VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SAMHITHA A(1BM23CS293)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019
Sep 2024-Jan 2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by **SAMHITHA A(1BM23CS293),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,

Department of CSE,

BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,

Department of CSE,

BMSCE, Bengaluru

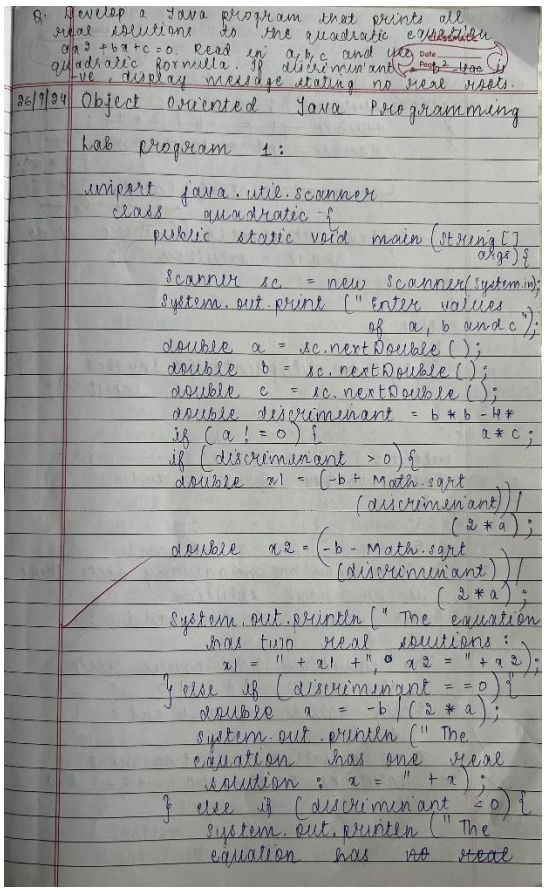
INDEX

GITHUB LINK: https://github.com/Samhithagit/1BM23CS293-OOJLAB

Program 1:

Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2 - 4ac is negative, display a message stating that there are no real solutions.

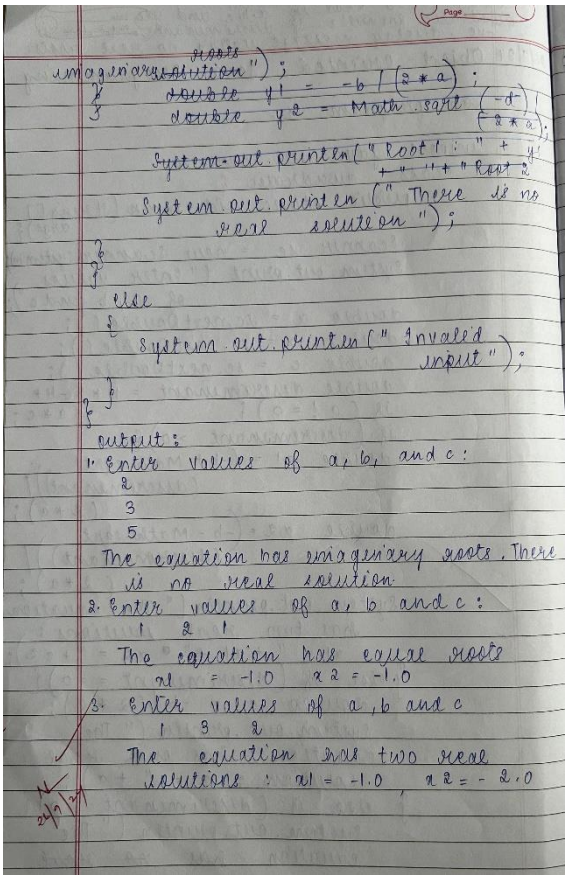
ALGORITHM:


Q. Develop a Java program that prints all real solutions to the quadratic equation ax2+bx+c=0. Read in a,b,c and use the quadratic formula. If discriminant b2-4ac is -ve, display message stating no real roots.

26/7/24 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 solution");
        double y1 = -b / (2*a);
        double y2 = Math.sqrt(-d) / (2*a);
        System.out.println("Root 1 : " + y1 + " i" + " Root 2 : " + y1 + " -i");
        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

CODE:

```
import java.util.Scanner;

class quadratic {

public static void main(String args[])

Scanner sc = new Scanner(System.in);

System.out.println("Enter values of a, b, and c:");

double a = sc.nextDouble();

double b = sc.nextDouble();

double c = sc.nextDouble();

double d = b * b - 4 * a * c;

  if (a != 0) {
```

```
    if (d > 0) {
        double x1 = (-b + Math.sqrt(d)) / (2 * a);
        double x2 = (-b - Math.sqrt(d)) / (2 * a);
        System.out.println("The equation has two real solutions: x1 = " + x1 + ", x2 = " + x2);
        } else if (d == 0) {
           double x = -b / (2 * a);
           System.out.println("The equation has one real solution: x = " + x);
        } else {
           System.out.println("The equation has imaginary roots. There is no real solution.");
        }
      } else {
        System.out.println("Invalid input: 'a' cannot be zero.");
      }
   }
}
```

OUTPUT:

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

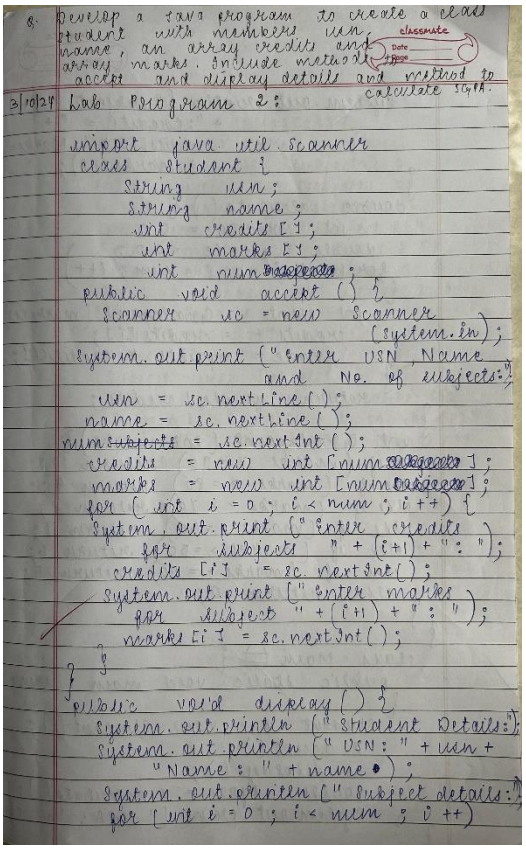
```
Enter values of a, b, and c:
2 3 5
The equation has imaginary roots. There is no real solution.
```

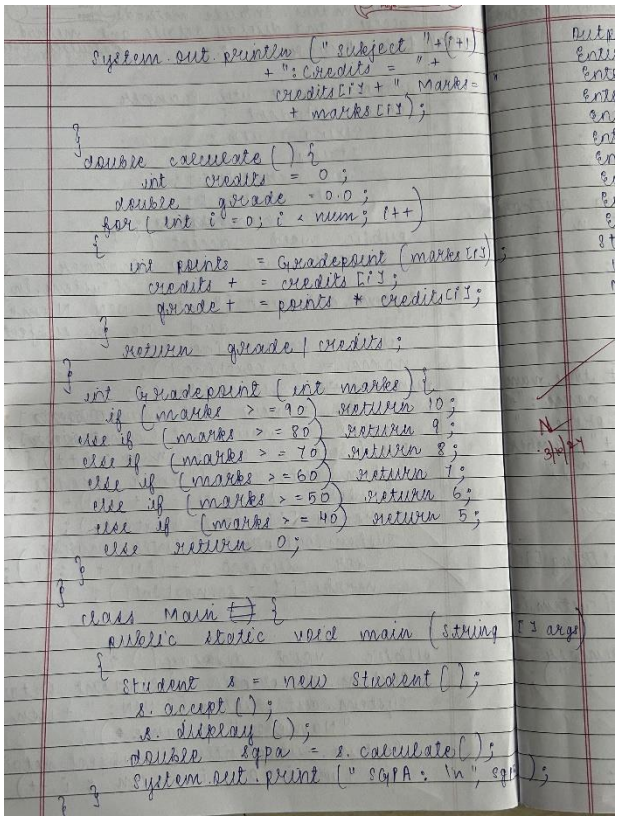
```
Enter values of a, b, and c:
1 3 2
The equation has two real solutions: x1 = -1.0, x2 = -2.0
```

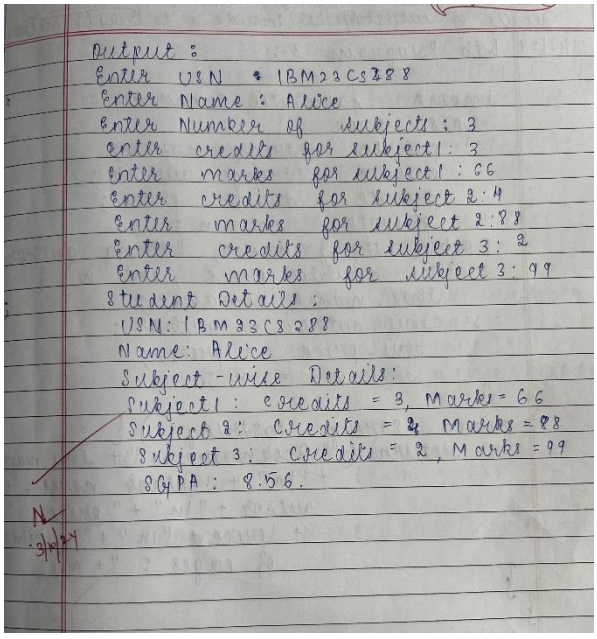
Program 2:

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

ALGORITHM:







CODE:

```
import java.util.Scanner;
class Student {
String usn;
String name;
int[] credits;
int[] marks;
int num;
public void accept() {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter USN: ");
usn = scanner.nextLine();
System.out.print("Enter Name: ");
name = scanner.nextLine();
System.out.print("Enter number of subjects: ");
num = scanner.nextInt();
credits = new int[num];
marks = new int[num];
for (int i = 0; i < num; i++) {
System.out.print("Enter credits for subject " + (i + 1) + ": ");
credits[i] = scanner.nextInt();
System.out.print("Enter marks for subject " + (i + 1) + ": ");
marks[i] = scanner.nextInt();
}
}
```

```
public void display() {
System.out.println("Student Details:");
System.out.println("USN: " + usn);
System.out.println("Name: " + name);
System.out.println("Subject-wise Details:");
for (int i = 0; i < num; i++) {
System.out.println("Subject " + (i + 1) + ": Credits = " + credits[i] + ", Marks = " + marks[i]);
}
}
public 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;
}
public 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;

}

}

public class Studentmain {

public static void main(String[] args) {

Student student = new Student();

student.accept();

student.display();

double sgpa = student.calculate();

System.out.printf("SGPA: %.2f\n", sgpa);

}

}
```

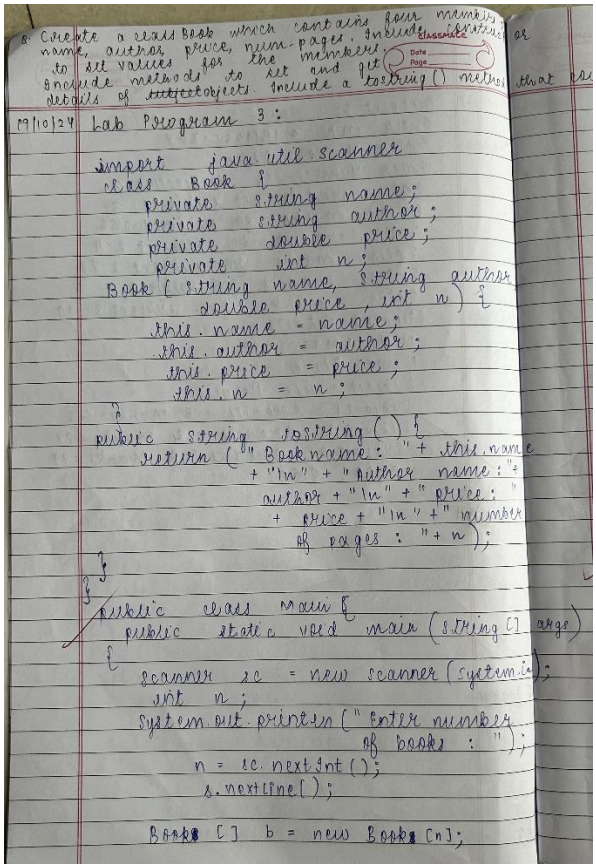
OUTPUT:

```
Enter USN: 1BM23CS003
Enter Name: Alice
Enter number of subjects: 5
Enter credits for subject 1: 4
Enter marks for subject 1: 78
Enter credits for subject 2: 4
Enter marks for subject 2: 88
Enter credits for subject 3: 3
Enter marks for subject 3: 66
Enter credits for subject 4: 3
Enter marks for subject 4: 45
Enter credits for subject 5: 2
Enter marks for subject 5: 100
Student Details:
USN: 1BM23CS003
Name: Alice
Subject-wise Details:
Subject 1: Credits = 4, Marks = 78
Subject 2: Credits = 4, Marks = 88
Subject 3: Credits = 3, Marks = 66
Subject 4: Credits = 3, Marks = 45
Subject 5: Credits = 2, Marks = 100
SGPA: 7.75
```

Program 3:

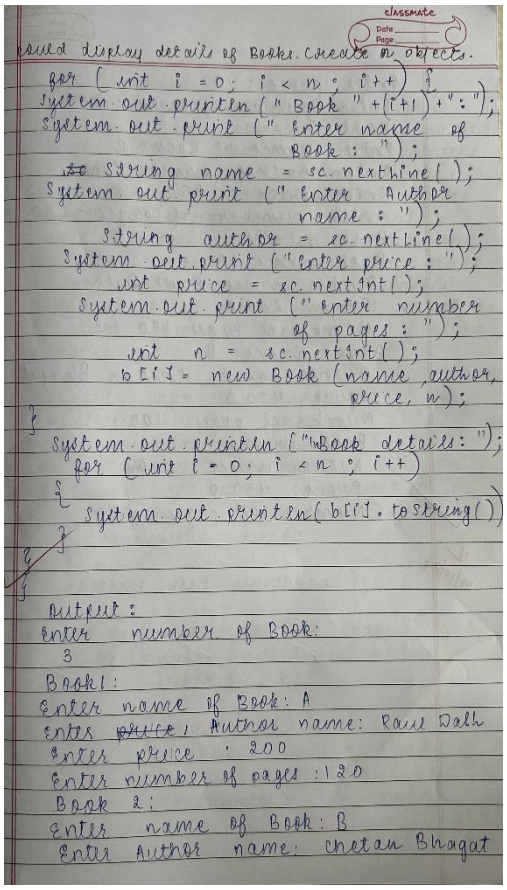
Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program to create n book objects.

ALGORITHM:

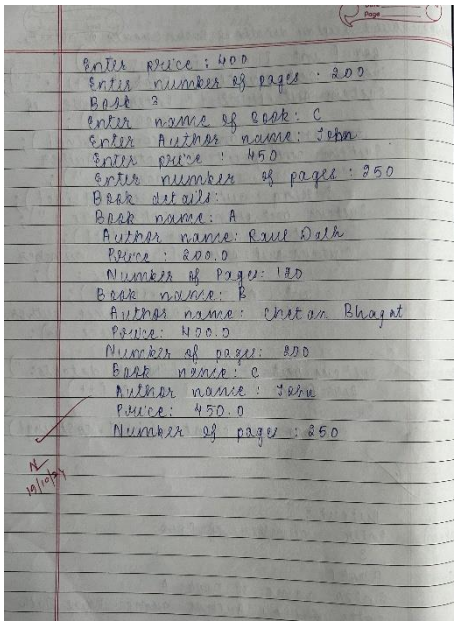


```
Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString() method that
```

```
19/10/24  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];
```



```
Read 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("Book 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 Author name: Paul Walk
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: Paul Walk
Price : 200.0
Number of Page: 120
Book name: B
Author name: Chetan Bhagat
Price: 400.0
Number of page: 200
Book name: C
Author name : John
Price: 450.0
Number of page : 250
```

CODE:

```
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: " + this.author + "\n" + "Price:
" + this.price + "\n" + "Number of pages: " + n;

    }

}

public class Bookdemo {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n;

        System.out.println("Enter number of books: ");

        n = sc.nextInt();

        sc.nextLine();

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

```
 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: ");
            double price = sc.nextDouble();
            System.out.print("Enter number of pages: ");
            int numPages = sc.nextInt();
            sc.nextLine();
            b[i] = new Book(name, author, price, numPages);
        }
System.out.println("Book details:");
        for (int i = 0; i < n; i++) {
            System.out.println(b[i].toString());
        }
sc.close();
}
}
```

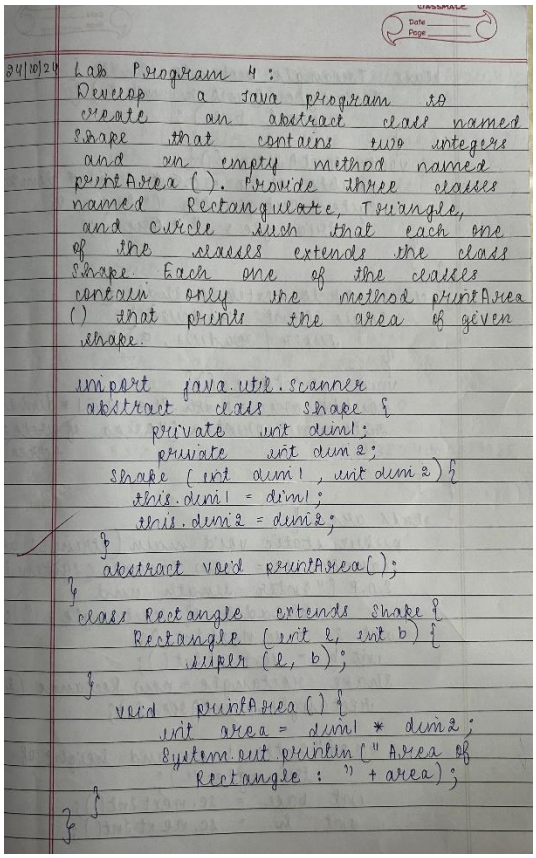
OUTPUT:

```
Enter number of books:
3
Book 1:
Enter name of Book: Brave New World
Enter Author name: Aldous Huxley
Enter price: 500
Enter number of pages: 250
Book 2:
Enter name of Book: Merchant of Venice
Enter Author name: Shakespeare
Enter price: 750
Enter number of pages: 400
Book 3:
Enter name of Book: The Da Vinci Code
Enter Author name: Dan Brown
Enter price: 999
Enter number of pages: 689
Book details:
Book name: Brave New World
Author name: Aldous Huxley
Price: 500.0
Number of pages: 250
Book name: Merchant of Venice
Author name: Shakespeare
Price: 750.0
Number of pages: 400
Book name: The Da Vinci Code
Author name: Dan Brown
Price: 999.0
Number of pages: 689
```

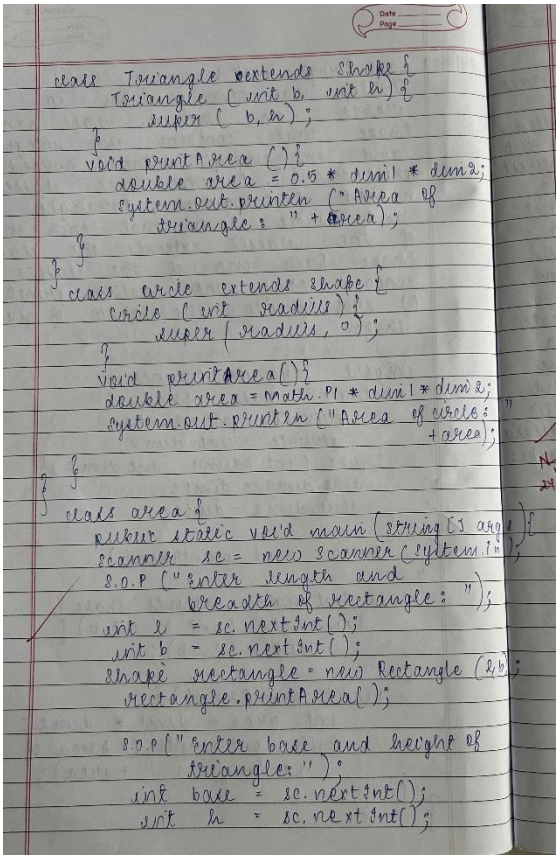
Program 4:

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

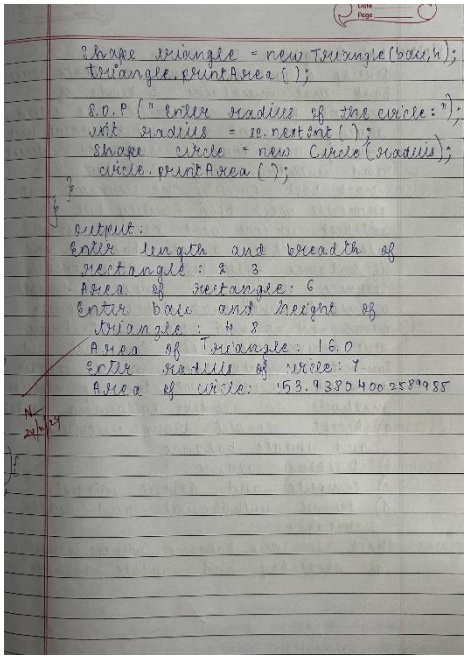
ALGORITHM:



```
3/4/10/24 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.println ("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
```

CODE:

```
import java.util.Scanner;
abstract class Shape {
    private int dimension1;
    private int dimension2;
public Shape(int dimension1, int dimension2) {
        this.dimension1 = dimension1;
        this.dimension2 = dimension2;
    }
 abstract void printArea();
}
class Rectangle extends Shape {
    public Rectangle(int length, int breadth) {
        super(length, breadth);
    }
 void printArea() {
        int area = dimension1 * dimension2;
        System.out.println("Area of Rectangle: " + area);
    }
}
class Triangle extends Shape {
    public Triangle(int base, int height) {
        super(base, height);
    }
void printArea() {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}
class Circle extends Shape {
```

```
    public Circle(int radius) {
        super(radius, 0);
    }
void printArea() {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}
public class area {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
System.out.print("Enter length and breadth of the rectangle: ");
        int length = sc.nextInt();
        int breadth = sc.nextInt();
        Shape rectangle = new Rectangle(length, breadth);
        rectangle.printArea();
System.out.print("Enter base and height of the triangle: ");
        int base = sc.nextInt();
        int height = sc.nextInt();
        Shape triangle = new Triangle(base, height);
        triangle.printArea();
System.out.print("Enter radius of the circle: ");
        int radius = sc.nextInt();
        Shape circle = new Circle(radius);
        circle.printArea();

    }
}
```

OUTPUT:

```
Enter length and breadth of the rectangle: 3 6
Area of Rectangle: 18
Enter base and height of the triangle: 6 4
Area of Triangle: 12.0
Enter radius of the circle: 7
Area of Circle: 153.93804002589985
```

Program 5:

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

a) Accept deposit from customer and update the balance.

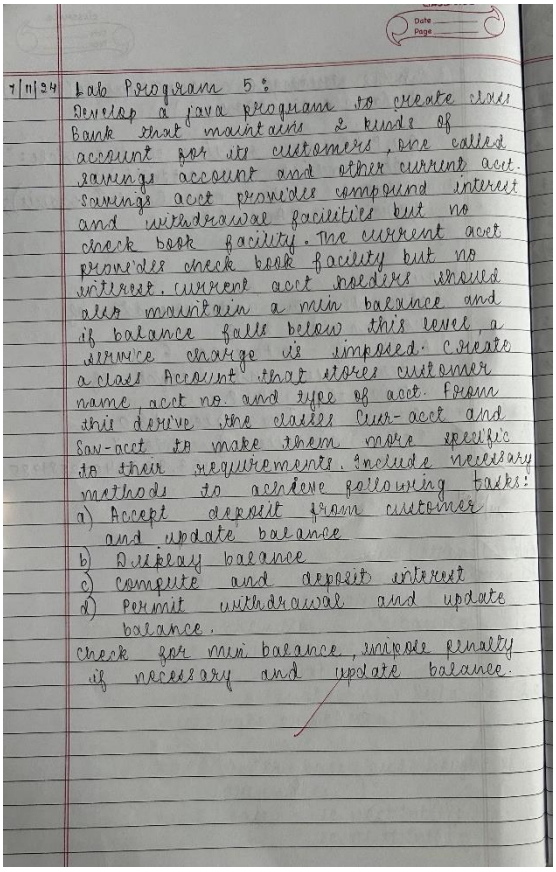
b) Display the balance.

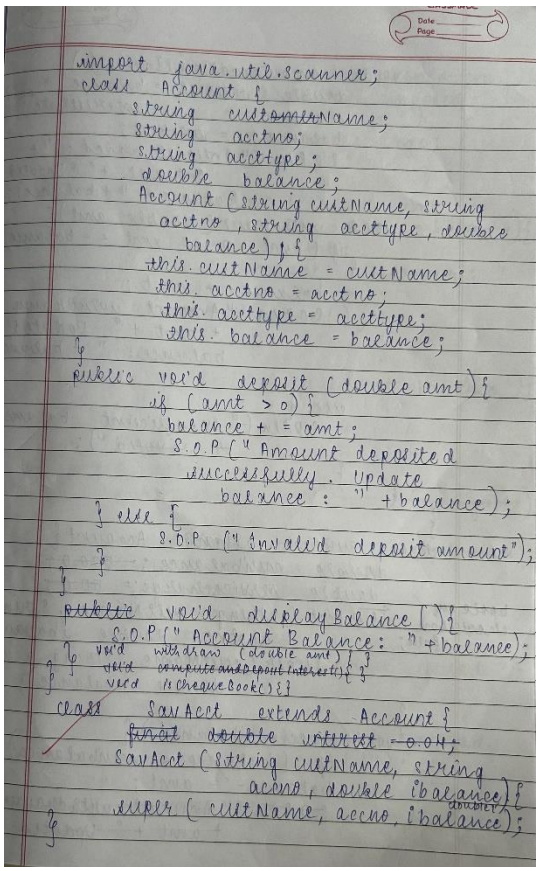
c) Compute and deposit interest

d) Permit withdrawal and update the balance

Check for the minimum balance, impose penalty if necessary and update the balance.

ALGORITHM:



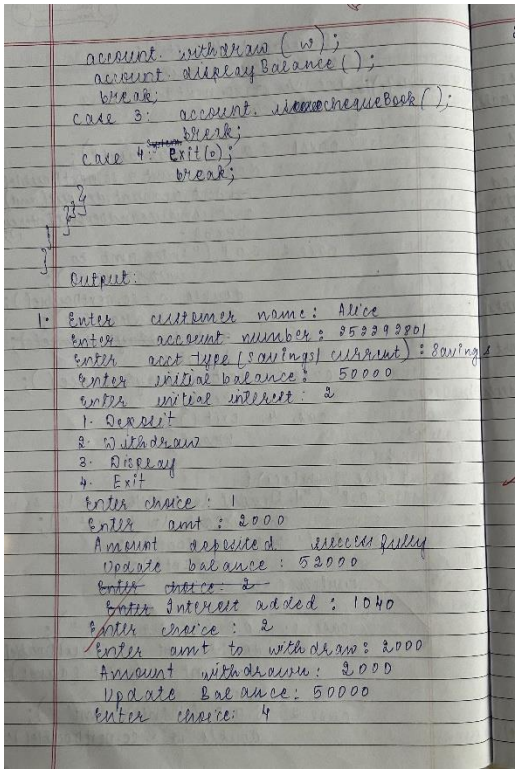


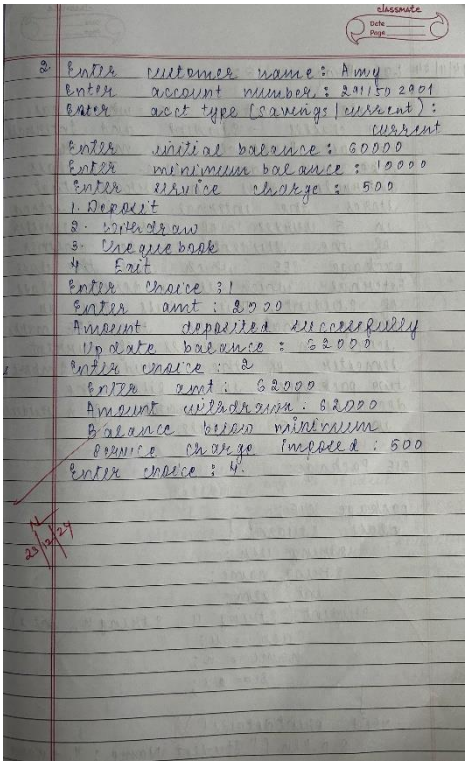
```
void computeAndDepositInterest() {
    double interest = balance * interestRate;
    balance += interest;
    S.O.P.ln("Interest added: " + interest + " Updated Balance = " + balance);
}
void withdraw(double amt) {
    if (amt > 0 && amt <= balance)
    {
        balance -= amt;
        S.O.P.ln("Amount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

```
S.O.P.ln("Balance below minimum. Service charge imposed = " + servicecharge");
    balance -= servicecharge;
}
void withdraw(double amt) {
    if (amt > 0 && amt <= balance) {
        balance -= amt;
        S.O.P.ln("Amount withdrawn: " + amt);
        checkminbalance();
    }
    else {
        S.O.P.ln("Insufficient balance");
    }
}
class Bank {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        Account a = new Acc
    void isChequeBook() {
        if (chequebook == false) {
            chequebook = true;
            S.O.P.ln("Cheque Book issued");
        }
        else {
            S.O.P.ln("Cheque has already been issued");
        }
    }
}
```

```
class Bank {
    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.nextDouble();
            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 2. Withdraw
```

```
3. Display 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 2. Withdraw 3. ChequeBook 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();
```





CODE:

```
import java.util.Scanner;


class Account {
    String customerName;
    String accountNumber;
    String accountType;
    double balance;


    Account(String customerName, String accountNumber, String accountType, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = initialBalance;
    }


    public void deposit(double amount) {
```

```
        if (amount > 0) {
            balance += amount;
            System.out.println("Amount deposited successfully. Updated balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

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

class SavAcct extends Account {
    double interestRate;

    SavAcct(String customerName, String accountNumber, double initialBalance, double
interestRate) {
        super(customerName, accountNumber, "Savings", initialBalance);
        this.interestRate = interestRate;
    }

    public void computeAndDepositInterest() {
        double interest = balance * interestRate / 100;
        balance += interest;
        System.out.println("Interest added: " + interest + ". Updated balance: " + balance);
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
```

```
            System.out.println("Amount withdrawn: " + amount + ". Updated balance: " +
balance);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}
class CurAcct extends Account {
    double minimumBalance;
    double serviceCharge;
    boolean chequebookIssued = false;

    CurAcct(String customerName, String accountNumber, double initialBalance, double
minimumBalance, double serviceCharge) {
        super(customerName, accountNumber, "Current", initialBalance);
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }

    void checkMinBalance() {
        if (balance < minimumBalance) {
            balance -= serviceCharge;
            System.out.println("Balance below minimum. Service charge of " + serviceCharge + "
imposed. Updated balance: " + balance);
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Amount withdrawn: " + amount + ". Updated balance: " +
balance);
```

```
            checkMinBalance();
        } else {
            System.out.println("Insufficient balance.");
        }
    }

    public void issueChequeBook() {
        if (!chequebookIssued) {
            chequebookIssued = true;
            System.out.println("Cheque book issued.");
        } else {
            System.out.println("Cheque book has already been issued.");
        }
    }
}

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

        System.out.println("Enter customer name:");
        String custName = sc.nextLine();
        System.out.println("Enter account number:");
        String accNo = sc.nextLine();
        System.out.println("Enter account type (Savings/Current):");
        String accType = sc.nextLine();

        Account account;

        if (accType.equalsIgnoreCase("Savings")) {
```

```
        System.out.println("Enter initial balance:");
        double balance = sc.nextDouble();
        System.out.println("Enter interest rate:");
        double interestRate = sc.nextDouble();
        account = new SavAcct(custName, accNo, balance, interestRate);

        while (true) {
            System.out.println("\n1. Deposit\n2. Withdraw\n3. Compute Interest\n4. Display Balance\n5. Exit");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter amount to deposit: ");
                    double depositAmount = sc.nextDouble();
                    account.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter amount to withdraw: ");
                    double withdrawAmount = sc.nextDouble();
                    ((SavAcct) account).withdraw(withdrawAmount);
                    break;
                case 3:
                    ((SavAcct) account).computeAndDepositInterest();
                    break;
                case 4:
                    account.displayBalance();
                    break;
                case 5:
                    System.out.println("Exiting Savings Account menu...");
```

```
                return;
            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
} else if (accType.equalsIgnoreCase("Current")) {
    System.out.println("Enter initial balance:");
    double balance = sc.nextDouble();
    System.out.println("Enter minimum balance:");
    double minBalance = sc.nextDouble();
    System.out.println("Enter service charge:");
    double serviceCharge = sc.nextDouble();
    account = new CurAcct(custName, accNo, balance, minBalance, serviceCharge);

    while (true) {
        System.out.println("\n1. Deposit\n2. Withdraw\n3. Issue Cheque Book\n4. Display Balance\n5. Exit");
        System.out.print("Enter choice: ");
        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                System.out.print("Enter amount to deposit: ");
                double depositAmount = sc.nextDouble();
                account.deposit(depositAmount);
                break;
            case 2:
                System.out.print("Enter amount to withdraw: ");
                double withdrawAmount = sc.nextDouble();
                ((CurAcct) account).withdraw(withdrawAmount);
                break;
```

```
                case 3:
                    ((CurAcct) account).issueChequeBook();
                    break;
                case 4:
                    account.displayBalance();
                    break;
                case 5:
                    System.out.println("Exiting Current Account menu...");
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    } else {
        System.out.println("Invalid account type. Exiting...");
    }
 sc.close();
  }
}
```

OUTPUT:

```
Enter customer name:
Alice
Enter account number:
8997237904994830
Enter account type (Savings/Current):
Savings
Enter initial balance:
50000
Enter interest rate:
2

1. Deposit
2. Withdraw
3. Compute Interest
4. Display Balance
5. Exit
Enter choice: 1
Enter amount to deposit: 2000
Amount deposited successfully. Updated balance: 52000.0

1. Deposit
2. Withdraw
3. Compute Interest
4. Display Balance
5. Exit
Enter choice: 3
Interest added: 1040.0. Updated balance: 53040.0
```

```
1. Deposit
2. Withdraw
3. Compute Interest
4. Display Balance
5. Exit
Enter choice: 2
Enter amount to withdraw: 60000
Insufficient balance.

1. Deposit
2. Withdraw
3. Compute Interest
4. Display Balance
5. Exit
Enter choice: 2
Enter amount to withdraw: 50000
Amount withdrawn: 50000.0. Updated balance: 3040.0

1. Deposit
2. Withdraw
3. Compute Interest
4. Display Balance
5. Exit
Enter choice: 5
Exiting Savings Account menu...
```

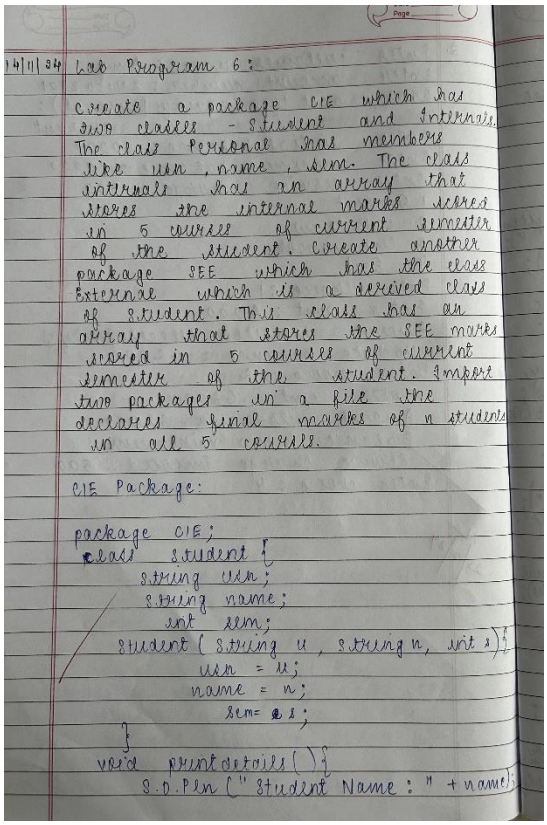
```
Enter account number:
7803789976502185
Enter account type (Savings/Current):
Current
Enter initial balance:
60000
Enter minimum balance:
10000
Enter service charge:
500

1. Deposit
2. Withdraw
3. Issue Cheque Book
4. Display Balance
5. Exit
Enter choice: 1
Enter amount to deposit: 2000
Amount deposited successfully. Updated balance: 62000.0

1. Deposit
2. Withdraw
3. Issue Cheque Book
4. Display Balance
5. Exit
Enter choice: 3
Cheque book issued.

1. Deposit
2. Withdraw
3. Issue Cheque Book
4. Display Balance
5. Exit
Enter choice: 2
Enter amount to withdraw: 52000
Amount withdrawn: 52000.0. Updated balance: 10000.0
```

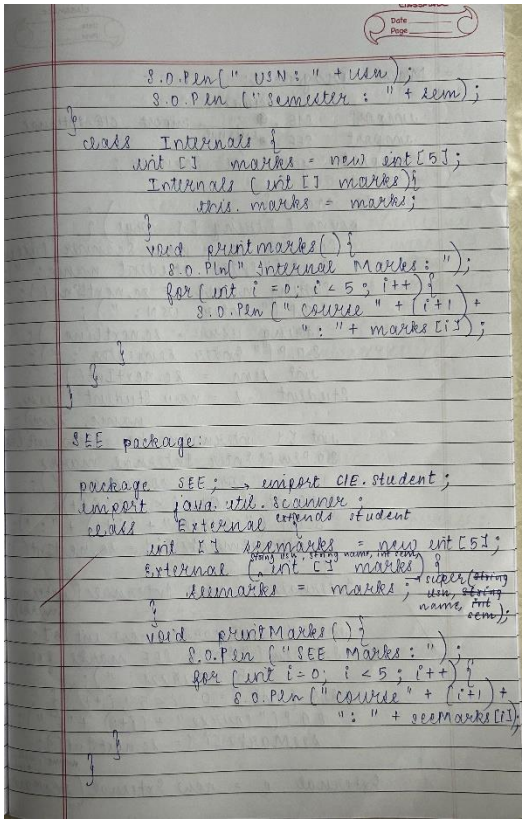
```
1. Deposit
2. Withdraw
3. Issue Cheque Book
4. Display Balance
5. Exit
Enter choice: 2
Enter amount to withdraw: 2000
Amount withdrawn: 2000.0. Updated balance: 8000.0
Balance below minimum. Service charge of 500.0 imposed. Updated balance: 7500.0

1. Deposit
2. Withdraw
3. Issue Cheque Book
4. Display Balance
5. Exit
Enter choice: 4
Account Balance: 7500.0

1. Deposit
2. Withdraw
3. Issue Cheque Book
4. Display Balance
5. Exit
Enter choice: 5
Exiting Current Account menu...
```

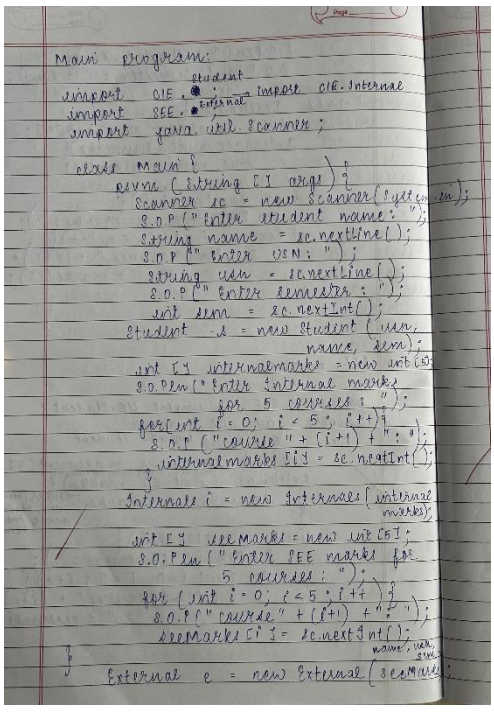
Program 6:

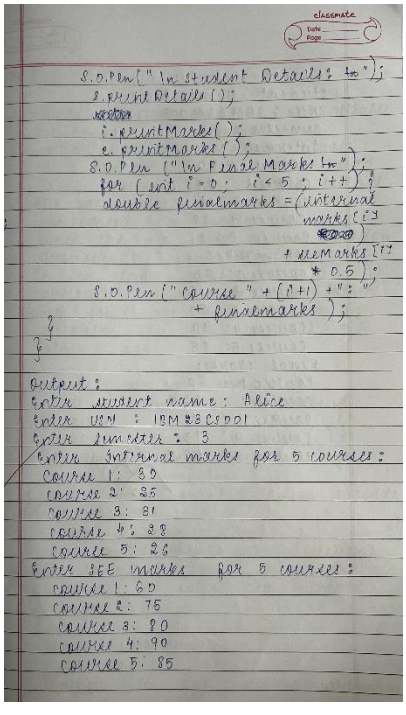
Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External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

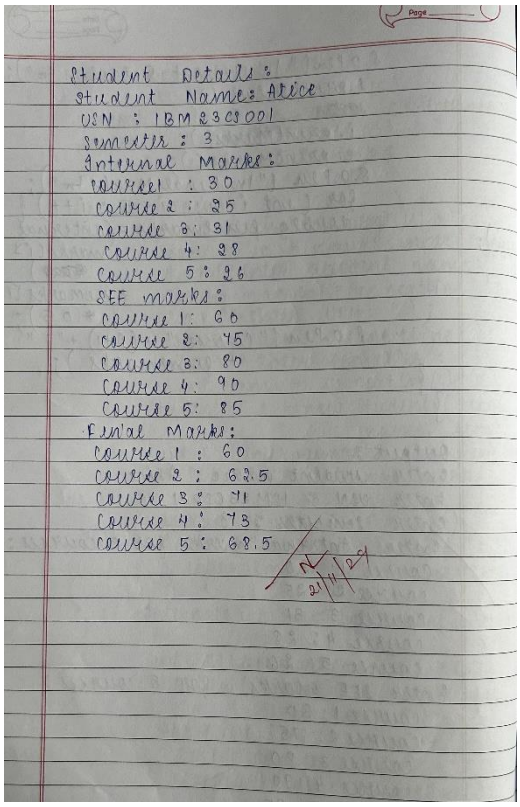
ALGORITHM:











CODE:

//package CIE

package CIE;

public class Student {

String usn;

String name;

int sem;

public Student(String u, String n, int s) {

usn = u;

name = n;

sem = s;

}

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

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

public Internals(int[] marks) {
this.marks = marks;
}

public void printMarks() {
System.out.println("Internal Marks: ");
for (int i = 0; i < 5; i++) {
System.out.println("Course " + (i + 1) + ": " + marks[i]);
}
}
}

//package SEE
package SEE;
```

```
import CIE.Student;
import java.util.scanner;

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

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

public void printMarks() {
System.out.println("SEE Marks: ");
for (int i = 0; i < 5; i++) {

System.out.println("Course " + (i + 1) + ": " + seeMarks[i]);
}
}
}

//main program
import CIE.Student;
import CIE.Internals;
import SEE.External;
import java.util.Scanner;
```

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

System.out.print("Enter Student Name: ");
String name = sc.nextLine();
System.out.print("Enter USN: ");
String usn = sc.nextLine();
System.out.print("Enter Semester: ");
int sem = sc.nextInt();

int[] internalMarks = new int[5];
System.out.println("Enter Internal Marks for 5 subjects: ");
for (int i = 0; i < 5; i++) {
internalMarks[i] = sc.nextInt();

}

int[] seeMarks = new int[5];
System.out.println("Enter SEE Marks for 5 subjects: ");
for (int i = 0; i < 5; i++) {
seeMarks[i] = sc.nextInt();
}
```

```
Student student = new Student(usn, name, sem);
Internals internals = new Internals(internalMarks);
External external = new External(name,usn,sem,seeMarks);

System.out.println("Student Details: ");
student.printDetails();
System.out.println("Internal Marks: ");
internals.printMarks();
System.out.println("SEE Marks: ");
external.printMarks();

System.out.println("Final Marks (50% Internals + 50% SEE): ");
for (int i = 0; i < 5; i++) {
int finalMark = (internalMarks[i]) + (seeMarks[i] / 2);
System.out.println("Course " + (i + 1) + ": " + finalMark);
}
}
}
```

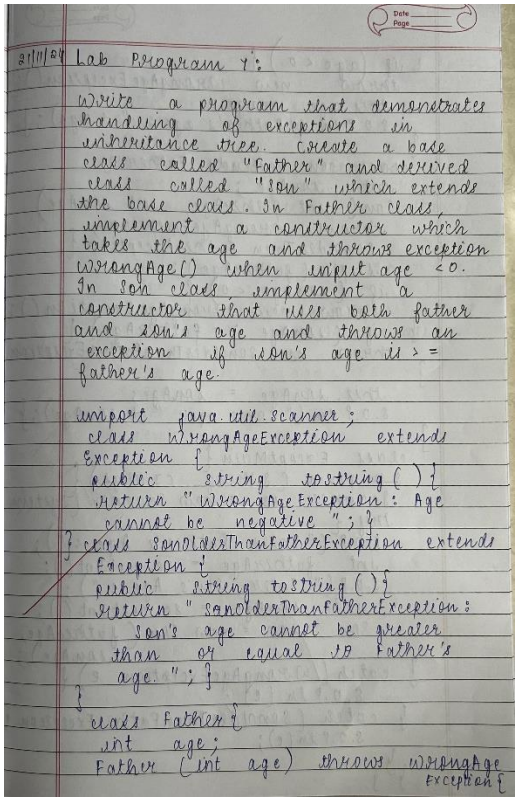
OUTPUT:

```
Enter Student Name: Rachel
Enter USN: 1BM23CS004
Enter Semester: 3
Enter Internal Marks for 5 subjects:
45
30
42
38
49
Enter SEE Marks for 5 subjects:
97
77
89
90
100
Student Details:
Student Name: Rachel
USN: 1BM23CS004
Semester: 3
Internal Marks:
Internal Marks:
Course 1: 45
Course 2: 30
Course 3: 42
Course 4: 38
Course 5: 49
SEE Marks:
SEE Marks:
Course 1: 97
Course 2: 77
Course 3: 89
Course 4: 90
Course 5: 100
Final Marks (50% Internals + 50% SEE):
Course 1: 93
Course 2: 68
Course 3: 86
Course 4: 83
Course 5: 99
```

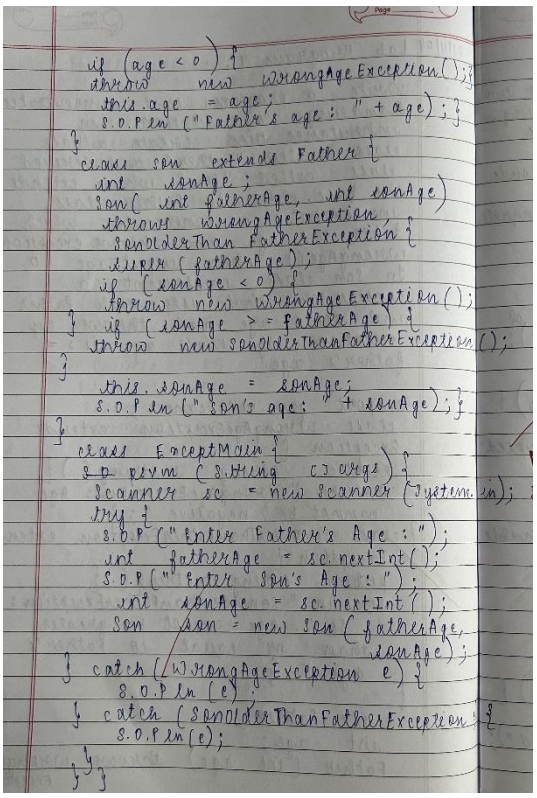
Program 7:

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

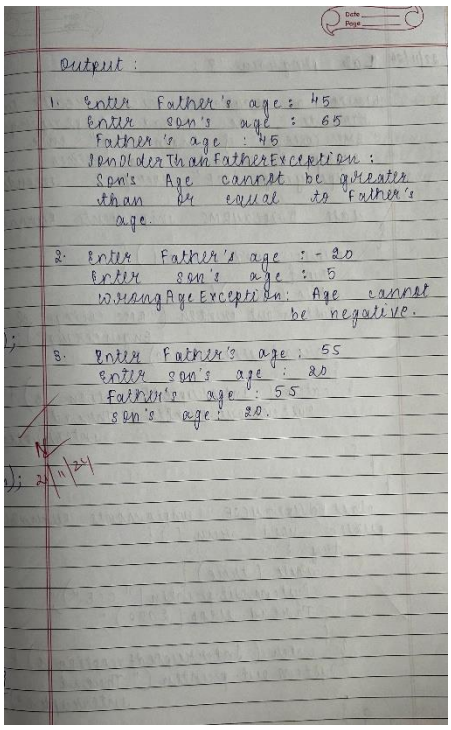
ALGORITHM:



```
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 in ("Father's age: " + age);}
}
class Son extends Father{
int sonAge;
Son(int fatherAge, int sonAge)
throws WrongAgeException,
SonOlderThanFatherException{
super(fatherAge);
if (sonAge < 0){
throw new WrongAgeException();
} if (sonAge >= fatherAge){
throw new SonOlderThanFatherException();
}
this.sonAge = sonAge;
S.O.P in ("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 in (e);
} catch (SonOlderThanFatherExcepton e){
S.O.P in (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.
```

CODE:

```
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 WrongAgeException {

if (age < 0) {

throw new WrongAgeException();

}

this.age = age;

System.out.println("Father's age: " + age);

}

}

class Son extends Father {

int sonAge;
```

```
Son(int fatherAge, int sonAge) throws WrongAgeException, SonOlderThanFatherException
{
super(fatherAge);
if (sonAge < 0) {
throw new WrongAgeException();
}
if (sonAge >= fatherAge) {
throw new SonOlderThanFatherException();
}
this.sonAge = sonAge;
System.out.println("Son's age: " + sonAge);
}
}
class ExceptionMain {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
try {
System.out.println("Enter Father's Age:");
int fatherAge = sc.nextInt();
System.out.println("Enter Son's Age:");
int sonAge = sc.nextInt();
Son son = new Son(fatherAge, sonAge);
} catch (WrongAgeException e) {
System.out.println(e);
} catch (SonOlderThanFatherException e) {
System.out.println(e);
```

}

}

}

OUTPUT:

```
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.
```

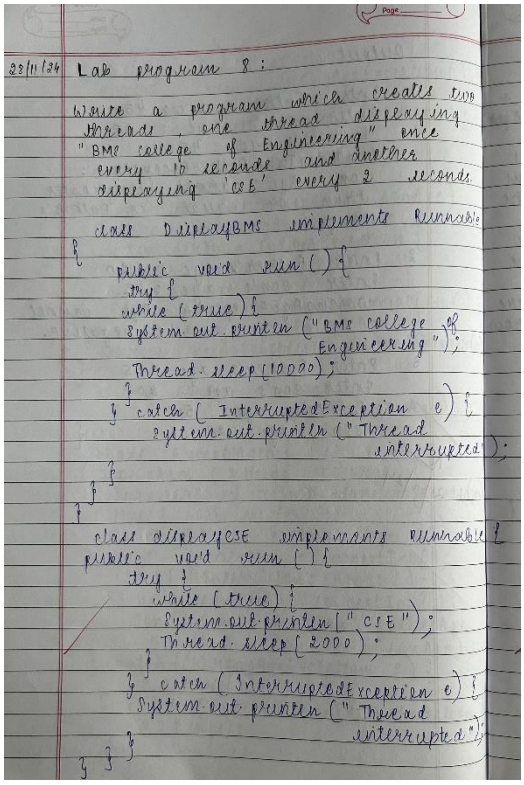
```
Enter Father's Age:
-20
Enter Son's Age:
5
WrongAgeException: Age cannot be negative.
```

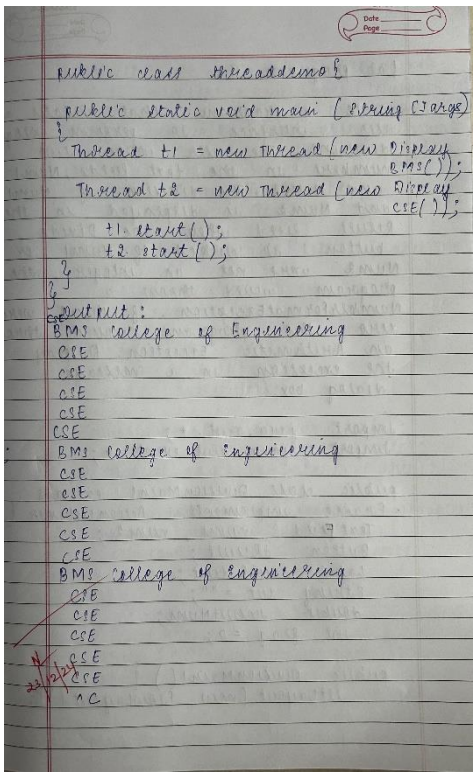
```
Enter Father's Age:
55
Enter Son's Age:
20
Father's age: 55
Son's age: 20
```

Program 8:

Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

ALGORITHM:





CODE:

```
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 DisplayBMS());
Thread t2 = new Thread(new DisplayCSE());
t1.start();
t2.start();
}
}
```

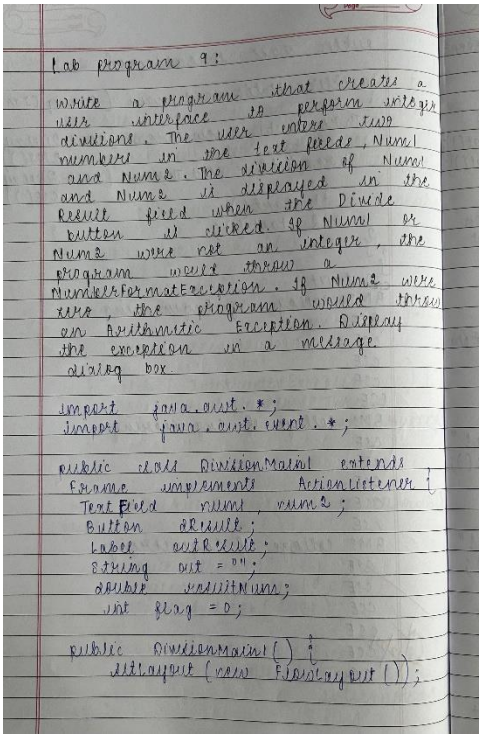
OUTPUT:

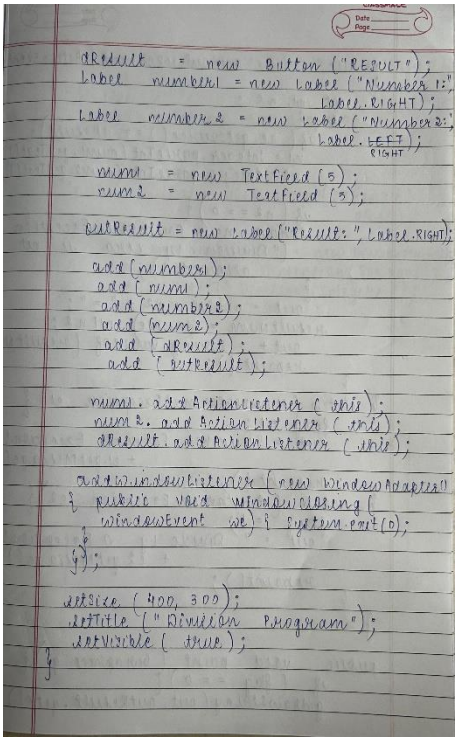
```
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
^C
```

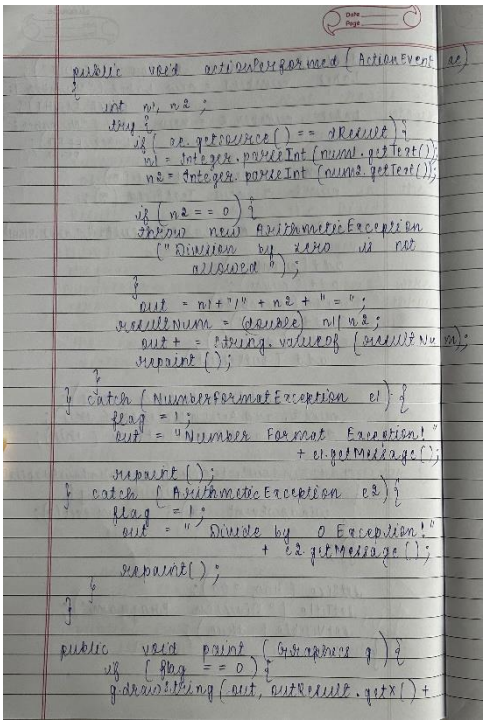
Program 9:

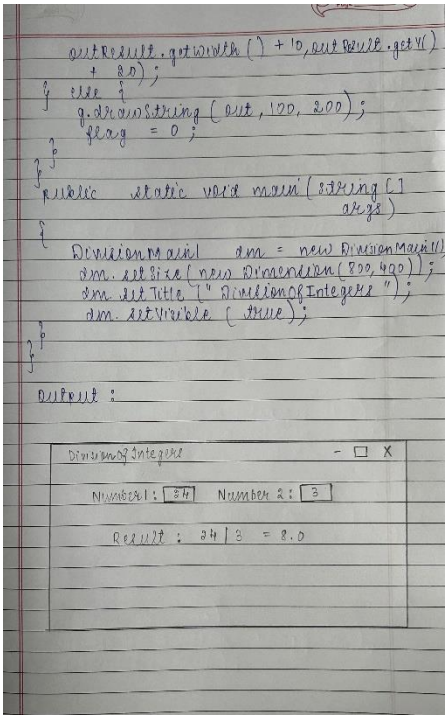
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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

ALGORITHM:









CODE:

```
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);
        }
    });

    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() + 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);
}
}
```

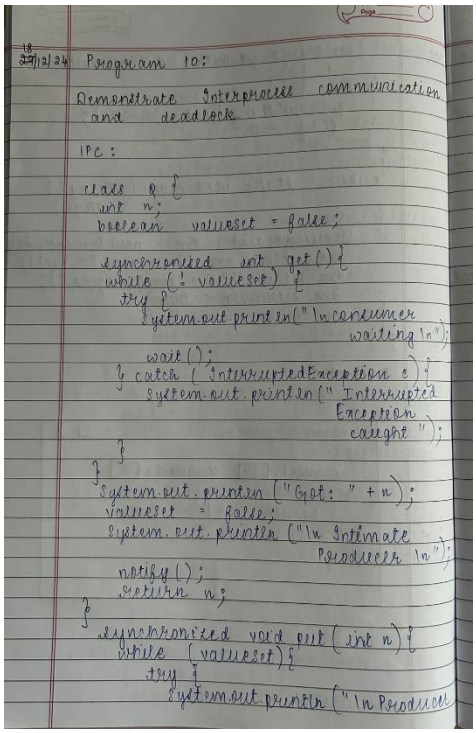
OUTPUT:

Program 10:

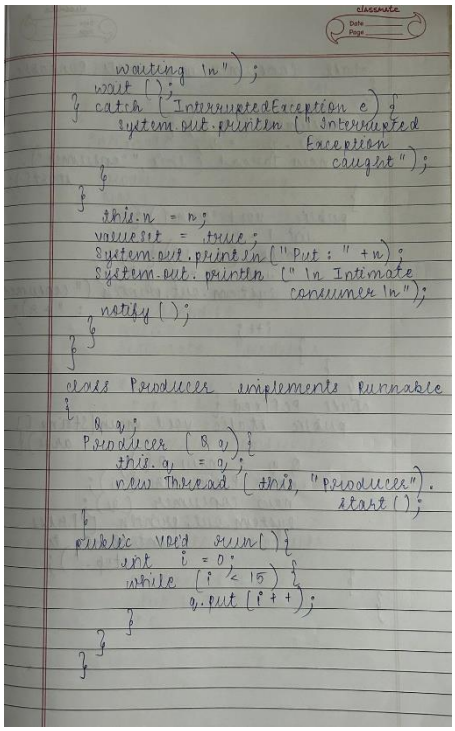
Demonstrate Inter process Communication and deadlock.

IPC:

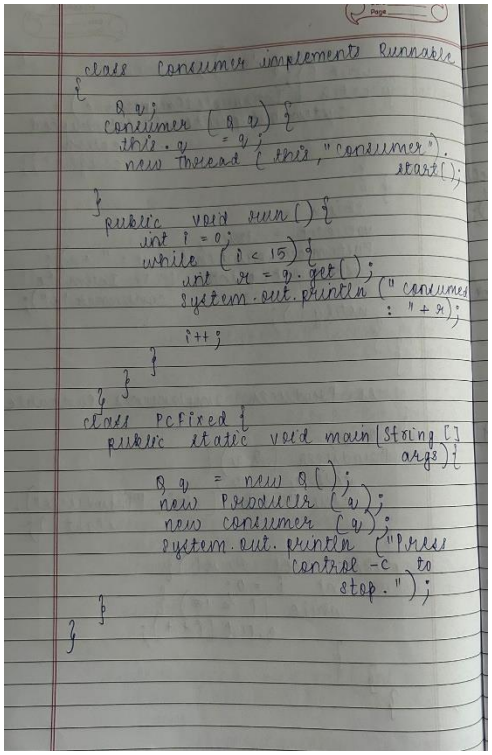
ALGORITHM:



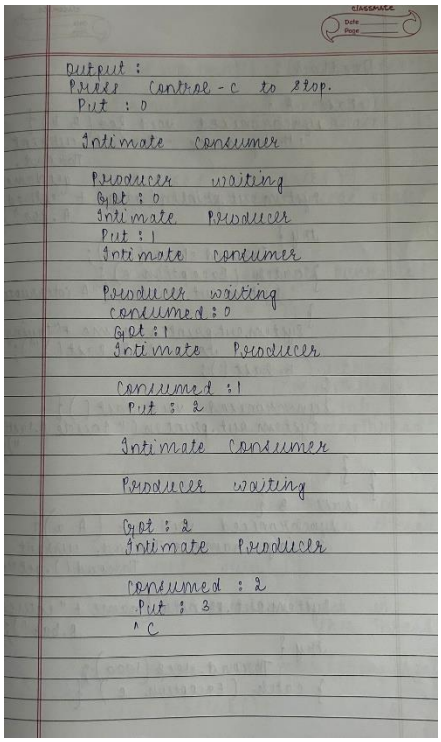
```
Program 10:

Demonstrate Interprocess communication and deadlock

IPC:

class Q {
  int n;
  boolean valueSet = false;

  synchronized int get() {
  while (!valueSet) {
    try {
      System.out.println("In consumer waiting \n");
      wait();
    } catch (InterruptedException e) {
      System.out.println("Interrupted Exception 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("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
```

CODE:

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

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

    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting\n");
```

```
            wait();
        } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
        }
      }
      this.n = n;
      valueSet = true;
      System.out.println("Put: " + n);
      System.out.println("\nIntimate Consumer\n");
      notify();
   }
}

class Producer implements Runnable {
   Q q;

   Producer(Q q) {
      this.q = q;
      new Thread(this, "Producer").start();
   }

   public void run() {
      int i = 0;
      while (i < 15) {
         q.put(i++);
```

```
        }
    }
}

class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

    public void run() {
        int i = 0;
        while (i < 15) {
            int r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

class PCFixed {
    public static void main(String args[]) {
        Q q = new Q();
```

```
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

OUTPUT:

```
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
```

```
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
```

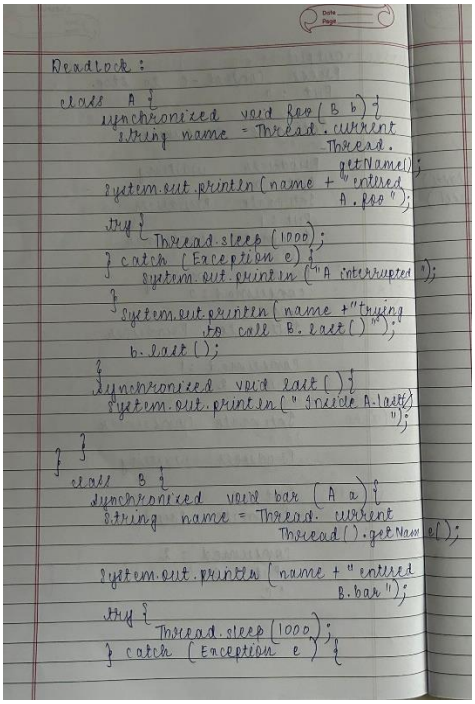
```
Got: 6

Intimate Producer

Consumed: 6
Put: 7

Intimate Consumer


Producer waiting

Got: 7

Intimate Producer

Put: 8

Intimate Consumer


Producer waiting

Consumed: 7
Got: 8

Intimate Producer

Consumed: 8
Put: 9

Intimate Consumer


Producer waiting
```

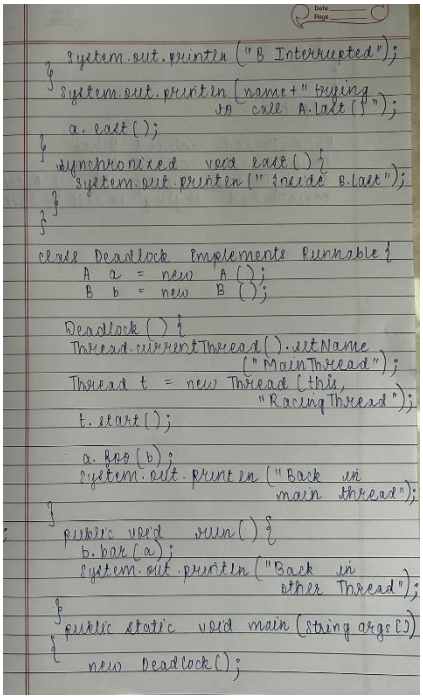
```
Got: 9

Intimate Producer

Consumed: 9
Put: 10

Intimate Consumer


Producer waiting

Got: 10

Intimate Producer

Consumed: 10
Put: 11

Intimate Consumer


Producer waiting

Got: 11

Intimate Producer

Consumed: 11
Put: 12

Intimate Consumer


Producer waiting
```

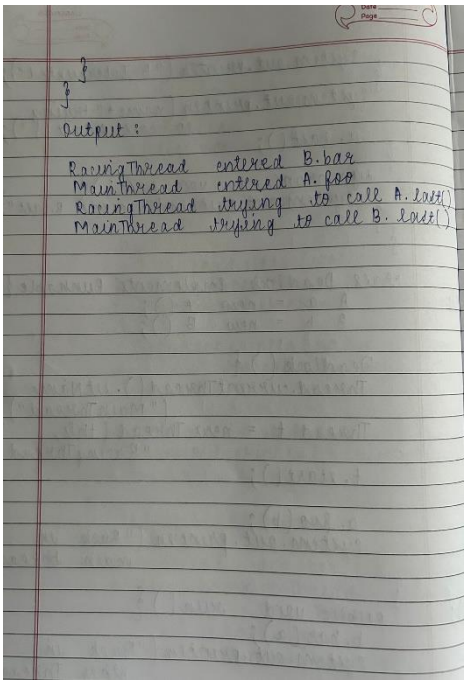
```
Got: 12

Intimate Producer

Put: 13

Intimate Consumer


Producer waiting

Consumed: 12
Got: 13

Intimate Producer

Consumed: 13
Put: 14

Intimate Consumer

Got: 14

Intimate Producer

Consumed: 14
```

DEADLOCK:

ALGORITHM:







CODE:

```
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()
```