BMS COLLEGE OF ENGINEERING

(AUTONOMOUS COLLEGE, AFFILIATED TO VTU)

Bull Temple Road, Basavangudi, Bangalore -560019

2023-2024

Lab Observation

IN

Object Oriented Programming

In Java

BY

SARVESH RASTOGI

USN-1BM22CS247

INDEX

1) Sample Prog. - 5/12/23 (10)
2) Quadratic eq. - 12/12/23 (10)
3) SGPA - 19/12/23 (10)
4) Book details. - 26/12/2310 — 26/12/2023
5) Area & Shapes - 2/1/24 — 2/1/2024
6) Bank - 9/1/24 - 10
7) Strings - 16/1/24 - 10
8) Packages - 23/1/24 - 10
9) ~~Se~~ father-son - 30/1/24 - 10 — 6-2-24
10) ~~Ex~~. Multithreading (BMSCE) - 6-2-24
11) IPC and Deadlock - 13-2-24 — 13/2/24
12) Diwali - 20-2-24 — 20-2-24

demo1.java

```
import java.util.*;
class demo1
{
    public static void main(String args[])
    {
        System.out.println("hello world");
    }
}
```

hello world.

---

sum.java

```
import java.util.*;
class sum
{
    public static void main(String args[])
    {
        int n1=5, n2=10, sum;
        sum = n1 + n2;
        System.out.println(sum);
    }
}
```

15

---

multiply.java

```
import java.util.*;
class multiply
{
    public static void main(String args[])
    {
        int n1=5, n2=10, mul;
        mul = n1 * n2;
        System.out.println(mul);
    }
}
```

50

rel_opt.java

```
import java.util.*;
class rel_opt {
    public static void main (String args[])
    {
        int n1 = 5, n2 = 10;
        if (n1 > n2) {
            System.out.println("n1 is greater");
        }
        else if (n1 < n2) {
            System.out.println("n2 is greater");
        }
        else {
            System.out.println("both are equal");
        }
    }
}
```

⇒ n2 is greater

~~Date~~ Develop a Java prog. that prints all real solution of the quad. eqn. $ax^2+bx+c=0$. Read in a, b, c and use the quad. formula.

```
import java.util.Scanner;
class Quadratic
{
    int a,b,c;
    double x1, x2, d;
    void getd()
    {
        Scanner s= new Scanner(System.in);
        System.out.println("Enter the coefficients of
                                a, b, c");
        a= s.nextInt();
        b= s.nextInt();
        c= s.nextInt();
    }
    void compute()
    {
        while(a==0)
        {
            System.out.println("not a quadratic");
            System.out.println("Enter a non zero
                                value for a:");
            Scanner s= new Scanner(System.in);
            a= s.nextInt();
        }
        d= b*b -4*a*c;
        if (d==0)
        {
            x1= (-b)/(2*a)
            System.out.println("Root are real and
                                equal");
            System.out.println("Root 1 = root 2= "+x1);
        }
```

```
else if (d>0)
{
    x1 = ((-b)+(Math.sqrt(d)))/(double)(2*a);
    x2 = ((-b)-(Math.sqrt(d)))/(double)(2*a);
    System.out.println("Roots are real and distinct");
    System.out.println("Root1 = " + x1 + "Root2 = " + x2);
}
else if (d<0)
{
    System.out.println("Roots are Imaginary");
    x1 = (-b)/(2*a);
    x2 = Math.sqrt(-d)/(2*a)
    System.out.println("Root1 = " + x1 + " + i " + x2);
    System.out.println("Root2 = " + x1 + " - i " + x2);
}
}
class QuadraticMain
{
public static void main(String args[])
{
    Quadratic q = new Quadratic();
    q.getd();
    q.Compute();
}
}
```

output

Enter the coefficients of a, b, c
1
2
-1

roots are equal
root1 = root2 = 1

Sourav Rastogi / 1BM22CS241

rectangle.java

```
class rectangle {
    public static void main(String args[]) {
        int l, b;
        l = Integer.parseInt(args[0]);
        b = Integer.parseInt(args[1]);
        int a = l*b;
        System.out.println("length of rectangle=" + l);
        System.out.println("breadth of rectangle=" + b);
        System.out.println("area of rectangle=" + a);
    }
}
```

⇒ javac rectangle.java

java rectangle 10 12

length of rectangle = 10

breadth of rectangle = 12

area of rectangle = 120.

---

hello.java

```
import java.util.Scanner;
class hello {
public static void main(String args[]) {
    int a; float b; String s;
    Scanner in = new Scanner(System.in);
    System.out.println("Enter a string");
    s = in.nextLine();
    System.out.println("Enter an integer");
    a = in.nextInt();
    System.out.println("You entered integer " + a);
    System.out.println("Enter a float");
    b = in.nextFloat();
```

```
        System.out.println("You entered float "+b);
    }
}
```

⇒
```
Enter a string
Ramesh
You entered string Ramesh
Enter an integer
10
You entered integer 10
Enter an float
23.5
You entered integer 23.5
```

Q. Develop a Java prog. to create a class Student with member 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 Subject {
    int subjectMarks;
    int credits;
    int grade;

Class Student {
    String name;
    String usn;
    double SGPA;
    Scanner in;
    Subject[] subjects;
```

```
Student() {
    int i;
    subjects = new Subject[8];
    for (i=0; i<8; i++)
        subject[i] = new Subject();
    s = new Scanner(System.in);
}
void getStudentDetails() {
    System.out.print(s:"Enter student
                        name: ");
    name = s.nextLine();
    System.out.print(s:"Enter student
                        USN: ");
    usn = s.nextLine();
}
void getMarks() {
    for(int i=0; i<8; i++) {
        System.out.println("Enter details
            for Subject" + (i+1));
        System.out.println(s:"Enter
                        marks:");
        subjects[i].subjectMarks = s.nextInt();
        System.out.print(s:"Enter credits:");
        subjects[i].credits = s.nextInt();

        If (subjects[i].subjectMarks >=90)
            ~~{subject.grade~~
            { subjects[i].grade = 10; }
        else if (subjects[i].subjectMarks >=80)
            { subjects[i].grade = 9; }
        else if (subjects[i].subjectMarks >= 70)
            { subjects[i].grade = 8; }
        else if(subjects[i].subjectMarks >=60)
            { subjects[i].grade = 7; }
```

```
        else if (subjects[i].subjectMarks >= 50){
            subjects[i].grade = 6; }
        else if (subjects[i].subjectMarks >= 40){
            subjects[i].grade = 5; }
        else {
            subjects[i].grade = 0; }
    }
}

void computeSGPA() {
    double totalCredits = 0
    double weightedSum = 0
    for(int i=0; i<8; i++) {
        totalCredits += subjects[i].credits;
        weightedSum += subjects[i].grade *
                       subjects[i].credits; }
    SGPA = weightedSum / totalCredits;
}

void displayResult() {
System.out.println("\n Student Details:");
                   ("Name:" + name);
                   ("USN:" + usn);
                   ("SGPA:" + SGPA);
}
}
```

```
Public class Main {
    public static void main (String[] args) {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        s1.displayResults();
    }
}
```

output:-

Enter student name: Sanvath
Enter student usn: 1BM22CS244
Enter details for subject 1
Enter marks: 91
Enter credits: 4
:
: 91
: 4
: 85
: 3
: 85
: 3
: 92
: 3
: 95
: 1
: 95
: 1
: 86
: 1

Student Details:

Name : Sanvak

USN: 1BM22CS214

SGPA: 9.45

14-12-23

BookMain.java

```
import java.util.Scanner;
Class Book {
    String name;
    String author;
    int price;
    int numPages;
    Public Book(String name, String author, int price,
                int numPages) {
        this.name = name;
        this.author = author;
        this.price = price
        this.numPages = numPages;
    }
    Public String toString() {
        String bookDetails = "Book name:" + this.
            name + "\n" + "Author name:" +
        this.author + "\n" + "Prices:" +
        this.price + "\n" + "Number of
        Pages:" + this.numPages + "\n";
        return bookDetails;
    }
}
public class BookMain {
    public static void main(String args[]) {
        Scanner s = new Scanner(System.in);
        int n;
        String name;
        String author;
```

```
int price;
int numPages;

System.out.println("number of books");
n = s.nextInt();

Book b[] = new Book[n];
for(int i=0; i<n; i++){
    System.out.println("Enter details for
        Book" + (i+1));
    System.out.print("Name: ");
    name = s.next();
    System.out.print("Author: ");
    author = s.next();
    System.out.print("Price: ");
    price = s.nextInt();
    System.out.print("No. of pages: ");
    numPages = s.nextInt();
    b[i] = new Book(name, author, price,
                    numPages);
}

System.out.println("Details of the books:");
for(int i=0; i<n; i++){
    System.out.println("Book" + (i+1) +
        ":\n" + b[i].
        toString());
}
}
```

Output :-

number of books

2

Enter details for Book 1

Name : Apple

Author : Newton

Price : 256

No. of Pages : 250

Enter details for Book 2

Name : India

Author : Sarvesh

Price : 999

No. of Pages : 100

Details of the Books :

Book 1 :

Book name : Apple

Author name : Newton

Price : 256

No. of pages : 250

Book 2 :

Book name : India

Author name : Sarvesh

Price : 999

No. of Pages : 100

26/12/23

To create an abstract class named Shape that contains two integers and an empty method 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;
class InputScanner {
    Scanner scanner = new Scanner(System.in);
    int getIntInput(String message) {
        System.out.println(message);
        return scanner.nextInt();
    }
}
abstract class Shape extends InputScanner {
    int side1;
    int side2;
    abstract void printArea;
}
class Rectangle extends Shape {
    void printArea() {
        side1 = getInput("Enter length: ");
        side2 = getInput("Enter width: ");
        int area = side1 * side2;
        System.out.println("Area of the rectangle: " +
                           area);
    }
}
```

```
class Triangle extends Shape {
    void printArea() {
        side1 = getIntInput("Enter base: ");
        side2 = getIntInput("Enter height: ");
        double area = 0.5 * side1 * side2;
        System.out.println("Area of the
                            triangle: " + area);
    }
}
class Circle extends Shape {
    void printArea() {
        side1 = getIntInput("Enter radius of the
                             circle: ");
        double area = Math.PI * side1 * side1;
        System.out.println("Area of the circle:"+
                           area);
    }
}
public class Area {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle();
        Triangle triangle = new Triangle();
        Circle circle = new Circle();
        rectangle.printArea();
        System.out.println();
        triangle.printArea();
        System.out.println();
        circle.printArea();
    }
}
```

Output:

```
Enter length: 12
Enter width: 5
```

Area of the rectangle : 60

Enter base : 10
Enter height :
Area of the triangle : 25

Enter radius :
Area of the Circle : 78.539981

2/12/2024

Bank.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class Account {
    String customerName;
    long accountNumber;
    String accountType;
    double balance;
    public Account(String customerName, long
        accountNumber, String accountType) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType
        this.balance = 0.0 }
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit success.
            Updated balance: $" + balance);
    }
    public void displayBalance() {
        System.out.println("Current balance:
            $" + balance); }
    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawal
            successful. Updated balance: $" +
                balance); }
```

```
        else {
            System.out.println("Insufficient funds
                for withdrawal");
        }
    }

class CurrAcct extends Account {
    double minBalance = 500.00;
    public CurrAcct(String customerName, long
            accountNumber) {
        super(customerName, accountNumber,
            "Savings");
    }
    public void computeAndDepositInterest(int timePeriod)
    { double interest =
    public void checkMinBalance() {
        if (balance < minBalance) {
            double penality = 50.0;
            balance -= penality;
            System.out.println("Minimum
                balance not
                maintained");
        }
    }
}

class SavAcct extend Account{
    double interestRate = 0.05;
    public SavAcct(String customerName, long
            accountNumber) {
        super(customerName, accountNumber,
            "Savings");
```

```
public void computeAndDepositInterest(int timePeriod)
{ double interest = balance * interestRate * timePeriod;
  balance += interest;
  System.out.println("Interest computed and
      deposited. Upated balance: $" + balance); }
public void withdraw(double amount) {
  if (amount <= balance) {
    balance -= amount;
    System.out.println("withdrawal
      successful. updated balance: $"
      + balance); }
  else {
    System.out.println(" Insufficient funds
        for withdrawal.");
  }
}
}
Public class Bank {
  public static void main (String[] args)
  throws IOException {
    BufferedReader reader = new BufferedReader(
        new InputStreamReader(System.in));
    System.out.print("Enter S for saving and
        C for current account : ");
    String accountType = reader.readLine();
    if(accountType.equalsIgnoreCase("C")) {
      System.out.print("Enter your name:");
      String name = reader.readLine();
```

```
System.out.print("Enter your account number:");
long accNo = Long.parseLong(reader.readLine());

CurrAcct currentAccount = new CurrAcct(
                              name, accNo);

System.out.print("Do you want to deposit?
                  Type 'yes' :");
String depositChoice = reader.readLine();
if(depositChoice.equalsIgnoreCase("yes")){
    System.out.print("Enter the amount
                      to deposit:");
    double depositAmmount = Double.parseDouble
                            (reader.readLine());
    currentAccount.deposit(depositAmount);
}
System.out.print("Do you want to withdraw?
                  Type 'yes':");
String withdrawChoice = reader.readLine();
if(withdrawChoice.equalsIgnoreCase("yes")){
    System.out.print("Enter the amount to
                      withdraw:");
    double withdrawAmount = Double.parse
    Double(reader.readLine());
    currentAccount.withdraw(withdrawAmount);
}
currentAccount.displayBalance();
currentAccount.checkMinBalance();}
```

```
else if (accountType.equalsIgnoreCase("S")) {
    System.out.print("Enter your name:");
    String name = reader.readLine();
    System.out.print("Enter your acc. no.:");
    long accNo = Long.parseLong(reader.readLine());
    SavAcct savingAccount = new SavAcct(name,
                                        accNo);
    System.out.print("Do you want to deposit?
                      Type 'yes':");
    String depositChoice = reader.readLine();
    if(depositChoice.equalsIgnoreCase("yes")) {
        System.out.print("Enter the amount to
                          deposit:");
        double depositAmount = Double.parseDouble
                               (reader.readLine());
        savingsAccount.deposit(depositAccount);
    }
    System.out.print("Enter the time period (in
                      years) the money is kept:");
    int timePeriod = Integer.parseInt(reader.
                     readLine());
    savingsAccount.computeAndDepositInterest
                   (timePeriod);
    System.out.print("Do you want to withdraw?
                      Type 'yes':");
    String withdrawChoice = reader.readLine();
```

```
if (withdrawChoice.equalsIgnoreCase("yes")) {
    System.out.print("Enter the amount to
                      withdraw :");
    double withdrawAmount = Double.parseDouble
                            (reader.readLine());
    savingsAccount.withdraw(withdrawAmount);
}
savingsAccount.displayBalance();
}
else {
    System.out.println("Invalid account type.");
}
reader.close();
}
```

2

Output

Enter S for saving and C for current account : S

Enter your name : Souvik

" " account number : 123

Do you want to deposit? Type yes : yes

Enter the amount to deposit : 600

Deposit successful. Updated balance: $600

Enter the time period (in years) the money is kept: 3

Interest computed and deposited.
Updated balance: $690.0

Do you want to withdraw? Type 'yes': yes

Enter the amount to withdraw: 600
Withdraw successful.. Updated balance: $90.0
Current balance: $90.0

Enter s for saving and c for current account: c
Enter your name: Sarath
account number: 123
Do you want to deposit: yes
Enter the amount: $400
Deposit successful. updated balance: $400
Minimum balance not maintained.
Penality imposed.
Updated balance: $350.0

6-2-24

String

Output :-

1) Hello World
   Java
   Hello
   String Buffer Demo.
2) Length of the String : 11
   String literals are equal : true
   Concatenated String : Hello World
3) Converted to String : 42
4) Extracted : Bmsce
5) Bytes : 72 101 108 111 32 87 111 114 108
   Chars : Hello World
6) Bmsce equals Bmsce → true
   Bmsce equals College → false
   Bmsce equals BMSCE → false
   Bmsce equalsIgnoreCase BMSCE → true
7) Substring is matched
8) true
   false
9) true
   false.

10) using equals(): true
using ==: false

11) Sorted words: [apple, ball, cat, dog, ent, free, gun, hen, ice, jug, kite, life, man, net, orange, parrot, queen, ring, stoc, tree, under, van]

12) Sorted Numbers: [1, 2, 3, 4, 5, 6, 7]

13) Original String: Thwas was a test.
Thwas was, too.

String after replacement:
This is a test. This is too.

14) Concatenated String: hellowworld

15) Original: Welcome to Bmsce College
Replacement: Welcome to Bmsce Commege.

6-2-24

Packages (CIE, SEE)

// Student.java

```
package CIE;
import java.util.*;
public class Student
{
    protected String usn = new String();
    protected String name = new String();
    protected int sem;

    public void inputStudentDetails()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Student
                            USN");
        usn = sc.next();
        System.out.println("Enter Student
                            name");
        name = sc.next;
        System.out.println("Enter sem");
        sem = sc.nextInt();
    }
    public void displayStudentDetails()
    {
        System.out.println("Student usn:"
                           + usn);
```

```
System.out.println("Student name:" +
                    name);
System.out.println("Student Dept:" +
                    dept);
}
}
```

```
// Internals.java
package CIE;
import java.util.*;
public class Internals extends Student {
    protected int marks[] = new int[5];
    public void inputCIEmarks() {
        Scanner sc = new Scanner(
                        System.in);
        for (int i = 0; i < 5; i++)
        {
            System.out.println("Enter " +
                "subject marks");
            marks[i] = sc.nextInt();
        }
    }
}
```

```
// Externals.java
package SEE;
import CIE.Internals;
import java.util.Scanner;
public class Externals extends Internals {
    protected int marks[];
    protected int finalmarks[];
    public Externals() {
        marks = new int[5];
        finalmarks = new int[5];
    }
    public void inputSEEmarks()
    {
        Scanner sc = new Scanner(
                        System.in);
        for(int i=0; i<5; i++)
        {
            System.out.println("Subject" +
            (i+1) + " marks");
            marks[i] = sc.nextInt();
        }
    }
    public void calculateFinalMarks() {
        for(int i=0; i<5; i++) {
            finalMarks[i] = marks[i]/2 +
                        Super.marks[i];
        }
    }
```

```
    public void displayFinalMarks() {
        displayStudentDetails();
        for (int i=0; i<5; i++) {
            System.out.println("Subject" +
            (i+1) + ":" + finalMarks[i]);
        }
    }
}
```

// Main.java

```
import SEE.Externals
class Main {
    public static void main(String args[]) {
        int numofStudents = 2
        External finalMarks[] = new
                Externals[numofStudents];
        for (int i=0; i < numofStudents; i++)
        {
            finalMarks[i] = new Externals();
            finalMarks[i].inputStudentDetails()
            System.out.println("Enter CIE
                                marks");
            finalMarks[i].input CIE marks;
            System.out.println("Displaying
                                data:");
            for (int i=0; i < numofStudents; i++)
            {
```

finalMarks[i].CalculateFinalMarks();
finalMarks[i].displayFinalMarks();

}

}

}

Output

```
Enter student usn
  250
Enter student name
 Sahu
Enter student sem
 3
Enter CIE Marks
Enter 5 : marks : 39
   "   "      : 38
   "          : 26
   "          : 40
   "          : 36

Enter SEE marks
Subject 1  marks : 78
   "    2    "  : 89
   "    3    "  : 90
   "    4    "  : 87
   "    5    "  : 92

Enter student usn
 247
Enter student name
 Sanvesh
```

Enter student rem
3
Enter CIE marks
Enter 5 marks : 39
" " : 38
" " : 30
" " : 30
" " : 24

Enter SEE marks
Subject 1 marks : 76
" 2 " : 78
" 3 " : 89
" 4 " : 90
" 5 " : 92

Displaying data
Student name : Dch
Student usn : 250
" sem : 3

Subject 1 : 78
" 2 : 82
" 3 : 77
" 4 : 83
" 5 : 89

Student name : Sawan
" usn : 247
" sem : 3

Subject 1 : 97
" 2 : 77
" 3 : 80
" 4 : 70
" 5 : 80

6.2.24

Exception handling (Father Son).

```
import java.util.Scanner;
class WrongAge extends Exception {
    public WrongAge() {
        super("Age Error");
    }
    public WrongAge(String message) {
        super(message);
    }
}
class InputScanner {
    protected Scanner s;
    public InputScanner() {
        s = new Scanner(System.in);
    }
}
class Father extends InputScanner {
    protected int fatherAge;
    public Father() throws WrongAge {
        System.out.println("Enter father's
                            age:");
        fatherAge = s.nextInt();
        if (fatherAge < 0) {
            throw new WrongAge("message:
            "Age cannot be negative");
        }
```

```
    public void display() {
        System.out.println("Father's age: " +
                           fatherAge);
    }
}

class Son extends Father {
    private int sonAge;
    public Son() throws WrongAge {
        super();
        System.out.println("Enter son's age:");
        sonAge = s.nextInt();
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater");
        }
        else if (sonAge < 0) {
            throw new WrongAge("age can't be negative");
        }
    }
    public void display() {
        super.display();
        System.out.println("Son's age: " +
                           sonAge);
    }
}
```

Multithreading (BMSCE)

```
class BMSCE extends Thread
{
    public void run()
    {
        for(int i=1, i<=5, i++)
            System.out.println("BMS College of Engineering" + i);
        try {
            Thread.sleep(10000);
        }
        Catch(InterruptedException e) {
            System.out.println("Interrupted");
        }
    }
}
class CS extends Thread
{
    public void run()
    {
        for(int i=1; i<=5; i++)
            System.out.println("CSE" + i);
        try {
            Thread.sleep(2000);
        }
        Catch(InterruptedException e) {
            System.out.println("Interrupted");
        }
    }
}
```

```
public class BMSCE_CS_ThreadProgram {
    public static void main(String args[])
    {
        BMSCE C1 = new BMSCE();
        C1.start();

        CS i1 = new CS();
        i1.start();

    }
}
```

Output:

BMS College of Engineering 1
2
CSE 1
2
3
4
5
BMS College of Engineering 3
4
5

IPC

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while(!valueSet)
            try {
                System.out.println("\n
                    Consumer waiting");
                wait();
            }
            catch(InterruptedException e)
            {
                System.out.println("Interruption
                    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");
```

```
public class Father_son {
    public static void main(String[] args)
    {
        try {
            Son son = new Son();
            son.display();
        }
        catch (WrongAge e) {
            System.out.println("Exception:"
                    + e.getMessage());
        }
    }
}
```

Output

Enter father's age:
48
Enter Sons age:
21
Father's age: 48
Son's age: 21

Enter father's age: 12
Enter Son's age: 56
Error: Son's age cannot be greater.

Enter father's age: -12
Error: Age can't be negative.

```
        wait();
    }
    catch(InterruptedException e){
        System.out.println("Interrupted
                Exception caught");
    }
    this.n = n;
    valueSet = true;
    System.out.println("\n Intimate
                Consumer\n");
    notify();
}
}

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

```
class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").
                start();
    }
    public void run() {
        int i=0
        while(i<15) {
            int r = q.get();
            System.out.println("Consumed:"
                    + r);
            i++;
        }
    }
}
class PCFixed {
    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C
                to stop.");
    }
}
```

Output

Put : 0

Intimate Consumer

Producer waiting

Press Control-C to stop.

Got: 0

Intimate Producer

Consumed : 0

Put : 1

Intimate Consumer

Producer waiting

Got: 1

Intimate Producer

Consumed : 1

Put : 2

Intimate Consumer

Producer waiting

Got: 2

Intimate Producer

Consumed : 2

Put : 3

Intimate Consumer

Producer waiting

Got: 3

Intimate Producer

Consumed: 3

Put: 4

Intimate Consumer

Producer waiting

Got: 4

Intimate Producer

Consumed: 4

Put: 5

Intimate Consumer

Producer waiting

Got: 5

Intimate Producer

Consumed: 5

Put: 6

Intimate Consumer

Producer waiting

Got: 6

.....

Got: 14

Intimate Producer

Consumed: 14

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();
    }
    void last() {
        System.out.println("Inside A.last");
    }
}
```

```
class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().
                      getName();
        System.out.println(name + " entered
                           B.bar");
        try {
            Thread.sleep(1000);
        }
        catch (Exception e) {
            System.out.println(" B interrupted");
        }
        System.out.println(name + " trying to
                           call A.last()");
        a.last();
    }
    void last() {
        System.out.println("Inside A.last");
    }
}
class Deadlock implements Runnable {
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName
                           ("Main Thread");
        Thread t = new Thread(this,
                           "Racing Thread");
```

```
        t.start();
        a.foo(b);
        System.out.println("Back in
                main thread");
    }
    public void run() {
        b.bar(a);
        System.out.println("Back in
                other thread");
    }
    public static void main(String args[]) {
        new Deadlock();
    }
}
```

Output :-

```
Main Thread entered A.foo
Racing Thread entered B.foo
Racing Thread trying to call A.last()
Inside A.last
Back in other thread
Main Thread trying to call B.last()
Inside B.last
Back in main thread
```

13/2/2024

DivisionMain1.java

```
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(number2);
        add(num1);
        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;
result/num = 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, 100, 200);
    flag = 0;
}
```

```
public static void main(String[] args)
{
    DivisionMain1 dm = new DivisionMain1();
    dm.setSize(new Dimension(800, 400));
    dm.setTitle("Division of Integers");
    dm.setVisible(true);
}
```

Number 1: [55] Number 2: [45]

[Result] Result: 55 45 1.0

20/2/2024

PROGRAM-01

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

```
import java.util.Scanner;

class Quadratic

{

int a,b,c;

double r1,r2,d;

void getd()

{

Scanner s= new Scanner(System.in);

System.out.println("Enter the coefficients of a,b,c");

a=s.nextInt();

b=s.nextInt();

c=s.nextInt();

}

void compute()

{

while (a==0)

{

System.out.println("Not a quadratic equation");

System.out.println("Enter a non zero value for a:");

Scanner s=new Scanner(System.in);

a=s.nextInt();

}
```

```
d=b*b-4*a*c;
if (d==0)
{
r1=(-b)/(2*a);
System.out.println("Roots are real and equal");
System.out.println("Root1=Root2= "+r1);
}
else if(d>0)
{
r1=((-b)+(Math.sqrt(d)))/(double)(2*a);
r1=((-b)-(Math.sqrt(d)))/(double)(2*a);
System.out.println("Roots are real and distinct");
System.out.println("Root1= "+r1+ "Root2= "+r2);
}
else if(d<0)
{
System.out.println("Roots are imaginary");
r1=(-b)/(2*a);
r2=Math.sqrt(-d)/(2*a);
System.out.println("Root1= "+r1+ "+i"+r2);
System.out.println("Root1= "+r1+ "-i"+r2);
}
}
}
class QuadraticMain
{
```

```
public static void main(String args[])
{
Quadratic q=new Quadratic();
q.getd();
q.compute();
}
}
```

PROGRAM-02

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 Subject
{
int subjectMarks;
int credits;
int grade;
}
class Student
{
Subject subject[];
String name;
String usn;
double sgpa;
```

```
Scanner s;
Student()
{
int i;
subject =new Subject[9];
for(i=0;i<9;i++)
subject[i]=new Subject();
s=new Scanner(System.in);
}
void getStudentDetails()
{
System.out.println("Enter student name");
name=s.next();
System.out.println("Enter student USN");
usn=s.next();
}
void getMarks()
{
for(int i=0;i<9;i++)
{
System.out.println("Enter marks");
subject[i].subjectMarks=s.nextInt();
System.out.println("Enter number of +credits");
subject[i].credits=s.nextInt();
subject[i].grade=(subject[i].subjectMarks/10)+1;
if(subject[i].grade==11)
```

```
subject[i].grade=10;
if(subject[i].grade<=4)
subject[i].grade=0;
}
}
void computeSGPA()
{
int effectiveScores=0;
int totalCredits=0;
for(int i=0;i<9;i++)
{
effectiveScores+=subject[i].grade*subject[i].credits;
totalCredits+=subject[i].credits;
}
sgpa=(double)effectiveScores/(double)totalCredits;
}
}
class Main
{
public static void main(String args[])
{
Student s1=new Student();
s1.getStudentDetails();
s1.getMarks();
s1.computeSGPA();
System.out.println("Student name:"+s1.name);
```

```
System.out.println("Student usn:"+s1.usn);
System.out.println("Student SGPA:"+s1.sgpa);
}
}
```

PROGRAM-03

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

```
return "Book name: " + name + "\n" +
"Author name: " + author + "\n" +
"Price: " + price + "\n" +
"Number of pages: " + numPages + "\n";
}
}
class BooksMain {
public static void main(String args[]) {
Scanner s = new Scanner(System.in);
int n;
System.out.println("Enter the number of books: ");
n = s.nextInt();
Book[] books = new Book[n];
for (int i = 0; i < n; i++) {
System.out.println("Enter the name of the book");
String name = s.next();
System.out.println("Enter the author of the book");
String author = s.next();
System.out.println("Enter the price of the book");
int price = s.nextInt();
System.out.println("Enter the pages of the book");
int numPages = s.nextInt();
books[i] = new Book(name, author, price, numPages);
}
for (int i = 0; i < n; i++) {
System.out.println(books[i].toString());
```

```
}
}
}
```

PROGRAM-04

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.*;
abstract class Shape{
double dim1;
double dim2;
double rad;
Shape(double a,double b)
{
dim1=a;
dim2=b;
}
Shape(double a)
{
rad=a;
}
```

```
abstract void area();
}
class Rectangle extends Shape
{
Rectangle(double a,double b)
{
super(a,b);
}
void area()
{
System.out.println("Area of rectangle is:"+(dim1*dim2));
}
}
class Triangle extends Shape
{
Triangle(double a,double b)
{
super(a,b);
}
void area()
{
System.out.println("Area of triangle is"+(dim1*dim2)/2);
}
}
class Circle extends Shape
{
```

```
Circle(double a)
{
super(a);
}
void area()
{
System.out.println("Area of circle is:"+(3.14*rad*rad));
}
}
class AbstractMain{
public static void main(String args[])
{
Scanner s=new Scanner(System.in);
System.out.println("Enter the dimensions of rectangle");
double l=s.nextInt();
double b=s.nextInt();
System.out.println("Enter the base and height of traingle");
double h1=s.nextInt();
double h2=s.nextDouble();
System.out.println("Enter the radius of circle");
double r=s.nextInt();
Shape sh;
Rectangle rect=new Rectangle(l,b);
Triangle tri=new Triangle(h1,h2);
Circle cir=new Circle(r);
sh=rect;
```

```
sh.area();
sh=tri;
sh.area();
sh=cir;
sh.area();
}
}
```

PROGRAM-05

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.*;
class Account{
String name;
int accno;
String acc_type;
double balance;
Account(String name,int accno,String acc_type,double balance)
{
this.name=name;
this.accno=accno;
this.acc_type=acc_type;
this.balance=balance;
}
void deposit(double amount)
{
balance+=amount;
}
void withdraw(double amount)
{
if((balance-amount)>=0)
{
balance-=amount;
}
else
{
System.out.println("Insufficient balance! Cannot withdraw");
```

```
}
}
void display()
{
System.out.println("Name:"+" "+name+"Account number:"+"
"+accno+"Account type:"+"
"+acc_type+"Balance:"+" "+balance);
}
}
class Sav_acct extends Account
{
static double rate=5.0;
Sav_acct(String name,int accno,double balance)
{
super(name,accno,"Savings",balance);
}
void interest()
{
balance+=balance*(rate)/100;
System.out.println("Balance:"+balance);
}
}
class Curr_account extends Account
{
private double minBal=500;
private double serviceCharges=50;
Curr_account(String name,int accno,double balance)
```

```
{
super(name,accno,"Current",balance);
}
void checkMin()
{
if (balance<minBal)
{
System.out.println("Balance is less than minimum, penlaty is
imposed"+serviceCharges);
balance-=serviceCharges;
System.out.println("Balance is:"+balance);
}
}
}
class Bank
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
System.out.println("Enter customer name");
String name=s.next();
System.out.println("Enter account number");
int accno=s.nextInt();
System.out.println("Enter the type of account-Current or Savings");
String acc_type=s.next();
System.out.println("Enter the initial balance");
double balance=s.nextDouble();
```

```
Account ob1=new Account(name,accno,acc_type,balance);
Sav_acct sa=new Sav_acct(name,accno,balance);
Curr_account cu=new Curr_account(name,accno,balance);
while (true)
{
if(acc_type.equals("Savings"))
{
System.out.println("Menu");
System.out.println("1.Deposit");
System.out.println("2.Withdraw");
System.out.println("3.Compute Interest for savings account");
System.out.println("4.Display account details");
System.out.println("5.Exit");
System.out.println("Enter your choice:");
int choice=s.nextInt();
switch(choice)
{
case 1:
{
System.out.println("Enter the amount to be deposited");
double amt1=s.nextDouble();
sa.deposit(amt1);
break;
}
case 2:
{
```

```
System.out.println("Enter the amount to be withdrawn");
double amt2=s.nextDouble();
sa.withdraw(amt2);
break;
}
case 3:
{
sa.interest();
break;
}
case 4:
{
sa.display();
break;
}
case 5:
{
break;
}
default:
{
System.out.println("Enter valid input");
break;
}
}
}
```

```
else
{
System.out.println("Menu");
System.out.println("1.Deposit");
System.out.println("2.Withdraw");
System.out.println("3.Display account details");
System.out.println("4.Exit");
System.out.println("Enter your choice:");
int choice=s.nextInt();
switch(choice)
{
case 1:
{
System.out.println("Enter the amount to be deposited");
double amt1=s.nextDouble();
cu.deposit(amt1);
break;
}
case 2:
{
System.out.println("Enter the amount to be withdrawn");
double amt2=s.nextDouble();
cu.withdraw(amt2);
break;
}
case 3:
```

```
{
cu.display();
cu.checkMin();
break;
}
case 4:
{
break;
}
default:
{
System.out.println("Enter valid input");
break;
}
}
}
}
}
}
```

PROGRAM-06

Create a package CIE which has two classes- Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in 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.

```
//Student.java
package CIE;
import java.util.Scanner;
public class Student {
protected String usn = new String();
protected String name = new String();
protected int sem;
public void inputStudentDetails()
{
Scanner sc=new Scanner(System.in);
System.out.println(&quot;Enter student usn&quot;);
usn=sc.next();
System.out.println(&quot;Enter student name&quot;);
name=sc.next();
System.out.println(&quot;Enter student semester&quot;);
sem=sc.nextInt();
}
public void displayStudentDetails() {
System.out.println(&quot;student usn:&quot;+ usn);
```

```
System.out.println(&quot;student name:&quot;+name);
System.out.println(&quot; student sem:&quot;+ sem);
}
}
//Internals.java
package CIE;
import java.util.Scanner;
public class Internals extends Student {
protected int marks[] = new int[5];
public void inputCIEmarks()
{
Scanner sc=new Scanner(System.in);
for (int i=0;i&lt;5;i++)
{
System.out.println(&quot;Enter 5 subject marks&quot;);
marks[i]=sc.nextInt();
}
}
}
//Externals.java
package SEE;
import CIE.Internals;
import java.util.Scanner;
public class Externals extends Internals {
protected int marks[];
protected int finalMarks[];
```

```
public Externals() {
marks = new int[5];
finalMarks = new int[5];
}
public void inputSEEmarks()
{
Scanner sc = new Scanner(System.in);
for(int i=0;i&amp;lt;5;i++)
{
System.out.print(&amp;quot;Subject &amp;quot;+(i+1)+&amp;quot; marks: &amp;quot;);
marks[i] = sc.nextInt();
}
}
public void calculateFinalMarks() {
for(int i=0;i&amp;lt;5;i++)
finalMarks[i] = marks[i]/2 + super.marks[i];
}
public void displayFinalMarks() {
displayStudentDetails();
for(int i=0;i&amp;lt;5;i++)
System.out.println(&amp;quot;Subject &amp;quot; + (i+1) + &amp;quot;: &amp;quot; +
finalMarks[i]);
}
}
//Main1.java
import SEE.Externals;
class Main1 {
```

```
public static void main(String args[])
{
int numOfStudents = 2;
Externals finalMarks[] = new
Externals[numOfStudents];
for(int i=0;i&amp;lt;numOfStudents;i++)
{
finalMarks[i] = new Externals();
finalMarks[i].inputStudentDetails();
System.out.println(&amp;quot;Enter CIE marks&amp;quot;);
finalMarks[i].inputCIEmarks();
System.out.println(&amp;quot;Enter SEE marks&amp;quot;);
finalMarks[i].inputSEEmarks();
}
System.out.println(&amp;quot;Displaying data:\n&amp;quot;);
for(int i=0;i&amp;lt;numOfStudents;i++)
{
finalMarks[i].calculateFinalMarks();
finalMarks[i].displayFinalMarks();
} //end of for loop
} // end of public main
} //end of class main
```

PROGRAM-07

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

is >=father’s age.

```
import java.util.*;
class WrongAge extends Exception
{
public WrongAge(String s)
{
super(s);
}
}
class Father
{
Scanner sc=new Scanner(System.in);
public int F_age;
public Father() throws WrongAge
{
System.out.println("Enter Father's age");
F_age=sc.nextInt();
if (F_age<0)
{
throw new WrongAge("Age cannot be negative");
}
```

```
}
public void display()
{
System.out.println("Father's age="+F_age);
}
}
class Son extends Father
{
private int S_age;
public Son() throws WrongAge
{
System.out.println("Enter son's age");
S_age=sc.nextInt();
if (S_age>F_age)
{
throw new WrongAge("Son's age cannot be greater than father's age");
}
else if (S_age<0)
{
throw new WrongAge("Age cannot be negative");
}
else if (S_age==F_age)
{
throw new WrongAge("Age cannot be same");
}
else
```

```
{
System.out.println("Valid age");
}
}
public void display()
{
System.out.println("Father's age="+F_age);
System.out.println("Son's age="+S_age);
}
}
class Lab7
{
public static void main(String args[])
{
try
{
Son son=new Son();
son.display();
}
catch(WrongAge e)
{
System.out.println("Exception caught");
System.out.println(e.getMessage());
}
}
}
```

PROGRAM-08

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 BMS extends Thread{
public void run(){
for(int i=1;i<=5;i++){
System.out.println("bms college of engineering"+i);
try
{
Thread.sleep(10000);
}
catch(InterruptedException e){
e.printStackTrace();
}
}
}
}
class CSE extends Thread{
public void run(){
for(int i=1;i<=10;i++){
System.out.println("Cse"+i);
try
{
Thread.sleep(2000);
}
catch(InterruptedException e){
```

```
e.printStackTrace();
}
}
}
}
class threadMain{
public static void main(String a[]){
BMS obj1=new BMS();
CSE obj2=new CSE();
obj1.start();
obj2.start();
}
}
```

PROGRAM-09

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(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()+outR
esult.getHeight()-8);
else
g.drawString(out,100,200);
flag=0;
}
public static void main(String[] args)
{
DivisionMain1 dm=new DivisionMain1();
dm.setSize(new Dimension(800,400));
dm.setTitle("DivionOfIntegers");
dm.setVisible(true);
}
}
```

PROGRAM-10

10.A)Demonstrate Inter process Communication and deadlock

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

```
System.out.println("InterruptedException caught");
}
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("Intimate Consumer\n");
notify();
}
}
class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(i<5) {
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<5) {
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.");
}
}
```

10B)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();
}
void last() {
System.out.println("Inside A.last");
}
}
class B {
synchronized void bar(A a) {
String name =Thread.currentThread().getName();
System.out.println(name + " entered B.bar");
try {
Thread.sleep(1000);
} catch(Exception e) {
System.out.println("B Interrupted");
}
System.out.println(name + " trying to call A.last()");
a.last();
}
void last() {
System.out.println("Inside A.last");
```

```
}
}
class Deadlock implements Runnable
{
A a = new A();
B b = new B();
Deadlock() {
Thread.currentThread().setName("MainThread");
Thread t = new Thread(this,"RacingThread");
t.start();
a.foo(b); // 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();
}
}
```