8. 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 quadratic formula. If discriminant b^2-4ac is -ve, display message stating no real roots.

26/9/24 Object Oriented Java Programming

Lab program 1:

```
import java.util.scanner
class quadratic {
    public static void main (String [] args) {
        Scanner sc = new scanner(system.in);
        System.out.print ("enter values of a, b and c");
        double a = sc.nextDouble ();
        double b = sc.nextDouble ();
        double c = sc.nextDouble ();
        double discriminant = b * b - 4 * a * c;
        if (a != 0) {
        if (discriminant > 0) {
        double x1 = (-b + Math.sqrt (discriminant)) / (2 * a);
        double x2 = (-b - Math.sqrt (discriminant)) / (2 * a);
        System.out.println (" The equation has two real solutions : x1 = " + x1 + ", & x2 = " + x2);
        } else if (discriminant == 0) {
        double x = -b / (2 * a);
        system.out.println (" The equation has one real solution : x = " + x);
        } else if (discriminant < 0) {
        System.out.println (" The equation has no real
```

imaginary roots");
}
double y1 = -b / (2 * a);
double y2 = Math.sqrt(-d) / (2 * a);
System.out.println("Root 1 : " + y1 + " " + "Root 2
System.out.println("There is no real solution");
}
}
else
{
System.out.println("Invalid input");
}
}

output:

1. Enter values of a, b, and c:
2
3
5
The equation has imaginary roots. There is no real solution.

2. Enter values of a, b and c:
1 2 1
The equation has equal roots
x1 = -1.0 x2 = -1.0

3. Enter values of a, b and c
1 3 2
The equation has two real solutions : x1 = -1.0, x2 = -2.0

24/9/21

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

3/10/24 Lab Program 2:

```
import java.util.Scanner
class Student {
    String usn;
    String name;
    int credits[];
    int marks[];
    int numSubjects;
public void accept() {
    Scanner sc = new Scanner
                (System.in);
System.out.print("Enter USN, Name
                and No. of subjects:");
    usn = sc.nextLine();
    name = sc.nextLine();
numSubjects = sc.nextInt();
    credits = new int[numSubjects];
    marks = new int[numSubjects];
for (int i = 0; i < num; i++) {
System.out.print("Enter credits
        for subjects " + (i+1) + ": ");
    credits[i] = sc.nextInt();
System.out.print("Enter marks
        for subject " + (i+1) + ": ");
    marks[i] = sc.nextInt();
    }
}
public void display() {
System.out.println("Student Details:");
System.out.println("USN: " + usn +
        "Name: " + name);
System.out.println("Subject details:");
for (int i = 0; i < num; i++)
```

```
System.out.println ("subject "+(i+1)
        +": Credits = "+
        credits[i]+" Marks= "
        + marks[i]);
}
double calculate(){
    int credits = 0;
    double grade = 0.0;
    for(int i=0; i<num; i++)
    {
        int points = Gradepoint(marks[i]);
        credits += credits[i];
        grade += points * credits[i];
    }
    return grade/credits;
}
int Gradepoint(int marks){
    if (marks >=90) return 10;
    else if (marks >=80) return 9;
    else if (marks >=70) return 8;
    else if (marks >=60) return 7;
    else if (marks >=50) return 6;
    else if (marks >=40) return 5;
    else return 0;
}
}
class Main {
    public static void main(String[] args)
    {
        Student s = new Student();
        s.accept();
        s.display();
        double sgpa = s.calculate();
        System.out.print("SGPA: \n", sg);
    }
}
```

Output :

Enter USN : 1BM23CS288
Enter Name : Alice
Enter Number of subjects : 3
Enter credits for subject 1 : 3
Enter marks for subject 1 : 66
Enter credits for subject 2 : 4
Enter marks for subjeet 2 : 88
Enter credits for subjeet 3 : 2
Enter marks for subjeet 3 : 99
Student Details :
USN : 1BM23CS288
Name : Alice
Subject-wise Details :
Subject 1 : credits = 3, Marks = 66
Subject 2 : Credits = 4 Marks = 88
Subjeet 3 : Credits = 2, Marks = 99
SGPA : 8.56.

N
3/10/24

[] args)

);

Q. Create a class Book which contains four members: name, author, price, num_pages. Include constructor to set values for the members. Include methods to set and get details of objects. Include a toString() method that

[9/10/24] Lab Program 3:

```
import java.util.scanner
class Book {
    private string name;
    private string author;
    private double price;
    private int n;
Book ( string name, string author,
        double price, int n) {
    this.name = name;
    this.author = author;
    this.price = price;
    this.n = n;
}
public string toString() {
    return ("Book name: " + this.name
        + "\n" + "Author name: " +
        author + "\n" + "price: "
        + price + "\n" + "number
        of pages: " + n);
}
}
public class Main {
    public static void main (string[] args)
{
    scanner sc = new scanner (system.in);
    int n;
    System.out.println("Enter number
                        of books: ");
        n = sc.nextInt();
        s.nextLine();

    Books [] b = new Books [n];
```

ired display details of Books. Create n objects.

```
for (int i = 0; i < n; i++) {
System.out.println("Book " + (i+1) + ":");
System.out.print("Enter name of
                  Book: ");
   String name = sc.nextline();
System.out.print("Enter Author
                  name: ");
    String author = sc.nextLine();
System.out.print("enter price: ");
    int price = sc.nextInt();
System.out.print("enter number
                  of pages: ");
    int n = sc.nextInt();
    b[i] = new Book(name, author,
                    price, n);
}
System.out.println("\nBook details:");
  for (int i = 0; i < n; i++)
  {
    System.out.println(b[i].toString());
  }
}
```

Output:

```
Enter number of Book:
3
Book1:
Enter name of Book: A
Enter price: Author name: Raul Dalh
Enter price : 200
Enter number of pages : 120
Book 2:
Enter name of Book: B
Enter Author name: Chetan Bhagat
```

Enter price : 400
Enter number of pages : 200
Book 3
Enter name of Book: C
Enter Author name: John
Enter price : 450
Enter number of pages : 250
Book details:
Book name: A
Author name: Raul Dalh
Price : 200.0
Number of Pages: 120
Book name: B
Author name: Chetan Bhagat
Price: 400.0
Number of pages: 200
Book name: C
Author name : John
Price: 450.0
Number of pages : 250

N
19/10/24

24/10/24

Lab Program 4:

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

```
import java.util.Scanner
abstract class Shape {
    private int dim1;
    private int dim2;
    Shape (int dim1, int dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }
    abstract void printArea();
}
class Rectangle extends Shape {
    Rectangle (int l, int b) {
        super (l, b);
    }
    void printArea () {
        int area = dim1 * dim2;
        System.out.println ("Area of
            Rectangle : " + area);
    }
}
```

```
class Triangle extends Shape {
    Triangle (int b, int h) {
        super (b, h);
    }
    void printArea () {
        double area = 0.5 * dim1 * dim2;
        System.out.printlen ("Area of
            triangle : " + area);
    }
}

class circle extends shape {
    Circle (int radius) {
        super (radius, 0);
    }
    void printarea () {
    double area = Math.PI * dim1 * dim2;
    System.out.println ("Area of circle: "
                        + area);
    }
}

class area {
    public static void main (String [] args) {
    Scanner sc = new Scanner (System.in);
    S.O.P ("Enter length and
            breadth of rectangle: ");
    int l = sc.nextInt ();
    int b = sc.nextInt ();
    Shape rectangle = new Rectangle (l, b);
    rectangle.printArea ();

    S.O.P ("Enter base and height of
            triangle: ");
    int base = sc.nextInt ();
    int h = sc.nextInt ();
```

```
Shape triangle = new Triangle(base,h);
triangle.printArea();

S.O.P("Enter radius of the circle:");
int radius = sc.nextInt();
Shape circle = new Circle(radius);
circle.printArea();
}
}
```

output:

```
Enter length and breadth of
rectangle: 2 3
Area of rectangle: 6
Enter base and height of
triangle: 4 8
Area of Triangle: 16.0
Enter radius of circle: 7
Area of circle: 153.93804002589985
```

7/11/24 Lab Program 5:

Develop a java program to create class Bank that maintains 2 kinds of account for its customers, one called savings account and other current acct. savings acct provides compound interest and withdrawal facilities but no check book facility. The current acct provides check book facility but no interest. current acct holders should also maintain a min balance and if balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, acct no. and type of acct. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include necessary methods to achieve following tasks:

a) Accept deposit from customer and update balance
b) Display balance
c) Compute and deposit interest
d) Permit withdrawal and update balance.

Check for min balance, impose penalty if necessary and update balance.

```
import java.util.Scanner;
class Account {
    String customerName;
    String acctno;
    String acctype;
    double balance;
    Account (String custName, String
        acctno, String acctype, double
            balance) ; {
        this.custName = custName;
        this.acctno = acctno;
        this.acctype = acctype;
        this.balance = balance;
}
public void deposit (double amt){
        if (amt > 0) {
            balance + = amt;
            S.O.P ("Amount deposited
                successfully. Update
                    balance : " + balance);
    } else {
            S.O.P ("Invalid deposit amount");
    }
}
public void displayBalance ( ){
    S.O.P ("Account Balance: " + balance);
} } void withdraw (double amt) { }
} void computeandDeposit Interest() { }
  void isChequeBook() { }

class SavAcct extends Account {
    final double interest = 0.04;
    SavAcct (String custName, String
            accno, double ibalance) {
    super (custName, accno, ibalance);
}
```

```
void computeandDepositInterest(){
    double interest = balance *
                      interestRate;
    balance += interest;
    S.O.Pln("Interest added: " +
            interest + " Updated
            Balance = " + balance);
void withdraw(double amt){
    if(amt > 0 && amt <= balance)
    {
        balance -= amt;
        S.O.Pln("Amount withdrawn:
                + amt + " Updated
                balance: " + balance);
    }
    else{
        S.O.Pln("Insufficient balance
                or invalid");
    }
}
}
class curAcct extends Account{
    double minbalance = 500.0;
    double servicecharge = 50.0;
boolean
checkbook
= false;
    curAcct(String custName, String
            acctno, double ibalance,
        super(custName, String acctno,
              double ibalance);
    }
    void checkminbalance(){
        if(balance < minbalance){
        balance -= amt;
        S.O.Pln("Amount withdrawn:
                + amt + " Updated
```

```
S.O.Pln ("Balance below minimum.
        service charge imposed = "
        + servicecharge");
    balance -= servicecharge;
  }
}

void withdraw (double amt) {
  if (amt > 0 && amt <= balance) {
    balance -= amt;
    S.O.Pln ("Amount withdrawn: "
            + amt);
    checkminibalance();
  }
  else {
    s.o.Pln ("Insufficient balance");
  }
}
}

class Bank {
  public static void main (String a[])
  {
    Scanner sc = new Scanner (System.in);
    Account a = new Acc
void ischequeBook() {
  if (chequebook == false) {
    chequebook = true;
    S.O.Pln ("cheque Book issued");
  }
  else {
    S.O.Pln ("cheque has already
            been issued");
  }
}
}
}
```

```
class Bank{
psvm{ Scanner sc = new Scanner(System.in);
public static void main (String [] args){
    Scanner sc = new Scanner (System.in);
    S.O.P ("Enter customer name: ");
    String custname = sc.nextLine();
    S.O.P ("Enter account number : ");
    String accno = sc.nextLine();
    S.O.P ("Enter acct type (savings/
            current) : : ");
    String acctype = sc.nextLine();
    Account account;
    if (acctype.equals ("savings")){
      S.O.P ("Enter initial balance:");
      double balance = sc.nextLine();
                       Double
      S.O.P ("Enter interest rate: ");
      double interestRate = sc.nextDouble();
      account = new SavAcct (custname,
        accno, balance, interestRate);
} else if (acctype.equals ("current")){
      S.O.P ("Enter initial balance: ");
      double balance = sc.nextDouble();
      S.O.P ("Enter minimum balance:");
      double mbalance = sc.nextDouble();
      S.O.P ("Enter service charge");
      double scharge = sc.nextDouble();
      account = new CurAcct (custname,
        accno, balance, mbalance,
        scharge);
}
if (acctype.equals ("Savings"))
{ while(1){
    S.O.P ("1. Deposit \n 2. Withdraw
```

```
3. Display \n 4. Exit");
S.O.P (" Enter choice ");
int ch = sc.nextInt();
switch (ch)
{
  case1: S.O.P ("Enter amt");
         double damt = sc.nextDouble();
         savAcct account.deposit(amt);
         acct.computeandDepositInterest
                                    ();
         break;
  case 2: S.O.P ("Enter amt to
                  withdraw: ");
          double w = sc.nextDouble();
          acct.withdraw(w);
          acct.displayBalance();
          break;
  case 3: acct.displayBalance();
          break;
  case 4: exit();
          break;
}
} else { while(1) {
  S.O.P ("1. Deposit \n 2. Withdraw \n 3.
          ChequeBook \n 4. Exit ");
  S.O.P (" Enter choice ");
  int i = sc.nextInt();
  switch (ch)
  {
    case 1: S.O.P ("Enter amt");
            double damt = sc.nextDouble();
            account.deposit(damt);
            break;
    case 2: S.O.P ("Enter amt");
            double w = sc.nextDouble();
```

```
account.withdraw(w);
account.displayBalance();
break;
case 3: account.issueChequeBook();
        break;
case 4: exit(0);
        break;
}}
}
```

Output:

1. Enter customer name: Alice
Enter account number: 55331901
Enter acct type (saving/current): Savings
Enter initial balance: 50000
Enter initial interest: 2
1. Deposit
2. Withdraw
3. Display
4. Exit
Enter choice: 1
Enter amt: 2000
Amount deposited successfully
Update balance: 52000
~~Enter choice: 2~~
~~Enter~~ Interest added: 1040
Enter choice: 2
Enter amt to withdraw: 2000
Amount withdrawn: 2000
Update Balance: 50000
Enter choice: 4

2. Enter customer name: Amy
Enter account number: 2911502901
Enter acct type (savings/current):
current
Enter initial balance: 60000
Enter minimum balance: 10000
Enter service charge: 500
1. Deposit
2. Withdraw
3. Cheque book
4. Exit
Enter choice: 1
Enter amt: 2000
Amount deposited successfully
Update balance: 62000
Enter choice: 2
Enter amt: 62000
Amount withdrawn: 62000
Balance below minimum
service charge imposed: 500
Enter choice: 4.

23/12/24

14/11/24 Lab Program 6:

Create a package CIE which has two classes - Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in 5 courses of 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 5 courses of current semester of the student. Import two packages in a file that declares final marks of n students in all 5 courses.

CIE Package:

```
package CIE;
class student {
    string usn;
    string name;
    int sem;
student ( string u, string n, int s){
        usn = u;
        name = n;
        sem= s;
    }
void printdetails( ){
    S.O.Pln (" Student Name : " + name);
```

```
        S.O.Pln(" USN: " + usn);
        S.O.Pln(" Semester : " + sem);
    }
class Internals {
    int [] marks = new int[5];
    Internals (int [] marks){
        this.marks = marks;
    }
    void printmarks(){
        S.O.Pln(" Internal Marks: ");
        for(int i =0; i < 5; i++){
            S.O.Pln(" course " + (i+1) +
                    " : " + marks[i]);
    }
}
}
```

SEE package:

```
package SEE; → import CIE.student;
import java.util.scanner;
class External extends student
    int [] seemarks = new int[5];
External (String usn, String name, int sem, int [] marks) {
    seemarks = marks; → super(usn, name, sem);
}
void printMarks(){
    S.O.Pln(" SEE Marks: ");
    for(int i=0; i<5; i++){
        S.O.Pln(" course " + (i+1) +
                " : " + seeMarks[i]);
    }
}
```

Main program:

```
import CIE.Student;
import SEE.External;
import java.util.Scanner;

class Main {
    psvm(String[] args) {
        Scanner sc = new Scanner(System.in);
        S.O.P("Enter student name: ");
        String name = sc.nextLine();
        S.O.P("Enter USN: ");
        String usn = sc.nextLine();
        S.O.P("Enter semester: ");
        int sem = sc.nextInt();
        Student s = new Student(usn,
                    name, sem);
        int[] internalmarks = new int[5];
        S.O.Pln("Enter Internal marks
                for 5 courses: ");
        for(int i = 0; i < 5; i++){
            S.O.P("course" + (i+1) + ": ");
            internalmarks[i] = sc.nextInt();
        }
        Internals i = new Internals(internal
                      marks);

        int[] seeMarks = new int[5];
        S.O.Pln("Enter SEE marks for
                5 courses: ");
        for(int i = 0; i < 5; i++){
            S.O.P("course" + (i+1) + ": ");
            seeMarks[i] = sc.nextInt();
        }
        External e = new External(seeMarks);
    }
}
```

```
S.O.Pln("\n Student Details: \n");
s.printDetails();
i.printMarks();
e.printMarks();
S.O.Pln("\n Final Marks \n");
for(int i=0; i<5; i++){
double finalmarks = (internal
                     marks[i]
                     )
                     + seeMarks[i]
                     * 0.5);
S.O.Pln("course "+(i+1)+": "
                     + finalmarks);
}
}
```

Output:

Enter student name: Alice
Enter USN : 1BM23CS001
Enter Semester : 3
Enter Internal marks for 5 courses:
Course 1: 30
course 2: 25
Course 3: 31
course 4: 28
Course 5: 26
Enter SEE marks for 5 courses:
course 1: 60
Course 2: 75
course 3: 80
course 4: 90
course 5: 85

Student Details:
Student Name: Alice
USN : 1BM23CS001
Semester : 3
Internal Marks:
course1 : 30
course 2 : 25
course 3: 31
course 4: 28
course 5: 26
SEE marks:
course 1: 60
course 2: 45
course 3: 80
course 4: 90
course 5: 85
Final Marks:
course 1 : 60
course 2 : 62.5
course 3 : 71
course 4 : 73
course 5 : 68.5

21/4/24

21/11/24 Lab Program 7:

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

```
import java.util.Scanner;
class WrongAgeException extends
Exception {
public String toString() {
return "WrongAgeException: Age
cannot be negative";}
} class SonOlderThanFatherException extends
Exception {
public String toString() {
return "SonOlderThanFatherException:
Son's age cannot be greater
than or equal to Father's
age.";}
}
class Father {
int age;
Father (int age) throws WrongAge
Exception {
```

```
if (age < 0) {
throw new WrongAgeException();}
this.age = age;
S.O.P ln ("Father's age : " + age);}
}
class son extends Father {
int sonAge;
Son( int fatherAge, int sonAge)
throws WrongAgeException,
SonOlderThan FatherException {
super (fatherAge);
if (sonAge < 0) {
throw new WrongAge Exception();
} if (sonAge >= fatherAge) {
throw new SonOlderThanFather Exception();
}
this.sonAge = sonAge;
S.O.P ln (" Son's age: " + sonAge);}
}
class ExceptMain {
S.O. psvm (String [] args) {
Scanner sc = new Scanner (System.in);
try {
S.O.P (" Enter Father's Age : ");
int fatherAge = sc.nextInt();
S.O.P (" Enter Son's Age : ");
int sonAge = sc.nextInt();
Son son = new Son (fatherAge,
sonAge);
} catch (WrongAgeException e) {
S.O.P ln (e);
} catch (SonOlderThanFatherException e) {
S.O.P ln (e);
}}}
```

Output :

1. Enter Father's age : 45
   Enter son's age : 65
   Father's age : 45
   SonOlderThanFatherException :
   Son's Age cannot be greater than or equal to Father's age.

2. Enter Father's age : -20
   Enter son's age : 5
   WrongAgeException : Age cannot be negative.

3. Enter Father's age : 55
   Enter son's age : 20
   Father's age : 55
   son's age : 20.

28/11/24 Lab program 8:

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

```
class DisplayBMS implements Runnable
{
    public void run(){
        try {
        while (true){
        System.out.println("BMS college of
                                Engineering");
        Thread.sleep(10000);
        }
    } catch ( InterruptedException e) {
        System.out.println("Thread
                                interrupted");
    }
}
}

class displayCSE implements Runnable {
public void run(){
    try {
        while (true) {
        System.out.println("CSE");
        Thread.sleep(2000);
        }
    } catch (InterruptedException e) {
        System.out.println("Thread
                                interrupted");
}}}
```

```
public class threaddemo{

public static void main (String [] args)
{
  Thread t1 = new Thread (new Display
                          BMS());
  Thread t2 = new Thread (new Display
                          CSE());
    t1.start();
    t2.start();
  }
}
```

Output:

```
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
^C
```

23/12/22

18
29/12/24

Lab program 9:

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were zero, the program would throw an 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.LEFT
                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);
  }
});

setSize (400, 300);
setTitle ("Division Program");
setVisible (true);
}
```

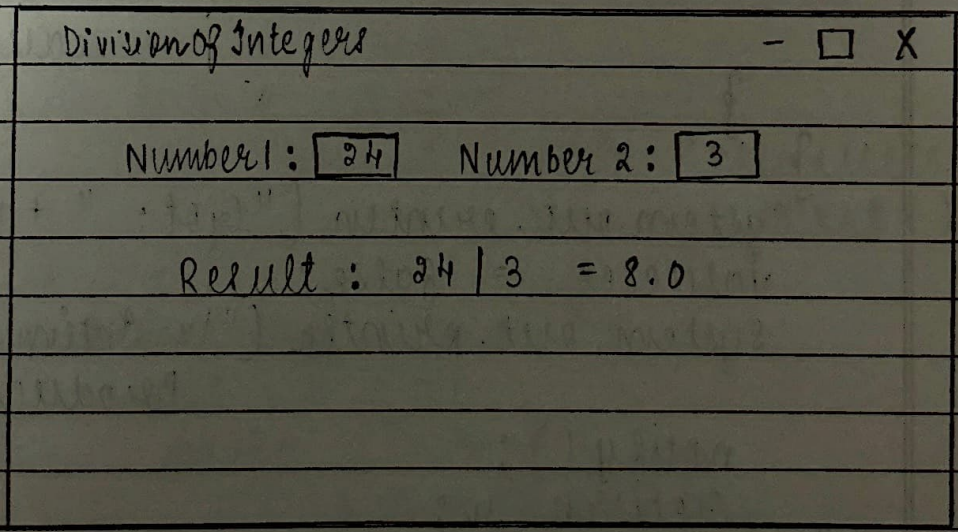
```
public void actionPerformed (ActionEvent ae)
{
    int n1, n2 ;
    try {
        if ( ae.getSource() == dResult ) {
        n1 = Integer.parseInt (num1.getText());
        n2 = Integer.parseInt (num2.getText());

        if (n2 == 0) {
        throw new ArithmeticException
            ("Division by zero is not
                allowed");
        }
        out = n1 + "/" + n2 + " = ";
      resultNum = (double) n1 / n2 ;
        out += String.valueOf (resultNum);
        repaint ();
    }
} catch ( NumberFormatException e1) {
    flag = 1;
    out = "Number Format Exception!"
                + e1.getMessage();
    repaint ();
} catch ( ArithmeticException e2) {
    flag = 1;
    out = " Divide by 0 Exception!"
                + e2.getMessage ();
    repaint();
}
}

public void paint ( Graphics g ) {
    if ( flag == 0 ) {
    g.drawString (out, outResult.getX() +
```

```
ae)        outResult.getWidth() + 10, outResult.getY()
             + 20);
        } 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("DivisionOfIntegers");
        dm.setVisible(true);
    }
m);
}
```

Output :



18
~~24~~/12/24 Program 10:

Demonstrate Interprocess communication and deadlock.

IPC:

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

  synchronised int get() {
  while (! valueset) {
    try {
      System.out.println("In consumer
                          waiting In");

      wait();
    } catch ( InterruptedException e) {
      System.out.println(" Interrupted
                          Exception
                          caught ");

    }
  }
  System.out.println ("Got: " + n);
  valueset = false;
  System.out.println ("In Intimate
                       Producer In");

  notify();
  return n;
}

synchronized void put (int n) {
  while (valueset) {
    try {
      System.out.println ("In Producer
```

```
            waiting \n");
            wait();
        } catch (InterruptedException e) {
            System.out.println("Interrupted
                                Exception
                                caught");
        }
    }
    this.n = n;
    valueset = true;
    System.out.println("Put : " + n);
    System.out.println("\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 :
Press Control - C to stop.
Put : 0

Intimate consumer

Producer waiting
Got : 0
Intimate Producer
Put : 1
Intimate consumer

Producer waiting
consumed : 0
Got : 1
Intimate Producer

Consumed : 1
Put : 2

Intimate consumer

Producer waiting

Got : 2
Intimate Producer

consumed : 2
Put : 3
^C

Deadlock :

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

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

```
        System.out.println("B Interrupted");
    }
    System.out.println(name + " trying
                        to call A.last()");
    a.last();
}
synchronized void last() {
    System.out.println("Inside B.last");
}
}

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() {
    Thread.currentThread().setName
                        ("MainThread");
    Thread t = new Thread(this,
                        "RacingThread");
    t.start();

    a.foo(b);
    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 :

```
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last()
MainThread trying to call B.last()
```