B. M. S. COLLEGE OF ENGINEERING
(AUTONOMOUS COLLEGE UNDER VTU, BELGAUM)
BANGALORE – 560019

2022-23

LAB RECORD

OBJECT ORIENTED JAVA PROGRAMMING(23CS3PCOOJ)

Submitted by :

NAME : Sakshi Shetty
USN : 1BM22CS234
SECTION : 3E

Submitted to
Dr. Seema Patil
Assistant Professor
Dept. of CSE, BMSCE

SAKSHI SHETTY 1BM22CS234

LAB PROGRAM 1

Develop a dab program that prints all real solutions to the quadratic $ax^2+bx+c=0$. 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
        a= s.nextInt();
        b= s.nextInt();
        c = s.nextInt();
    }
void compute()
{
    while(a==0)
    {
        System.out.println("Not a quadratic equati
        System.out.println("Enter a non-zero value fo
        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);
        r2 = ((-b)-(Math.sqrt(d)))/(double)(2*a);
        System.out.println("Roots are real and distinct");
        System.out.println("Root1 =" +r1+ "Root 2 =" +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();
    }
}
```

~~Conversions (Widening, Narrowing, Promotion)~~

Output:

Enter the coefficients of a, b, c

1

2

3

Roots are imaginary

Root1 = -1.0 + i1.4142135623730951

Root1 = -1.0 * 1

Output:

Enter the coefficients of a, b, c

1

2

3

Roots are imaginary

Root1 = -1.0 + i1.4142135623730951

Root1 = -1.0 - i1.4142135623730951

Enter the coefficients of a, b, c

1

2

1

Roots are real and equal

Root1 = Root2 -1.0

Enter the coefficients a, b, c
1
4
1
Roots are real and distinct
Root1 = 3.7320508075688772 Root2 = 0.2679491924311228

SAKSHI SHETTY
1BM22CS234.

CGPA → excluding F grade
SGPA → including F grade.

Pgm 2

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

$$SGPA = \frac{\sum[(\text{Course credits})(\text{Grade points})]}{\sum[(\text{Course credits})]}$$

```
import java.util.Scanner;
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 your name:");
        name = s.next();
        System.out.println("Enter your usn:");
        usn = s.next();
    }
```

```
void getMarks()
{
    for (int i=0; i<9; i++)
    {
        System.out.print("Enter marks for subject "+(i+1)+" : ");
        subject[i].subjectMarks = S.nextInt();
        System.out.print("Enter credits for subject "+(i+1)+" : ");
        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 effectiveScore = 0;
    int totalCredits = 0;
    for (int i=0; i<9; i++)
    {
        effectiveScore += (subject[i].grade
            * subject[i].credits);
        totalcredits += subject[i].credits;
    }
    SGPA = (double)effectiveScore/(double)totalCredits;
}
}
```

```
class Main
{
    public static void main(String args[])
    {
        Student S1 = new Student();
        S1.getStudentDetails();
        S1.getMarks();
        S1.computeSGPA();
        System.out.println("Name:" + S1.name);
        System.out.println("USN:" + S1.usn);
        System.out.println("SGPA:" + S1.SGPA);
    }
}
```

Output:

```
Enter your name: sakshi
Enter your usn : 1bm22cs234
Enter marks for subject1: 89
Enter credits for subject 1: 4
Enter marks for subject 2 : 67
Enter credits for subject 2 : 2
Enter marks for subject 3 : 82
Enter credits for subject 3 : 3
Enter marks for subject 4: 78
Enter credits for subject 4: 2
Enter marks for subject 5: 90
Enter credits for subject 5: 4
Enter marks for subject 6: 78
Enter credits for subject 6: 4
Enter marks for subject 7: 78
Enter credits for subject 7: 1
```

Bafna Gold
Date: Page:

Enter marks for subject 8: 95
Enter credits for subject 8: 2
Enter marks for subject 9: 78
Enter credits for subject 9: 4
Name: sakshi
USN: 1bm22cs234
SGPA: 8.653846153846153

Sakshi Shetty
1BM22CS234

19-12-2023

Lab pgm 3.
26/12/23

Create a class book which contains four members name, author, price, numpages. Include a constructor to set the values for the members. Include methods to set and get the details for 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 Books {
    String name, author; int price, numPages;
    Books(String name, String author, int price,
          int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }
    public String toString() {
        String name, author, price, numPages;
        name = "Book name s: " + this.name + "\n";
        author = "Author name: " + this.author + "\n";
        price = "Price: " + this.price + "\n";
        numPages = "Number of pages: " + this.numPages + "\n";
        return name + author + price + numPages;
    }
}
class BookMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n;
        String name, author;
```

```
int price, numPages;
System.out.println("Enter the number of books:");
n = sc.nextInt();
Books b[] = new Books[n];
System.out.println("Enter Name, author, price
    and number of pages:");
for(int i=0; i<n; i++){
    name = sc.next();
    author = sc.next();
    price = sc.nextInt();
    numPages = sc.nextInt();
    b[i] = new Books(name, author, price, numPages);
}
System.out.println("Book details:");
for(int i=0; i<n; i++){
    System.out.println(b[i].toString());
}
}
}
```

```
Enter the number of books:
1
Enter Name, author, price and number of pages:
Ikigai
Hector
400
350
Book details:
Book name: Ikigai
Author name: Hector
Price: 400
Number of Pages: 350
```

Enter the number of Books:
2
Enter Name, author, price and number of pages:
Ikigai
hector
400
350
little women
May
350
400
Book details:
Book name : Ikigai
Author name : hector
Price : 400
Number of pages : 350

Book name : little women
Author name : May
Price : 350
Number of pages : 400

26/12/2023

SAKSHI SHETTY
1BM22CS234

Bafna Gold
Date: Page:

Lab Prgm 4.
2/01/24

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 contain only the method printArea() that prints the area of the given shape.

```
import java.util.*;

class InputScanner {
    Scanner sc;
    InputScanner(){
        sc = new Scanner(System.in);
    }
abstract class Shape extends InputScanner {
    double a
    double b
    abstract void getInput();
    abstract void displayArea();
}

class Rectangle extends Shape {
    void getInput() {
        System.out.println("Enter the length and breadth:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }
    void displayArea() {
        System.out.println("Area of rectangle is:"
                           + (a*b));
    }
}
```

```
class Triangle extends Shape {
    void getInput() {
        System.out.println("Enter the length and height:");
        a = sc.nextDouble();
        b = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of rectangle is:"
                + (a*b*0.5));
    }
}

class Circle extends shape {
    void getInput() {
        System.out.println("Enter the radius:");
        a = sc.nextDouble();
    }

    void displayArea() {
        System.out.println("Area of rectangle is:"
                + (a*a*3.14));
    }
}

class ShapeMain {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.getInput();
        r.displayArea();
        t.getInput();
        t.displayArea();
        c.getInput();
        c.displayArea();
    }
}
```

Bafna Gold
Date: Page:

Output:

Enter the length and breadth:
23
12
Area of rectangle is: 276.0
Enter the length and height:
23
12
Area of rectangle is: 23.0
Enter the radius:
4
Area of rectangle is: 50.24

Sakshi Shetty
1BM22CS234

Lab programs 5
9/01/24

- 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 name;
    int accno;
    String type;
    double balance;
```

```
account ( String name, int accno, String type,
          double balance)
{
    this.name = name;
    this.accno = accno;
    this.type = 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,
                             can't withdraw");
    }
}
void display ()
{
    System.out.println ("name:" + name + "accno:"
        + accno + "type:" + type + "balance:"
        + balance);
}
```

```
class savAcct extends account
{
    private static double rate = 5;
    savAcct (String name, int accno, double balance)
    {
        super (name, accno, "savings", balance);
    }
    void interest()
    {
        balance += balance * (rate)/100;
        System.out.println("balance:" + balance);
    }
}
class curAcct extends account
{
    private double minBal = 500;
    private double serviceCharges = 50;
    curAcct( String name, int accno, double balance)
    {
        super (name, accno, "current", balance);
    }
    void checkmin()
    {
        if (balance < minBal)
        {
            System.out.println("balance is less than min balance, service charges imposed:"
            + serviceCharges);
            balance -= serviceCharges;
            System.out.println("balance is:" + balance);
        }
    }
}
```

```
class accountMain
{
    public static void main (String a[])
    {
        Scanner s = new
        Scanner (System.in);
        System.out.println (" enter the name:");
        String name = s.next();
        System.out.println (" Enter the type
            ( current / savings):");
        String type = s.next();
        System.out.println (" Enter the account number:");
        int aceno = s.nextInt();
        System.out.println (" Enter the initial balance:");
        double balance = s.nextDouble();
        int ch;
        double amount1, amount2;
        account acc = new
        account (name, aceno, type, balance);
        savAcct sa = new
        savAcct (name, aceno, balance);
        curAcct ca = new
        curAcct (name, aceno, balance);
        while (true)
        {
            if (acc.type.equals ("savings"))
            {
                System.out.println (" Enter your choice
                1. deposit 2. withdraw 3. compute interest
                4. display ");
                System.out.println (" Enter the choice:");
                ch = s.nextInt();
```

```
switch (ch)
{
    case 1: System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            sa.deposit(amount1);
            break;
    case 2: System.out.println("Enter the amount:");
            amount2 = s.nextInt();
            sa.withdraw(amount2);
            break;
    case 3: sa.interest();
            break;
    case 4: sa.display();
            break;
    case 5: System.exit(0);
    default: System.out.println("invalid input:");
            break;
    }
}
else
{
    System.out.println("\n Menu \n 1. deposit
        2. withdraw 3. display");
    System.out.println("Enter the choice:");
    ch = s.nextInt();
    switch(ch)
    {
    case 1: System.out.println("Enter the amount:");
            amount1 = s.nextInt();
            sa.deposit(amount1);
            break;
```

```
            case 2: System.out.println("Enter the amount:");
                    amount2 = s.nextInt();
                    ca.withdraw(amount2);
                    ca.checkmin();
                    break;
            case 3: ca.display();
                    break;
            case 4: System.exit(0);
            default: System.out.println("Invalid input");
                     break;
        }
    }
}
```

Output:

Enter the name, type current/saving(); account number, initial balance;

Sakshi

Savings

123

50000

Menu

1. Deposit 2. Withdraw 3. Compute Interest

4. Display

Enter the choice:

1.

Enter the amount:

5000

Menu

1. Deposit 2. Withdraw 3. Compute Interest 4. Displ
2
Enter the amount:
500

Menu:
1. Deposit 2. Withdraw 3. Compute Interest 4. Display
3
Balance: 57225.0

Menu:
1. Deposit 2. Withdraw 3. Compute Interest 4. Display
Enter the choice:
4.
Name: Sakshi
Accno: 123
Type: Savings
Balance: 57225.0

Enter the name, type (Current/saving), account numbe
Initial balance,
Sakshi
current
101
500000

Menu
1. Deposit 2. Withdraw 3. display
Enter the choice
1.
Enter the amount:
5000

Bafna Gold
Date: Page:

Menu
1. Deposit 2. Withdraw 3. display
Enter the choice:
2
Enter the amount:
500

Menu:
1. Deposit 2. Withdraw 3. display
Enter the choice:
3
Name: Sakshi
Accno: 101
Type: current
Balance: 50500.0.

SAKSHI SHETTY
1BM22CS234.

Pgm 6:

1/24

package

Create 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 stored 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 SEE marks scored in five courses of the current semester of the student. Import the two packages in a five 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("Enter USN :");
        USN = sc.next();
        System.out.print("Enter name:");
        name = sc.next();
        System.out.print("Enter semester:");
        sem = sc.nextInt();
    }
```

```
public void displayStudentDetails(){
    System.out.println("USN:" + USN);
    System.out.println("Name:" + Name);
    System.out.println("Semester:" + Sem);
  }
}
```

```
// Internals.java
package CIE;

import java.util.Scanner;

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

    public Internals(){}

    public void inputCIEmarks(){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter internal marks for" + name);
        for(int i=0; i<5; i++){
            System.out.print("Subject" + (i+1) + " 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);
        System.out.println("Enter SEE marks for" + name);
        for(int i=0; i<5; i++){
            System.out.print("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;

public class Main {
    public static void main (String args[]) {
        int numOfStudents = 2;
        Externals finalMarks[] = new Externals
                                 (numOfStudents);
        for (int i=0; i< numOfStudents; i++) {
            finalMarks[i] = new External();
            finalMarks[i].inputStudentDetails();
            System.out.println(" Enter CIE marks :");
            finalMarks[i].inputCIEmarks();
            System.out.println(" Enter SEE marks :");
            finalMarks[i].inputSEEmarks();
        }

        System.out.println (" Displaying data :\n");

        for (int i=0 ; i< numOfStudents ; i++) {
            finalMarks[i].calculateFinalMarks();
            finalMarks[i].displayFinalMarks();
        }
    }
}
```

Output:

Enter USN : 111
Enter Name : Santosh
Enter Semester : 3
Enter CIE Marks

Enter Internal marks for santosh
Subject 1 marks : 33
Subject 2 marks : 36
Subject 3 marks : 28
Subject 4 marks : 31
Subject 5 marks : 40
Enter see marks for santosh
Subject 1 marks : 89
Subject 2 marks : 91
Subject 3 marks : 78
Subject 4 marks : 84
Subject 5 marks : 90
Enter USN : 112
Enter Name : Siri
Enter Semester : 3
Enter CIE marks :
Enter Internal marks for Siri :
Subject 1 marks : 33
Subject 2 marks : 28
Subject 3 marks : 23
Subject 4 marks : 27
Subject 5 marks : 36
Enter see marks
Enter See marks for Siri.
Subject 1 marks : 54
Subject 2 marks : 85
Subject 3 marks : 89
Subject 4 marks : 92
Subject 5 marks : 97

displaying data:

USN : 111
Name : Santosh
Semester : 3
Subject 1 : 77
Subject 2 : 81
Subject 3 : 67
Subject 4 : 73
Subject 5 : 85

USN : 112
Name : Siri
Semester : 3
Subject 1 : 75
Subject 2 : 76
Subject 3 : 72
Subject 4 : 73
Subject 5 : 74

Sakshi Shetty
1BM22CS234

2-4

STRINGS:

(1) BMSCE
BMSCE

(2) 3
3
Roll no 10 is present

(3) Dimensions are 10.0 by 14.0 by 12.6
box b: Dimensions are 10.0 by 14.0 by 12.0

(4) bmsce

(5) 65
66
67
Welcome to bmsce college.

(6) Bmsce equals Bmsce → true
Bmsce equals college → false
Bmsce equals Ignorecase BMSCE → true.

(7) Substring is matched
S1 = " Bmsce college"
S2 = " Welcome to Bmsce College of Engineering".

(8) true
false

(9) false
true.

(10) Hello equals Hello → true
Hello == Hello → false.

(11) The names in alphabetical order are:
apple
ball
cat
lion
watch.

(12) Sorted Numbers (Ascending order):
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

(13) Thwas was a test. Thwas was too

(14) hello world

(15) commege

(16) Hello friends.

(17) Student 1
name: Sakshi
Reg no: 123
Semester: 3
CGPA: 8.87

Student 2
name: Priyanka
Reg no: 124
Semester: 4
CGPA: 9.05

(18) charAt 3 is 'A'
reverse of SAS is SAS

(19) Eagle is flying
Eagle makes a sound.

Hawk is moving
Hawk is making a sound.

(20) Circle - Area : 78.5398 perimeter - 31.4259
Triangle - Area : 5.0 perimeter - 12.0.

Sakshi Shetty
1BM22CS234

Lab pgm 4.
30/1/24

WAP to demonstrate 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 takes both father and son's age and throws an exception if son's age is >= father's age.

```
import java.util.Scanner;

class WrongAge extends Exception{
    public WrongAge(String s){
        super(s);
    }
}
class InputScanner {
    Scanner sc;
    public InputScanner(){
        sc= new Scanner(System.in);
    }
}

class Father extends InputScanner {
    private int FAge;

    public Father() throws Exception{
        System.out.println("Enter father's Age:");
        this.FAge = sc.nextInt();
        if(this.FAge<0){
            throw new WrongAge("Age cannot be -ve");
        }
```

```
    public void display(){
        System.out.println("Father Age:" + this.FAge);
    }
}
class son extends father
{
    private int SAge;

    public son() throws exception {
        System.out.println("Enter son age:");
        this.SAge = sc.nextInt();
        if (this.SAge<0)
            throw new WrongAge("Age cannot be -ve");

        if (this.SAge >= Super.Age)
            throw new WrongAge("Son age cannot be
                greater than father");

        public void display(){
            Super.display();
            System.out.println("Son age:" + this.SAge);
        }
}
class Main{
    public static void main(String[] args){
        try{
            Son s = new son();
            S.display();
        }
        catch(WrongAge){
            System.out.println("Error:" + e.getMessage());
        }
    }
}
```

Output:

Enter Father Age: -5
Age cannot be -ve

Enter Father Age: 40
Enter son age: -5
Age cannot be -5

Enter Father Age: 40
Enter son Age: 45
Son age cannot be greather that father.

Enter Father Age: 40
Enter Son Age: 12

Father Age: 40
Son Age: 12

Sakshi Shetty
1BM22CS234

Lab Pgm 3
6/2/24

WAP which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

```
class DisplayMessageThread extends Thread {
    private final String message;
    private final long interval;

    DisplayMessageThread (String message, long interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run(){
        try {
            while (true){
                System.out.println(message);
                Thread.sleep(interval);
            }
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().
                getName() + " interrupted.");
        }
    }
}

public class TwoThreadDemo{
    public static void main(String[] args) {
        DisplayMessageThread thread1 = new
DisplayMessageThread("BMS College of Engineering", 10000);
        DisplayMessageThread thread2 = new
DisplayMessageThread("CSE", 2000);
```

```
thread1.setName("Thread 1");
thread2.setName("Thread 2");

thread1.start();
thread2.start();

try{
    Thread.sleep(30000);
} catch(InterruptedException e){
    System.out.println("Main thread interrupted");
}

thread1.interrupt();
thread2.interrupt();

System.out.println("Main thread exiting.");
}
```

o/p

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

BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
Thread 1 interrupted
Thread 2 interrupted
Main thread existing

Sakshi Shetty
1BM22CS234

13/2/24
Lab Pgm 10.

Bafna Gold
Date: Page:

Implementation of producer and consumer.

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

```
class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run(){
int i=0;
while (i<15){
q.put(i++);
}
}
}
class Consumer implements Runnable{
Q q;
Consumer (Q q){
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
int r=0;
while (r<15) {
int r = q.get();
System.out.println("consumed : "+r);
r++;
}
}
}
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 : 1
Got : 1
Put : 2
Got : 2
Put : 3
Got : 3
Put : 4
Got : 4
Put : 5
Got : 5
```

Sakshi Shetty
1BM22CS234

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();
}
}
```

o/p:

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

Sakshi Shetty
1BM22CS234

Lab pgm 9.

20/2/24

WAP 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 integers, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException Display the exception in a message dialog box.

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

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

    public DivisionMain1()
    {
        setLayout(new FlowLayout());
        dResult = new Button("RESULT");
        Label number1 = new Label("Number 1:",
                                  Label.RIGHT);
        Label number2 = new Label("Number 2:",
                                  Label.RIGHT);
        num1 = new TextField(5);
        num2 = new TextField(5);
```

```
outResult = new Label ("Result:", Label.RIGHT);

add (number1);
add (num1);
add (number2);
add (num2);
add (dResult);
add (outResult);

num1.addActionListener (this);
num2.addActionListener (this);
dResult.addActionListener (this);
addWindowListener (new WindowAdapter()
{
    public void windowClosing (WindowEvent we)
    {
        System.exit (0);
    }
});
}

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

Bafna Gold
Date: Page:

Output:

Number 1: | 10 | Number 2: | 5 | RESULT Result: 2.0.

Functions used:

A Button is basically a control component with a label that generates an event when pushed.

SetLayout() method allows you to set the layout of the container.

addActionListener() is an type of class in Java that receieves a notification whenever any action is performed.

repaint() method is an asynchronous method of applet class.

setSize() sets the size of this Dimension object to the specified width and height.

setTitle() function defines the title to appear at the top of the sketch window.

setVisible() method makes the frame appear on the screen.

20/2/2024

Sakshi Shetty
1BM22CS234

JAVA LAB PROGRAMS
1BM22CS234

1) Quadratic

```
import java.util.*;
class Quadratic {
    int a,b,c;
    double r1,r2,d;

    void getd() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the coefficients of a,b,c : ");
        a = sc.nextInt();
        b = sc.nextInt();
        c = sc.nextInt();
    }

    void compute() {
        while(a == 0) {
            System.out.println("Not a quadratic equation :(");;
            System.out.println("Enter a non zero : ");
            Scanner sc = new Scanner(System.in);
            a = sc.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);
            r2 = ((-b) - Math.sqrt(d)) / (double)(2*a);
            System.out.println("Roots are real and distinct");
            System.out.println("Root1 = " + r1 + "Root2 = " + r2);
        }
        else {
            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("Root2 = "+r2+"-i"+r2);
        }
    }
}
class QuadraticMain {
    public static void main(String[] args) {
        Quadratic q = new Quadratic();
        q.getd();
        q.compute();
    }
}
```

2) Student CGPA calculator

```
import java.util.Scanner;
```

```
class Subject {
    int subjectMarks;
    int credits;
    int grade;
}
class Student {
    Subject subject[];
    String name;
    String USN;
    double SGPA;
    Scanner sc;

    public Student() {
        subject = new Subject[8];
        for(int i = 0; i < 8; i++) {
            subject[i] = new Subject();
        }
    }

    void getStudentDetails() {
        sc = new Scanner(System.in);
        System.out.print("Enter name : ");
        this.name = sc.next();

        System.out.print("Enter your USN : ");
        this.USN = sc.next();
    }
```

```
void getMarks() {
    sc = new Scanner(System.in);
    for(int i = 0; i < 8; i++) {
        System.out.print("Enter " + (i+1) + " subject marks : ");
        subject[i].subjectMarks = sc.nextInt();

        System.out.print("Enter number of credits : ");
        subject[i].credits = sc.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 totalCredits = 0;
    for(int i = 0; i < 8; i++) {
        totalCredits += subject[i].credits;
    }

    int totalGradeAndCredit = 0;
    for(int i = 0; i < 8; i++) {
        totalGradeAndCredit += (subject[i].credits * subject[i].grade);
    }
```

```
        this.SGPA = ((float)totalGradeAndCredit/totalCredits);

        System.out.println("SGPA of the student is : " + this.SGPA);
    }
}


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

3) Book

```
import java.util.*;
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 void set(String name, String author, int price, int numPages) {
    this.name = name;
    this.author = author;
    this.price = price;
    this.numPages = numPages;
}

public String toString() {
    String name = "Book name : " + this.name + "\n";
    String author = "Author name : " + this.author + "\n";
    String price = "Price : " + this.price + "\n";
    String numPages = "Number of pages : " + this.numPages + "\n";
    return name+author+price+numPages;
}

public String getName() {
    return this.name;
}
public String getAuthor() {
    return this.author;
}
public int getPrice(){
    return this.price;
}
public int getNumberOfPages() {
```

```
        return this.numPages;
    }
}

public class Main3 {
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of books : ");
        int n = sc.nextInt();

        Book[] b = new Book[n];
        for(int i = 0; i < n; i++) {
            System.out.print("Enter " + (i+1) + " book name, author name, price and
number of pages : ");
            String name = sc.next();
            String author = sc.next();
            int price = sc.nextInt();
            int numPages = sc.nextInt();

            b[i] = new Book(name, author, price, numPages);
        }

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

}
```

4) Abstract Demo of class Shape

```
import java.util.*;


class InputScanner {
    Scanner sc;
    public InputScanner () {
        sc = new Scanner(System.in);
    }
}


abstract class Shape extends InputScanner {
    double a, b;
    abstract void getInput();
    abstract void displayArea();
}


class Rectangle extends Shape {


    @Override
    void getInput() {
        System.out.print("Enter length and breath of a Rectangle : ");
        this.a = sc.nextDouble();
        this.b = sc.nextDouble();
    }


    @Override
    void displayArea() {
        System.out.println("The area of the Rectangle is : " + (this.a * this.b));
```

```
    }
}
class Triangle extends Shape {

    @Override
    void getInput() {
        System.out.print("Enter Height and Base of a Triangle : ");
        this.a = sc.nextDouble();
        this.b = sc.nextDouble();
    }

    @Override
    void displayArea() {
        System.out.println("The area of the Triangle is : " + ((this.a * this.b)/2));
    }
}
class Circle extends Shape {

    @Override
    void getInput() {
        System.out.print("Enter the radius of the Circle : ");
        this.a = sc.nextDouble();
    }

    @Override
    void displayArea() {
        System.out.println("The area of the Circle is : " + (Math.PI*this.a*this.a));
    }
```

```
}

public class AbstractDemo {
    public static void main(String[] args) {
        Shape s;
        s = new Circle();
        s.getInput();
        s.displayArea();

        s = new Rectangle();
        s.getInput();
        s.displayArea();

        s = new Triangle();
        s.getInput();
        s.displayArea();
    }
}
```

5) Bank

```
import java.util.Scanner;

abstract class Account {
    String name;
    int accno;
    String type;

    abstract void deposit(double amount);
```

```
    abstract void display();
    abstract void withdraw(double amount);
    abstract void menu();

}
class Current extends Account {
    double balance;
    private static int MIN_BALANCE = 500;
    private static int SER_CHARGE = 10;



    public Current(String name, int accno, String type) {
        this.name = name;
        this.accno = accno;
        this.type = type;
    }

    @Override
    void deposit(double amount) {

        this.balance += amount;
        System.out.println("amount deposited ");

              if(this.balance < MIN_BALANCE) {
            System.out.println("Penalty");
            this.balance -= SER_CHARGE;
        }
    }
```

```
@Override
void display() {
    System.out.println("Name : " + this.name);
    System.out.println("Account number : " + this.accno);
    System.out.println("Balance : " + this.balance);
}

@Override
void withdraw(double amount) {
    if(this.balance < amount) {
        System.out.println("insufficient balance");
    }
    else {
        this.balance -= amount;
    }
}

@Override
void menu() {
    System.out.println("---MENU---");
    int choice;
    double amount;
    do {
        System.out.println("1. deposit 2. Withdraw 3. details 4. Exit");
        Scanner sc = new Scanner(System.in);
        choice = sc.nextInt();
            switch(choice) {
```

```
                case 1 : System.out.print("Enter the deposit amount : ");
                    amount = sc.nextDouble();
                    this.deposit(amount);
                    break;
                case 2 : System.out.println("Enter withdrawal amount : ");;
                    amount = sc.nextDouble();
                    this.withdraw(amount);
                    break;
                case 3 : this.display();
                    break;
            case 4 : return;

        }
        }while(choice != 4);
    }

}
class Savings extends Account {
    int balance;
    final float interest = 5f;

        public Savings(String name, int accno, String type) {
        this.name = name;
        this.accno = accno;
                this.type = type;
        }
    @Override
    void deposit(double amount) {
```

```
    float interestAmount = (float) (amount*(this.interest/100));
    System.out.println("Inetrest of : " + interestAmount + " is added");
    this.balance += amount + interestAmount;
    System.out.println("amount deposited ");
}

@Override
void display() {
    System.out.println("Name : " + this.name);
    System.out.println("Account number : " + this.accno);
    System.out.println("Balance : " + this.balance);
}

@Override
void withdraw(double amount) {
    if(this.balance < amount) {
        System.out.println("insufficient balance");
    }
    else {
        this.balance -= amount;
    }
}

@Override
void menu() {
    System.out.println("---MENU---");
    int choice;
```

```
        double amount;
        do {
            System.out.println("1. deposit 2. Withdraw 3. details 4. Exit");
            Scanner sc = new Scanner(System.in);
            choice = sc.nextInt();
                switch(choice) {
                    case 1 : System.out.print("Enter the deposit amount : ");
                        amount = sc.nextDouble();
                        this.deposit(amount);
                        break;
                    case 2 : System.out.println("Enter withdrawal amount : ");;
                        amount = sc.nextDouble();
                        this.withdraw(amount);
                        break;
                    case 3 : this.display();
                        break;
                 case 4 : return;

        }
        }while(choice != 4);
    }

}

public class Bank {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter name, accno, type : ");
```

```
        String name = sc.next();
        int accno = sc.nextInt();
        String type = sc.next();

        if(type.equals("current")) {
            Account c = new Current(name, accno, type);
            c.menu();
        }
        else {
            Account s = new Savings(name, accno, type);
            s.menu();
        }
    }
}
```

6) 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.

```
package CIE;

import java.util.Scanner;

public class Internals extends Student {
```

```
    protected int marks[] = new int[5];

    public Internals() {
        // Constructor for Internals
    }

    public void inputCIEmarks() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter Internal Marks for " + super.name);
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + " marks: ");
            this.marks[i] = scanner.nextInt();
        }
    }
}
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.print("Enter USN: ");
        this.usn = sc.next();
        System.out.print("Enter Name: ");
```

```
        this.name = sc.next();
        System.out.print("Enter Semester: ");
        this.sem =  sc.nextInt();
    }

    public void displayStudentDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}
package SEE;
import CIE.Internals;


import java.util.Scanner;

public class External extends Internals {
    protected int marks[];
    protected int finalMarks[];

    public External() {
        marks = new int[5];
        finalMarks = new int[5];
    }

    public void inputSEEmarks() {
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.println("Enter SEE Marks for " + name);
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + " marks: ");
            marks[i] = scanner.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]);
    }
}
package SEE;
import CIE.Internals;



import java.util.Scanner;

public class External extends Internals {
    protected int marks[];
    protected int finalMarks[];
```

```
    public External() {
        marks = new int[5];
        finalMarks = new int[5];
    }

    public void inputSEEmarks() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter SEE Marks for " + name);
        for (int i = 0; i < 5; i++) {
            System.out.print("Subject " + (i + 1) + " marks: ");
            marks[i] = scanner.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]);
    }
}
```

7) Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called

“Father” and derived class called “Son” which extends the base class. In Father class, implement a

constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In Son class,

implement a constructor that cases both father and son's age and throws an exception if son's age is

>=father's age.

```
import java.util.Scanner;


class WrongAge extends Exception
{
    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("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 than father's age");
        }
        else if (sonAge < 0)
        {
            throw new WrongAge("Age cannot be negative");
        }
    }

    public void display()
    {
        super.display();
        System.out.println("Son's Age: " + sonAge);
    }
}

public class Main
{
    public static void main(String[] args)
    {
        try
        {
            Son son = new Son();
            son.display();
        }
        catch (WrongAge e)
        {
```

```
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

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 DisplayMessageThread extends Thread {
    private final String message;
    private final long interval; // in milliseconds

    DisplayMessageThread(String message, long interval) {
        this.message = message;
        this.interval = interval;
    }

    public void run() {
        try {
            while (true) {
                System.out.println(message);
                Thread.sleep(interval);
            }
        } catch (InterruptedException e) {
            System.out.println(Thread.currentThread().getName() + " interrupted.");
        }
    }
}
```

```
public class TwoThreadDemo {

    public static void main(String[] args) {

        DisplayMessageThread thread1 = new DisplayMessageThread("BMS College
of Engineering", 10000); // 10 seconds

        DisplayMessageThread thread2 = new DisplayMessageThread("CSE", 2000); //
2 seconds



        thread1.setName("Thread 1");

        thread2.setName("Thread 2");



        thread1.start();

        thread2.start();



        try {

            // Let the threads run for a while

            Thread.sleep(30000); // Let the program run for 30 seconds

        } catch (InterruptedException e) {

            System.out.println("Main thread interrupted.");

        }



        // Interrupt both threads to stop them

        thread1.interrupt();

        thread2.interrupt();



        System.out.println("Main thread exiting.");

    }

}
```

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 javax.swing.*;

import java.awt.*;

import java.awt.event.*;


class UserInterface {

    UserInterface() {

        // create JFrame container

        JFrame jfrm = new JFrame("Divider App");

        jfrm.setSize(275, 150);

        jfrm.setLayout(new FlowLayout());

        // to terminate on close

        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


        // text label

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


        // add text field for both numbers

        JTextField ajtf = new JTextField(8);

        JTextField bjtf = new JTextField(8);


        // calc button

        JButton button = new JButton("Calculate");


        // labels

        JLabel err = new JLabel();

        JLabel alab = new JLabel();

        JLabel blab = new JLabel();
```

```
JLabel anslab = new JLabel();


// add in order :)
jfrm.add(err);  // to display error message
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);


ActionListener calculateListener = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        try {
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());
            if (b == 0) {
                throw new ArithmeticException();
            }
            int ans = a / b;


            alab.setText("\nA = " + a);
            blab.setText("\nB = " + b);
            anslab.setText("\nAns = " + ans);
            err.setText(""); // Clear any previous error message
        } catch (NumberFormatException e) {
            displayErrorMessage("Enter Only Integers!");
```

```
            } catch (ArithmeticException e) {
                displayErrorMessage("B should be non-zero!");
            }
        }

        private void displayErrorMessage(String message) {
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText(message);
        }
    };

    button.addActionListener(calculateListener);

    // display frame
    jfrm.setVisible(true);
  }

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

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++;
```

```
}

}

}
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.");

}

}
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("M
ainThread");

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();

}

}
```