE / Internal.java

```
package CIE;
public class internal {
    public int [] marksInternal = new int [5];
    public void set (int [] marks) {

        for (int i = 0; i < marks.length; i++) {
            marksInternal [i] = marks [i];
        }
    }
    public int [] get () {
        return marksInternal;
    }
}
```

SEE / Externals.java

```
package SEE;
import CIE.Student;
public class Externals extends Student {
    public int [] marksSEE = new int [5];
    public Externals (String usn, String name,
    int sem) {
        super (usn, name, sem);
    }
    public void set (int [] marks) {
        for (int i = 0; i < marks.length; i++) {
            marksSEE [i] = marks [i];
    }
}
```

```
public int[] get() {
    return marks SEE;
  }
}
```

LP6.java

```
import CIE.*;
import SEE.*;
import java.util.Scanner;
Public class LP6 {
    public static void main() {
     Scanner sc = new Scanner(System.in);
     int n;
     System.out.println("Enter the number of
      students:");
      n = sc.nextInt();
     student[] studs = new Student[n];
     Internals[] internals = new Internal[n];
     Externals[] external = new External[

     for(int i=0; i<n; i++) {
      sopln("Enter details");
      String usn = sc.next();
      String name = sc.next();
      int sem = sc.nextInt();
     studs[i] = new Student(usn, name, sem
     internals[i] = new Internals();
     externals[i] = new Externals(usn, name,
```

```
System.out.println("Enter marks of
internals");
int[] marksInt = new Int[5];
for (int j=0; j<5; j++) {
    marksInt[j] = sc.nextInt();
}
internals[i].set(marksInt);
System.out.println("Enter SEE marks");
int[] marksSEE = new Int[5];
for (int j=0; j<5; j++) {
    marksSEE[j] = sc.nextInt();
}
externals[i].set(marksSEE);
}
for (int i=0; i<n; i++) {
    System.out.println("USN" + students[i].usn);
    System.out.println("Name" + students[i].name);
    int[] totalMarks = new Int[5];
    int[] marksInt = internals[i].get();
    int[] marksSEE = externals[i].get();
    for (int j=0; j<5; j++) {
        totalMarks[j] = marksInt[j] +
        marksSEE[j]/2;
        System.out.println(totalMarks[j]);
    }
}
}
}
```

Output :-

```
Enter number of student : 3
Enter the USN :
1BM23CS298
Enter name :
Sanchit
Semester
3
Enter internal marks for 5 courses :
18 20 15 17 19
Enter marks of SEE
70 80 75 60 90

USN : 1BM23CS298
Name : Sanchit
        53
        60
        52
        47
        74
```

LABORATORY 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=father’s age.

```
class WrongAge extends
    Exception { String
    message;

    WrongAge(String
        message) {
        this.message =
        message;
    }

    public String toString() {
        return "WrongAge Exception: " + message;
    }
}

class Father
    { int fAge;
    Father(int age) throws
        WrongAge { if (age < 0) {
            throw new WrongAge("Father's age cannot be negative!");
        }
        fAge = age;
    }
}

class Son extends Father
    { int sAge;

    Son(int fAge, int sAge) throws
        WrongAge { super(fAge);
```

```
            if (sAge < 0) {
                throw new WrongAge("Son's age cannot be negative!");
            }
            if (sAge >= fAge) {
                throw new WrongAge("Son's age cannot be greater than
or equal to Father's age!");
            }

            this.sAge = sAge;
        }
}

public class LP7 { public
    static void
            main(String[] args) { try { Father
                father1 = new Father(40);
                Son son1 = new Son(40, 20);
                System.out.println("Father's age: " + father1.fAge
+ ", Son's age: " + son1.sAge);


                Father father2 = new Father(-5);
            }
            catch (WrongAge e) {
                System.out.println(e);
            }

            try {
                Son son2 = new Son(35, 40);
            }
            catch (WrongAge e) { System.out.println(e);
            }

            try {
                Son son3 = new Son(50, -10);
            }
            catch (WrongAge e) { System.out.println(e);
            }
    }
}
```

```
Father's age: 40, Son's age: 20
WrongAge Exception: Father's age cannot be negative!
WrongAge Exception: Son's age cannot be greater than or equal to Father's age!
WrongAge Exception: Son's age cannot be negative!
```

DATE: PAGE:

LAB-PROGRAM-7

WAP that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class 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.

```
class WrongAge extends Exception {
    String msg;
    WrongAge(String msg) {
        this.msg = msg;
    }

    public String toString() {
        return "WrongAge exception: " + msg;
    }
}

class Father {
    int fage;

    Father(int age) throws WrongAge {
        if(age < 0) {
            throw new WrongAge("Fathers age
            can't be negative");
        }
        fage = age;
    }
}
```

```
class Son extends Father {
    int sAge;

    Son (int fAge, int sAge) throws WrongAge {
        super (fAge);
        if (sAge < 0) {
            throw new WrongAge ("Son's Age cant
                                be negative");
        }

        if (sAge >= fAge) {
            throw new WrongAge ("Son's age cant
                                be negative");
        }
        this.sAge = sAge;
    }
}
public class TryCatch {
    public static void main (String[] args) {
        try {
            Father f1 = new Father (40);
            Son s1 = new Son (40, 20);
        System.out.println ("Fathers age: " + f1.fAge +
                            " Sons Age: " + s1.sAge);

        Father f2 = new Father (-5);
        }

        Catch (WrongAge e) {
            System.out.println (e);
        }
```

```
try {
    Son s2 = new Son (75, 40);
    }

    Catch (WrongAge e) {
        System.out.println (e);
    }

try {
    Son s3 = new Son (50, -10)
    }

    Catch (WrongAge e) {
        System.out.println (e);
    }
}
}
```

Output :-

```
Fathers Age : 40 , Son's Age : 20
WrongAge Exception: Fathers age can't be ne
WrongAge Exception: Son's age cant be greates
or equal to fathers age
WrongAge Exception: Son's age cannot be neg
```

LABORATORY 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.

```
class Threads extends
  Thread{ String s; int
  time; Threads(String
  s, int time){
    this.s = s;
    this.time =
    time;
  }
  public void
    run(){ try{
      while(true){
        System.out.println(s)
        ; Thread.sleep(time
        * 1000);
      }
    }
    catch(InterruptedException ie){
      System.out.println("Thread occurs:
      "
      + ie);
    }
  }
}
public class LP8{
  public static void main(String[] args) {
```

Threads t1 = new Threads("BMS College of

```
    Engineering", 10); Threads t2 = new
    Threads("CSE", 2);

    t1.start();
    t2.start();
  }
}
```

```
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
```

DATE: PAGE:

LAB PROGRAM-8

(Q) WAP which creates two threads, one thread display "BMS College of Engineering" every 10 seconds" and another display "CSE" one every three seconds.

```
class Threads extend Thread {
    String s; int time;
    Threads (String s, int time) {
        this.s = s;
        this.time = time;
    }
    public void run(){
        try {
            while (true) {
                System.out.println(s);
                Thread.sleep(time * 1000);
            }
        }
        catch (InterruptedException ie) {
            System.out.println("Thread occur"
            + ie );
        }
    }

public class LP8 {
    public static void main( String [] args ) {
        Threads t1 = new Threads ("BMS College
of Engineering", 10);
    Threads t2 = new Threads("CSE", 2);
        t1.start();
        t2.start();
    }
```

Output

CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
BMS College of Engineering.

LABORATORY PROGRAM - 9

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

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

public class DivisionMain1 extends Frame implements ActionListener
{
        TextField
        num1,num2;
        Button
        dResult; Label
        outResult;
        String
        out="";
        double
        resultNum
        ; int
        flag=0;

        public DivisionMain1()
        {
                setLayout(new FlowLayout());

                dResult = new Button("RESULT");
                Label number1 = new Label("Number
                1:",Label.RIGHT); Label number2 = new
                Label("Number          2:",Label.RIGHT);
                num1=new TextField(5);
                num2=new TextField(5);
                outResult = new Label("Result:",Label.RIGHT);
```

```
add(number1
);
add(num1);
add(number2
);
add(num2);
add(dResult);
add(outResul
t);

num1.addActionListener(this);
num2.addActionListener(this);
dResult.addActionListener(this)
; addWindowListener(new
WindowAdapter()
{
        public void windowClosing(WindowEvent we)
        {
```

```
                                        System.exit(0);
                                    }
                    });
            }

public void actionPerformed(ActionEvent ae)
{
        int
        n1,n2;
        try
        {
                if (ae.getSource() == dResult)
                {
                        n1=Integer.parseInt(num1.getText());
                        n2=Integer.parseInt(num2.getText());

                        /*if(n2==0)
                                throw new
                        ArithmeticException();*/
                        out=n1+" "+n2+" ";
                        resultNum=n1/n2;
                        out+=String.valueOf(result
```

```
                    Num); repaint();

                }
            }
            catch(NumberFormatException e1)
            {
                flag=1;
                out="Number Format Exception!
                "+e1; repaint();
            }
            catch(ArithmeticException e2)
            {
                flag=1;
                out="Divide by 0 Exception!
                "+e2; repaint();
            }

        }
        public void paint(Graphics g)
        {
            if(flag==0)
            g.drawString(out,outResult.getX()+outResult.getWidth(),outRes
            ult.getY()+outResult. getHeight()-8);
            else
            g.drawString(out,10
            0,200); flag=0;
        }
        public static void main(String[] args)
        {
            DivisionMain1 dm=new
            DivisionMain1(); dm.setSize(new
            Dimension(800,400));
            dm.setTitle("DivisionOfIntegers");
            dm.setVisible(true);
        }
    }
```

Lab-Program-9 24-12-2024

Write a program creates a GUI 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 0, the program would throw an ArithmeticException. display the exception in a message dialog box.

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

public class DivisionMain1 extends Frame implements ActionListener{
    TextField num1, num2;
    Button dResult;
    Label outResult;
    String out = " ";
    double resultNum;
    int flag = 0;
    public DivisionMain1(){
        setLayout(new FlowLayout());
        dResult = new Button("RESULT");
        Label number1 = new Label("Number 1:", Label.RIGHT);
        Label number2 = new Label("Number 2:", Label.RIGHT);
        num1 = new TextField(5);
        num2 = new TextField(5);
        outResult = new Label("Result:", Label.RIGHT);
        add(number1);
        add(num1);
        add(number2);
        add(num2);
        add(dResult);
        add(outResult);
        num1.addActionListener(this);
        num2.addActionListener(this);
        dResult.addActionListener(this);
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
```

```
public void actionPerformed(ActionEvent ae){
    int n1, n2;
    try{
        if(ae.getSource() == dResult){
            n1 = Integer.parseInt(num1.getText());
            n2 = Integer.parseInt(num2.getText());
            /* if(n2 == 0) throw new ArithmeticException(); */
            out = n1 + " " + n2 + " : ";
            resultNum = n1/n2;
            out += String.valueOf(resultNum);
            repaint();
        }
    }
    catch(NumberFormatException e1){
        flag = 1;
        out = "Number Format Exception! " + e1;
        repaint();
    }
    catch(ArithmeticException e2){
        flag = 1;
        out = "Divide by 0 Exception! " + e2;
        repaint();
    }
}
public void paint(Graphics g){
    if(flag == 0){
        g.drawString(out, outResult.getX() + outResult.getWidth(),
                     outResult.getY() + outResult.getHeight()-8);
    }
    else{
        g.drawString(out, 10, 0, 200);
        flag = 0;
    }
}
public static void main(String[] args){
    DivisionMain1 dm = new DivisionMain1();
    dm.setSize(new Dimension(800, 400));
    dm.setTitle("DivisionOfIntegers");
    dm.setVisible(true);
}
}
```

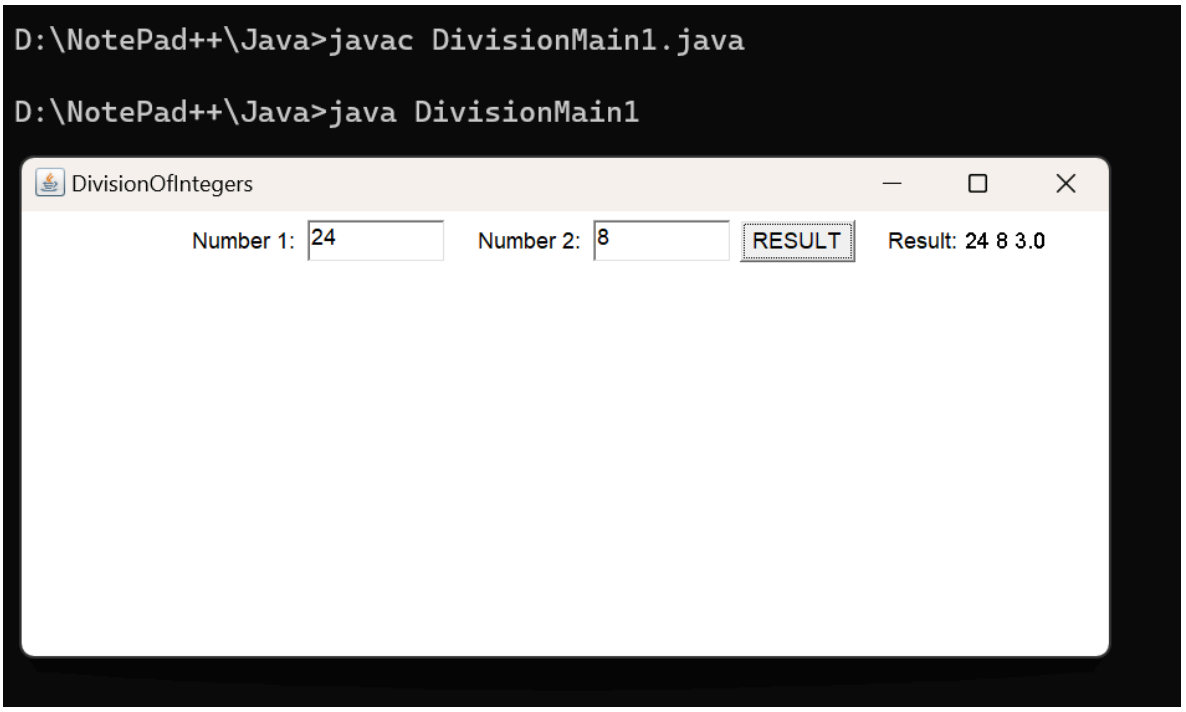
LABORATORY PROGRAM - 10

Demonstrate Interprocess communication and deadlock

```
class Q {
int n;
boolean valueSet = false;

synchronized int get() {
while(!valueSet)
try {
System.out.println("\nConsumer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("\nIntimate Producer\n");
notify();
return n;
}

synchronized void put(int n) {
while(valueSet)
try {
System.out.println("\nProducer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\nIntimate Consumer\n");
notify();
}
}

class Producer implements Runnable {
Q q;
```

```
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(i<15) {
q.put(i++);
}
}
}

class Consumer implements Runnable {
Q q;
Consumer(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
        int i=0;
while(i<15) {
int r=q.get();
System.out.println("consumed:"+r);
i++;
}
}
}

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

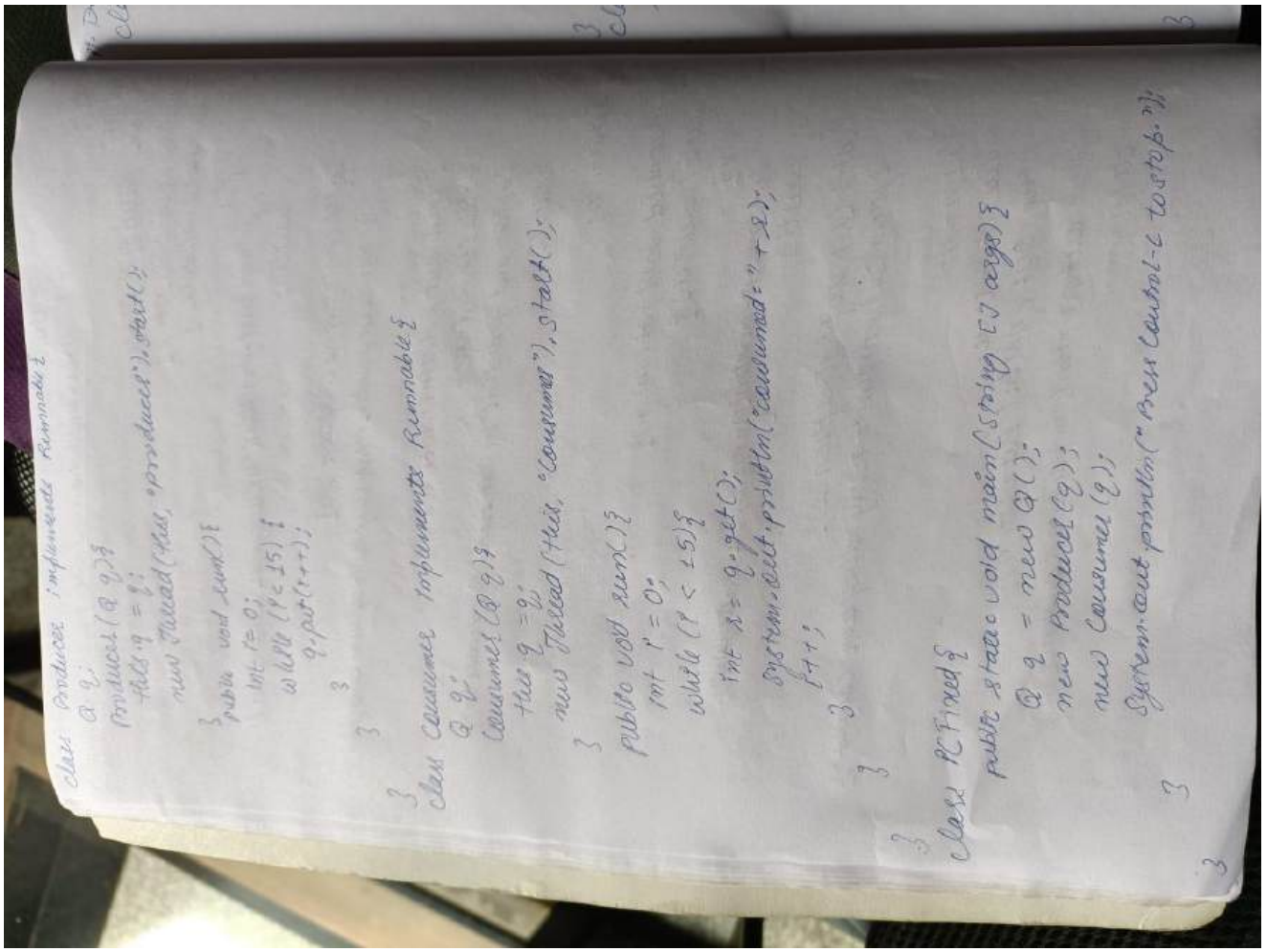
OUTPUT


D:\NotePad++\Java>javac DivisionMain1.java
D:\NotePad++\Java>java DivisionMain1
DivisionOfIntegers
Number 1:
24
Number 2:
8
RESULT
Result: 24 8 3.0

Lab-Program-10 24-12-2024

Demonstration of interprocess Communication

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while(!valueSet) {
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            }
            catch(InterruptedException e){
                System.out.println("Interrupted Exception caught");
            }
            System.out.println("Got: " + n);
            valueSet = false;
            System.out.println("\nIntimate Producer\n");
            notify();
        }
        return n;
    }
    synchronized void put(int n){
        while(valueSet) {
            try{
                System.out.println("\nProducer waiting\n");
                wait();
            }
            catch(InterruptedException e){
                System.out.println("InterruptedException caught");
            }
            this.n = n;
            valueSet = true;
            System.out.println("Put: " + n);
            System.out.println("\nIntimate Consumer\n");
            notify();
        }
    }
```



```
class Producer implements Runnable {
  Q q;
  Producer(Q q) {
    this.q = q;
    new Thread(this, "Producer").start();
  }
  public void run() {
    int r = 0;
    while (true) {
      q.put(r++);
    }
  }
}
class Consumer implements Runnable {
  Q q;
  Consumer(Q q) {
    this.q = q;
    new Thread(this, "Consumer").start();
  }
  public void run() {
    int r = 0;
    while (true) {
      int x = q.get();
      System.out.println("Consumed: " + x);
      r++;
    }
  }
}
class PCTest {
  public static void main(String[] args) {
    Q q = new Q();
    new Producer(q);
    new Consumer(q);
    System.out.println("Press Control-C to stop.");
  }
}
```

ii. Demonstration of deadlock

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

 class B {
  synchronized void bar(A a) {
  String name = Thread.currentThread().getName();
  System.out.println(name + " entered B.bar");
```

```
   try { Thread.sleep(1000); }
catch(Exception e) { System.out.println("B Interrupted"); }
System.out.println(name + " trying to call A.last()"); a.last(); }
synchronized void last() { System.out.println("Inside A.last"); }

}


class Deadlock implements Runnable
 {
  A a = new A(); B b = new B();
  Deadlock( ) {
    Thread.currentThread().setName("MainThread");
    Thread t = new Thread(this, "RacingThread");
    t.start(); a.foo(b); // get lock on a in this thread.
     System.out.println("Back in main thread");
   }
 public void run() { b.bar(a); // get lock on b in other thread.
  System.out.println("Back in other thread");
  }
public static void main(String args[]) { new Deadlock(); }
}
```

Demonstration of deadlock

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

```
class Deadlock implements Runnable {
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);   // get lock on a in this thread
        System.out.println("Back in main thread.");
    }
    public void run() {
        b.bar(a);   // get lock on b in other thread
        System.out.println("Back in other thread");
    }
    public static void main(String[] args) {
        new Deadlock();
    }
}
```