VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

UDAY SHANKAR Y (1BM24CS427)

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 **UDAY SHANKAR Y(1BM24CS427),** 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

Sl. No.	Date	Experiment Title	Page No.
1	26-9-24	Quadratic Equation	1-4
2	30-10-24	Student	5-10
3	19-10-24	Book	11-17
4	24-10-24	Shape Triangle,Rectangle,Circle	18-22
5	7-11-24	Bank	23-36
6	14-11-24	CIE/SEE Package	37-45
7	21-11-24	WrongAge Exception	46-49
8	28-11-24	Threads	49-53
9	-	GUI for Integer Division with Exception Handling	54-59
10	-	Demonstrate Inter process Communication and deadlock	60-69

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;
   class lab1{
      public static void main(String args[]){
         double a;
         double b;
         double c;
         Scanner sx = new Scanner(System.in);
         System.out.print("enter value a : ");
         a = sx.nextDouble();
         System.out.print("enter value b : ");
         b = sx.nextDouble();
         System.out.print("enter value c : ");
         c = sx.nextDouble();

         double discriminant = b*b-(4*a*c);
         if(a == 0){
            System.out.println("it is not a quadratic equation");
         }
         else{
         if(discriminant<0){
            System.out.println("no real roots...");
         }
         else if(discriminant == 0){
            double root = -b/(2*a);
            System.out.println("there is one real root "+root);
         }
         else{
            double sqrtdiscriminant = Math.sqrt(discriminant);
```

```
            double root1 = (-b+sqrtdiscriminant)/(2*a);
            double root2 = (-b-sqrtdiscriminant)/(2*a);
            System.out.println("real roots are "+root1+" and "+root2);
        }
        }
    }
}
```

O/P:

```
PS D:\3rd sem\JAVA\new practice> javac lab1.java
PS D:\3rd sem\JAVA\new practice> java lab1
enter value a : 1
enter value b : -5
enter value c : 6
real roots are 3.0 and 2.0
PS D:\3rd sem\JAVA\new practice> java lab1
enter value a : 1
enter value b : 3
enter value c : 4
no real roots...
```

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

→
```
import java.util.Scanner;
Public class Quadratic {
    Public static void main (String args[]) {
     int a,b,c;
     Scanner sx = new Scanner (System.in);
     System.out.print("enter the value of a :");
     a = sx.nextInt();
     System.out.print("enter the value of b :");
     b = sx.nextInt();
     System.out.print("enter the value of c :");
     c = sx.nextInt();
     sx.close();
     double discriminant = b*b-4*a*c;
     if (discriminant <0) {
          System.out.println("there are no real
     }          solutions...!");
     else {
     double quad = (-b + Math.Sqrt(discriminant))
                   /(2*a);
     double quad1 = (-b - Math.Sqrt(discriminant))
                   /(2*a);
     System.out.println("In Root1 : "+quad+
                   "In Root 2 : "+quad1);
     }
}
```

o/p : else

```
if (discriminant == 0)
{
    System.out.println("root 1 : " + (-b / (2.0
                                * a)));
}
```

-> Enter : a : 2
Enter : b : 3
Enter : c : 4
no real solutions ... !

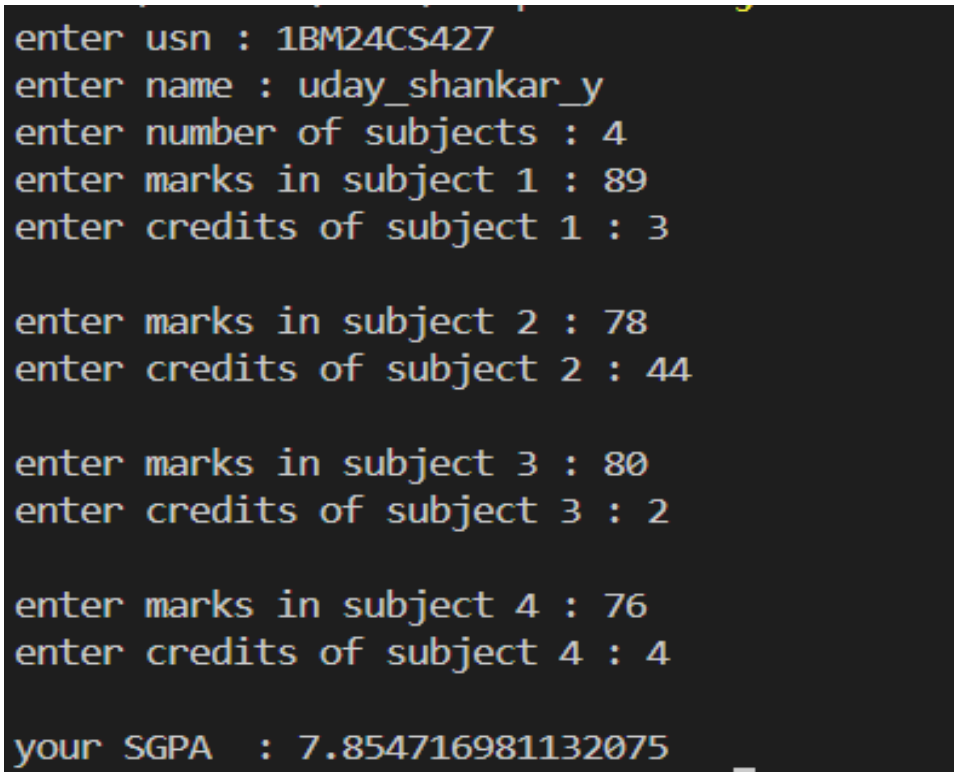
-> Enter : a : 3
Enter b : -7
Enter c : -3
root 1 : 2.70...
root 2 : -0.36...

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.Scanner;

class student{
   String usn;
   String name;
   int []credits;
   int []marks;
   double sgpa ;
   void putd(){
      Scanner sx = new Scanner(System.in);
      System.out.print("enter usn : ");
      usn = sx.next();
      System.out.print("enter name : ");
      name = sx.next();
      int n;
      System.out.print("enter number of subjects : ");
      n = sx.nextInt();
      credits = new int[n];
      marks = new int[n];
      for(int i = 0;i<n;i++){
         System.out.print("enter marks in subject "+(i+1)+" : ");
         marks[i] = sx.nextInt();
         System.out.print("enter credits of subject "+(i+1)+" : ");
         credits[i] = sx.nextInt();
         System.out.println();
      }
      int totalCredits = 0;
      double Wightedgradepoints = 0;
      for(int i = 0;i<n;i++){
```

```
        double gradepoint = marks[i] / 10.0 ;
        Wightedgradepoints += gradepoint * credits[i];
        totalCredits += credits[i];
      }
      sgpa = Wightedgradepoints/totalCredits;
   }
   void display()
      System.out.println("your SGPA : "+sgpa);
   }
}
class lab2{
   public static void main(String args[]){
      student s1 = new student();
      s1.putd();
      s1.display();
   }
}
```

O/P :



2 Develop a java program to creat a class. student with members usn, name, an array credits and an array marks; include methods to accepts and display and details and a method to calculate SGPA of a student.

->

```
import java.util.Scanner;
class stu{
    int max_credits[] = {4,3,4,2};
    int ttl_credits = 13;
    static string subjects[] = {"sdm", "oej", "dbms",
                                "ds"};
    String usn;
    String name;
    static int n = 4;
    int marks_earned[] = new int[n];
    int ~~marks_earn~~ gp[] = new int[4];
    int gp_c[] = new int[4];
    float tt_gp = 0;
    void put() {
```

```
Scanner sx = new Scanner(System.in);
System.out.println("enter the values of s1 :");
System.out.print("name : ");
name = sx.next();
System.out.println("usn : ");
usn = sx.next();
System.out.println("enter the marks:");
for(int i=0; i<stu.n; i++){
System.out.print(stu.subjects[i] + " :");
marks_earned[i] = sx.nextInt();
}
}
void display(){
tt_gp = 0;
System.out.println("\n\n");
System.out.println("name : "+ name + "\nusn
: "+usn);
for(int i=0; i<stu.n; i++){
if(marks_earned[i] >= 90 && marks_earned[i]
<90){
gp[i] = 9;
}
else if(marks_earned[i]>=80 && marks_earned.
[i]<90){
gp[i] = 9;
}
else if(marks_earned[i]>=70 && marks_earned
[i]<80){
gp[i]=8;
}
else if(marks_earned[i]>60 && marks_earned
[i]<70){
gp[i]=7;
}
else if(marks_earned[i]>=50 && marks_earned
[i]<60){
gp[i]=6;
```

```
else if (marks_earned[i] >= 40 && marks_earned[i] <
                50) {
        gp[i] = 5;
    }
else {
    gp[i] = 0;
    }
}
for (int i=0; i<n; i++) {
 gp_c[i] = g[i] * max_credits[i];
    }
 for (int i=0; i<4; i++) {
    tt_gp = tt_gp + gp_c[i];
    }
 float sgpa_r = tt_gp / ttl_credits;
 System.out.print("sgpa :" + sgpa_r);
    }
}
public class Sgpa {
    public static void main (String [] args) {
        Stu s1 = new Stu();
        Stu s2 = new Stu();
    s1.get();
    s1.display();
    }
}
```

Multiple Student object

o/p :

```
Enter number of subjects : 3.
Student name : uday
USN : 24 BECS403
Enter max credits :
    sub1 : 4
    sub2 : 4
    sub3 : 3
Enter marks for each subject :
    marks sub1 : 75
    marks sub2 : 67
```

marks for Sub 3 : 79

Student details :

Name : Uday

USN : 24BECS403

SGPA : ~~7.86~~

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.Scanner;
class book{
   String name;
   String author;
   int price;
   int num_pages;
  public book(String name,String author,int price,int num_pages){
      this.name = name;
      this.author = author;
      this.price = price;
      this.num_pages = num_pages;
   }
   book(){
      name = "";
      author = "";
      price = 0;
      num_pages = 0;
   }
   void set(){
      Scanner sx = new Scanner(System.in);
      System.out.print("enter the name of book : ");
      name = sx.next();
      System.out.print("enter the author of book : ");
      author = sx.next();
      System.out.print("enter the price of book : ");
      price = sx.nextInt();
      System.out.print("enter the number of pages of the book : ");
```

```
        num_pages = sx.nextInt();


    }
    void get(){
        System.out.println(" name : "+name+"\n author : "+author+"\n price : "+price+"\n num of pages : "+num_pages);
    }
    public String toString(){
        return " name : "+name+"\n author : "+author+"\n price : "+price+"\n num of pages : "+num_pages;
    }
}
class lab3{
    public static void main (String args[]){
        int n;
        System.out.print("enter number of books : ");
        Scanner sx = new Scanner(System.in);
        n = sx.nextInt();
        book B[] = new book[n];
        for(int i=0;i<n;i++){
            System.out.println("enter details of book "+(i+1)+" : ");
            B[i] = new book();
            B[i].set();
            // B[i].get();
            System.out.println();
        }
        // for(int i=0;i<n;i++){
        //     B[i].get();
        // }
        for(int i=0;i<n;i++){
            System.out.println("details of book "+(i+1)+" : ");
            System.out.println(B[i]);
```

```
    }
    book b1 = new book("rich dad poor dad","robert",250,150);
    System.out.println();
    System.out.println("parameterized constructor : ");
    System.out.println(b1);
  }
}
```

O/P :

```
PS D:\3rd sem\JAVA\new practice> javac lab3.java
PS D:\3rd sem\JAVA\new practice> java lab3
enter number of books : 2
enter details of book 1 :
enter the name of book : automic_habbits
enter the author of book : james_clear
enter the price of book : 280
enter the number of pages of the book : 145

enter details of book 2 :
enter the name of book : 1984
enter the author of book : george_orwell
enter the price of book : 300
enter the number of pages of the book : 153
```

```
details of book 1 :
 name : automic_habbits
 author : james_clear
 price : 280
 num of pages : 145
details of book 2 :
 name : 1984
 author : george_orwell
 price : 300
 num of pages : 153

parameterized constructor :
 name : rich dad poor dad
 author : robert
 price : 250
 num of pages : 150
```

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.Scanner;
class Book {
    private String name;
    private String author;
    private double price;
    private int numpages;
    Public Book(String name, String author, double
    price, int numpages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numpages = numpages;
    }
    public void setName(string name) {
        this.name = name;
    }
    Public void setAuthor(string author) {
        this.author = author;
```

```
public void setPrice(double price) {
    this.price = price;
}
public void setNumPages(int numPages) {
    this.numPages = numPages;
}

public String getName() {
    return name;
}
public String getAuthor() {
    return author;
}
public String getAuthor() {
    return author;
}
public double getPrice() {
    return price;
}
public int getNumPages() {
    return numPages;
}
@Override
public String toString() {
    return "Book Details : \n" +
           "Name : " + name + "\n" +
           "Author : " author + "\n" +
           "price : $ " + price + "\n" +
           "Number of pages : " + numPages
           + "\n";
}
}
public class Main {
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.println("enter number of Books");
        int n = scanner.nextInt();
        scanner.nextLine();
```

```
Book[] books = new Book [n];
for (int i=0 ; i<n; i++) {
    System.out.println("enter details for book<" +
            (i+1) + ":");
    System.out.println("Enter book name :");
    String name = Scanner.nextLine();
    System.out.println("enter author name :");
    String author = Scanner.nextLine();
    System.out.println("Enter price :");
    double price = Scanner.nextDouble();
    System.out.println("enter number of pages :");
    int numpages = Scanner.nextInt();
    Scanner.nextLine();
    books[i] = new Book(name, author, price,
                        numpages);
}
System.out.println("\n Details of all books:");
for (int i=0; i<n; i++) {
    System.out.println(books[i].toString());
}
Scanner.close();
}
}
```

o/p

```
number of books : 1
Enter details of Book 1 :
Enter book name : qwert
author name : zxcv
Enter price : 560
number of pages : 200
```

Details of all books :

Book details :

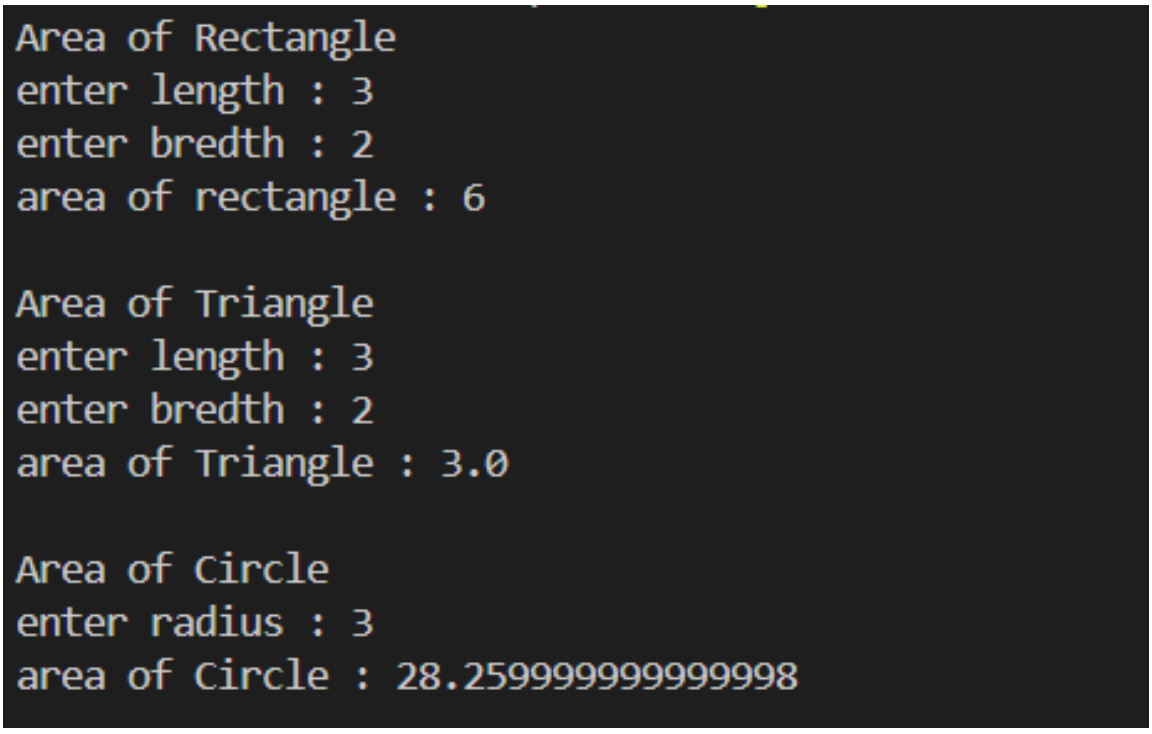
Name : QUERT
Author : ZXCV.
Price : $560.0.
page : 200

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.

```
import java.util.Scanner; abstract class Shape{
   abstract public void printArea(); int i1,i2;
}
class Rectangle extends Shape{ public void printArea(){
   Scanner sx = new Scanner(System.in); System.out.println("Area of
   Rectangle"); System.out.print("enter length : ");
   i1 = sx.nextInt();
   System.out.print("enter bredth : "); i2 = sx.nextInt();
      System.out.println("area of rectangle : "+(i1*i2)+"\n");
   }
}
class Triangle extends Shape{ public void printArea(){
   Scanner sx = new Scanner(System.in); System.out.println("Area of
   Triangle"); System.out.print("enter length : ");
   i1 = sx.nextInt();
   System.out.print("enter bredth : "); i2 = sx.nextInt();
      System.out.println("area of Triangle : "+(0.5*i1*i2)+"\n");
   }
}
class Circle extends Shape{
   public void printArea(){
   Scanner sx = new Scanner(System.in); System.out.println("Area of
   Circle"); System.out.print("enter radius : ");
```

```
    i1 = sx.nextInt();
        System.out.println("area of Circle : "+(3.14*i1*i1)+"\n");
    }
}
class  lab4{
    public static void main(String args[]){ Shape s = new Rectangle();
        s.printArea();
        s = new Triangle(); s.printArea();
        s = new Circle(); s.printArea();
    }
}
```

O/P :

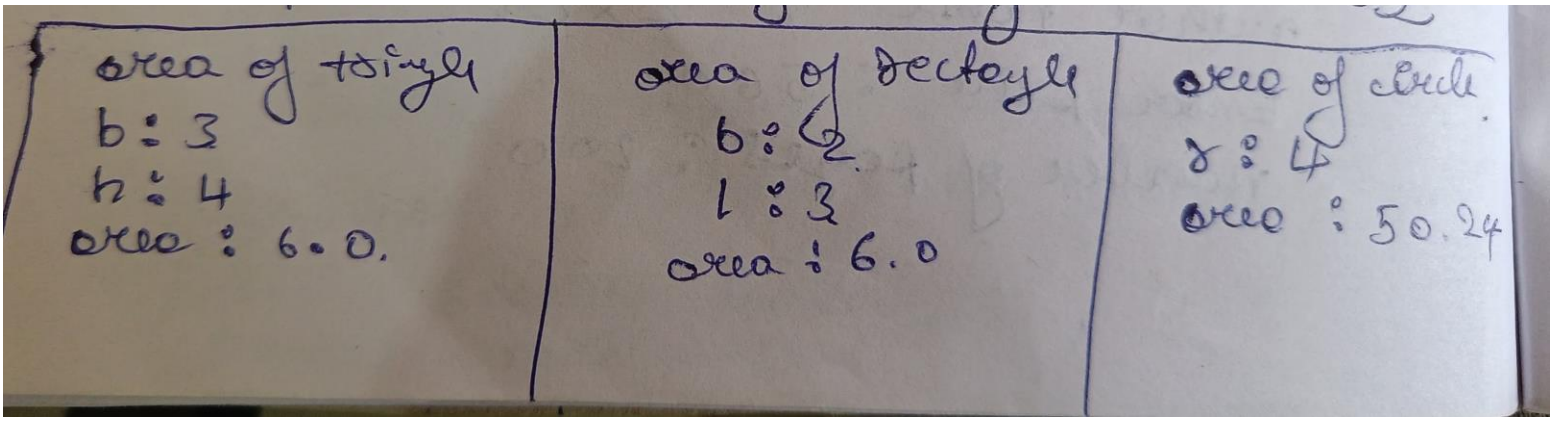


4) Develop a Java pgm to create an abstract class named shape that contains two integers and an empty method named printArea(). provide three classes name Rectangle, Triangle and Circle such that each one of the classes extends the class shape. Each one of the classes contain only the method printArea()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

```
import java.util.Scanner;
abstract class shape {
    abstract void printArea();
}
class triangle extends shape {
    void printArea() {
        System.out.println("\n area of
                Triangle");
Scanner sx = new Scanner(System.in);
int b, h;
double res;
System.out.println("enter base: ");
b = sx.nextInt();
System.out.println("enter height: ");
h = sx.nextInt();
res = (0.5 * b * h);
System.out.println("area of Triangle : " + res);
}
```

area of tri-gle	area of rectangle	area of circle
b: 3	b: 2	r: 4
h: 4	l: 3	area: 50.24
area: 6.0	area: 6.0	

```
class rectangle extends Shape {
    void printArea() {
        System.out.println("In area of rectangle");
        Scanner sx = new Scanner(System.in);
        int b, l;
        double res;
        System.out.print("enter breadth : ");
            b = sx.nextInt();
        System.out.print("enter length : ");
        l . sx.nextInt();
        res = (b*l);
        System.out.println("area of recangl : "+res);
    }
}
class circle extends Shape {
    void printArea() {
        System.out.println("In area of circle : ");
        Scanner sx = new Scanner(System.in);
        int r;
        double res;
        System.out.print("enter radius : ");
        r = sx.nextInt();
        res = (3.14 * r * r);
        System.out.println("area of circle :"+res);
    }
}
class ob_pgm4 {
    public static void main(String args[]) {
        Triangle t = new Triangle();
        rectangle r = new rectangle();
        circle c = new circle();
        t.printArea();
        r.printArea();
        c.printArea();
    }
}
```


area of triangle
b : 3
h : 4
area : 6.0.
area of rectangle
b : 2
l : 3
area : 6.0
area of circle
r : 4
area : 50.24

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 Customer_name;

   Long  account_number;

   String type;

   void getd() {

      Scanner sx = new Scanner(System.in);

      System.out.print("Enter Customer name: ");

      Customer_name = sx.next();

      System.out.print("Enter Customer Account number: ");

      account_number = sx.nextLong();

      System.out.print("Enter Account type (Savings/Current): ");

      type = sx.next();

   }

}

class Sav_acct extends Account {

   double balance;

   double interestRate = 0.05; // Annual interest rate


   void getad() {

      super.getd();
```

```
    }
    void deposit() {
        Scanner sx = new Scanner(System.in);
        System.out.print("Enter Amount to deposit: ");
        double dep_amt = sx.nextDouble();
        balance += dep_amt;
        System.out.println("Your balance after depositing: " + balance);
    }

    void balance() {
        System.out.println("Your current balance: " + balance);
    }
    void compute_deposit_interest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest added: " + interest);
        System.out.println("Your updated balance: " + balance);
    }
    void withdraw() {
        Scanner sx = new Scanner(System.in);
        System.out.print("Enter amount to withdraw: ");
        double wdrwl_amt = sx.nextDouble();
        if (wdrwl_amt > balance) {
            System.out.println("Insufficient balance!");
        } else {
            balance -= wdrwl_amt;
            System.out.println("Amount has been successfully withdrawn!");
            System.out.println("Your updated balance: " + balance);
        }
    }
}
class Curr_acct extends Account {
```

```
double balance;
final double minimumBalance = 500.0;
final double penalty = 50.0;
void getad() {
   super.getd();
}
void deposit() {
   Scanner sx = new Scanner(System.in);
   System.out.print("Enter Amount to deposit: ");
   double dep_amt = sx.nextDouble();
   balance += dep_amt;
   System.out.println("Your balance after depositing: " + balance);
}
void balance() {
   System.out.println("Your current balance: " + balance);
}
void withdraw() {
   Scanner sx = new Scanner(System.in);
   System.out.print("Enter amount to withdraw: ");
   double wdrwl_amt = sx.nextDouble();
   if (wdrwl_amt > balance) {
      System.out.println("Insufficient balance!");
   } else {
      balance -= wdrwl_amt;
      if (balance < minimumBalance) {
         System.out.println("Your balance is below the minimum. Penalty imposed!");
         balance -= penalty;
      }
      System.out.println("Your updated balance: " + balance);
   }
}
```

```
}
class lab5 {
    public static void main(String args[]) {
        Scanner sx = new Scanner(System.in);
        Sav_acct sav = new Sav_acct();
        Curr_acct cur = new Curr_acct();
        while (true) {
            System.out.println("\nPress accordingly:");
            System.out.println("1. New Savings Account");
            System.out.println("2. New Current Account");
            System.out.println("3. Deposit");
            System.out.println("4. Check Balance");
            System.out.println("5. Withdraw");
            System.out.println("6. Calculate Interest (Savings only)");
            System.out.println("7. Exit");
            System.out.print("Enter your choice: ");
            int choice = sx.nextInt();
            switch (choice) {
                case 1:
                    sav.getad();
                    System.out.println("New Savings account has been created!");
                    break;
                case 2:
                    cur.getad();
                    System.out.println("New Current account has been created!");
                    break;
                case 3:
                    System.out.print("Deposit to (1: Savings, 2: Current): ");
                    int depChoice = sx.nextInt();
                    if (depChoice == 1) sav.deposit();
```

```
            else if (depChoice == 2) cur.deposit();
            else System.out.println("Invalid choice!");
            break;
        case 4:
            System.out.print("Check balance for (1: Savings, 2: Current): ");
            int balChoice = sx.nextInt();
            if (balChoice == 1) sav.balance();
            else if (balChoice == 2) cur.balance();
            else System.out.println("Invalid choice!");
            break;
        case 5:
            System.out.print("Withdraw from (1: Savings, 2: Current): ");
            int wdrChoice = sx.nextInt();
            if (wdrChoice == 1) sav.withdraw();
            else if (wdrChoice == 2) cur.withdraw();
            else System.out.println("Invalid choice!");
            break;
        case 6:
            sav.compute_deposit_interest();
            break;
        case 7:
            System.out.println("Exiting. Thank you!");
            return;
        default:
            System.out.println("Invalid choice. Please try again.");
      }
   }
}
```

O/P :

```
Press accordingly:
1. New Savings Account
2. New Current Account
3. Deposit
4. Check Balance
5. Withdraw
6. Calculate Interest (Savings only)
7. Exit
Enter your choice: 1
Enter Customer name: uday
Enter Customer Account number: 7869574456
Enter Account type (Savings/Current): Savings
New Savings account has been created!

Press accordingly:
1. New Savings Account
2. New Current Account
3. Deposit
4. Check Balance
5. Withdraw
6. Calculate Interest (Savings only)
7. Exit
Enter your choice: 3
Deposit to (1: Savings, 2: Current): 1
Enter Amount to deposit: 25
Your balance after depositing: 25.0
```

```
Press accordingly:
1. New Savings Account
2. New Current Account
3. Deposit
4. Check Balance
5. Withdraw
6. Calculate Interest (Savings only)
7. Exit
Enter your choice: 5
Withdraw from (1: Savings, 2: Current): 1
Enter amount to withdraw: 26
Insufficient balance!
```

```
Press accordingly:
1. New Savings Account
2. New Current Account
3. Deposit
4. Check Balance
5. Withdraw
6. Calculate Interest (Savings only)
7. Exit
Enter your choice: 6
Interest added: 1.25
Your updated balance: 26.25

Press accordingly:
1. New Savings Account
2. New Current Account
3. Deposit
4. Check Balance
5. Withdraw
6. Calculate Interest (Savings only)
7. Exit
Enter your choice: 5
Withdraw from (1: Savings, 2: Current): 1
Enter amount to withdraw: 6
Amount has been successfully withdrawn!
Your updated balance: 20.25

Press accordingly:
1. New Savings Account
2. New Current Account
3. Deposit
4. Check Balance
5. Withdraw
6. Calculate Interest (Savings only)
7. Exit
Enter your choice:
```

5) Develop a Java Pgm 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 & 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;
        long accountNumber;
        double balance;
        Public void acceptDeposit(double amount) {
             balance += amount;
        }
```

```
public void display() {
    System.out.println("Balance : "+balance);
}
public void getDetails() {
    System.out.println("Customer Name "+customer
        Name + " In Account Number : "+account
                                    Number);
}
}
class currentAccount extends Account {
    private static final double MIN_BALANCE
                                = 2000;
    private static final double PENALTY=500;

    public void acceptDeposit(double amount) {
        super.acceptDeposit(amount);
        if (balance < MIN_BALANCE) {
            System.out.println("Penalty
        imposed for not maintaining minimum.
                balance.");
        balance -= PENALTY;
    }
}
}
class SavingsAccount extends Account {
    private static final double Interest_
                RATE = 0.05;
    Public void computeAndDepositInterest(int years)
    {
        double compoundInterest = balance *
            Math.
            pow(1+INTREST_RATE, years) -
                balance;
```

```
        balance += compoundInterest;
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal Successful.
            Amount withdrawn: " + amount);
        }
        else {
            System.out.println("Insufficient balance.
            Withdrawal unsuccessful.");
        }
    }
}

public class lab5 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        CurrentAccount currentAccount = new Current
                                        Account();
        SavingsAccount savingsAccount = new Savings
                                        Account();
        System.out.println("Enter customer name: ");
        currentAccount.customerName = savingsAccount.
                customerName = scanner.nextLine();
        System.out.println("Enter account number:");
        currentAccount.accountNumber = savingsAccount.
                accountNumber = scanner.nextLong();
        int choice;
        do {
```

```
System.out.println("\n Menu");
System.out.println(" 1. Deposit to current Account");
System.out.println(" 2. Deposit to Savings Account");
System.out.println(" 3. Display Balance for current Account");
System.out.println(" 4. Display Balance for Savings Account");
System.out.println(" 5. Compute Interest for Savings Account");
System.out.println(" 6. Withdraw from Savings Account");
System.out.println(" 7. Exit");
System.out.println(" Enter your choice:");
choice = Scanner.nextInt();

Switch (choice) {
    case 1:
        S.o.p("Enter amount to deposit to current Account : ");
        double currentDeposit = Scanner.nextDouble();
        currentAccount.acceptDeposit(current deposit);
        break;

    case 2:
        System.out.println("Enter amount to deposit to Savings Account : ");
        double SavingsDepost = Scanner.nextDouble;
        SavingsAccount.acceptDeposit(SavingsDeposit);
        break;
```

```
case 3:
    S.o.p ("Current Account Balance : ");
    currentAccount.DisplayBalance();
    break;

case 4:
    S.o.p ("Savings Account Balance : ");
    SavingsAccount.displayBalance();
    break;

case 5;
    System.out.print("Enter the years for
                    Interest calculation : ");
    int years = Scanner.nextInt();
    SavingsAccount.computeAndDepositInterest(
                                    years);
    S.o.p ("Interest Computed & Deposited");
    break;

case 6:
    S.o.p ("Enter amount to withdraw from
                    SavingsAccount : ");
    double withdrawAmount = Scanner.next
    SavingsAccount.withdraw (withdrawAmount);
                                    Double();
    break;

case 7:
    S.o.p ("Exiting program ...");
    break
default:
    S.o.p ("Invalid choice Try again");
}
while (choice != 7);
Scanner.close();
}
```

o/p: Enter customer name : vijay
Enter account number : 5 6 4 5 7 4 5 6

Menu:
1 - Deposit to current Account
2 - Deposit to Savings Account
3 Display Balance for current Account
4 Display Balance for Savings Account
5. Compute Interest for Savings Account
6. Withdraw from Savings Account
7 - Exit

Enter your choice : 1
Enter amount to deposit to current Accnt : 5000

Menu:
Enter your choice : 2.
Enter amount to deposit to Savings Accnt : 3000

Menu:
Enter your choice : 3.
current Accont Balance:
Balance : 5000.0

Menu:
Enter your choice : 4
Savings Account Balance:
Balance : 3000.0

Menu:
Enter your choice : 5
Enter the number of years for interest calculation : 5
Intrest computed and deposited

Menu:
Choice : 4

Savings Account Balance:

Balance: 3828.84468.

Menu:

choice: 6.

Enter Amount to withdraw: 3000

Successfully withdrawn: ! 300

Menu:

choice: 4.

Balance: 3528.84468

14/11/24

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.

```
package CIE;

public class Personal {
    protected String usn;
    protected String name;
    protected int sem;

    public Personal(String usn, String name, int sem) {
        this.usn = usn;
        this.name = name;
        this.sem = sem;
    }

    public String getUsn() {
        return usn;
    }

    public String getName() {
        return name;
    }

    public int getSem() {
        return sem;
    }
}
```

// CIE/Internals.java

```
package CIE;

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

    public Internals(String usn, String name, int sem, int[] internalMarks) {
        super(usn, name, sem);
        this.internalMarks = internalMarks;
    }

    public int[] getInternalMarks() {
        return internalMarks;
    }
```

```
}
```

// SEE/External.java

```
package SEE;

import CIE.Personal;

public class External extends Personal {
    private int[] seeMarks = new int[5];

    public External(String usn, String name, int sem, int[] seeMarks) {
        super(usn, name, sem);
        this.seeMarks = seeMarks;
    }

    public int[] getSeeMarks() {
        return seeMarks;
    }
}
```

// Main.java

```
import CIE.*;
import SEE.*;

import java.util.Scanner;

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

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

        Personal[] students = new Personal[n];

        // Input for students in CIE and SEE
        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for student " + (i + 1));

            // Input for Personal Information
            System.out.print("USN: ");
            String usn = scanner.next();
            System.out.print("Name: ");
            String name = scanner.next();
            System.out.print("Semester: ");
            int sem = scanner.nextInt();

            // Input for Internal marks (CIE)
```

```
    int[] internalMarks = new int[5];
    System.out.println("Enter internal marks for 5 courses:");
    for (int j = 0; j < 5; j++) {
        internalMarks[j] = scanner.nextInt();
    }

    // Create Internals object
    Internals internals = new Internals(usn, name, sem, internalMarks);

    // Input for External marks (SEE)
    int[] seeMarks = new int[5];
    System.out.println("Enter external marks for 5 courses:");
    for (int j = 0; j < 5; j++) {
        seeMarks[j] = scanner.nextInt();
    }

    // Create External object
    External external = new External(usn, name, sem, seeMarks);

    // Calculate and display the final marks
    System.out.println("\nStudent Details:");
    System.out.println("USN: " + internals.getUsn());
    System.out.println("Name: " + internals.getName());
    System.out.println("Semester: " + internals.getSem());

    System.out.println("\nInternal  Marks:");
    int[] internalMarksArr = internals.getInternalMarks();
    for (int j = 0; j < 5; j++) {
        System.out.print(internalMarksArr[j] + " ");
    }

    System.out.println("\nExternal  Marks:");
    int[] seeMarksArr = external.getSeeMarks();
    for (int j = 0; j < 5; j++) {
        System.out.print(seeMarksArr[j] + " ");
    }

    // Calculate final marks
    int totalMarks = 0;
    for (int j = 0; j < 5; j++) {
        totalMarks += internalMarksArr[j] + seeMarksArr[j];
    }

    System.out.println("\nFinal Marks (Total): " + totalMarks);
}

scanner.close();
```

```
    }
}
```

O/P :

```
Enter the number of students: 1

Enter details for student 1
USN: 1BM21CS001
Name: John
Semester: 5
Enter internal marks for 5 courses:
20 18 22 25 20
Enter external marks for 5 courses:
40 45 35 50 40

Student Details:
USN: 1BM21CS001
Name: John
Semester: 5

Internal Marks:
20 18 22 25 20
External Marks:
40 45 35 50 40
Final Marks (Total): 320
```

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

->

```
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;
    }
    public void displayDetails(){
        System.out.println("USN: "+usn);
        System.out.println("Name: "+name);
        System.out.println("Semester: "+sem);
    }
}
```

Internals.java

```
package CIE;
public class Internals{
    public int[] internalmarks = new int[5];
    public Internals(int[] marks){
        if(marks.length == 5){
            System.arraycopy(marks, 0, internals
                marks, 0, marks.length);
        }
        else{
            System.out.println("Error: Exactly
                5 internal marks are required.");
        }
    public void displayInternalmarks(){
        System.out.println("Internal marks
            :");
        for(int mark : internalmarks){
            System.out.print(mark+" ");
        }
        System.out.println();
    }
}
```

External.java

```
package SEE;
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, marks.length);
    }
    else {
        System.out.println("Error : Exactly
            5 SEE marks are required.");
    }
}
public void displayExternalMarks(){
    System.out.print("SEE marks : ");
    for(int mark : externalMarks){
        System.out.print(mark + " ");
    }
    System.out.println();
}
}
```

~~Main~~ FinalMarks.java

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

```
public class FinalMarks {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of students: ");
        int n = sc.nextInt();
        Student[] students = new Student[n];
        Internals[] internals = new Internals[n];
        External[] externals = new External[n];
        for (int i = 0; i < n; i++) {
            System.out.println("\n Enter details of 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 5 internal marks: ");
            int[] internalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = sc.nextInt();
            }
            System.out.println("Enter 5 SEE marks: ");
            int[] externalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                externalMarks[j] = sc.nextInt();
            }
            students[i] = new Student(usn, name, sem);
            internals[i] = new Internals(
```

```
internalmarks);
externals[i] = new External(usn, name, sem, externalmarks);
    }
System.out.println("\n Final marks of Students:");
        for(int i=0; i<n; i++){
        students[i].displayDetails();
        internals[i].displayInternalMarks();
        externals[i].displayExternalmarks();

System.out.print("Final marks :");
        for(int j=0; j<5; j++){
            int finalmarks = internals[i].internal
        externals[i].externalmarks[j] marks[j] +
        System.out.print(finalMark + " ");
        }
        System.out.println("\n");
        }
    }
```

(5) o/p. Enter no of Students : 1
Enter Student details.
Enter USN : 24BECS403
Enter Name : Uday
Enter Sem : 5
Enter CIE marks : 20 19 15 22 19
Enter 5 see marks : 60 70 65 55 75

Student marks details

USN : 24BECS403.
Name : Uday.
Sem : 5

Final marks :
50 53 58 49 56

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.

```
import java.util.Scanner;
class WrongAge extends Exception{
public WrongAge(String message){
      super(message);
}
}
class father{
   int age;
   father(){
      Scanner sx = new Scanner(System.in);
      System.out.print("enter father age : ");
      age = sx.nextInt();

      try{
         if(age<0){
            throw new WrongAge("age cannot be negative...!");
         }
      }
      catch(WrongAge we){
         System.out.print(we);
      }
   }
}
class son extends father{

      son(){
      super();
      int son_age;
```

```
        Scanner sx = new Scanner(System.in);
        System.out.print("enter son age : ");
        son_age = sx.nextInt();
        try{
           if(son_age > age){
              throw new WrongAge("son age cannot be greater than father age...!");
           }
        }
        catch(WrongAge we){
           System.out.print(we);
        }
        }
}
public class lab7{
   public static void main(String args[]){
      // father fa = new father();
      son sa = new son();
   }
}
```

O/P :

```
enter father age : 45
enter son age : 50
WrongAge: son age cannot be greater than father age...!
PS D:\3rd sem\JAVA\new practice> java lab7
enter father age : -2
WrongAge: age cannot be negative...!enter son age : 4
WrongAge: son age cannot be greater than father age...!
PS D:\3rd sem\JAVA\new practice> java lab7
enter father age : -4
WrongAge: age cannot be negative...!enter son age : 3
WrongAge: son age cannot be greater than father age...!
```

7) Write a program that demonstrates handling of Exception 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 marks class. ; Implement a constructor that uses both father and son's age and throw an Exception if son age >= father age.

-> class wrongAge extends Exception {
    wrongAge (String message) {
        super (message);

}
class Father {
    int age;
    Father (int age) throws wrongAge {
        if (age < 0) {
            throw new wrongAge ("Age cannot be negative");
        }
        this.age = age;
    }
}
class Son extends Father {
    Son (int fage, int sage) throws wrongAge {
        super (fage);
        if (sage >= fage) {

```
throw new ~~Exception~~ WrongAge("
        Son age cant greater than fatherage");
    }
    this.fage = fage;
    this.sage = sage;
  }
}
class reader {
  psvm (String Args[]) {
    for(;;) {
    Scanner sx = new Scanner (System.in);
  try{
    int fage;
    int sage;
    System.o.p("Enter father age");
    fage = sx.nextInt();
    S.o.p("enter son age:")
    sage = sx.nextInt();
```

```
Son s = new Son (fage, sage);
}
catch (WrongAge wa){
    System.out.println(wa);
}
}
}
}
```

op : Enter father's age : - 1
Enter son's age = 1
Error : Age cannot be negative
Enter father's age : 20
Enter son's age : 21
Error ! : son's age > father age

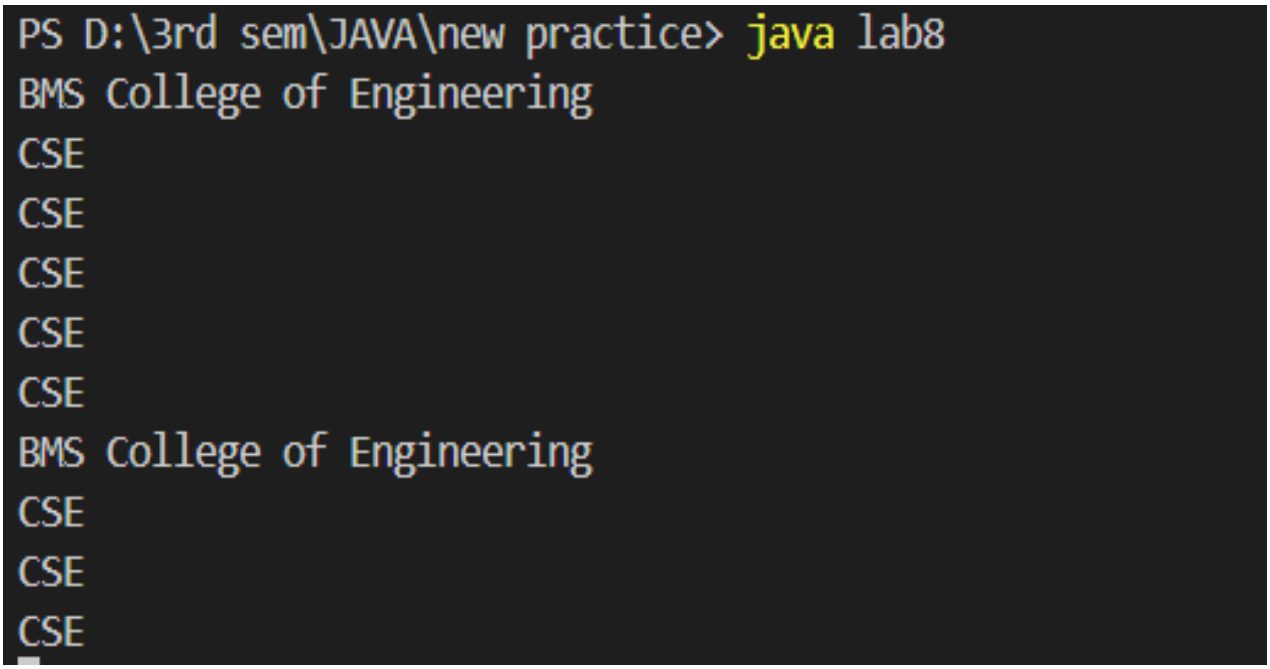
which creates two threads

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 thread1 extends Thread{
   public void run(){
while(true){
   System.out.println("BMS College of Engineering");
   try{
      Thread.sleep(10000);
   }
   catch(InterruptedException e){
      System.out.println(e);
   }
}
   }
}
class thread2 extends Thread{
   public void run(){
while(true){
   System.out.println("CSE");
   try{
      Thread.sleep(2000);
   }
   catch(InterruptedException e){
      System.out.println(e);
   }
}
}
}
class lab8{
   public static void main(String args[]){
```

```
        thread1 t1 = new thread1();
        thread2 t2 = new thread2();
        t1.start();

        t2.start();
    }
}
```

O/P :



8. Write a program which creates two threads, one thread displaying BMS college of Engineering" once every ten seconds and another displays "CSE" once every two seconds

->

```
class A extends Thread {
    public void run() {
        while (true) {
            try {
                System.out.println("BMS college
                        of Engineering");
                Thread.sleep(10000);
            }
            catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}
class B extends Thread {
    public void run() {
        while (true) {
            try {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
```

```
catch (InterruptedException e){
        System.out.println(e);
    }
  }
 }
}
class threads_pgm{
 public static void main(String args[]){
    A obj1 = new A();
    B obj2 = new B();
    obj1.start();
    obj2.start();
  }
}
```

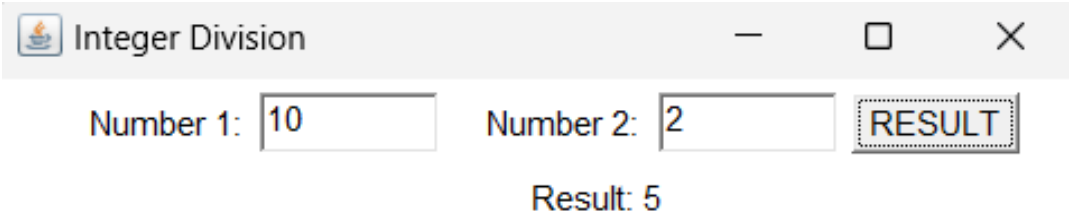
o/p: BMS college of Engineering
CSE
CSE
CSE
CSE 25/12/ten
CSE
BMS college of Engineering.

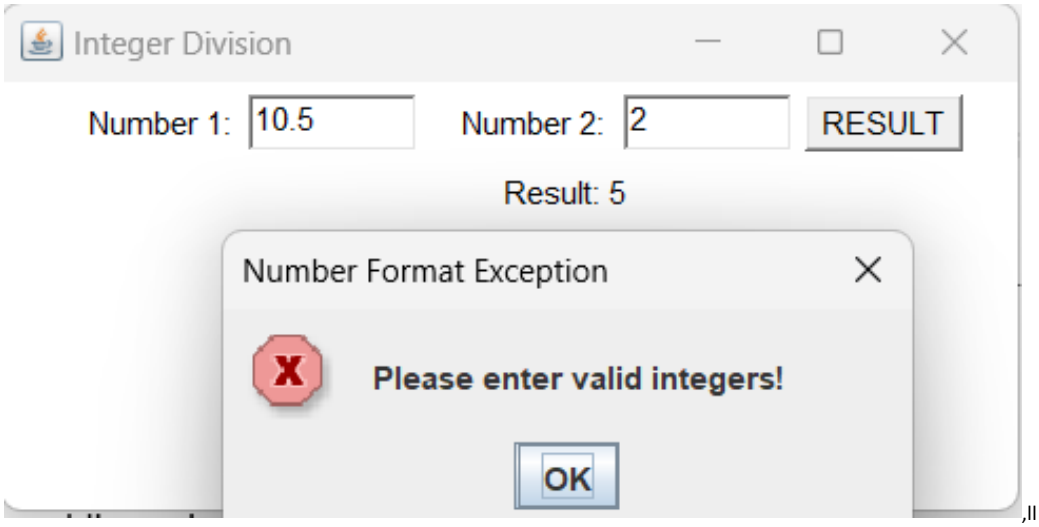
9.Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a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

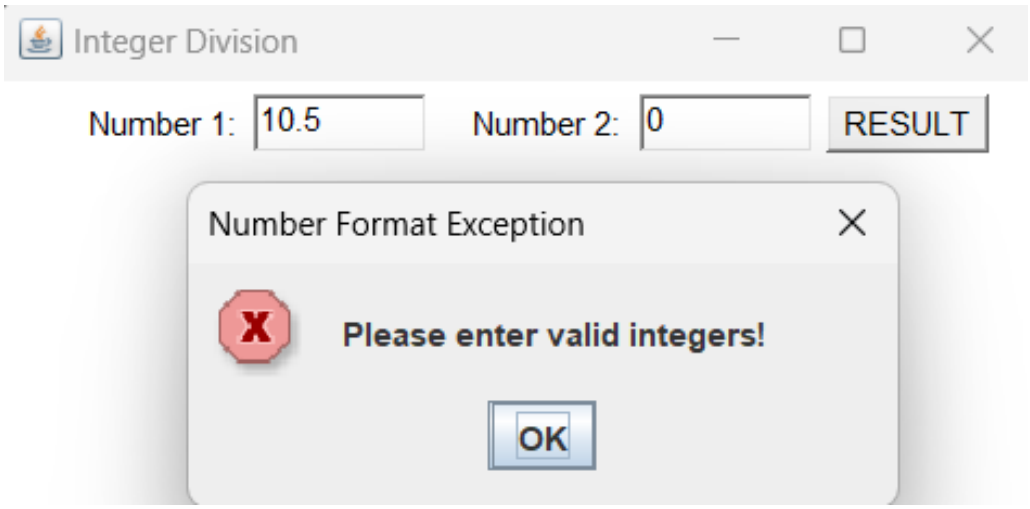
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class DivisionMain1 extends Frame implements ActionListener {
   TextField num1, num2;
   Button dResult;
   Label outResult;

   public DivisionMain1() {
      // Layout setup
      setLayout(new FlowLayout());

      // Components
      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);

      // Adding components to frame
      add(number1);
      add(num1);
      add(number2);
      add(num2);
      add(dResult);
      add(outResult);

      // Event listeners
      dResult.addActionListener(this);

      // Window close operation
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent we) {
            System.exit(0);
         }
      });

      // Frame properties
```

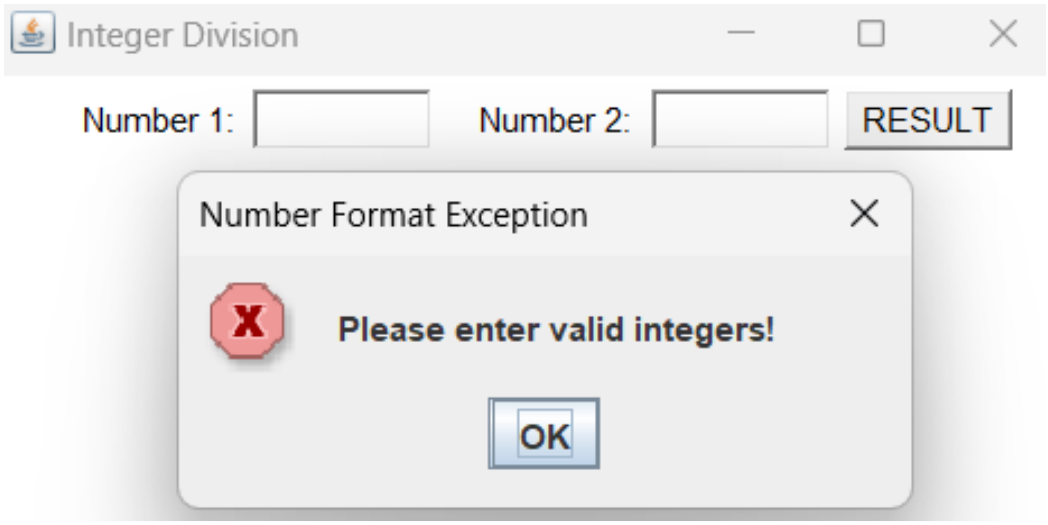
```
        setTitle("Integer Division");
        setSize(400, 200);
        setVisible(true);
    }

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

                // Check for division by zero
                if (n2 == 0) {
                    throw new ArithmeticException("Cannot divide by zero.");
                }

                // Perform division and display result
                int result = n1 / n2;
                outResult.setText("Result: " + result);
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Please enter valid integers!", "Number Format
Exception", JOptionPane.ERROR_MESSAGE);
        } catch (ArithmeticException e) {
            JOptionPane.showMessageDialog(this, e.getMessage(), "Arithmetic Exception",
JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void main(String[] args) {
        new DivisionMain1();
    }
}
```

o/p :




Integer Division
Number 1: 10.5
Number 2: 2
RESULT
Result: 5
Number Format Exception
Please enter valid integers!
OK


Integer Division
Number 1: 10.5
Number 2: 0
RESULT
Number Format Exception
Please enter valid integers!
OK


Integer Division
Number 1:
Number 2:
RESULT
Number Format Exception
Please enter valid integers!
OK

9. Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num 1 & Num 2. The division of Num 1 & Num 2 is displayed in the Result field when the divide Button is clicked. If Num 1 or Num 2 were not an Integer the program would throw a NumberFormat Exception if Num 2 were zero, the program would throw an Arithmetic Exception Display the Exception in a Dialog box

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class DivisionMain1 extends Frame
        implements ActionListener {
    TextField num1, num2;
    Button dResult;
    Lable outResult;

    public DivisionMain1 ( ) {
        setLayout (new FlowLayot ( ));
        dResult = new Button ("Result");
        Lable number1 = new Lable ("Number1
                : ", Label.RIGHT);
        Label number2 = new Lable ("Number2
                : ", 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);
dResult.addActionListener(this);
addWindowListener(new windowAdapter() {
    public void windowClosing(WindowEvent we
    {
        System.exit(0);
    }
});
setTitle("Integer Division");
setSize(400, 200);
setVisible(true);
}
public void actionPerformed(ActionEvent ae) {
try {
if(ae.getSource() = dResult) {
int n1 = Integer.parseInt(num1.getText());
int n2 = Integer.parseInt(num2.get
                          Text());
if(n2 == 0) {
throw new ArithmeticException("
        cannot divide by zero");
}
int result = n1 / n2;
outResult.setText("Result: " + result
);
} catch(NumberFormatException e) {
JOptionPane.showMessageDialog(this,
please enter valid Integers!", "Number Format
Exception", JOptionPane.ERROR_MESSAGE);
} catch(ArithmeticException e) {
JOptionPane.showMessageDialog(this, e.getMessage()
, "Arithmetic Exception", JOptionPane.
                    ERROR_MESSAGE);
```

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

o/p:

Entering valid integers and clicking "RESULT" display the result in the Result. Label.

Invalid inputs (non-integer values or division by zero) show appropriate error message in a dialog box.

10. Demonstrate Inter process 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++;
      }
   }
}
public class lab10 {
   public static void main(String[] args) {
      Q q = new Q();
      new Producer(q);
      new Consumer(q);
      System.out.println("Press Control-C to stop.");
   }
}
```

O/P :

```
PS D:\3rd sem\JAVA\new practice> java lab10
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
```

DeadLock :

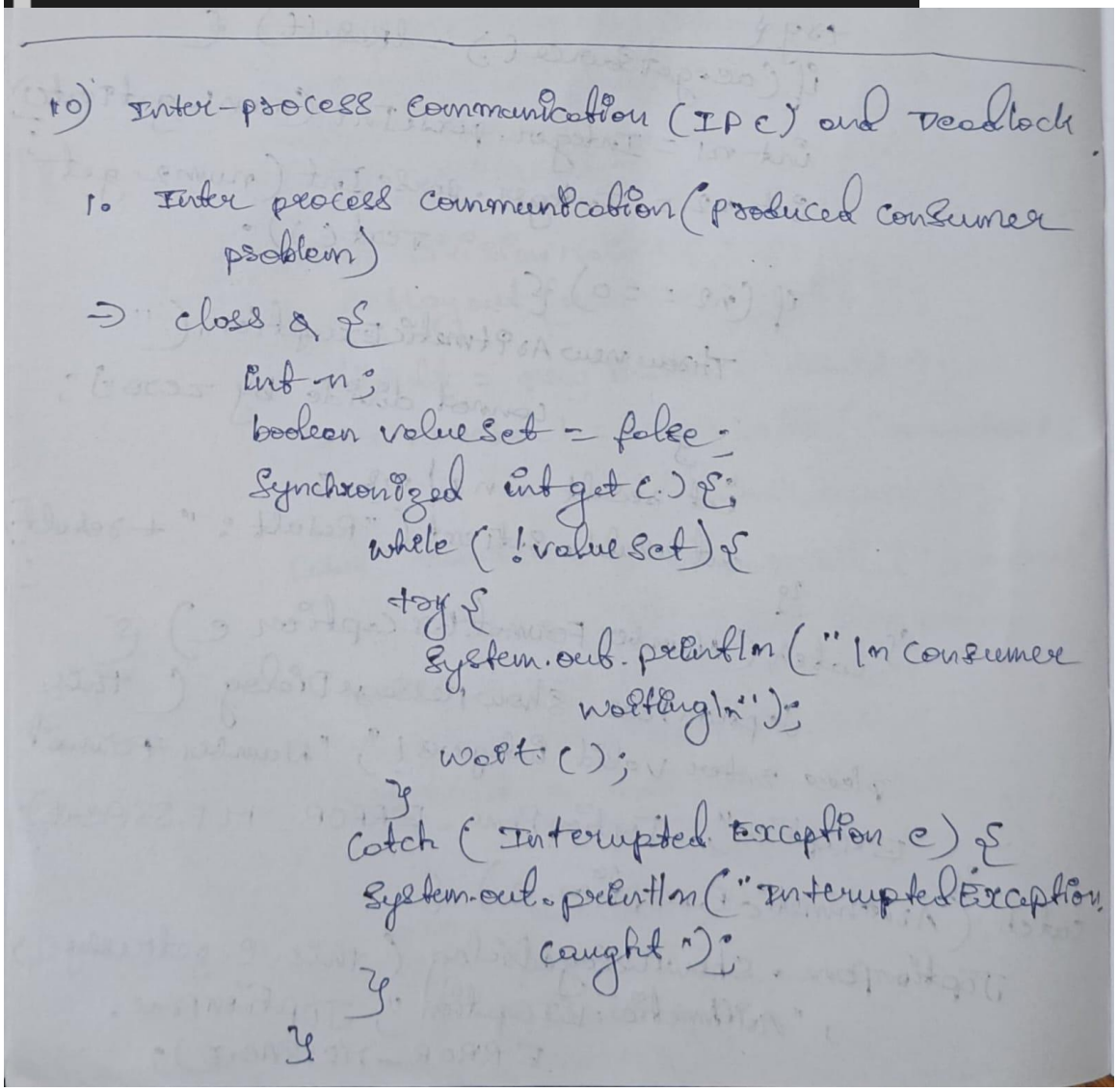
```
class A {
   synchronized void foo(B b) {
      String name = Thread.currentThread().getName();
      System.out.println(name + " entered A.foo");
      try {
```

```
            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 B.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); // Main thread locks A and tries to lock B.
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // RacingThread locks B and tries to lock A.
        System.out.println("Back in other thread");
    }

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

}

O/P :

```
PS D:\3rd sem\JAVA\new practice> java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
```



10) Inter-process communication (IPC) and Deadlock

1. Inter process communication (produced consumer problem)

⇒ class Q {

int n;

boolean valueSet = false;

synchronized int get() {

while (!valueSet) {

try {

System.out.println("In consumer waiting\n");

wait();

}

catch (InterruptedException e) {

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

}

}

```
System.out.println("Got : " + n);
valueSet = false;
System.out.println("In Intimate Producer \n");
notify();
return n;
}
Synchronized void put(int n).
{
while(valueSet){
try{
System.out.println("In produce waiting\n");
wait();
}
catch(InterruptedException e){
System.out.println("Interrupted.
Exception caught");
}}
this.n = n;
valueSet = true;
System.out.println("Put : "+n);
System.out.println("In Intimate consum
er \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 x = q.get();
            System.out.println("Consumed : "+x);
            i++;
        }
    }
}
public 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");
    }
}
```

o/p:
```
Put : 0
Interrupted Consumer
Got : 0
Consumed : 0
Interrupted Producer
...
```

2. 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(name + " 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 B.last");
}
}
class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() {
        Thread.currentThread().setName
            ("mainthread");
        Thread t = new Thread(this, "Rac
            -ngthread");
        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();
    }
}
```

o/p : Main thread entered A.foo.
Racing Thread entered B.bar
Main thread trying to call B.last()
Racing thread trying to call A.last()