# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

**LAB REPORT**
**on**

# Object Oriented Java Programming

# (23CS3PCOOJ)

***Submitted by***

**Ayush Ranjan (1BM23CS058)**

*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 to 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 (23CS3PCOOJ)" carried out by **Ayush Ranjan (1BM23CS058),** who is a 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. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

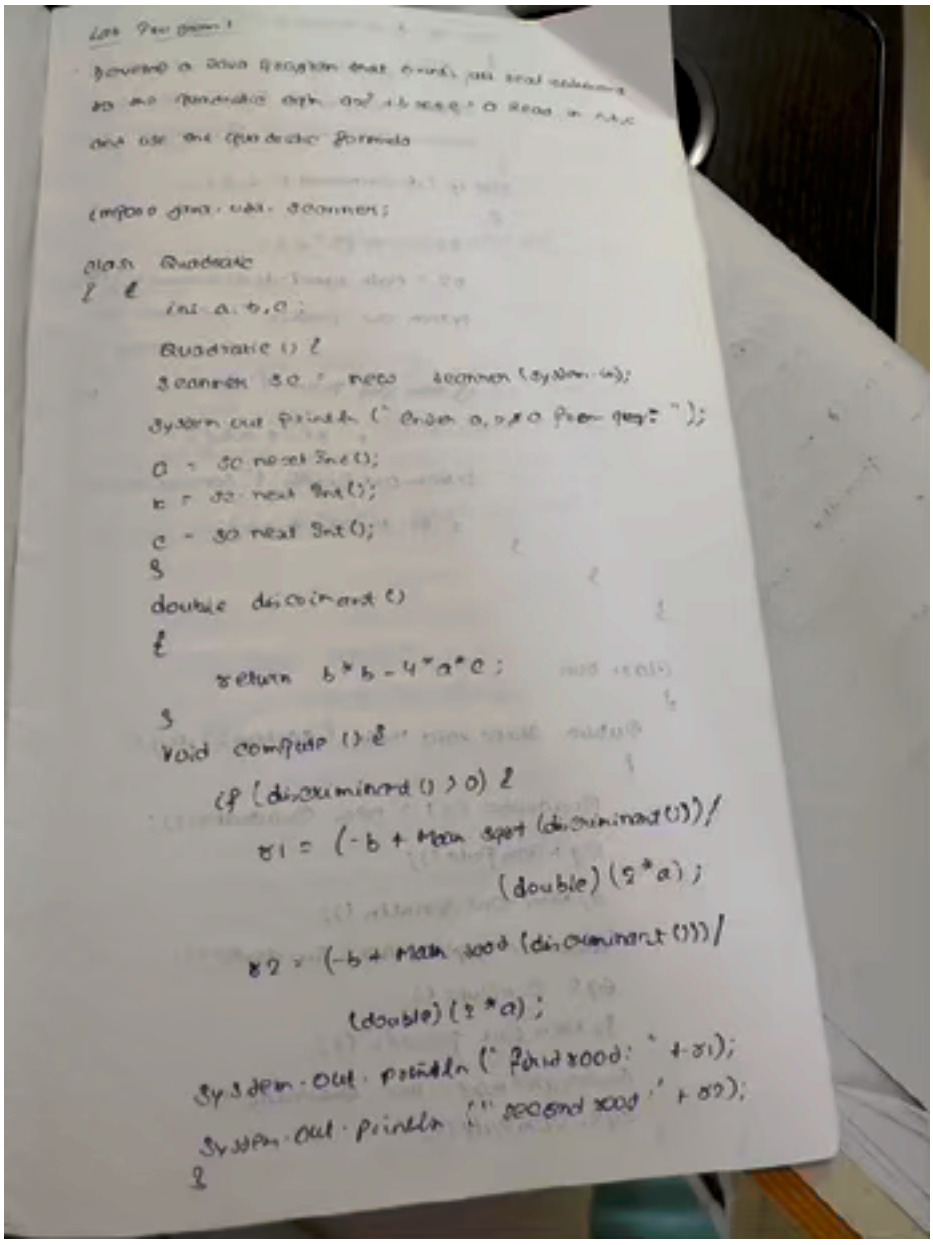
| | |
|---|---|
| Dr. Prasad Gr<br>Assistant Professor<br>Department of CSE, BMSCE | Dr. Jyothi S Nayak<br>Professor & HOD<br>Department of CSE, BMSCE |

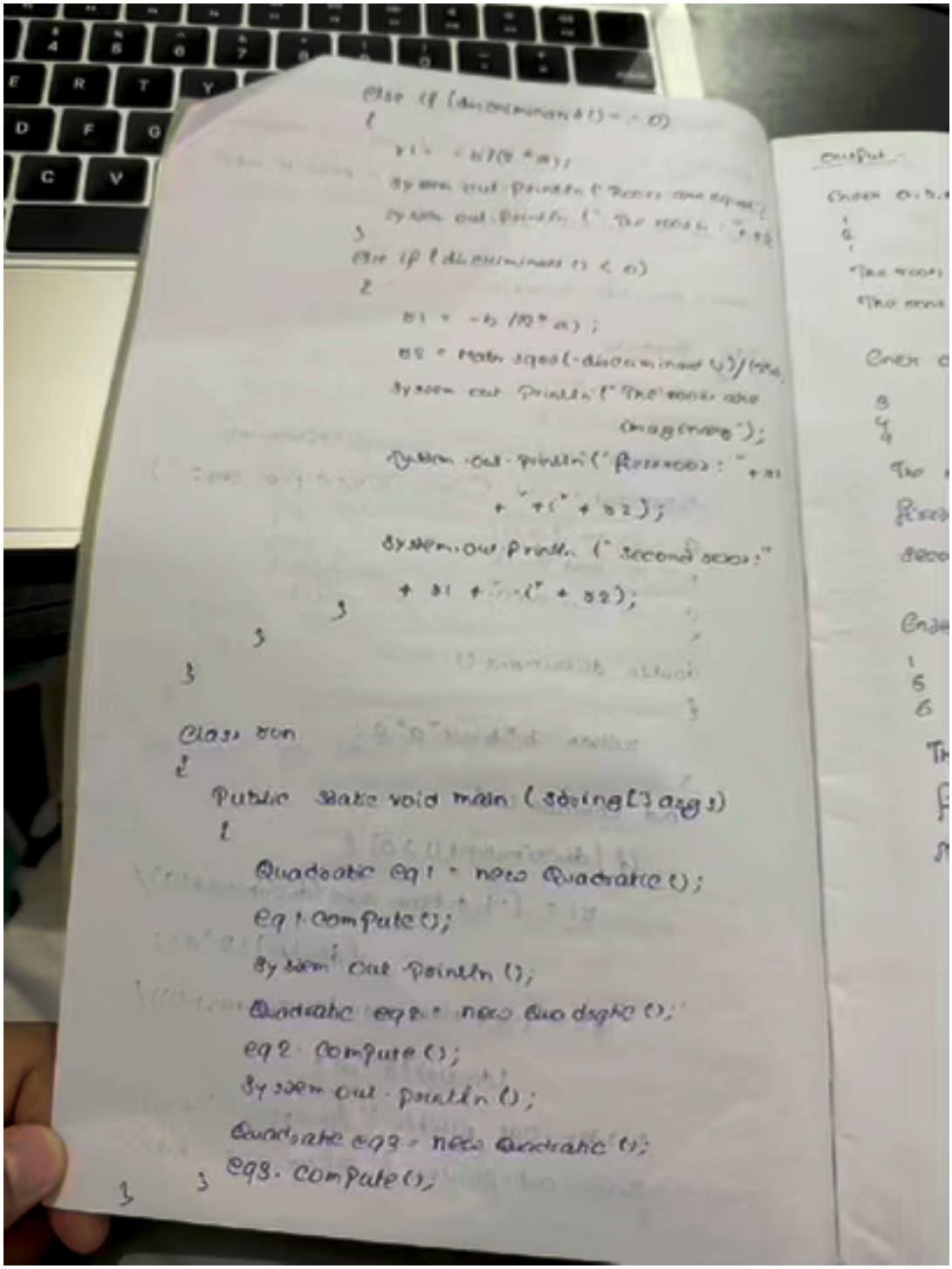
# Index

Github Link:
https://github.com/AyushRanjan-58/java_lab

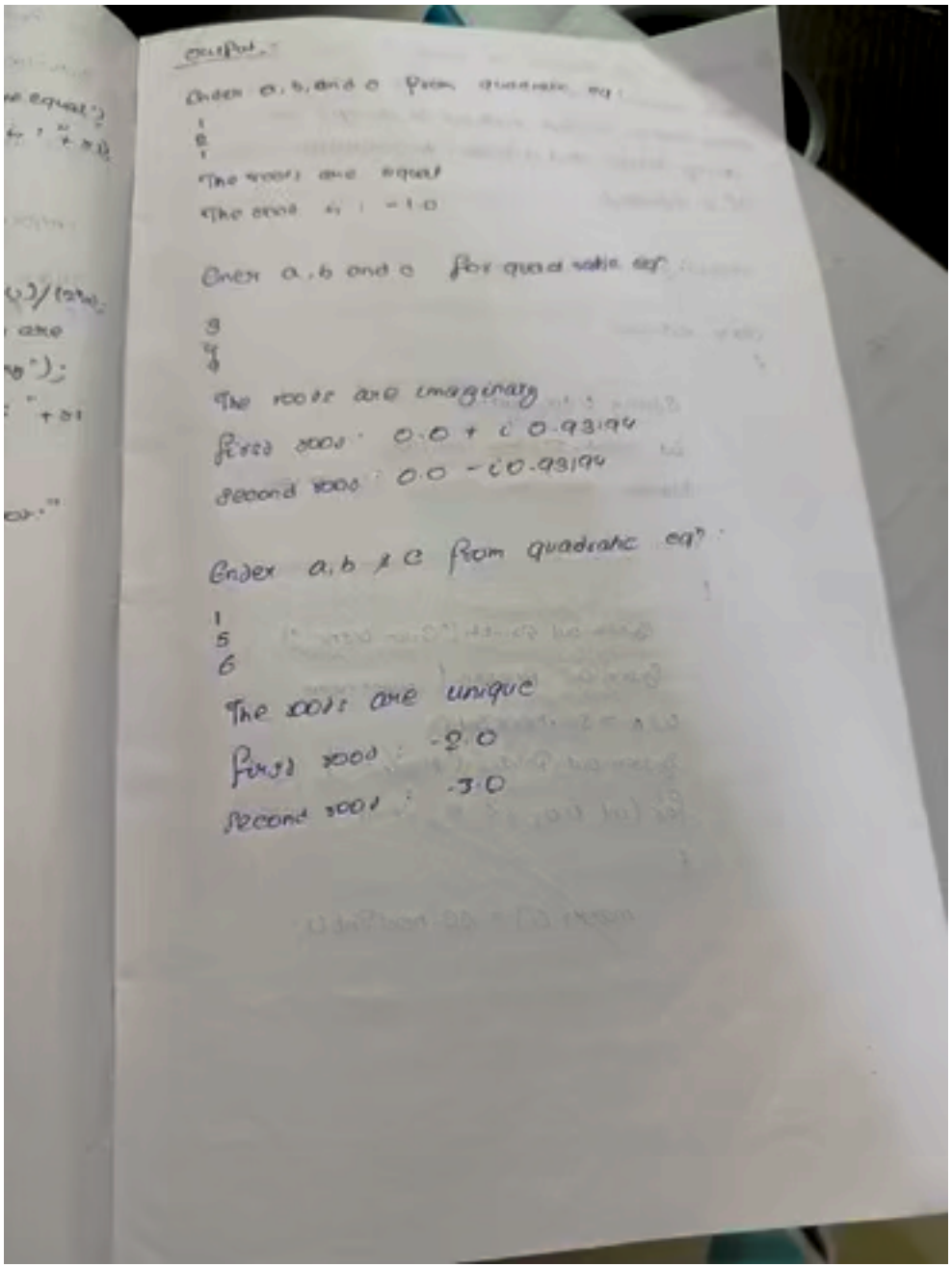
**Program 1**
Implement Quadratic Equation

**Algorithm:**





```
Else if (discriminant() == 0)
{
    r1 = -b/(2*a);
    System.out.println("Roots are equal");
    System.out.println("The root is: " + r1);
}
Else if (discriminant() < 0)
{
    r1 = -b/(2*a);
    r2 = Math.sqrt(-discriminant())/(2*a);
    System.out.println("The roots are
                        imaginary");
    System.out.println("Root1 is: " + r1
                        + "+(" + r2);
    System.out.println("Second root:"
                        + r1 + "-(" + r2);
}
}
}

Class Run
{
    Public static void main (String[] args)
    {
        Quadratic eq1 = new Quadratic();
        eq1.compute();
        System.out.println();
        Quadratic eq2 = new Quadratic();
        eq2.compute();
        System.out.println();
        Quadratic eq3 = new Quadratic();
        eq3.compute();
    }
}
```



output:

Enter a, b, and c from quadratic eq:
1
8
1
The roots are equal
The root is : -1.0

Enter a, b and c for quadratic eq:
3
4
8
The roots are imaginary
First root: 0.0 + i 0.9394
Second root: 0.0 - i 0.9394

Enter a, b & c from quadratic eq:
1
5
6
The roots are unique
First root : -2.0
Second root : -3.0

**Code:**

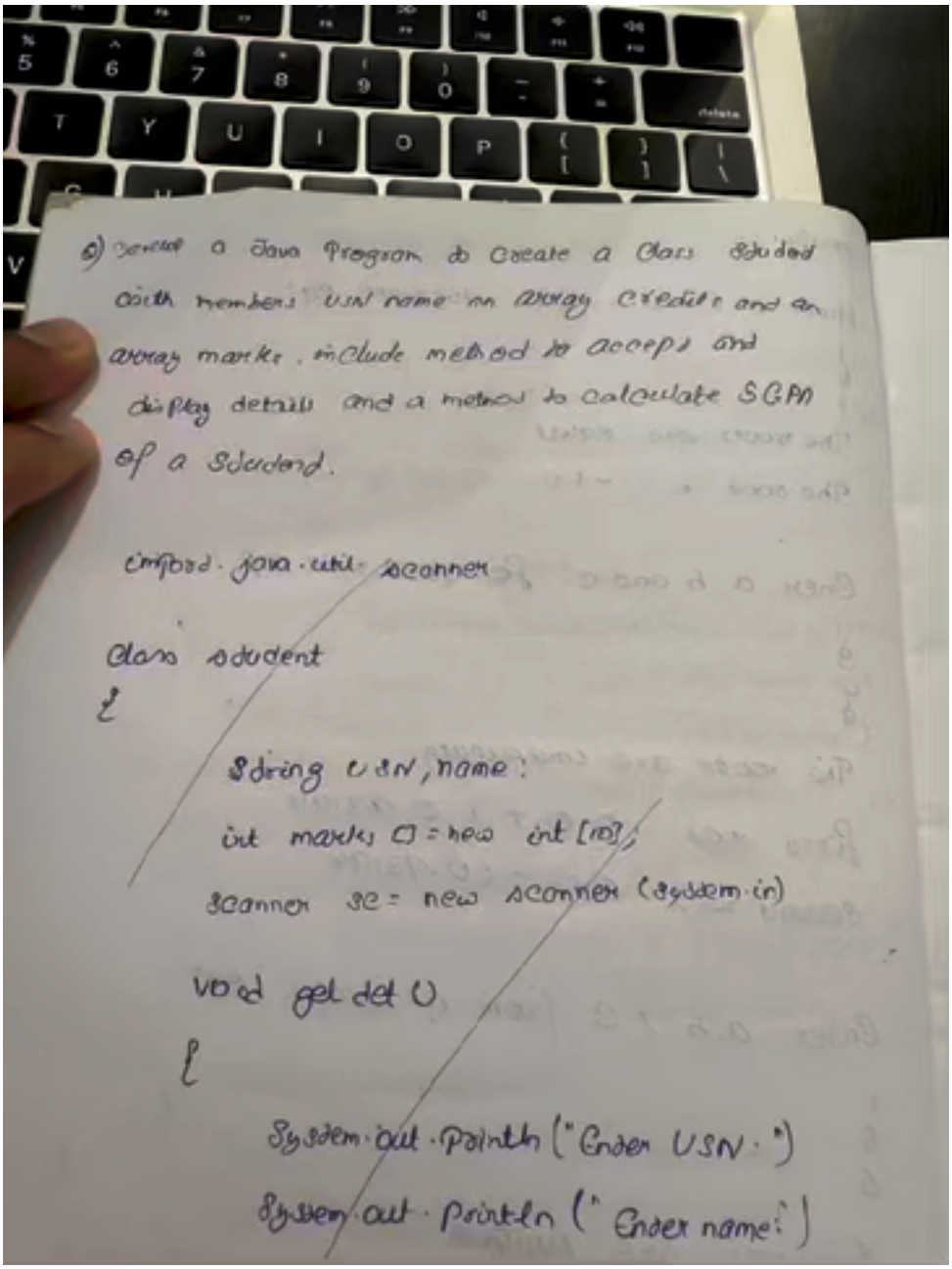
```
import java.util.Scanner;

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

      System.out.print("Enter coefficient a: ");
      double a = scanner.nextDouble();

      System.out.print("Enter coefficient b: ");
      double b = scanner.nextDouble();

      System.out.print("Enter coefficient c: ");
      double c = scanner.nextDouble();

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

      if (discriminant > 0) {
         double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
         double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
         System.out.println("The roots are real and different:");
         System.out.println("Root 1: " + root1);
         System.out.println("Root 2: " + root2);
      } else if (discriminant == 0) {
         double root = -b / (2 * a);
         System.out.println("The roots are real and the same:");
         System.out.println("Root: " + root);
      } else {
         System.out.println("The roots are complex:");
         double realPart = -b / (2 * a);
         double imaginaryPart = Math.sqrt(-discriminant) / (2 * a);
         System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
         System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
                  System.out.print(“Ayush Ranjan 1BM23CS058");
      }

      scanner.close();
   }
}
```

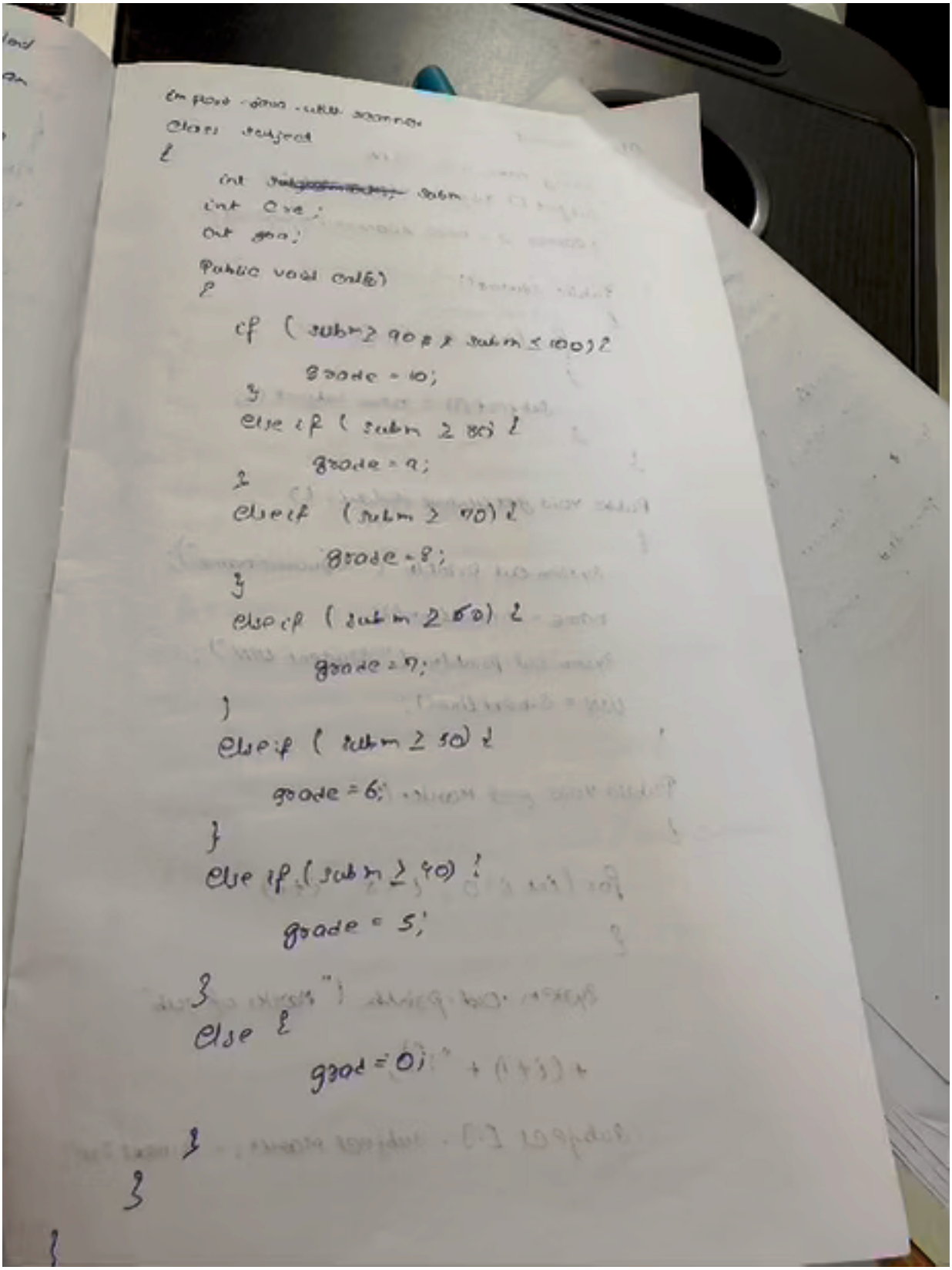
Output :

```
Enter a, b, c:
1
2
1
Equal roots. r1 = r2 = -1.0
Name: Ayush.r USN: 1BM23CS058

Enter a, b, c:
1
-7
12
r1 = 6.0 r2 = 2.0
Name: Ayush.r USN: 1BM23CS058

Enter a, b, c:
1
2
5
Roots are imaginary
Name: Ayush.r USN: 1BM23CS058
```

**Program 2 :**
**Implement SGPA Calculator**

**Algorithm :**





```
Employee emp = new Employee();
class subject
{
    int subm
    int sub;
    int grade;

    Public void call()
    {
        if ( subm >= 90 && subm <= 100){
            grade = 10;
        }
        else if ( subm >= 80) {
            grade = 9;
        }
        else if ( subm >= 70) {
            grade = 8;
        }
        else if ( subm >= 60) {
            grade = 7;
        }
        else if ( subm >= 50) {
            grade = 6;
        }
        else if ( subm >= 40) {
            grade = 5;
        }
        else {
            grad = 0;
        }
    }
}
```

**Code :**

```
import java.util.*;

class Stud_details {
   int marks[] = new int[8];
   int credit[] = new int[8];
   String usn, name;
   Scanner sc = new Scanner(System.in);

   void getdetails() {
      System.out.println("Enter the usn and name:");
      usn = sc.next();
      name = sc.next();
      System.out.println("Enter the marks for 8 subjects:");
      for (int i = 0; i < 8; i++) {
         marks[i] = sc.nextInt();
      }
      System.out.println("Enter the credits for 8 subjects:");
      for (int i = 0; i < 8; i++) {
         credit[i] = sc.nextInt();
      }
   }

   void display() {
      System.out.println("\nStudent Details:");
      System.out.println("USN: " + usn);
      System.out.println("Name: " + name);
      for (int i = 0; i < 8; i++) {
         System.out.println("Marks of Subject " + (i + 1) + ": " + marks[i]);
      }
      System.out.println("SGPA: " + calculateSGPA());
   }

   double getgradepoint(int mark) {
      if (mark >= 90) return 10.0;
      else if (mark >= 80) return 9.0;
      else if (mark >= 70) return 8.0;
```

```
        else if (mark >= 60) return 7.0;
        else if (mark >= 50) return 6.0;
        else if (mark >= 40) return 5.0;
        else return 0.0;
    }

    double calculateSGPA() {
        int totalcredits = 0;
        double gradepoint = 0;
        for (int i = 0; i < 8; i++) {
            totalcredits += credit[i];
        }
        for (int i = 0; i < 8; i++) {
            gradepoint += getgradepoint(marks[i]) * credit[i];
        }
        return (gradepoint / totalcredits);
    }
}

public class Student {
    public static void main(String args[]) {
        Stud_details s1[] = new Stud_details[3];
        for (int j = 0; j < 3; j++) {
            s1[j] = new Stud_details();
        }
        for (int j = 0; j < 3; j++) {
            System.out.println("\nEnter details of student " + (j + 1));
            s1[j].getdetails();
        }
        for (int j = 0; j < 3; j++) {
            s1[j].display();
        }
        System.out.println("\nName: Ayush Ranjan USN: 1BM23CS058");
    }
}
```

Output :

Enter details of student 1
Enter the usn and name:
1BM23CS001
John
Enter the marks for 8 subjects:
90
80
70
60
50
40
30
20
Enter the credits for 8 subjects:
4
3
3
3
2
3
2
3
Enter details of student 2
Enter the usn and name:
1BM23CS002
Alice
Enter the marks for 8 subjects:
80
85
75
65
55
45
35
25
Enter the credits for 8 subjects:

4
3
3
3
2
3
2
3
Enter details of student 3
Enter the usn and name:
1BM23CS003
Bob
Enter the marks for 8 subjects:
85
90
95
80
70
60
50
40
Enter the credits for 8 subjects:
4
3
3
3
2
3
2
3
Student Details:
USN: 1BM23CS001
Name: John
Marks of Subject 1: 90
Marks of Subject 2: 80
Marks of Subject 3: 70
Marks of Subject 4: 60
Marks of Subject 5: 50
Marks of Subject 6: 40
Marks of Subject 7: 30
Marks of Subject 8: 20
SGPA: 5.555555555555555

Student Details:

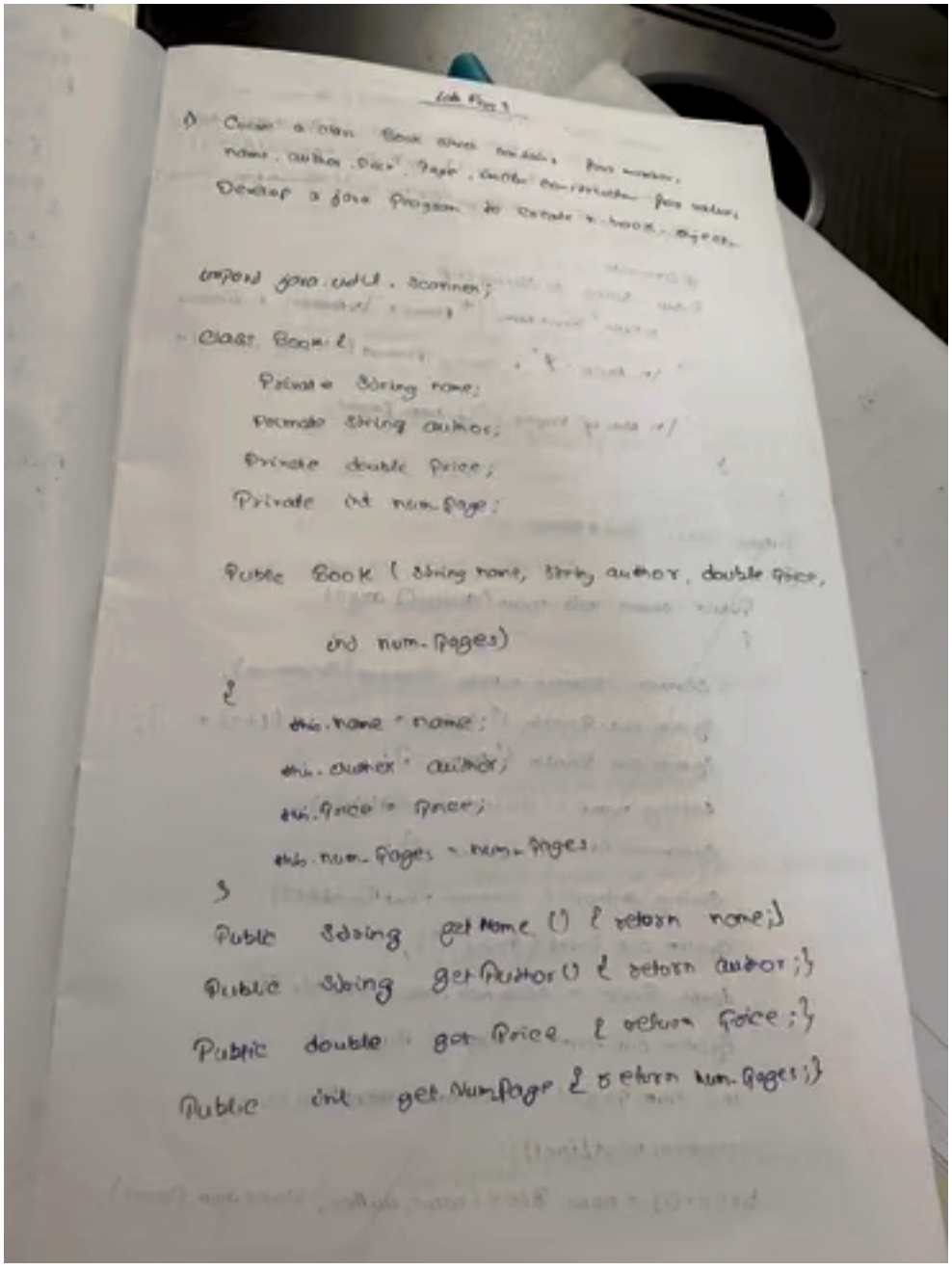
USN: 1BM23CS002
Name: Alice
Marks of Subject 1: 80
Marks of Subject 2: 85
Marks of Subject 3: 75
Marks of Subject 4: 65
Marks of Subject 5: 55
Marks of Subject 6: 45
Marks of Subject 7: 35
Marks of Subject 8: 25
SGPA: 5.5

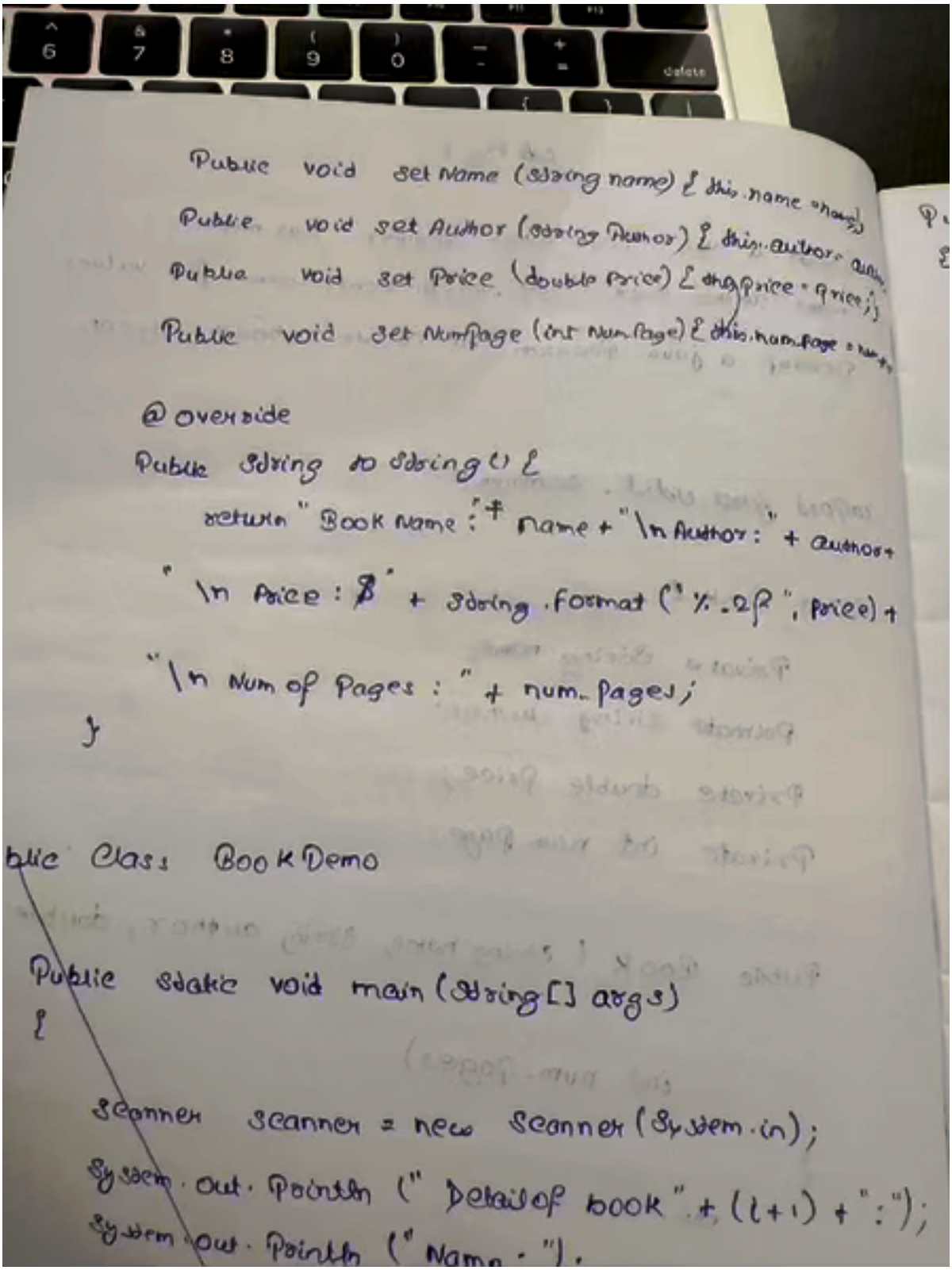
Student Details:
USN: 1BM23CS003
Name: Bob
Marks of Subject 1: 85
Marks of Subject 2: 90
Marks of Subject 3: 95
Marks of Subject 4: 80
Marks of Subject 5: 70
Marks of Subject 6: 60
Marks of Subject 7: 50
Marks of Subject 8: 40
SGPA: 7.722222222222222

Name: Ayush Ranjan USN: 1BM23CS058

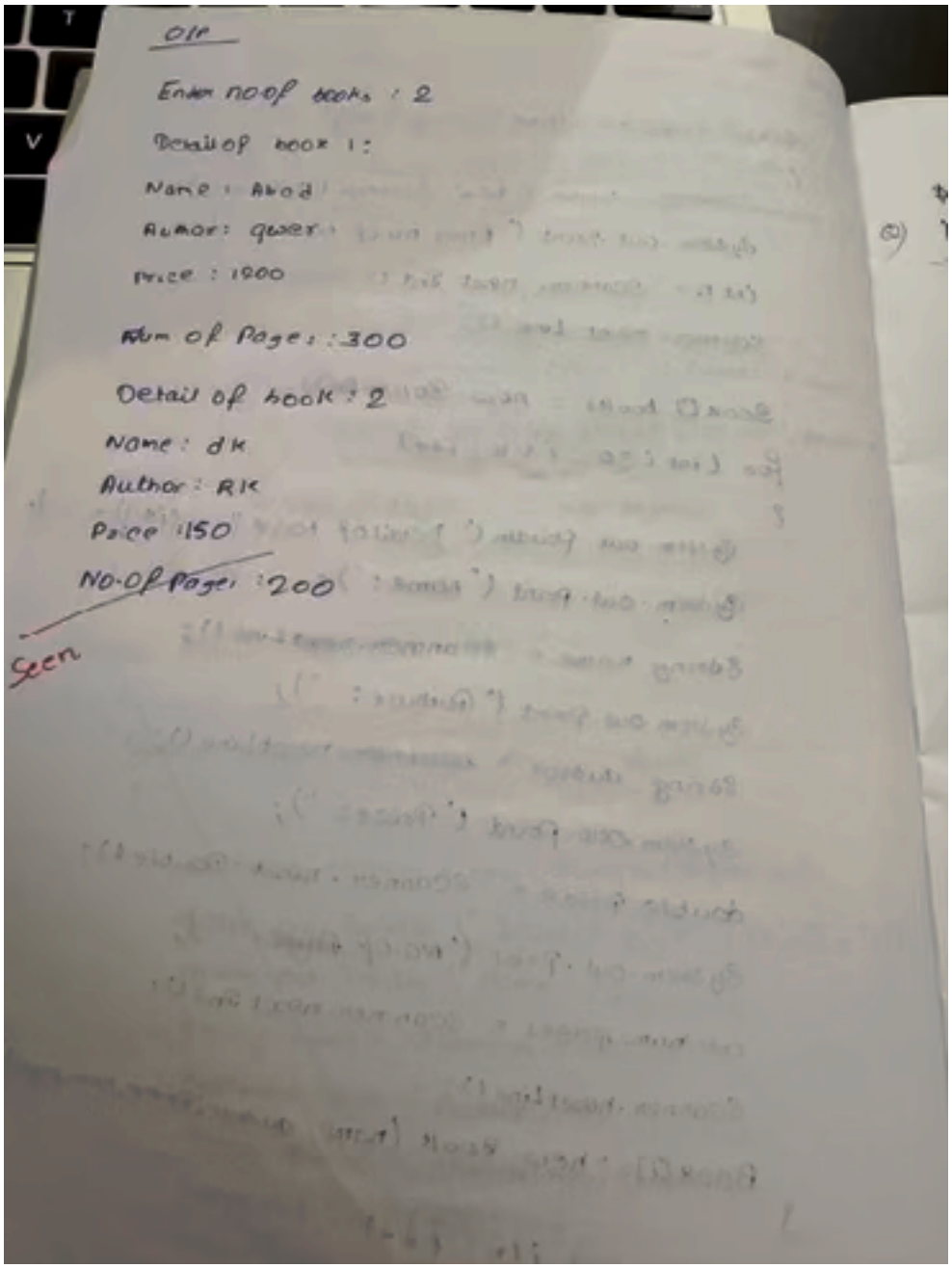
Program 3 :
Create Objects for Books

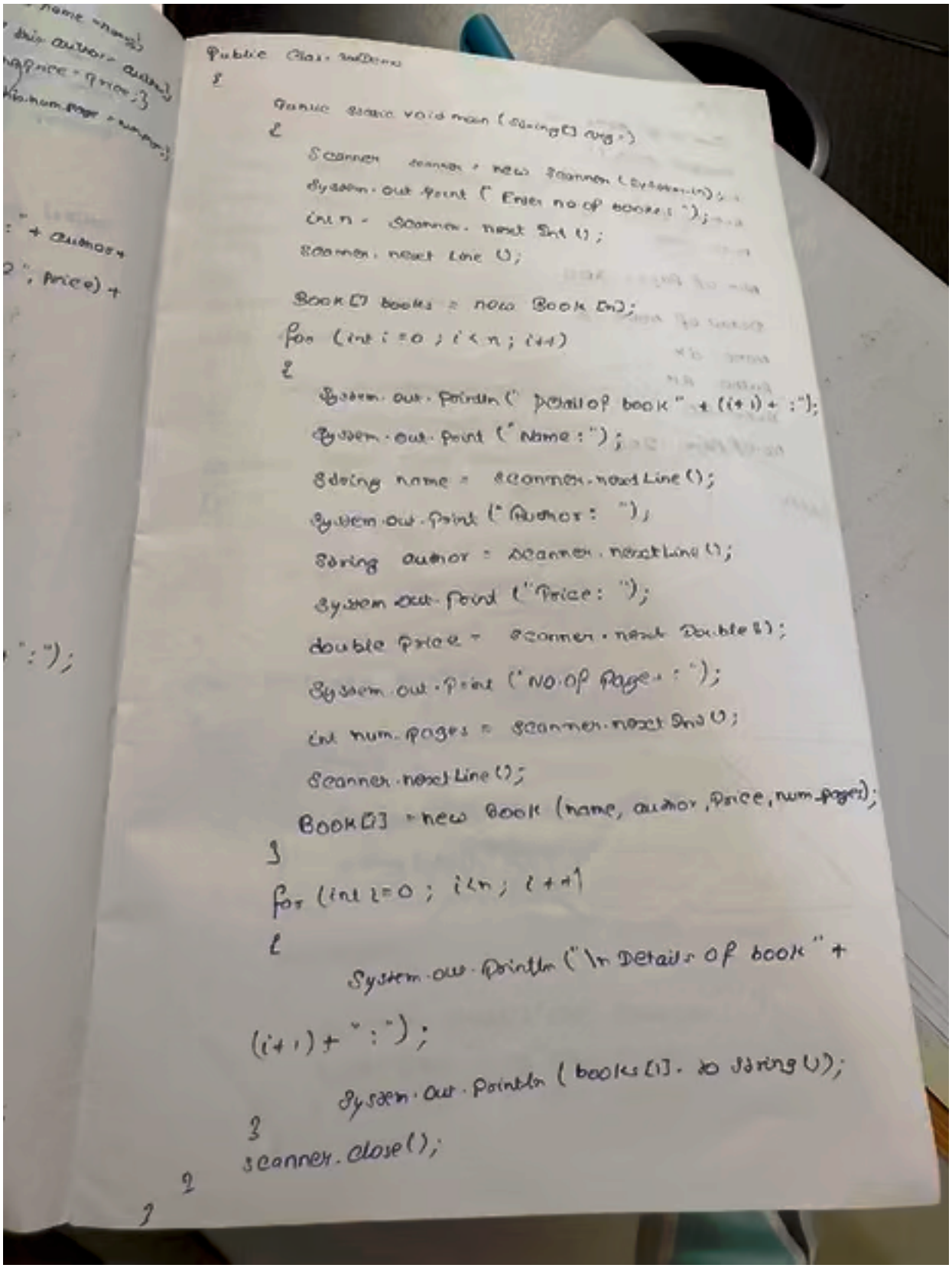
Algorithm:





```
Public void setName (String name) { this.name = name;
Public void setAuthor (String Author) { this.author = author;
Public void setPrice (double Price) { this.price = price;
Public void setNumPage (int NumPage) { this.numPage = numPage;

@Override
Public String toString() {
    return "Book Name :" + name + "\n Author :" + author +
    "\n Price : $" + String.format ("%.2f", price) +
    "\n Num of Pages :" + num_pages;
}

blic Class BookDemo

Public static void main (String[] args)
{
    Scanner scanner = new Scanner (System.in);
    System.out.Println (" Detail of book" + (i+1) + ":");
    System.out.Println ("Name :").
```


O/P
Enter no of books : 2
Detail of book 1:
Name : Abcd
Author: qwer
Price : 1200
Num of Page: 300
Detail of book : 2
Name : dk
Author : RK
Price 150
NO-of Page: 200
Seen



```
Public Class ...
{
    Public static void main (String[] args)
    {
        Scanner scanner = new Scanner (System.in);
        System.out.print (" Enter no of books: ");
        int n = scanner.nextInt ();
        scanner.nextLine ();

        Book[] books = new Book [n];
        for (int i=0 ; i<n ; i++)
        {
            System.out.println (" Detail of book " + (i+1) + ":");
            System.out.print (" Name : ");
            String name = scanner.nextLine ();
            System.out.print (" Author : ");
            String author = scanner.nextLine ();
            System.out.print (" Price : ");
            double price = scanner.nextDouble ();
            System.out.print (" No of Pages : ");
            int num_pages = scanner.nextInt ();
            scanner.nextLine ();
            Book[i] = new Book (name, author, price, num_pages);
        }
        for (int i=0 ; i<n ; i++)
        {
            System.out.println (" \n Details of book " +
(i+1) + ":");
            System.out.println ( books[i].toString ());
        }
        scanner.close ();
    }
}
```

Code :

```
import java.util.*;

class Book {
   String name;
   String author;
   int price;
   int num_pages;


   Book(String name, String author, int price, int num_pages) {
      this.name = name;
      this.author = author;
      this.price = price;
      this.num_pages = num_pages;
   }


   public String toString() {
      return "Name: " + name + ", Author: " + author + ", Price: " + price + ",
No of Pages: " + num_pages;
   }
}

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

      System.out.println("Enter number of books:");
      int n = sc.nextInt();
      sc.nextLine();

      Book[] book = new Book[n];


      System.out.println("Enter name, author, price and number of pages for
each book respectively:");
      for (int i = 0; i < n; i++) {
         System.out.println("Enter details of book " + (i + 1) + ":");


         System.out.print("Name: ");
         String name = sc.nextLine();
```

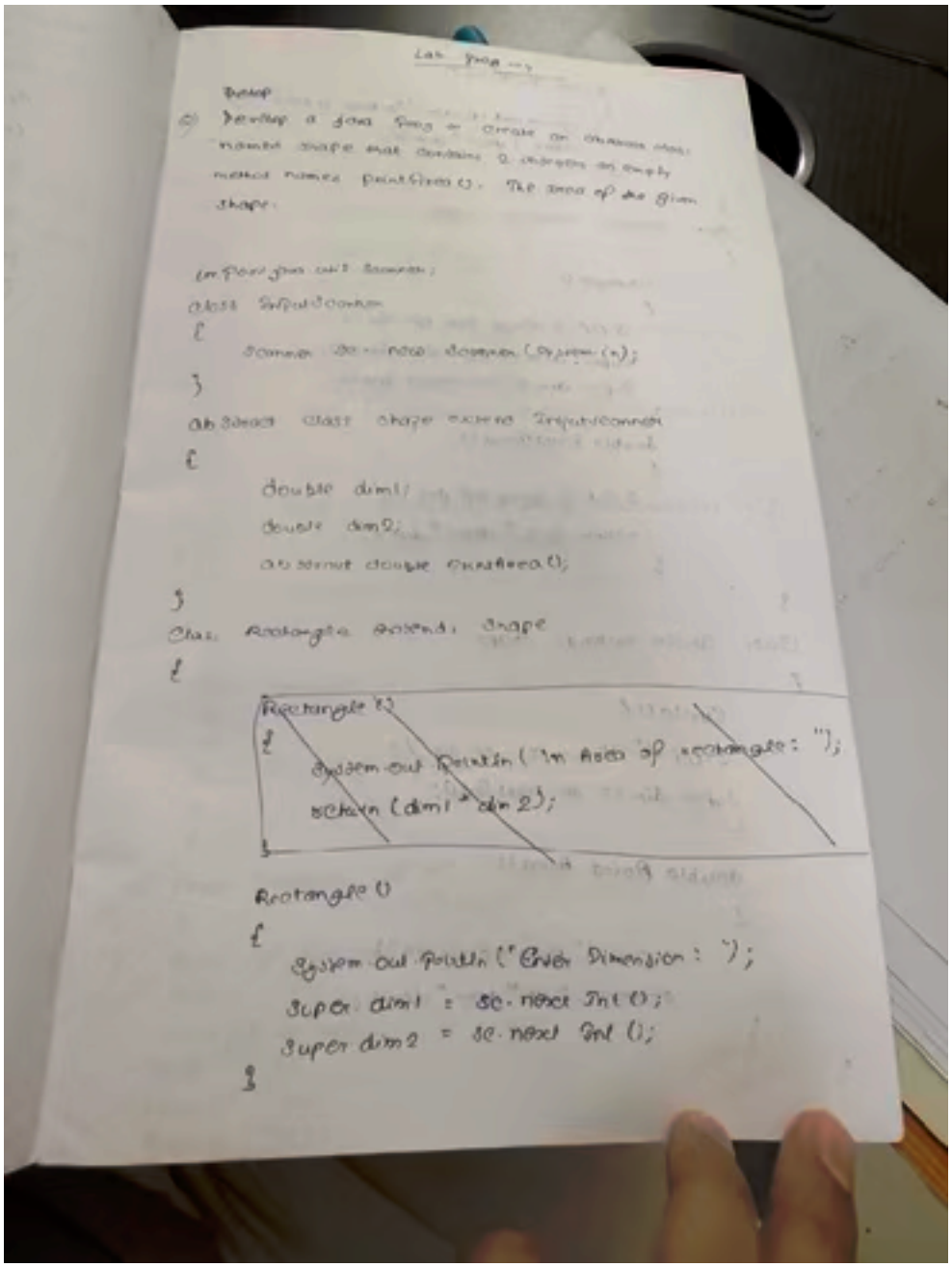
```
            System.out.print("Author: ");
            String author = sc.nextLine();

            System.out.print("Price: ");
            int price = sc.nextInt();

            System.out.print("Number of Pages: ");
            int num_pages = sc.nextInt();
            sc.nextLine();


            book[i] = new Book(name, author, price, num_pages);
        }


        for (int i = 0; i < n; i++) {
            System.out.println("\nBook " + (i + 1) + ":");
            System.out.println(book[i].toString());
        }

        sc.close();
    }
}
```

o/p

```
Enter number of books:
2
Enter name, author, price and number of pages for each book respectively:
Enter details of book 1:
Name: Java Programming
Author: James Gosling
Price: 500
Number of Pages: 300
Enter details of book 2:
Name: Data Structures
Author: Robert Lafore
Price: 400
Number of Pages: 450
Book 1:
Name: Java Programming, Author: James Gosling, Price: 500, No of Pages:
300
Book 2:
Name: Data Structures, Author: Robert Lafore, Price: 400, No of Pages: 450
```

Program 4 : Implement Abstract Class

Algorithm:



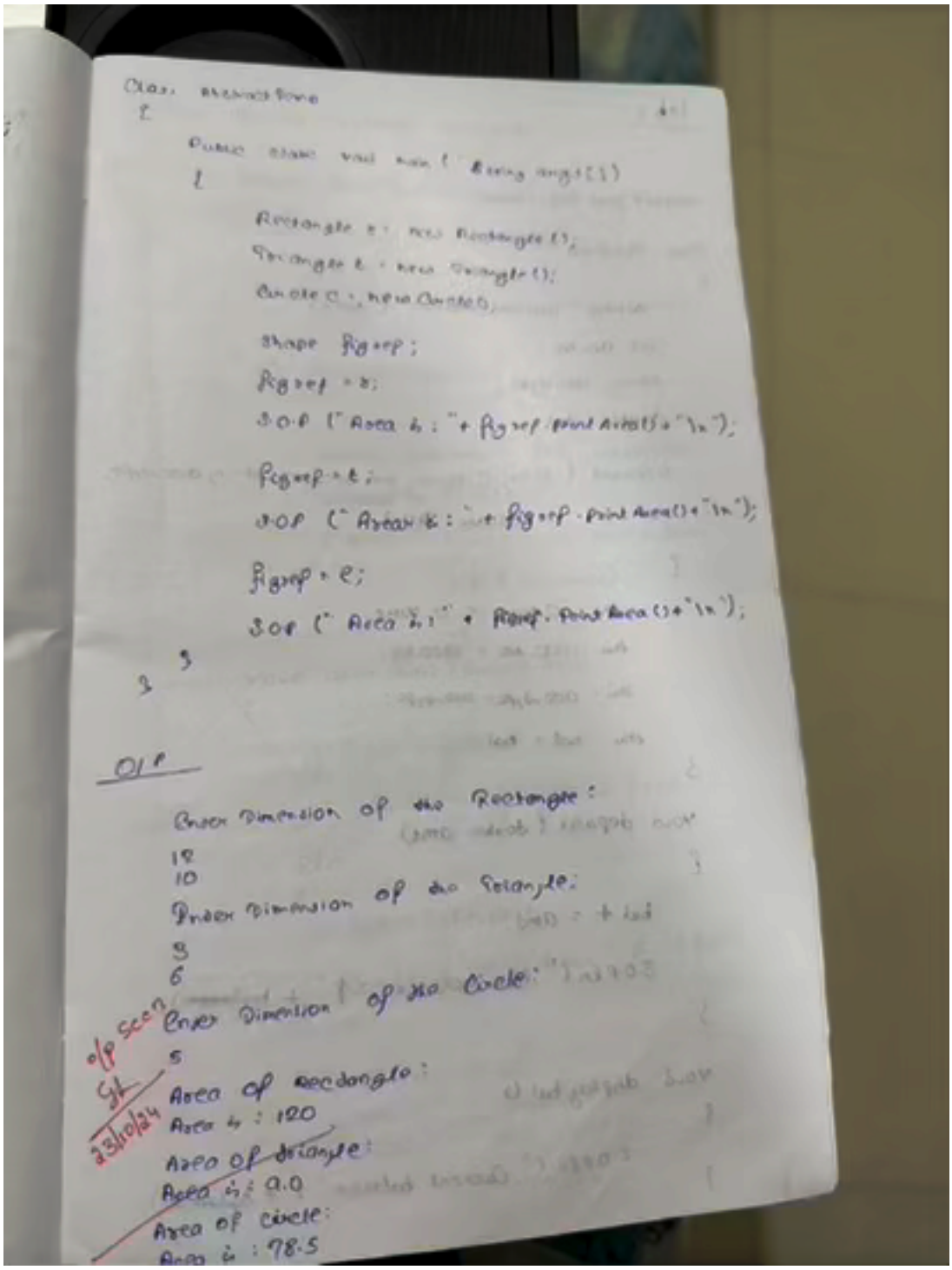
```
        double PrintArea()
        {
            System.out.println("In Area of Rect");
            return (dim1 * dim2);
        }
}
Class Triangle extends Shape
{
        Triangle()
        {
            S.O.P("Enter Dim of Tri")
            super.dim1 = sc.nextInt();
            super.dim2 = sc.nextInt();
        }
        double PrintArea()
        {
            S.O.P("Area of tri");
            return 0.5 * dim1 * dim2;
        }
}
Class Circle extends Shape
{
        Circle()
        {
        S.O.P("Dimension of cir");
        super.dim1 = sc.nextInt();
        }
        double PrintArea()
        {
            S.O.P("Area of circle")
            return 3.14 * dim1 * dim1;
        }
}
```

```
Class AbstractDemo
{
    Public class
    {
        Rectangle
        Triangle
        Circle
        Shape
        Figure
        S.O.P
        Figure
        S.O.P
        Figure
        S.O.P
    }
}
```

O/P

```
Enter
12
10
Enter
3
6
Enter
5
Area
Area
Area
Area
Area
Area
```

o/p screen
23/10/24



```
Class AbstractDemo
{
    public static void main(String args[])
    {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();

        Shape figref;
        figref = r;
        System.out.println("Area is: " + figref.printArea() + "\n");

        figref = t;
        System.out.println("Area is: " + figref.printArea() + "\n");

        figref = c;
        System.out.println("Area is: " + figref.printArea() + "\n");
    }
}
```

O/P

```
Enter Dimension of the Rectangle:
12
10
Enter Dimension of the Triangle:
3
6
Enter Dimension of the Circle:
5
Area of Rectangle:
Area is: 120
Area of Triangle:
Area is: 9.0
Area of Circle:
Area is: 78.5
```

o/p seen
23/10/24

Code :

```
import java.util.*;

abstract class Shape {
   int dim1, dim2;

   Shape(int a, int b) {
      dim1 = a;
      dim2 = b;
   }

   abstract void printArea();
}

class Rectangle extends Shape {
   Rectangle(int l, int b) {
      super(l, b);
   }

   void printArea() {
      System.out.println("Area of Rectangle: " + (double)(dim1 * dim2));
   }
}

class Triangle extends Shape {
   Triangle(int b, int h) {
      super(b, h);
   }

   void printArea() {
      System.out.println("Area of Triangle: " + (double)(dim1 * dim2 / 2));
   }
}

class Circle extends Shape {
   Circle(int r) {
      super(r, 0);
   }

   void printArea() {
      System.out.println("Area of Circle: " + (Math.PI * dim1 * dim1));
   }
}
```

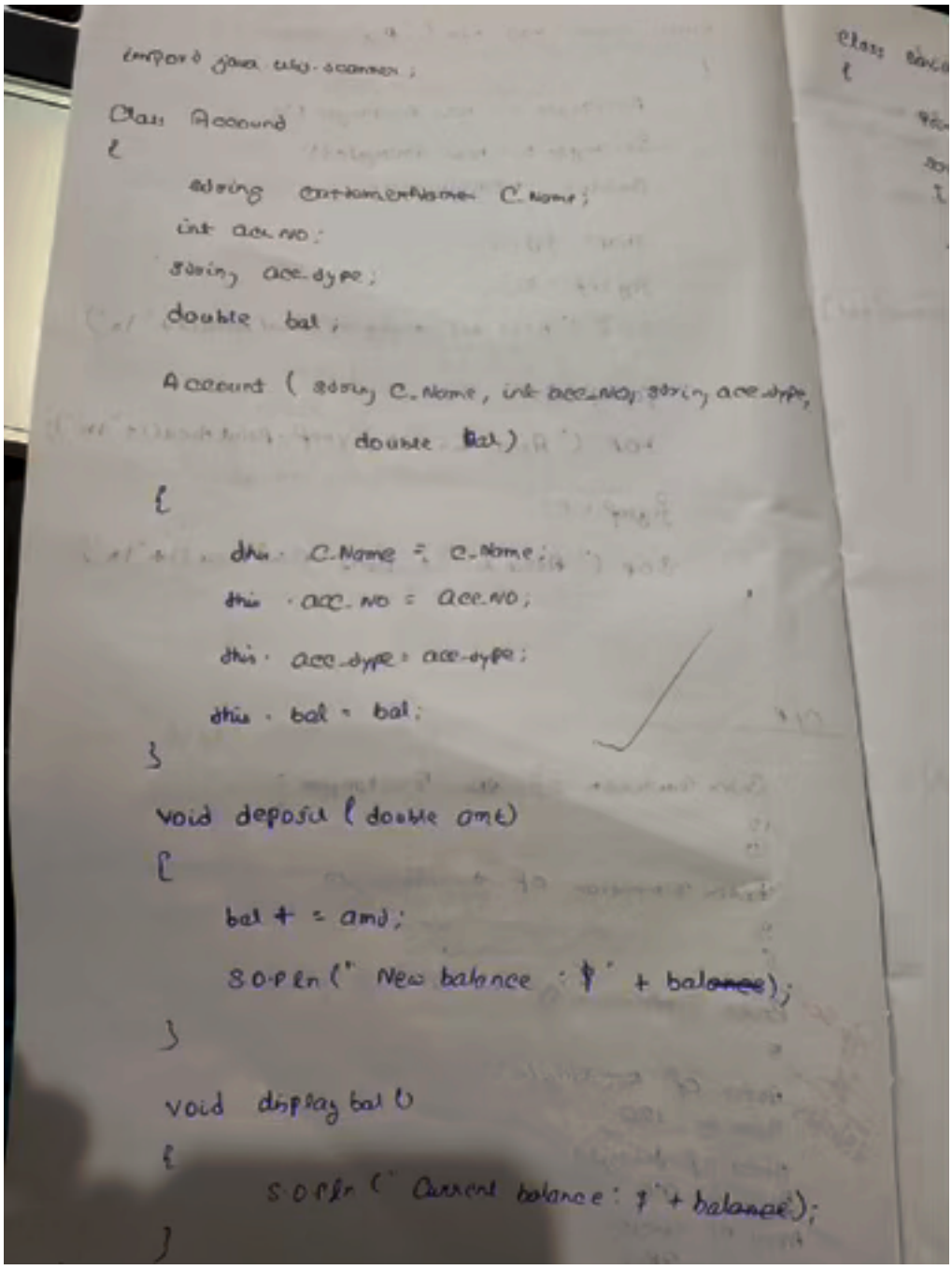
```
class ShapeArea {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.println("Enter length and width of rectangle:");
      int length = sc.nextInt();
      int breadth = sc.nextInt();

      System.out.println("Enter base and height of triangle:");
      int base = sc.nextInt();
      int height = sc.nextInt();

      System.out.println("Enter radius of circle:");
      int radius = sc.nextInt();

      Rectangle r = new Rectangle(length, breadth);
      Triangle t = new Triangle(base, height);
      Circle c = new Circle(radius);

      r.printArea();
      t.printArea();
      c.printArea():

      sc.close();
   }
}
```

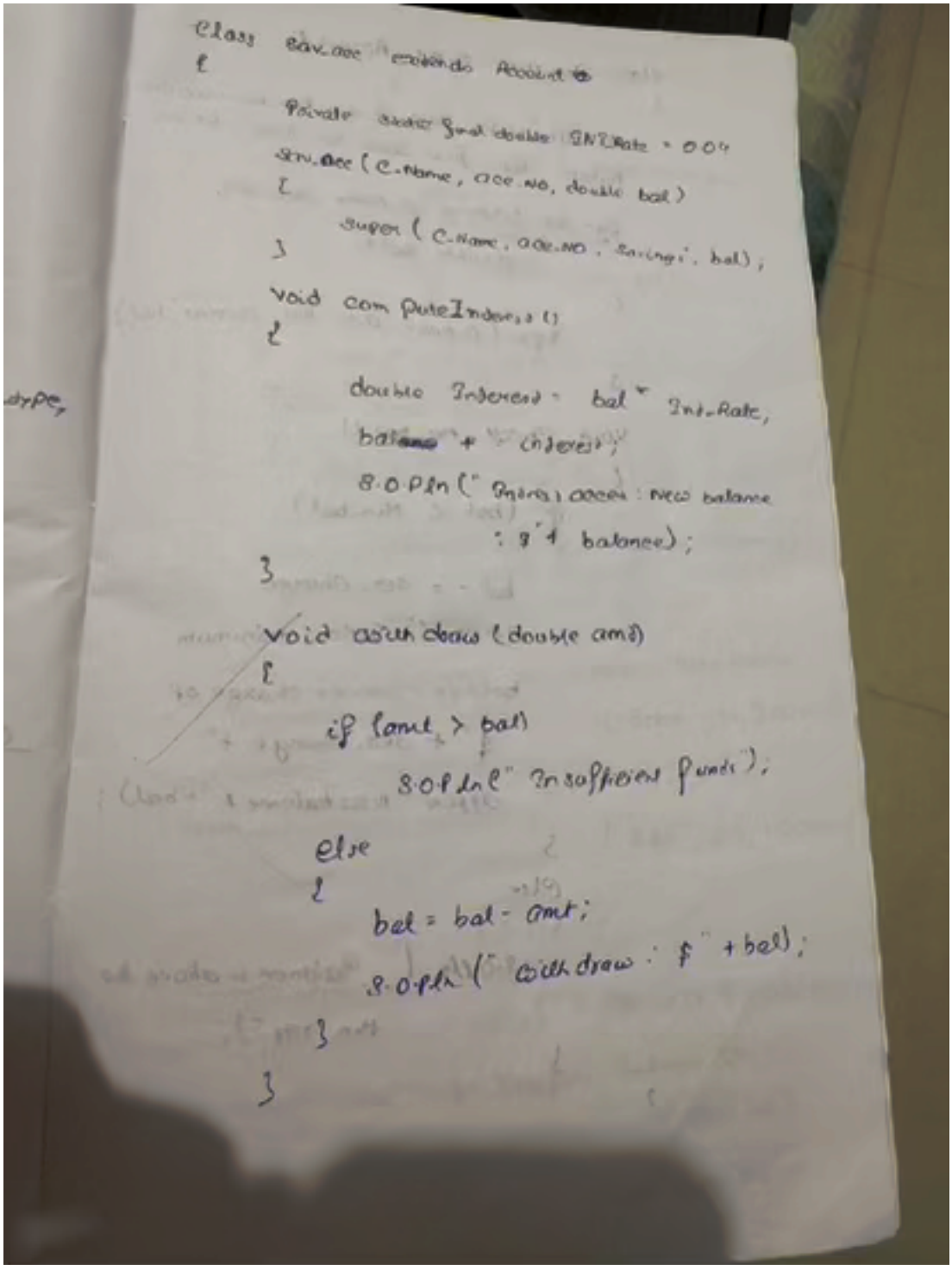
o/p

```
Enter length and width of rectangle:
10 5
Enter base and height of triangle:
8 6
Enter radius of circle:
7
Area of Rectangle: 50.0
Area of Triangle: 24.0
Area of Circle: 153.93804002589985
```

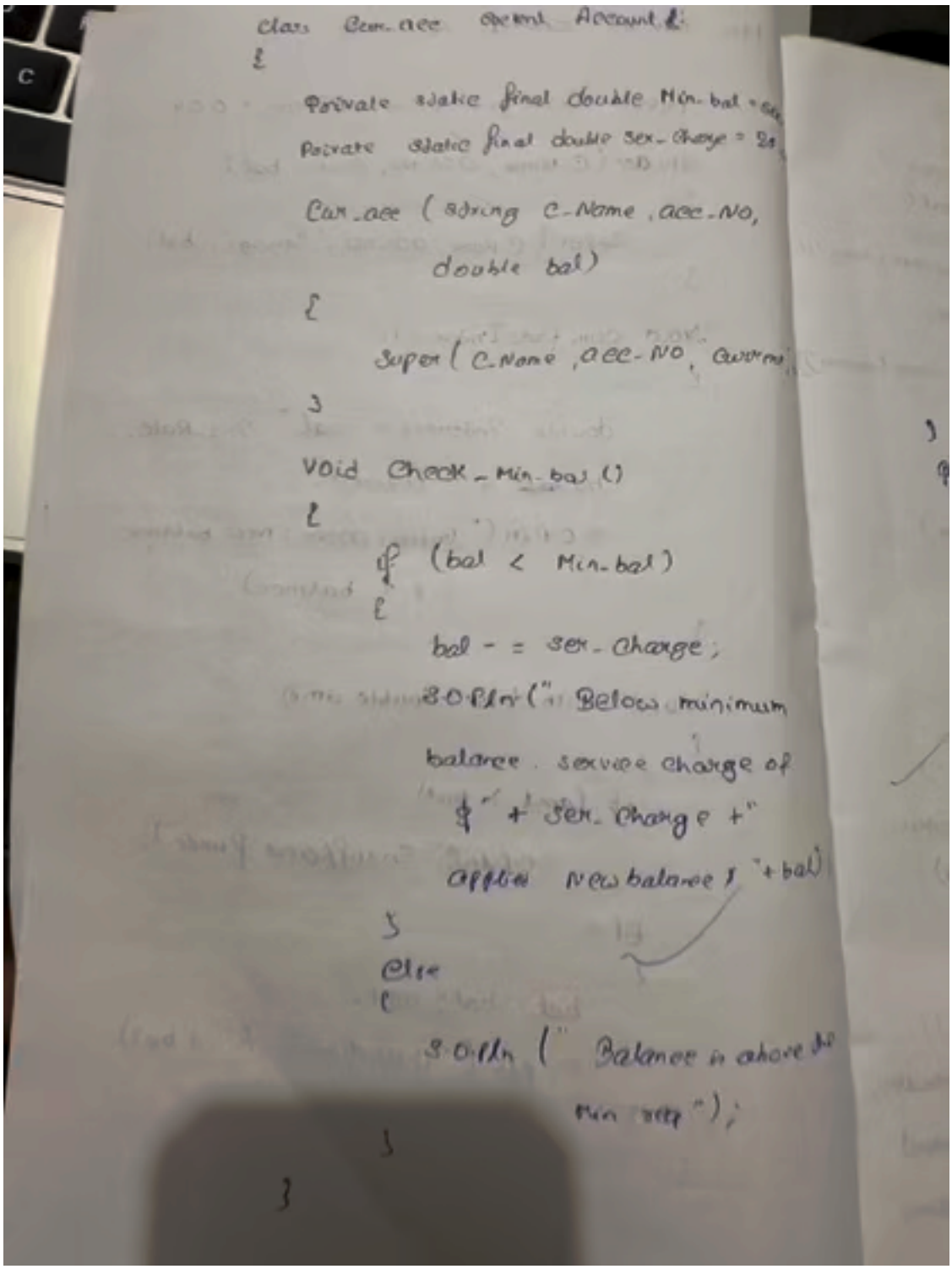
Program 5 : Bank Account Management

Algorithm :

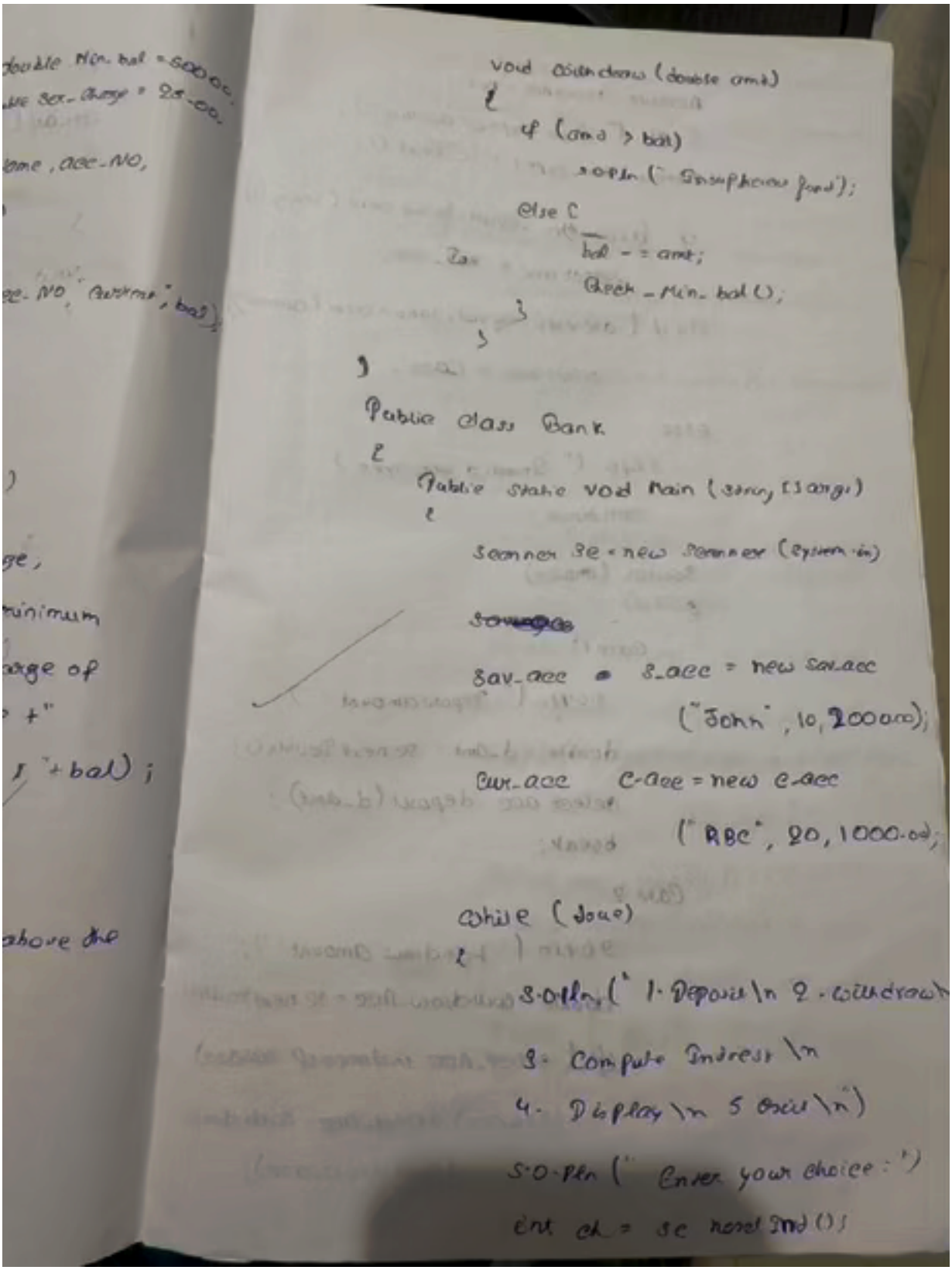




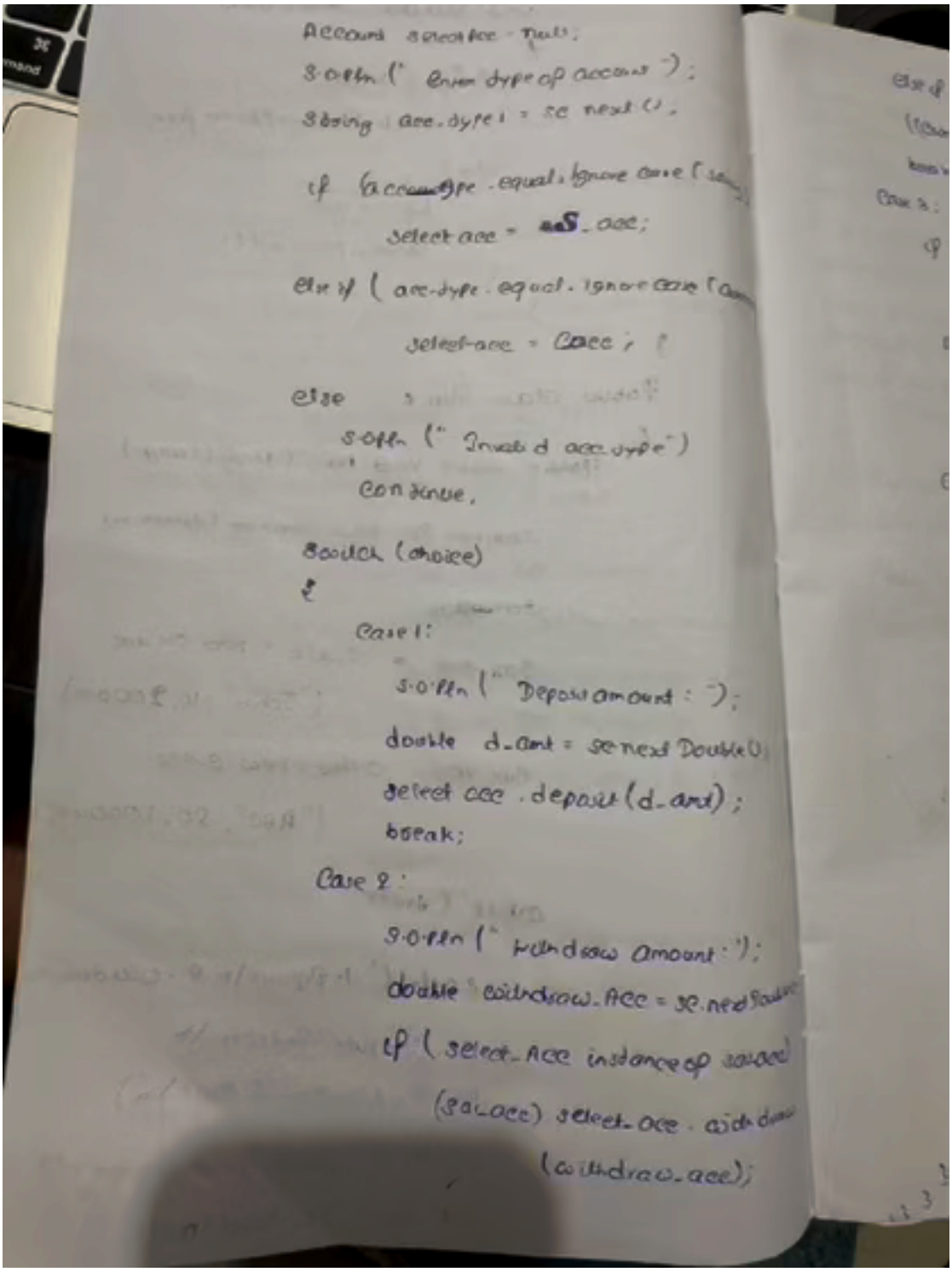
```
class Sav_acc extending Account
{
    Private static final double SN_Rate = 0.04
    Sav_Acc (c_Name, acc_no, double bal)
    {
        super (c_Name, acc_no, 'Savings', bal);
    }
    void computeInterest ()
    {
        double Interest = bal * Int_Rate;
        balance = balance + (Interest);
        S.O.println (" Interest accrued : New balance
                    : " + balance);
    }
    void withdraw (double amt)
    {
        if (amt > bal)
            S.O.println (" Insufficient funds");
        else
        {
            bal = bal - amt;
            S.O.println (" withdraw : " + bal);
        }
    }
```



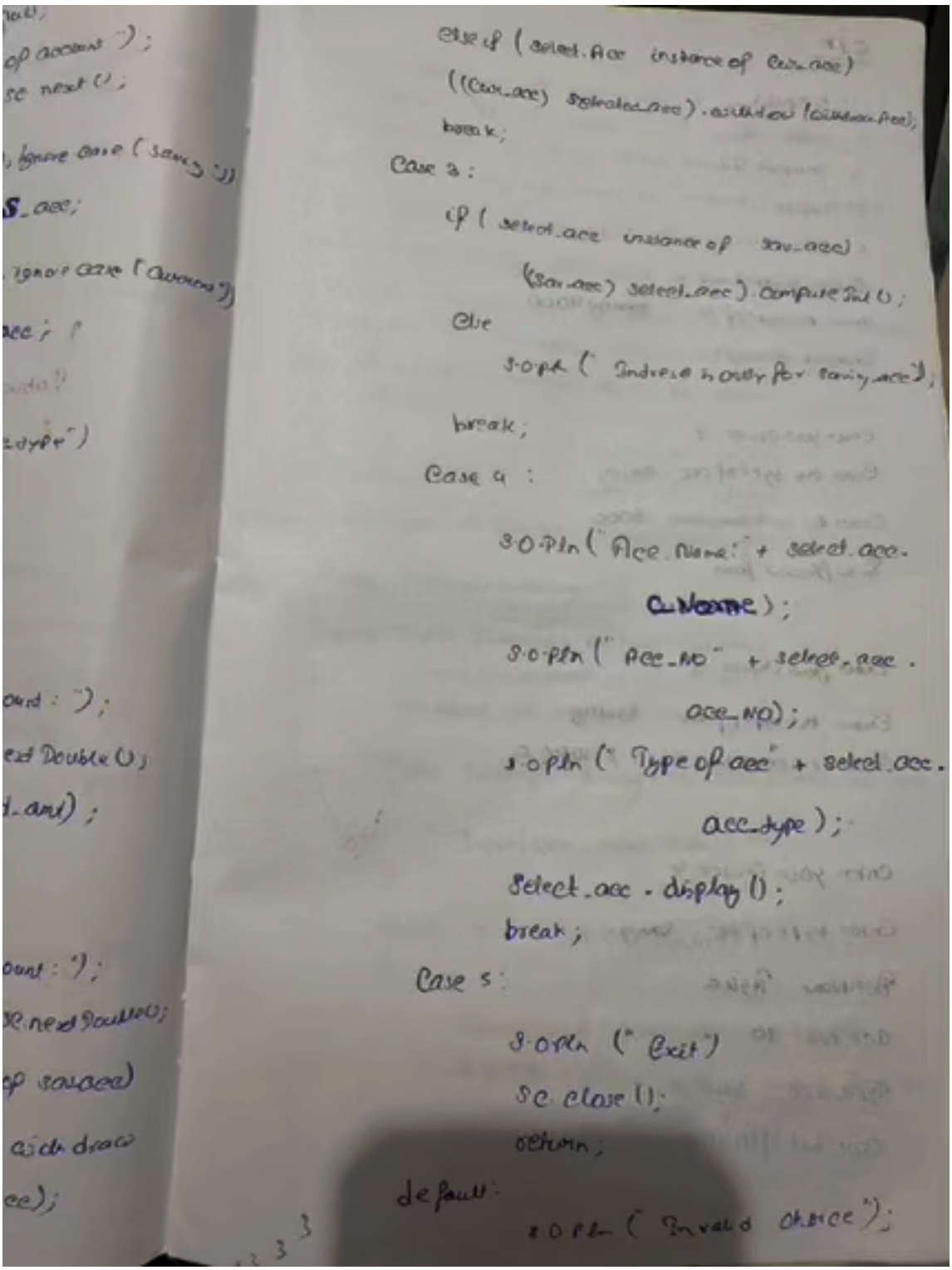
```
class Cur_acc extend Account {
{
    Private static final double Min_bal = 50
    Private static final double ser_charge = 25

    Cur_acc (String C_Name, acc_No,
             double bal)
    {
        super (C_Name, acc_No, curr
    }
    Void Check_Min_bal ()
    {
        if (bal < Min_bal)
        {
            bal - = ser_charge;
            SOPln ("Below minimum
            balance service charge of
            $" + ser_charge +"
            after New balance $ "+bal)
        }
        else
        {
            SOPln (" Balance is above
                    Min ");
        }
    }
```



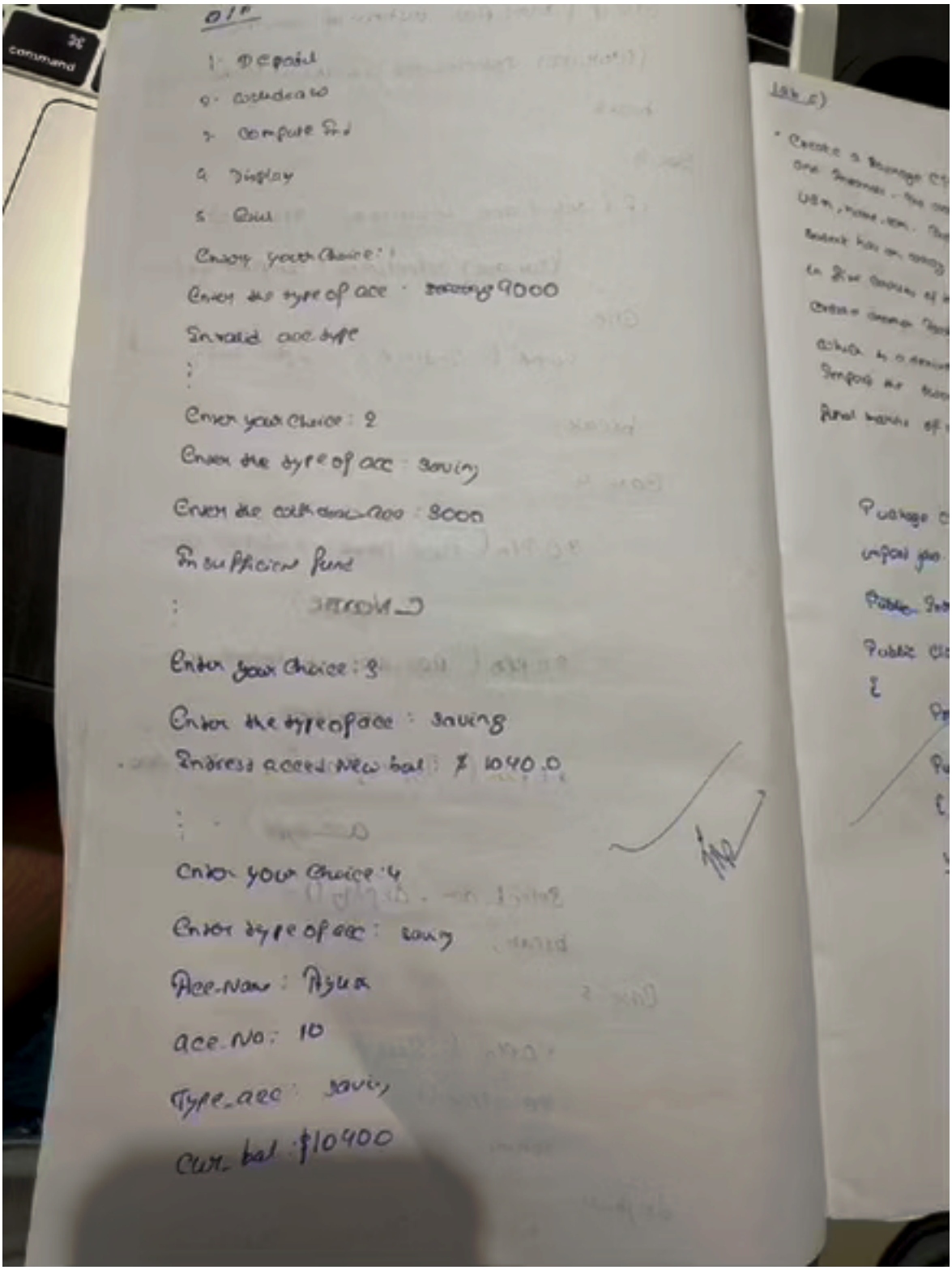
```
void withdraw (double amt)
{
    if (amt > bal)
        System.out.println ("Insufficient funds");
    else {
        bal -= amt;
        Check_Min_bal();
    }
}

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

        Sav_acc s_acc = new Sav_acc
                ("John", 10, 2000.00);

        Cur_acc c_acc = new c_acc
                ("RBC", 20, 1000.00);

        while (true)
        {
            System.out.println ("1. Deposit\n 2. Withdraw
            3. Compute Interest \n
            4. Display \n 5 Exit \n")

            System.out.println ("Enter your choice:")
            int ch = sc.nextInt();
```



```
Account selectAcc = null;
System.out.println("Enter type of account");
String acc_type1 = sc.next();

if (acc_type.equalsIgnoreCase("saving"))
    selectAcc = saS_acc;
else if (acc_type.equalsIgnoreCase("current"))
    selectAcc = Cacc;
else
    System.out.println("Invalid acc_type")
    continue;

Switch (choice)
{
    case 1:
        System.out.println("Deposit amount:");
        double d_amt = sc.nextDouble();
        selectAcc.deposit(d_amt);
        break;
    case 2:
        System.out.println("Withdraw amount:");
        double withdraw_Acc = sc.nextDouble
        if (selectAcc instanceof SavAcc)
            (SavAcc) selectAcc.withdraw
                (withdraw_acc);
```



```
                else if (select.Acc instanceof Cur_acc)
                ((Cur_acc) Selected_acc).withdraw (amount);
                break;
            Case 3 :
                if ( select_acc instanceof Sav_acc)
                    (Sav_acc) select_acc).computeInt();
                else
                    S.o.pln (" Interest is only for saving_acc");
                break;
            Case 4 :
                S.o.pln(" Acc_Name:" + select_acc.
                                    cName);
                S.o.pln (" Acc_NO" + select_acc.
                                    acc_NO);
                S.o.pln (" Type of acc" + select_acc.
                                    acc_type);
                Select_acc . display ();
                break;
            Case 5:
                S.o.pln (" Exit")
                Sc.close ();
                return;
            default:
                S.o.pln ( Invalid choice");
```



Output

1. Deposit
2. withdraw
3. Compute Int
4. Display
5. Quit

Enter your choice: 1

Enter the type of acc: saving 9000

Invalid acc type

:

Enter your choice: 2

Enter the type of acc: saving

Enter the withdraw amt: 8000

Insufficient fund

:

Enter your choice: 3

Enter the typeofacc: saving

Interest accrued New bal: $ 1040.0

:

Enter your choice: 4

Enter type of acc: saving

Acc_Name: Ajeya

acc_No: 10

Type_acc: saving

Cur_bal: $10400

Code :

```
import java.util.Scanner;

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

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

    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit successful. New balance: $" + balance);
    }

    void displayBalance() {
        System.out.println("Current balance: $" + balance);
    }
}

class SavingsAccount extends Account {
    private static final double INTEREST_RATE = 0.04;

    SavingsAccount(String customerName, int accountNumber, double
initialBalance) {
        super(customerName, accountNumber, "Savings", initialBalance);
    }

    void computeInterest() {
        double interest = balance * INTEREST_RATE;
        balance += interest;
        System.out.println("Interest added. New balance: $" + balance);
    }

    void withdraw(double amount) {
        if (amount > balance) {
```

```
            System.out.println("Insufficient funds.");
        } else {
            balance -= amount;
            System.out.println("Withdrawal successful. New balance: $" + balance);
        }
    }
}

class CurrentAccount extends Account {
    private static final double MIN_BALANCE = 500.00;
    private static final double SERVICE_CHARGE = 25.00;

    CurrentAccount(String customerName, int accountNumber, double
initialBalance) {
        super(customerName, accountNumber, "Current", initialBalance);
    }

    void checkMinimumBalance() {
        if (balance < MIN_BALANCE) {
            balance -= SERVICE_CHARGE;
            System.out.println("Below minimum balance. Service charge of $" +
SERVICE_CHARGE + " applied. New balance: $" + balance);
        } else {
            System.out.println("Balance is above the minimum required.");
        }
    }

    void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient funds.");
        } else {
            balance -= amount;
            checkMinimumBalance();
        }
    }
}

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


        SavingsAccount savingsAccount = new SavingsAccount("Ar", 1,
1000.00);
```

```
        CurrentAccount currentAccount = new CurrentAccount("Ayush", 2,
800.00);

        while (true) {
            System.out.println("\n---MENU---");
            System.out.println("1. Deposit");
            System.out.println("2. Withdraw");
            System.out.println("3. Compute interest for SavingsAccount");
            System.out.println("4. Display account details");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();

            Account selectedAccount = null;
            System.out.print("Enter the type of account (saving/current): ");
            String accountType = sc.next();

            if (accountType.equalsIgnoreCase("saving")) {
                selectedAccount = savingsAccount;
            } else if (accountType.equalsIgnoreCase("current")) {
                selectedAccount = currentAccount;
            } else {
                System.out.println("Invalid account type. Try again.");
                continue;
            }

            switch (choice) {
                case 1:
                    System.out.print("Enter the deposit amount: ");
                    double depositAmount = sc.nextDouble();
                    selectedAccount.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter the withdrawal amount: ");
                    double withdrawalAmount = sc.nextDouble();
                    if (selectedAccount instanceof SavingsAccount) {
                        ((SavingsAccount)
selectedAccount).withdraw(withdrawalAmount);
                    } else if (selectedAccount instanceof CurrentAccount) {
                        ((CurrentAccount)
selectedAccount).withdraw(withdrawalAmount);
                    }
                    break;
                case 3:
```

```
                if (selectedAccount instanceof SavingsAccount) {
                    ((SavingsAccount) selectedAccount).computeInterest();
                } else {
                    System.out.println("Interest computation is only for savings
accounts.");
                }
                break;
            case 4:
                System.out.println("Account name: " +
selectedAccount.customerName);
                System.out.println("Account number: " +
selectedAccount.accountNumber);
                System.out.println("Type of account: " +
selectedAccount.accountType);
                selectedAccount.displayBalance();
                break;
            case 5:
                System.out.println("Exiting program...");
                sc.close();
                return;
            default:
                System.out.println("Invalid choice. Try again.");

        }
    }
  }
}
```

Output:

```
 ---MENU---
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice: 1
Enter the type of account (saving/current): saving
Enter the deposit amount: 500
Deposit successful. New balance: $1500.0

---MENU---
1. Deposit
```

2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice: 2
Enter the type of account (saving/current): current
Enter the withdrawal amount: 400
Withdrawal successful. New balance: $400.0
Below minimum balance. Service charge of $25.0 applied. New balance:
$375.0

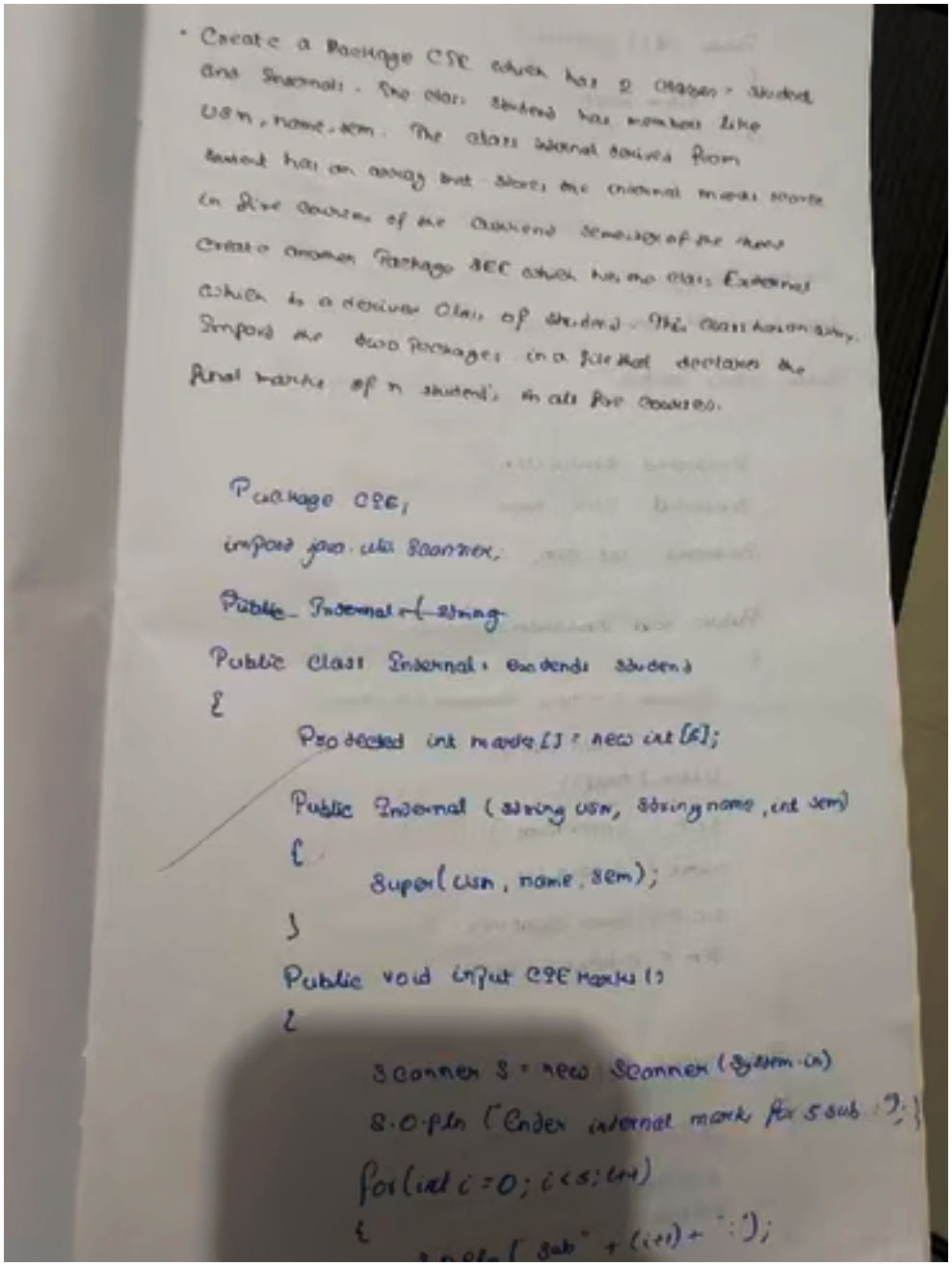
---MENU---
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): saving
Interest added. New balance: $1560.0

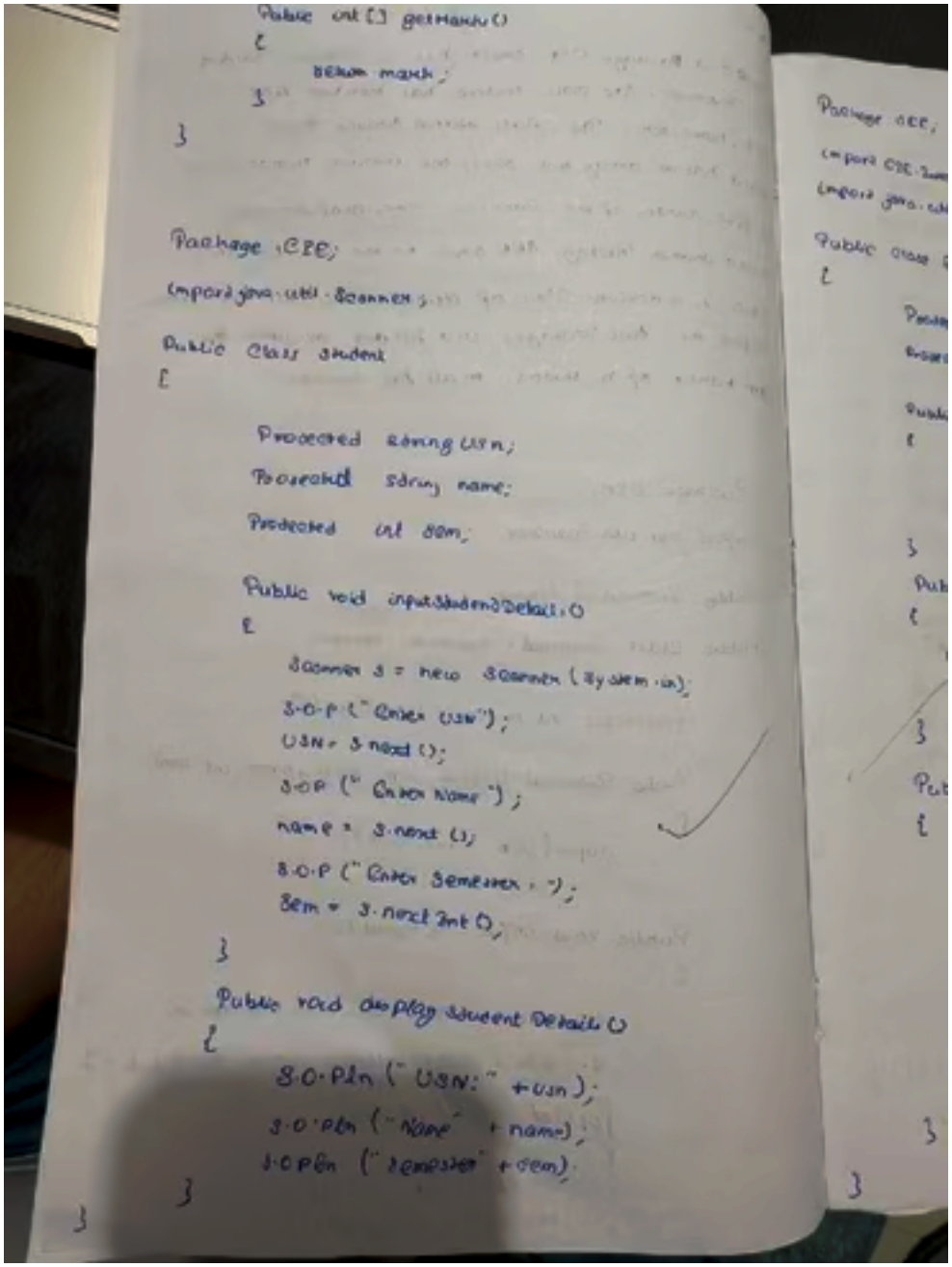
---MENU---
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice: 4
Enter the type of account (saving/current): current
Account name: Ayush
Account number: 2
Type of account: Current
Current balance: $375.0

---MENU---
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice: 5
Exiting

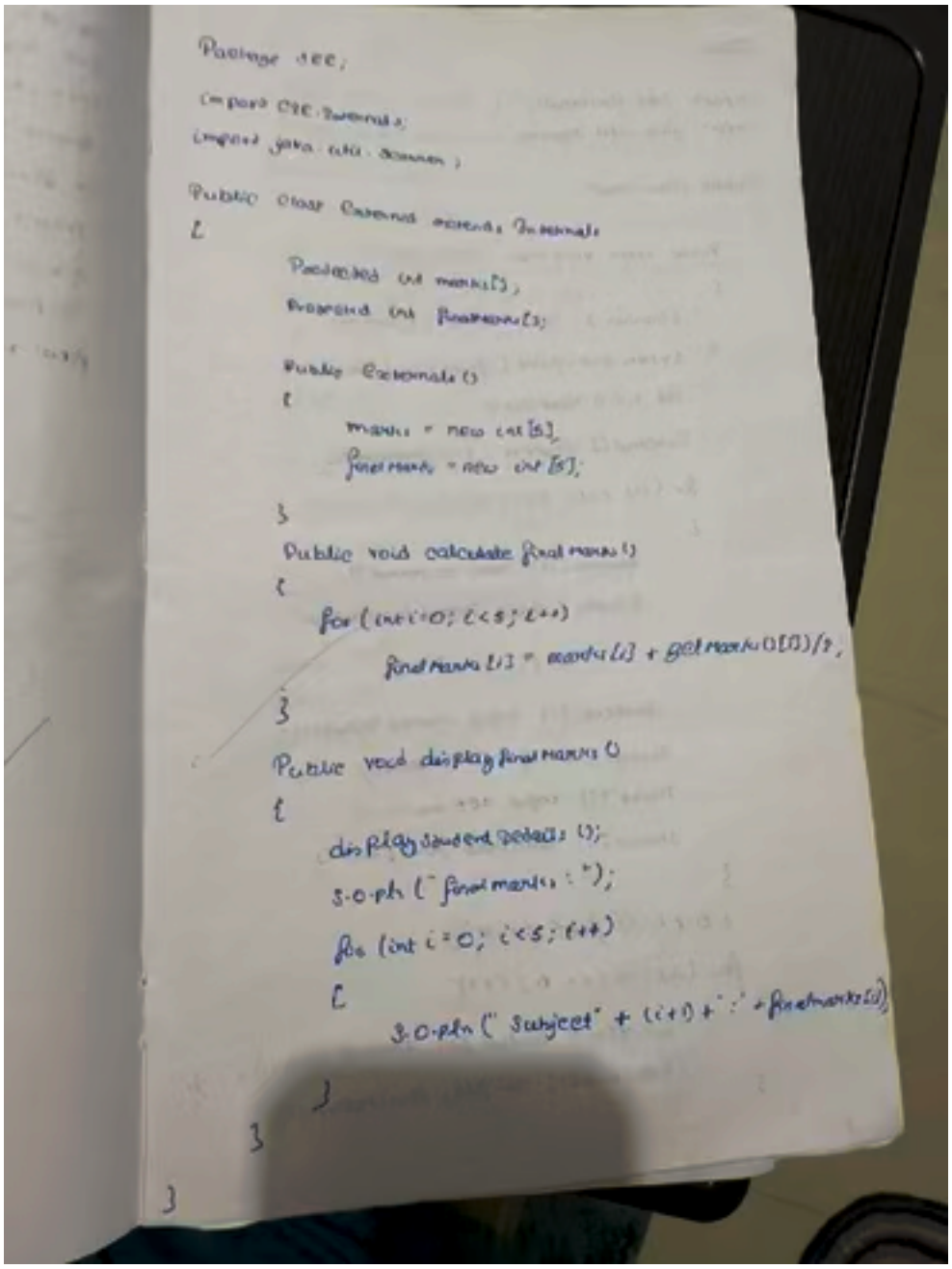
Program No: 6
Implement Packages

Algorithm :

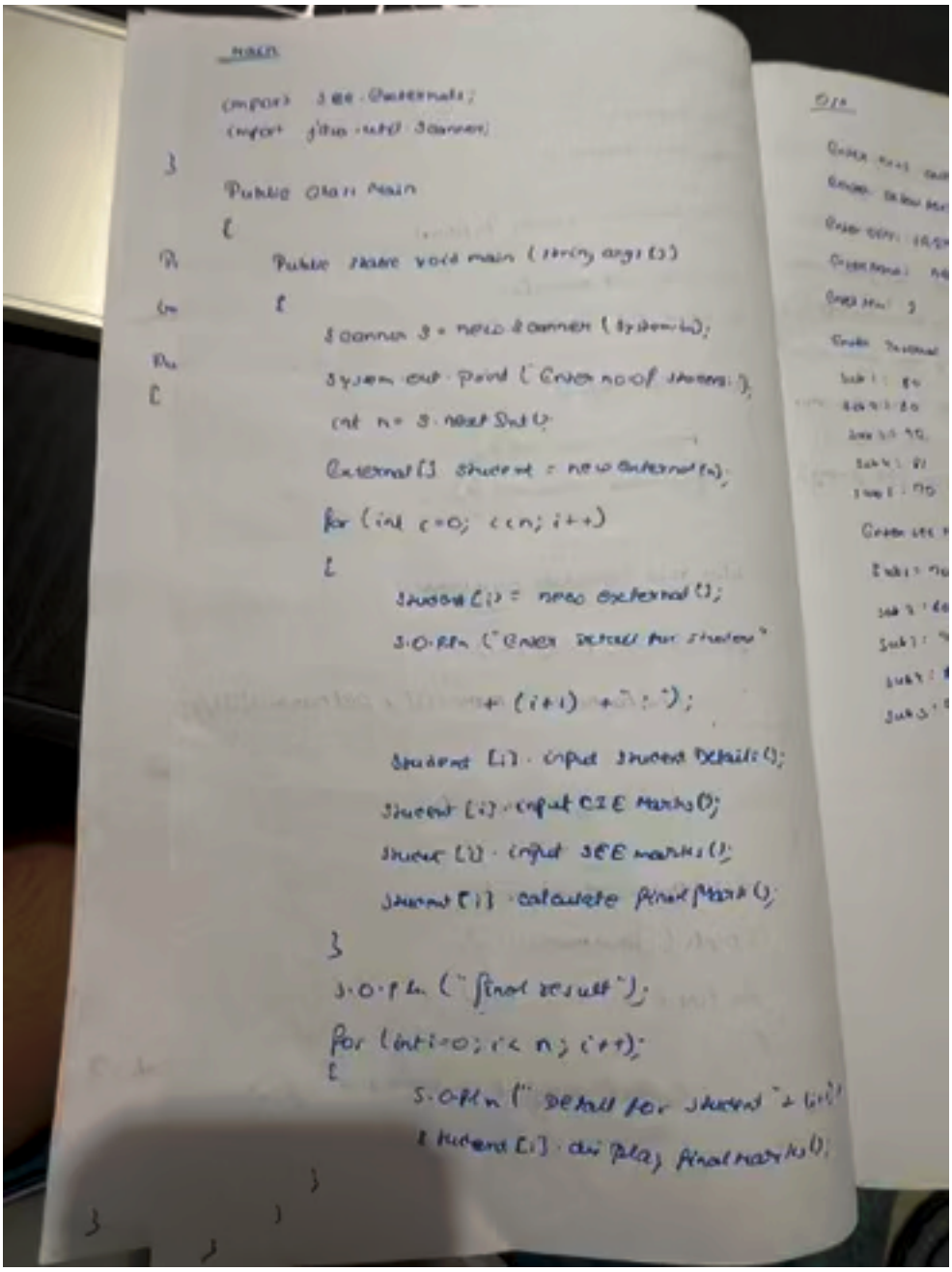




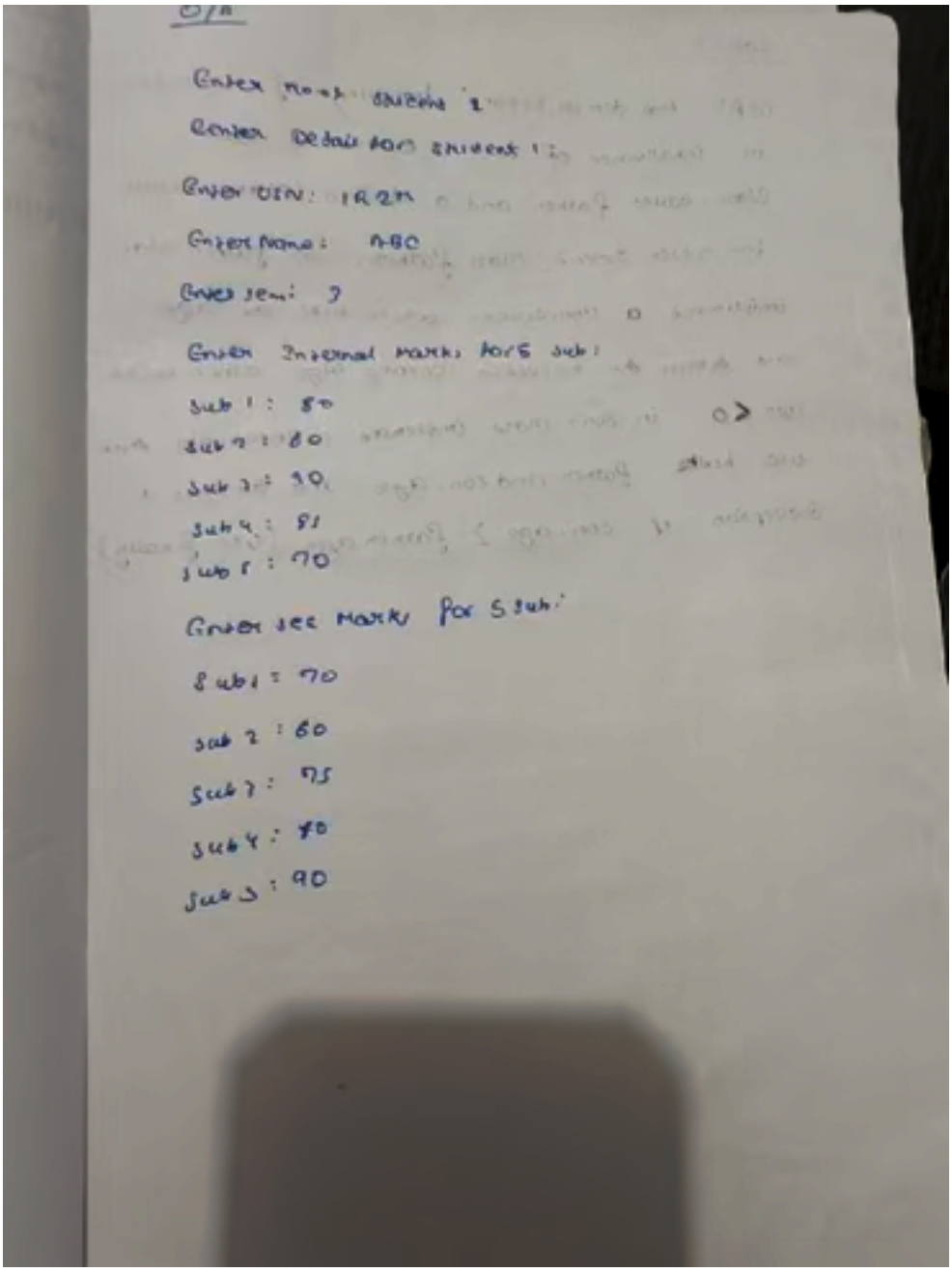
```
Public int [] getMarks()
{
    return marks;
}
}

Package CIE;

import java.util.Scanner;

Public class Student
{
    Protected String usn;
    Protected String name;
    Protected int sem;

    Public void inputStudentDetails()
    {
        Scanner s = new Scanner (System.in);
        S.O.P ("Enter USN");
        USN = S.next();
        S.O.P ("Enter Name");
        name = S.next();
        S.O.P ("Enter Semester:");
        Sem = S.nextInt();
    }

    Public void displayStudentDetails()
    {
        S.O.Pln ("USN:" +usn);
        S.O.Pln ("Name" + name);
        S.O.Pln ("Semester" + sem);
    }
}
```



```
Package sec;

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 calculate finalmarks()
        {
            for(int i=0; i<5; i++)
                finalmarks[i] = marks[i] + getmarks()[i])/2;
        }
        Public void displayfinalmarks()
        {
            displaystudentdetails();
            S.O.println("finalmarks:");
            for(int i=0; i<5; i++)
            {
                S.O.println("subject" + (i+1) + ":" + finalmarks[i]);
            }
        }
}
```



```
main

import java.io.*;
import java.util.Scanner;

Public class main
{
    Public static void main (String args[])
    {
        Scanner s = new Scanner (System.in);
        System.out.println ("Enter no of students");
        int n = s.nextInt();
        External[] student = new External[n];
        for (int i=0; i<n; i++)
        {
            student[i] = new external();
            s.println ("Enter details for student"
                + (i+1) + ":");
            student[i].inputStudentDetails();
            student[i].inputCIEMarks();
            student[i].inputSEEmarks();
            student[i].calculateFinalMarks();
        }
        s.println ("final result");
        for (int i=0; i<n; i++);
        {
            s.println ("details for student" + i);
            student[i].displayFinalMarks();
        }
    }
}
```




Code:

```
package SEE;

import CIE.Student;

public class External extends Student {
    public int[] externalMarks;
```

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


    public void printExternalMarks() {
        System.out.println("External Marks for " + name + " (" + usn + "):");
        for (int i = 0; i < externalMarks.length; i++) {
            System.out.println("Course " + (i + 1) + ": " + externalMarks[i]);
        }
    }
}


package CIE;

public class Internals extends Student {
    public int[] internalMarks;


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


    public void printInternalMarks() {
        System.out.println("Internal Marks for " + name + " (" + usn + "):");
        for (int i = 0; i < internalMarks.length; i++) {
            System.out.println("Course " + (i + 1) + ": " + internalMarks[i]);
        }
    }
}



import CIE.Internals;
import SEE.External;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
```

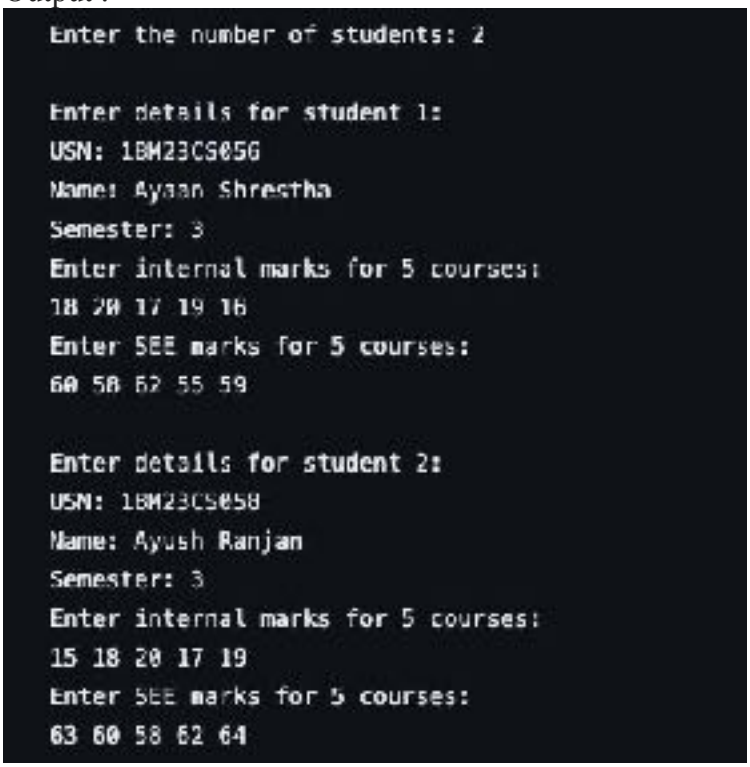
```
Scanner scanner = new Scanner(System.in);


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

for (int i = 0; i < n; i++) {

    System.out.println("\nEnter details for Student " + (i + 1) + ":");

    System.out.print("Enter USN: ");
    String usn = scanner.nextLine();

    System.out.print("Enter Name: ");
    String name = scanner.nextLine();

    System.out.print("Enter Semester: ");
    int sem = scanner.nextInt();


    int[] internalMarks = new int[5];
    System.out.println("Enter Internal Marks for 5 Courses:");
    for (int j = 0; j < 5; j++) {
        System.out.print("Course " + (j + 1) + ": ");
        internalMarks[j] = scanner.nextInt();
    }


    int[] externalMarks = new int[5];
    System.out.println("Enter External Marks for 5 Courses:");
    for (int j = 0; j < 5; j++) {
        System.out.print("Course " + (j + 1) + ": ");
        externalMarks[j] = scanner.nextInt();
    }
    scanner.nextLine();


    Internals internalStudent = new Internals(usn, name, sem, internalMarks);
    External externalStudent = new External(usn, name, sem, externalMarks);


    internalStudent.printInternalMarks();
    externalStudent.printExternalMarks();
```

```
            printFinalMarks(internalStudent, externalStudent);
        }

        scanner.close();
    }


    public static void printFinalMarks(Internals internal, External external) {
        int[] internalMarks = internal.internalMarks;
        int[] externalMarks = external.externalMarks;
        int totalMarks;

        System.out.println("Final Marks for " + internal.getName() + " (" + internal.getUsn() + "):");
        for (int i = 0; i < internalMarks.length; i++) {
            totalMarks = internalMarks[i] + externalMarks[i];
            System.out.println("Course " + (i + 1) + ": " + totalMarks);
        }
    }
}
```

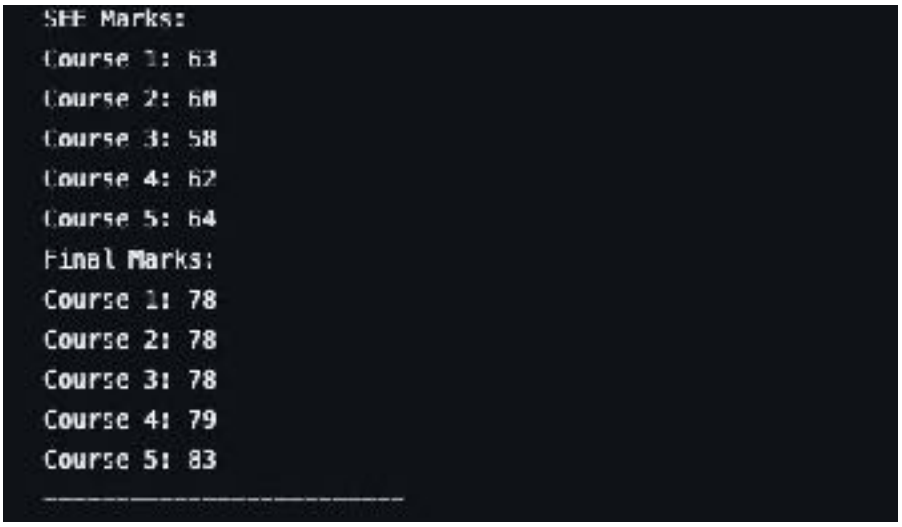
```
package CIE;

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


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

    // Getter methods
    public String getUsn() {
        return usn;
    }

    public String getName() {
        return name;
    }
```

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

Output :

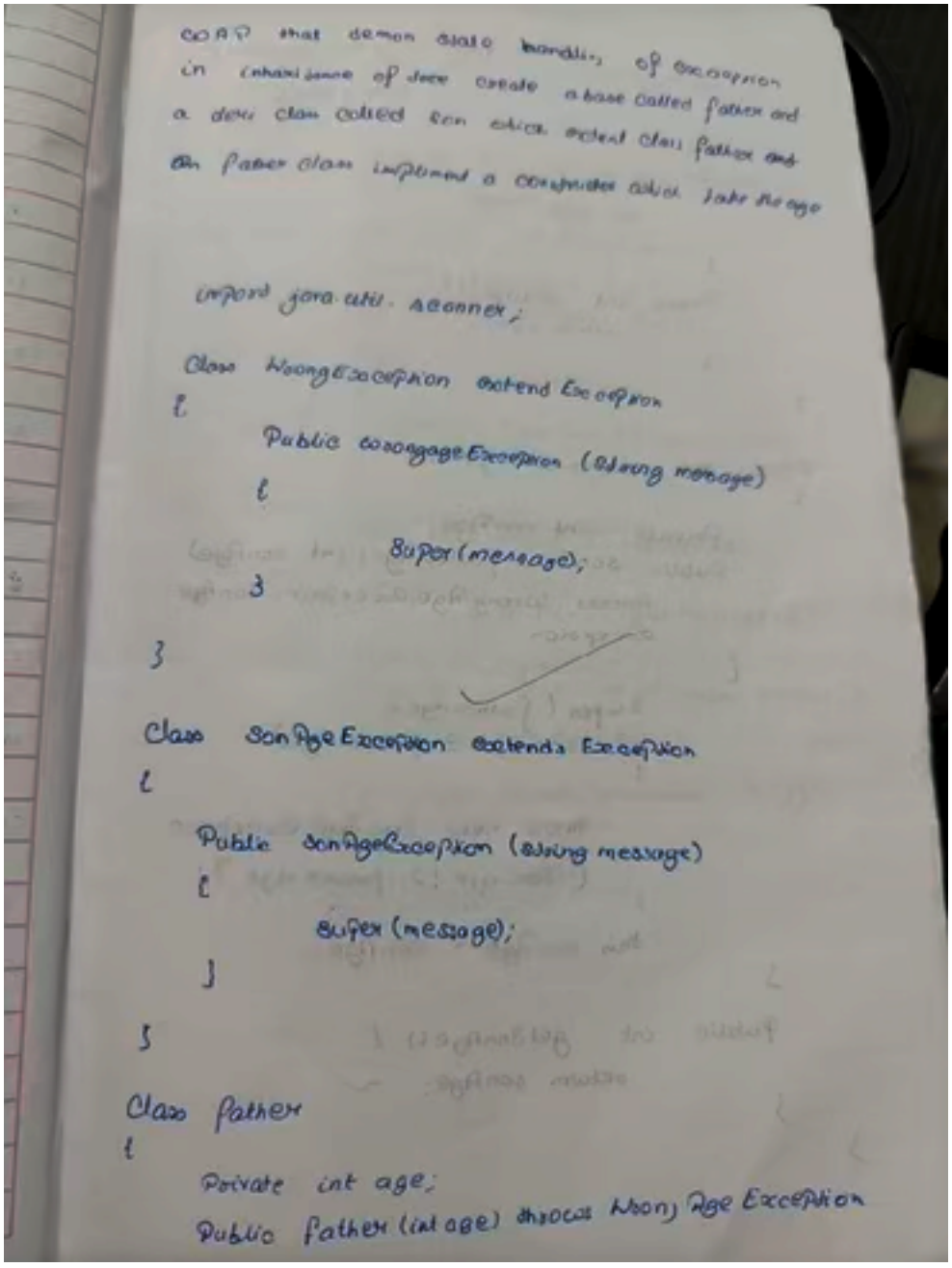


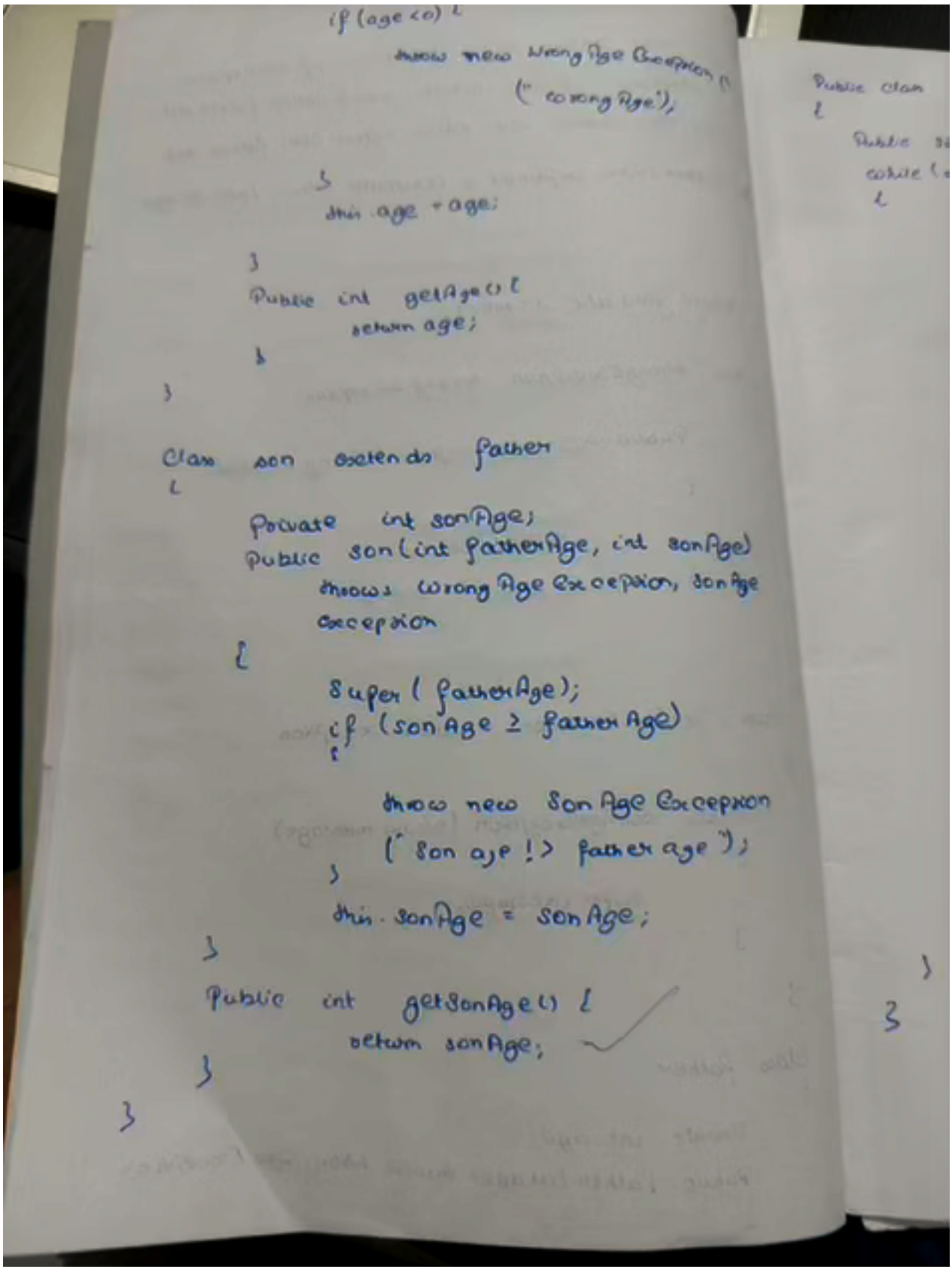
```
Final Marks of Students:

USN: 1BM23CS056
Name: Ayaan Shrestha
Semester: 3
Internal Marks:
Course 1: 18
Course 2: 20
Course 3: 17
Course 4: 19
Course 5: 16
SEE Marks:
Course 1: 60
Course 2: 58
Course 3: 62
Course 4: 55
Course 5: 59
Final Marks:
Course 1: 78
Course 2: 78
Course 3: 79
Course 4: 74
Course 5: 75
------------------------

USN: 1BM23CS058
Name: Ayush Ranjan
Semester: 3
Internal Marks:
Course 1: 15
Course 2: 18
Course 3: 20
Course 4: 17
Course 5: 19
SEE Marks:
Course 1: 63
Course 2: 60
Course 3: 58
Course 4: 62
Course 5: 64
```


SEE Marks:
Course 1: 63
Course 2: 60
Course 3: 58
Course 4: 62
Course 5: 64
Final Marks:
Course 1: 78
Course 2: 78
Course 3: 78
Course 4: 79
Course 5: 83

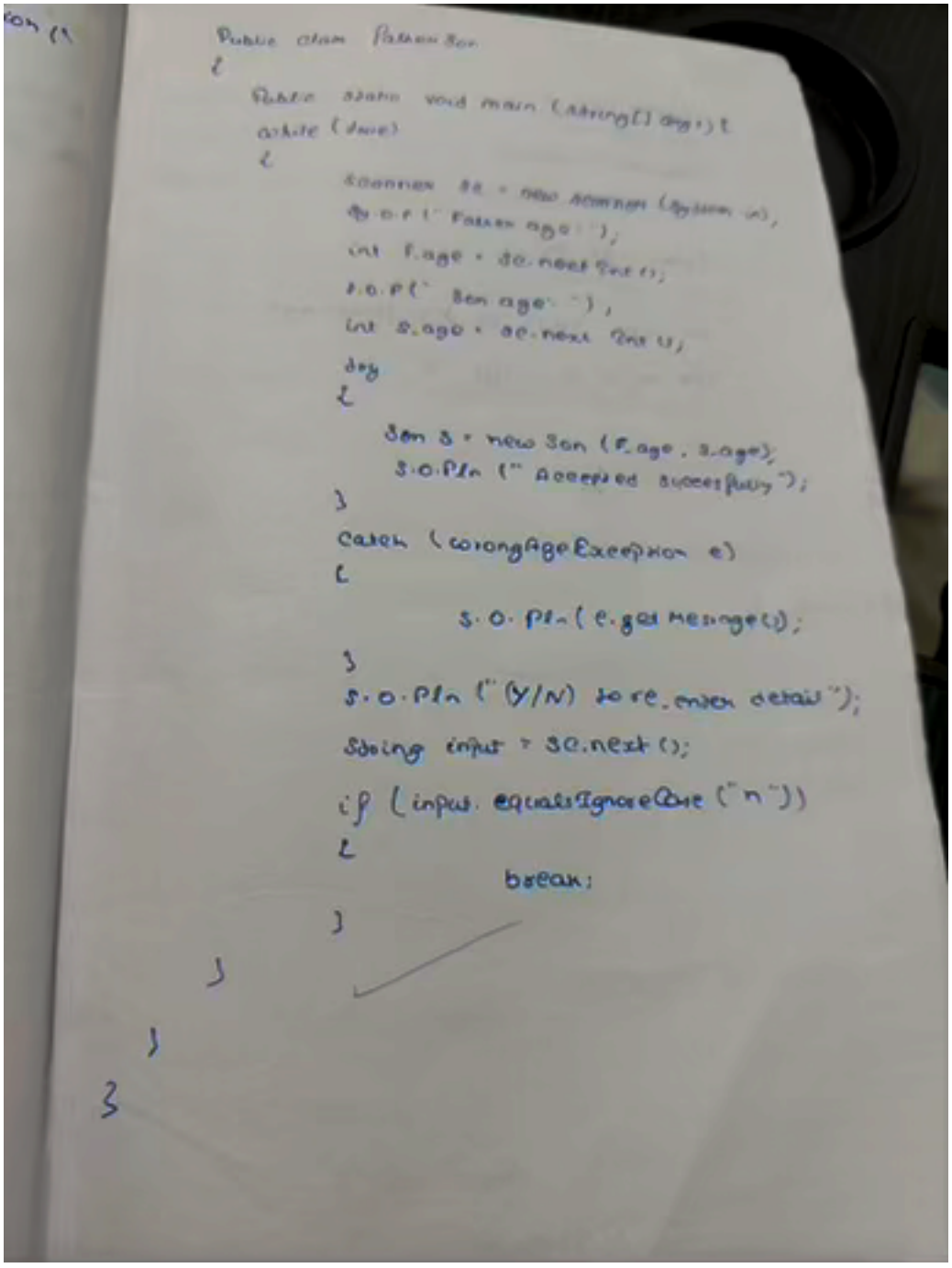
Program No : 7
Implement Exception Handling

Algorithm :

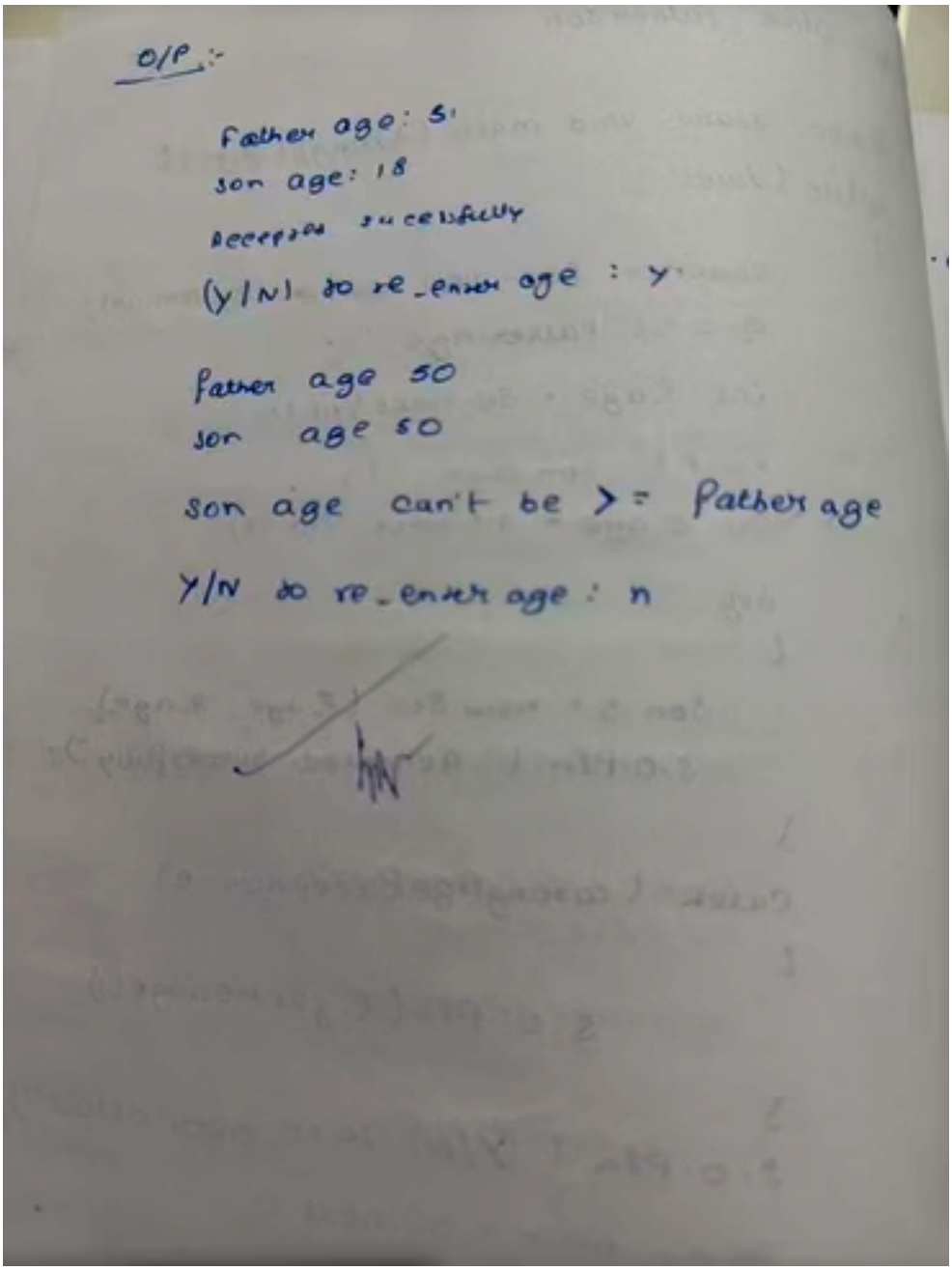




```
        if (age < 0) {
            throw new WrongAgeException
                ("wrong Age");
        }
        this.age = age;
    }
    Public int getAge() {
        return age;
    }
}

Class son extends father
{
    Private int sonAge;
    Public son(int fatherAge, int sonAge)
        throws WrongAgeException, SonAge
        Exception
    {
        super(fatherAge);
        if (sonAge ≥ fatherAge)
        {
            throw new SonAgeException
            ("Son age !> father age");
        }
        this.sonAge = sonAge;
    }
    Public int getSonAge() {
        return sonAge;
    }
}
```



```
Public class FatherSon
{
    Public static void main (String[] args) t
    while (true)
    {
        Scanner sc = new Scanner (System.in);
        System.out.println(" Father age: ");
        int F_age = sc.nextInt();
        System.out.println(" Son age: ");
        int S_age = sc.nextInt();
        try
        {
            Son s = new Son (F_age, S_age);
            System.out.println(" Accepted successfully");
        }
        Catch (WrongAgeException e)
        {
            System.out.println(e.getMessage());
        }
        System.out.println(" (Y/N) to re-enter details");
        String input = sc.next();
        if (input.equalsIgnoreCase("n"))
        {
            break;
        }
    }
}
}
```



O/P :-

Father age: 5
son age: 18
Exception successfully
(Y/N) do re_enter age : Y

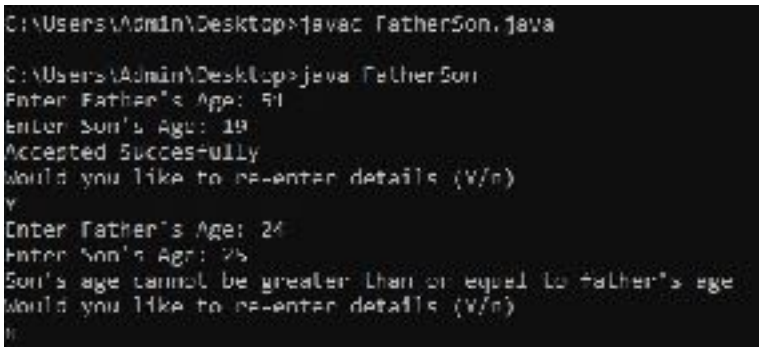
Father age 50
son age 50
son age can't be >= Father age
Y/N do re_enter age : n

Code :

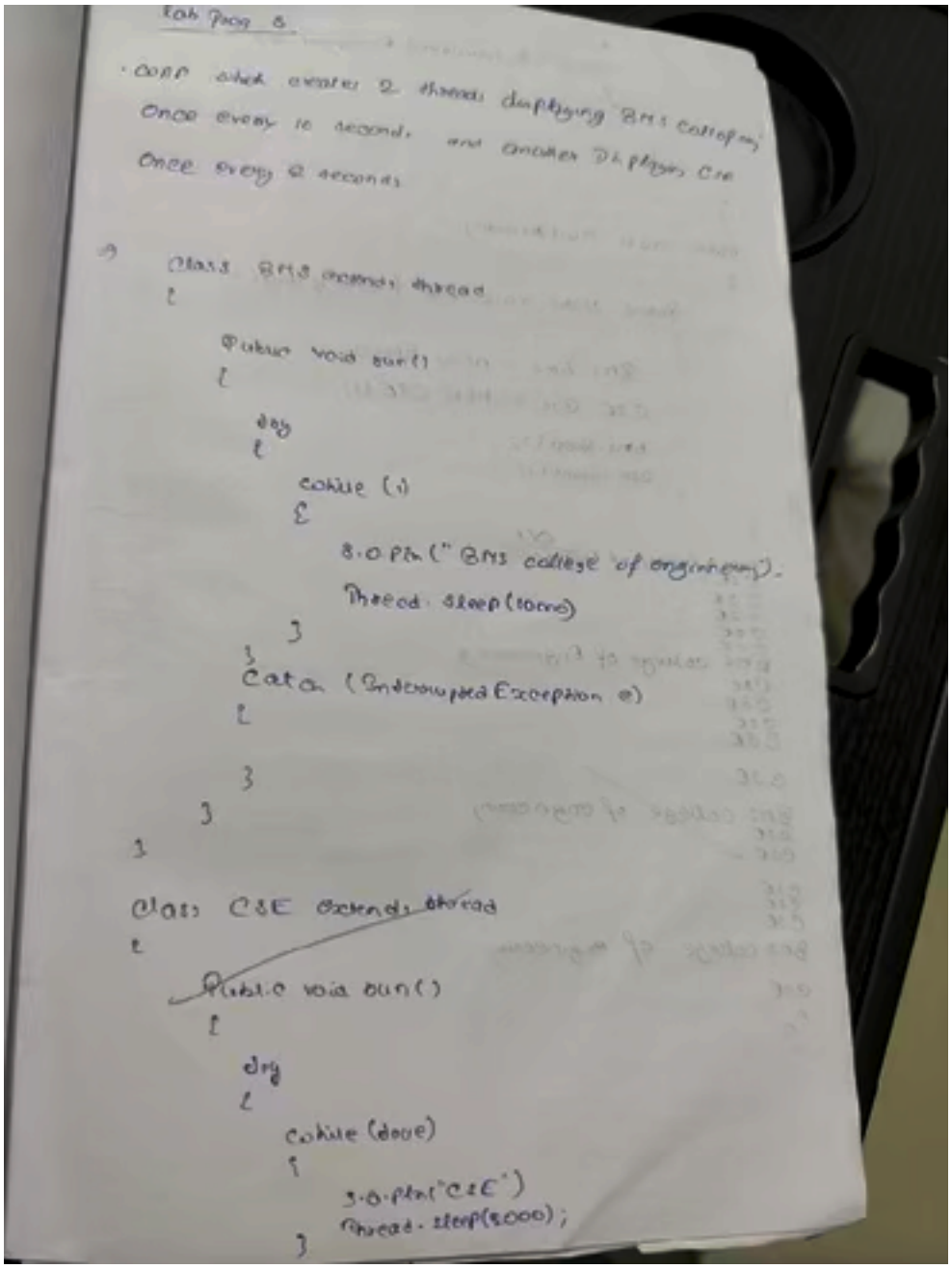
```
import java.util.Scanner;
class WrongAgeException extends Exception {
   public WrongAgeException(String message) {
      super(message);
   }
}

class SonAgeException extends Exception {
   public SonAgeException(String message) {
      super(message);
   }
}

class Father {
   private int age;
   public Father(int age) throws WrongAgeException {
      if (age < 0) {
         throw new WrongAgeException("Wrong age");
      }
      this.age = age;
   }
   public int getAge() {
      return age;
   }
}

class Son extends Father {
   private int sonAge;
   public Son(int fatherAge, int sonAge) throws WrongAgeException,
SonAgeException {
      super(fatherAge);
      if (sonAge >= fatherAge) {
         throw new SonAgeException("Son's age cannot be greater than or equal
to father's age");
      }
      this.sonAge = sonAge;
   }
   public int getSonAge() {
      return sonAge;
   }
}
public class FatherSon{
```

```
    public static void main(String[] args) {
        while(true){
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter Father's Age: ");
            int fatherAge = sc.nextInt();
            System.out.print("Enter Son's Age: ");
            int sonAge = sc.nextInt();
            try {
                Son son = new Son(fatherAge, sonAge);
                System.out.println("Accepted Succesfully");
            }
            catch (WrongAgeException e) {
                System.out.println(e.getMessage());
            }
            catch (SonAgeException e) {
                System.out.println(e.getMessage());
            }
            System.out.println("Would you like to re-enter details (Y/n)");
            String input = sc.next();
            if (input.equalsIgnoreCase("n")) {
                break;
            }
        }
    }
}
```

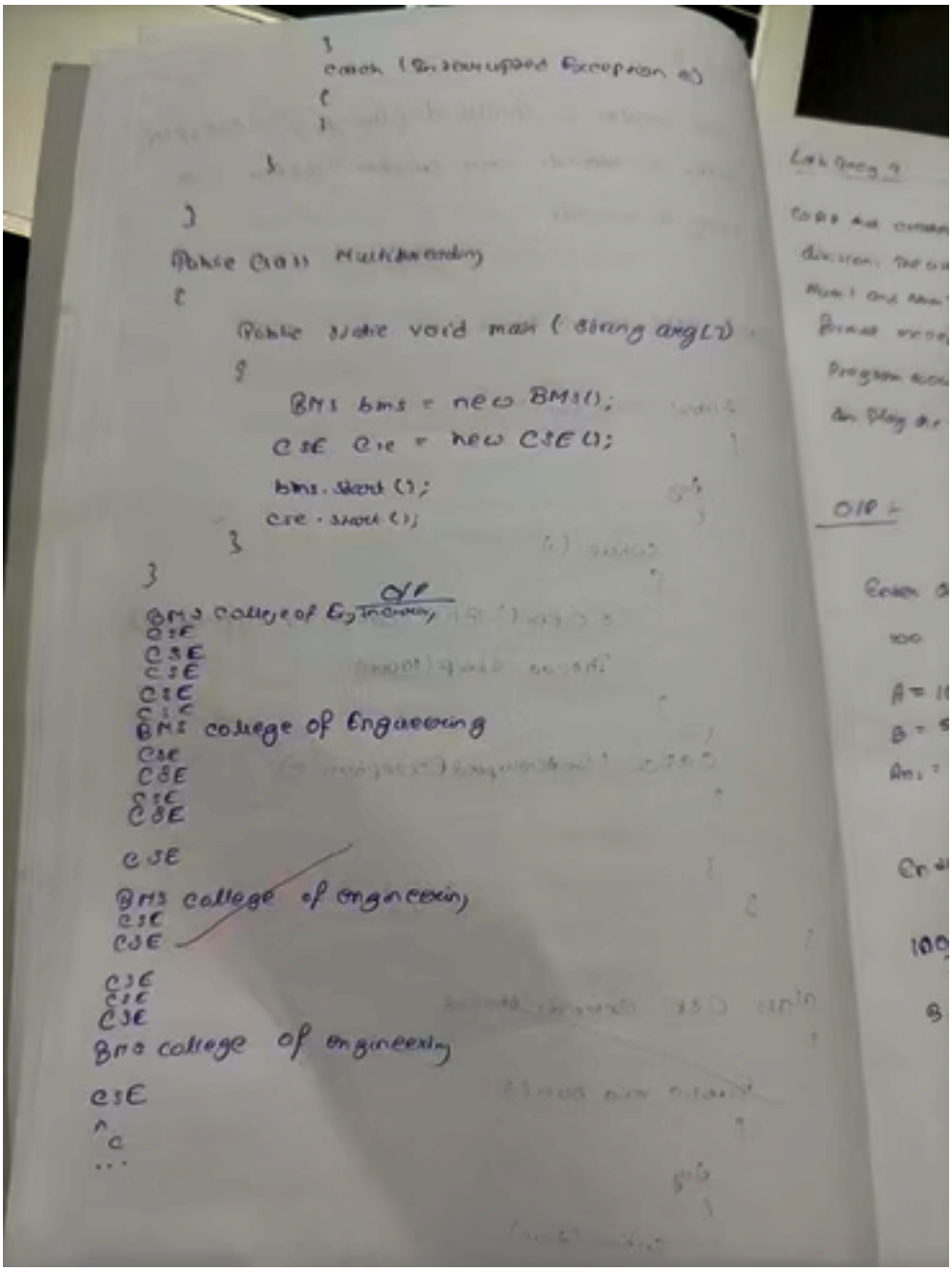
Output:




Program No : 8
Multithreading, Creating Threads in Java


Lab Prog 8
COAP which creates 2 threads displaying BMS College
Once every 10 second. and another Displays CSE
Once every 2 seconds
class BMS extends thread
public void run()
try
while (1)
System.out.println("BMS college of engineering");
Thread.sleep(10000)
catch (InterruptedException e)
class CSE extends thread
public void run()
try
while (true)
System.out.println("CSE")
Thread.sleep(2000);

Algorithm :



Code :

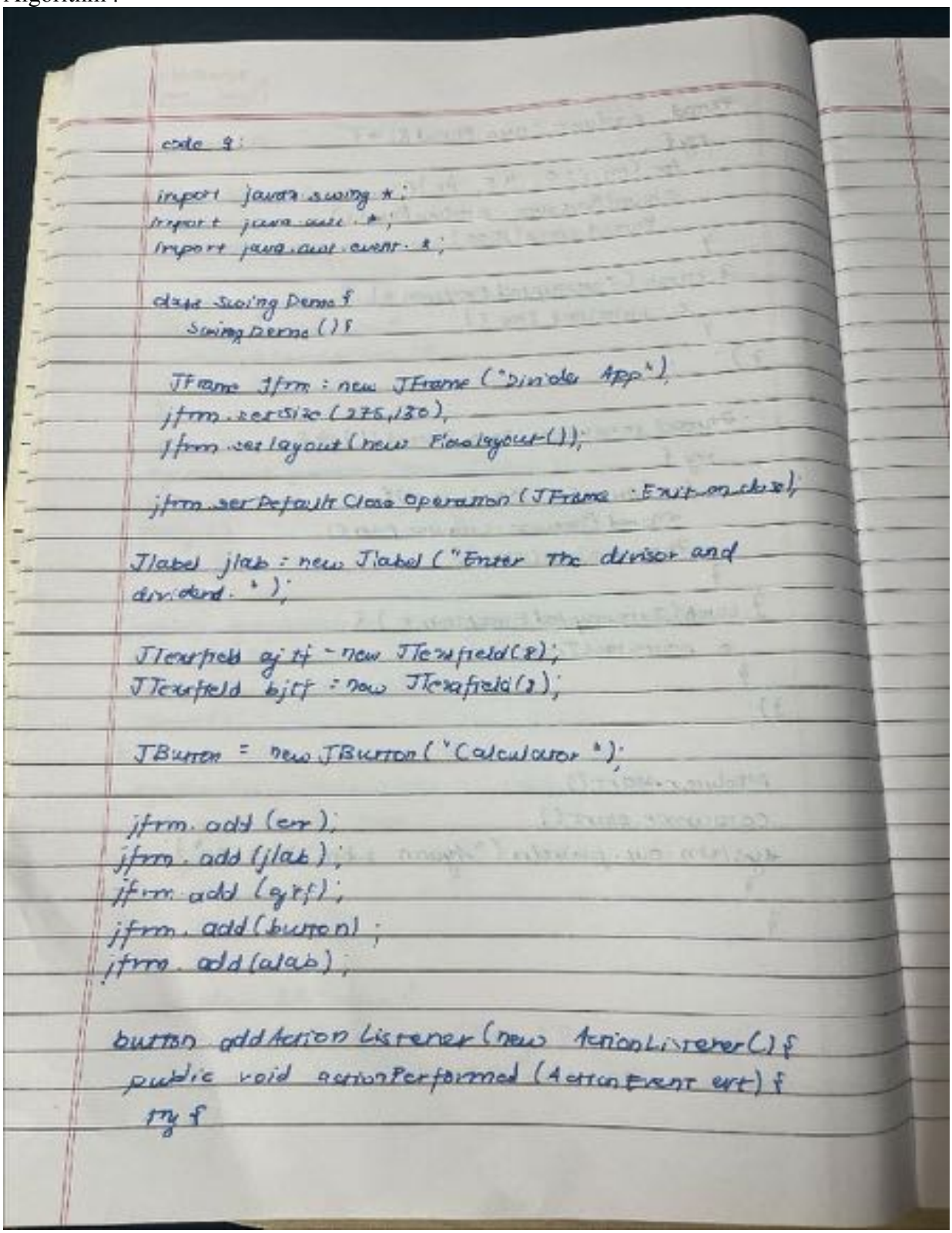
```
class BMS extends Thread {
   public void run() {
      try {
         while (true) {
```

```
            System.out.println("BMS College of Engineering");
            Thread.sleep(10000); // Sleep for 10 seconds
        }
    }catch (InterruptedException e) {}
  }
}
class CSE extends Thread {
  public void run() {
    try {
      while (true) {
        System.out.println("CSE");
        Thread.sleep(2000); // Sleep for 2 seconds
      }
    }catch (InterruptedException e) {}
  }
}

public class Multithreading{
  public static void main(String[] args) {
    BMS bms = new BMS();
    CSE cse = new CSE();
    bms.start();
    cse.start();
  }
}
```

Output:


Program No : 9
Interface to Perform Integer Division

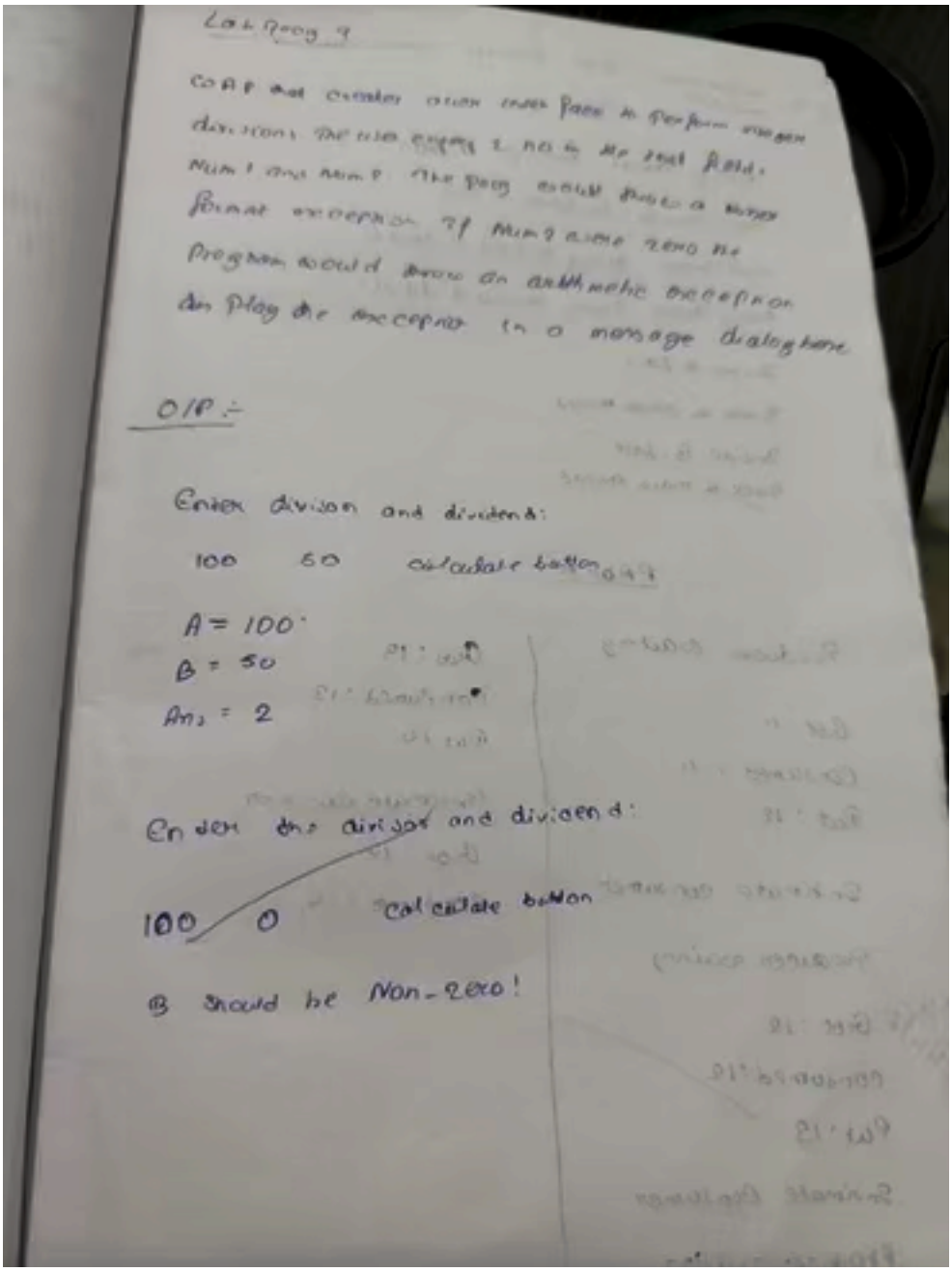
Algorithm :



code :

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

class SwingDemo {
  SwingDemo() {

  JFrame jfrm = new JFrame("Divide App");
  jfrm.setSize(275,130);
  jfrm.setLayout(new FlowLayout());

  jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

JLabel jlab = new JLabel("Enter the divisor and
dividend.");

JTextField ajtf = new JTextField(8);
JTextField bjtf = new JTextField(8);

JButton = new JButton("Calculate");

jfrm.add(err);
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(button);
jfrm.add(alab);

button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
  try {
```

```
int a = Integer.parseInt(ajtf.getText());
int b = Integer.parseInt(bjtf.getText());

int ans = a / b;
alab.setText("A = " + a);
blab.setText("B = " + b);
ans lab.setText("Ans: " + ans);
err.setText("");

} catch(NumberFormatException e) {

  alab.setText("");
  blab.setText("");
  ans lab.setText("");
  err.setText("B should be Non-zero!");
  }
 }
});

  jfrm.setVisible(true);
}

public static void main(String args[]){

SwingUtilities.invokeLater(new Runnable() {
    public void run(){
      new SwingDemo();
    }
  });
 }
}
```



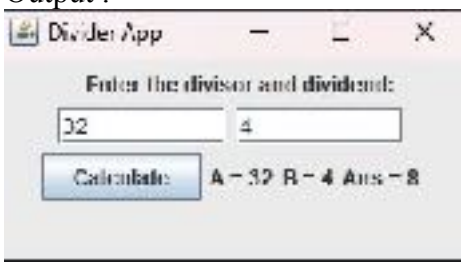

Code :

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

class SwingDemo {
   SwingDemo() {

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

```
jfrm.setSize(275, 150);
jfrm.setLayout(new FlowLayout());

jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


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


JTextField ajtf = new JTextField(8);
JTextField bjtf = new JTextField(8);


JButton button = new JButton("Calculate");


JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();
JLabel anslab = new JLabel();


jfrm.add(err);
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);


button.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      try {

         int a = Integer.parseInt(ajtf.getText());
         int b = Integer.parseInt(bjtf.getText());


         int ans = a / b;
         alab.setText("A = " + a);
         blab.setText("B = " + b);
         anslab.setText("Ans = " + ans);
```

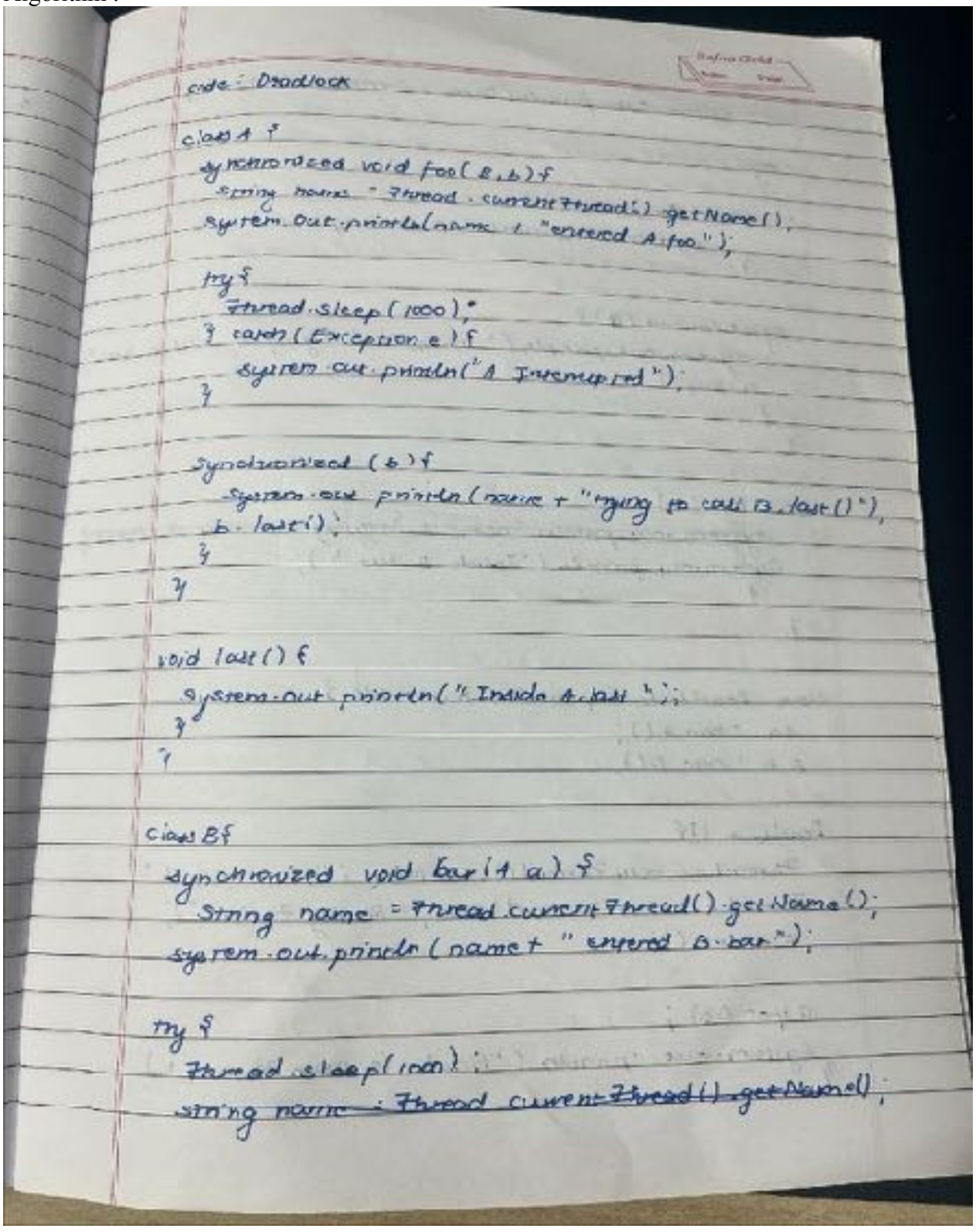
```
                err.setText("");

            } catch (NumberFormatException e) {

                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("Enter Only Integers!");
            } catch (ArithmeticException e) {

                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("B should be NON-zero!");
            }
        }
    });


    jfrm.setVisible(true);
  }

  public static void main(String args[]) {

    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new SwingDemo();
      }
    });
  }
}
```

Output :

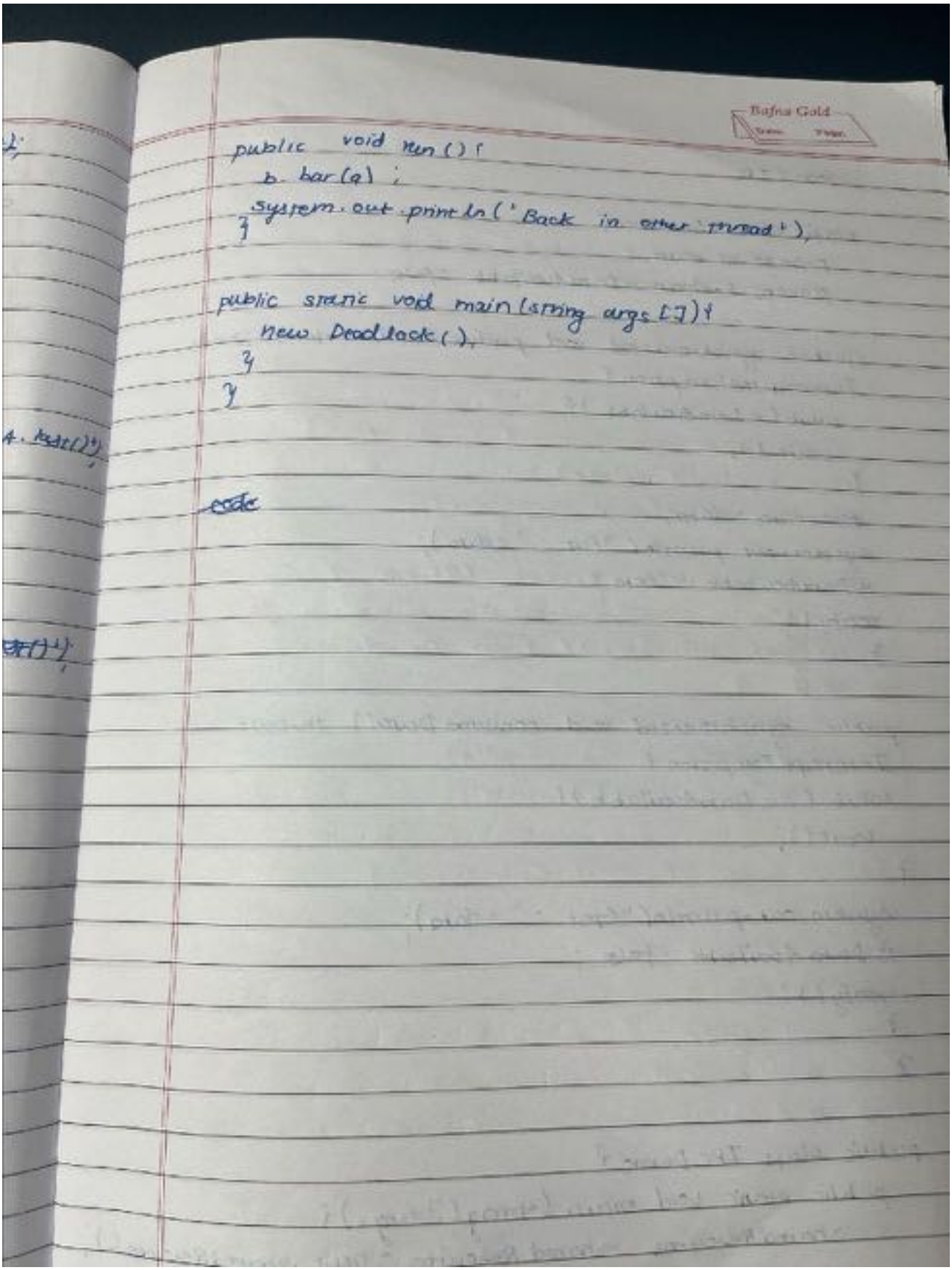


Program 10.1
Implement Deadlock

Algorithm :

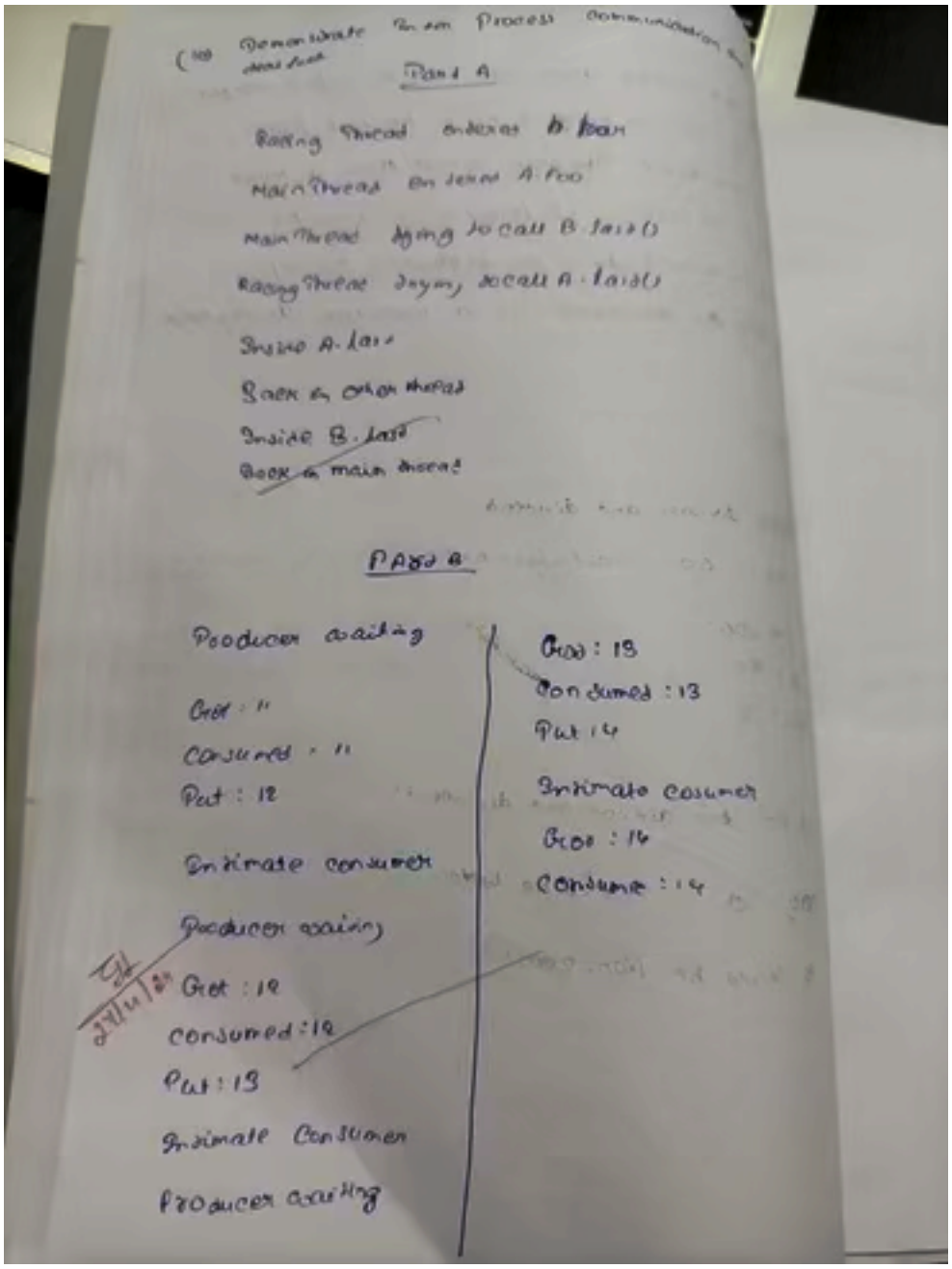


```
    system.out.println(name + "entered B.bar");
    try{
    }
    catch(Exception e){
        system.out.println("B Interrupted");
    }

Synchronized (A){
    system.out.println(" name + " trying to call A.last()
    a.last();
    }
}

void last(){
    system.out.println(name + "trying to call A.last()");
    System.out.println("Inside B.bar");
    }
}

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

Deadlock(){
    Thread.currentThread().setName("MainThread");
    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();
}
}
```

code



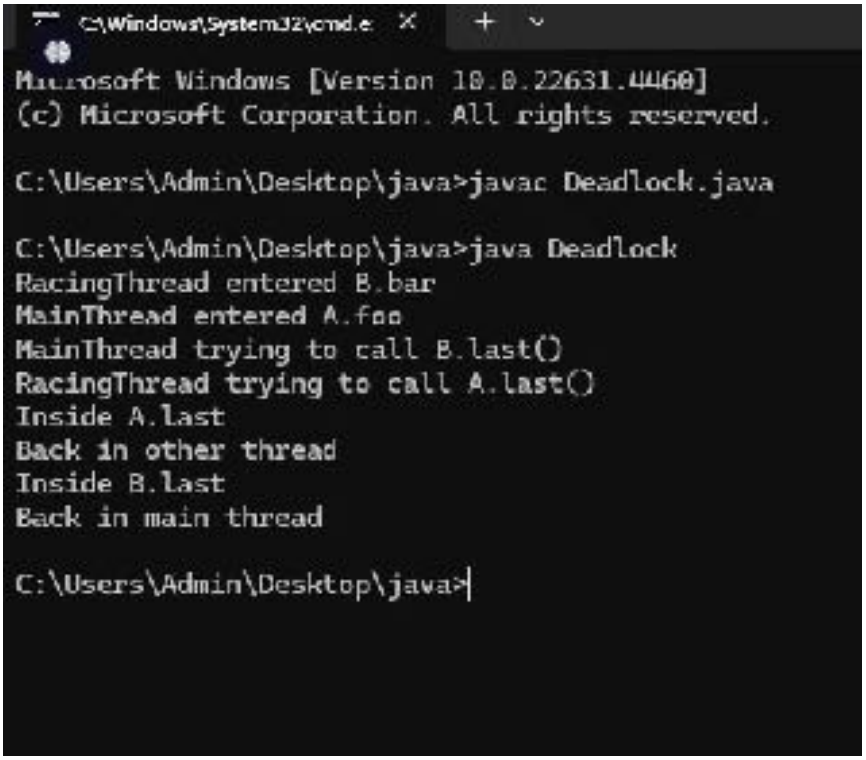
Code :

```
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");
        }


        synchronized (b) {
            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");
        }


        synchronized (a) {
            System.out.println(name + " trying to call A.last()");
            a.last();
        }
    }

    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");
```

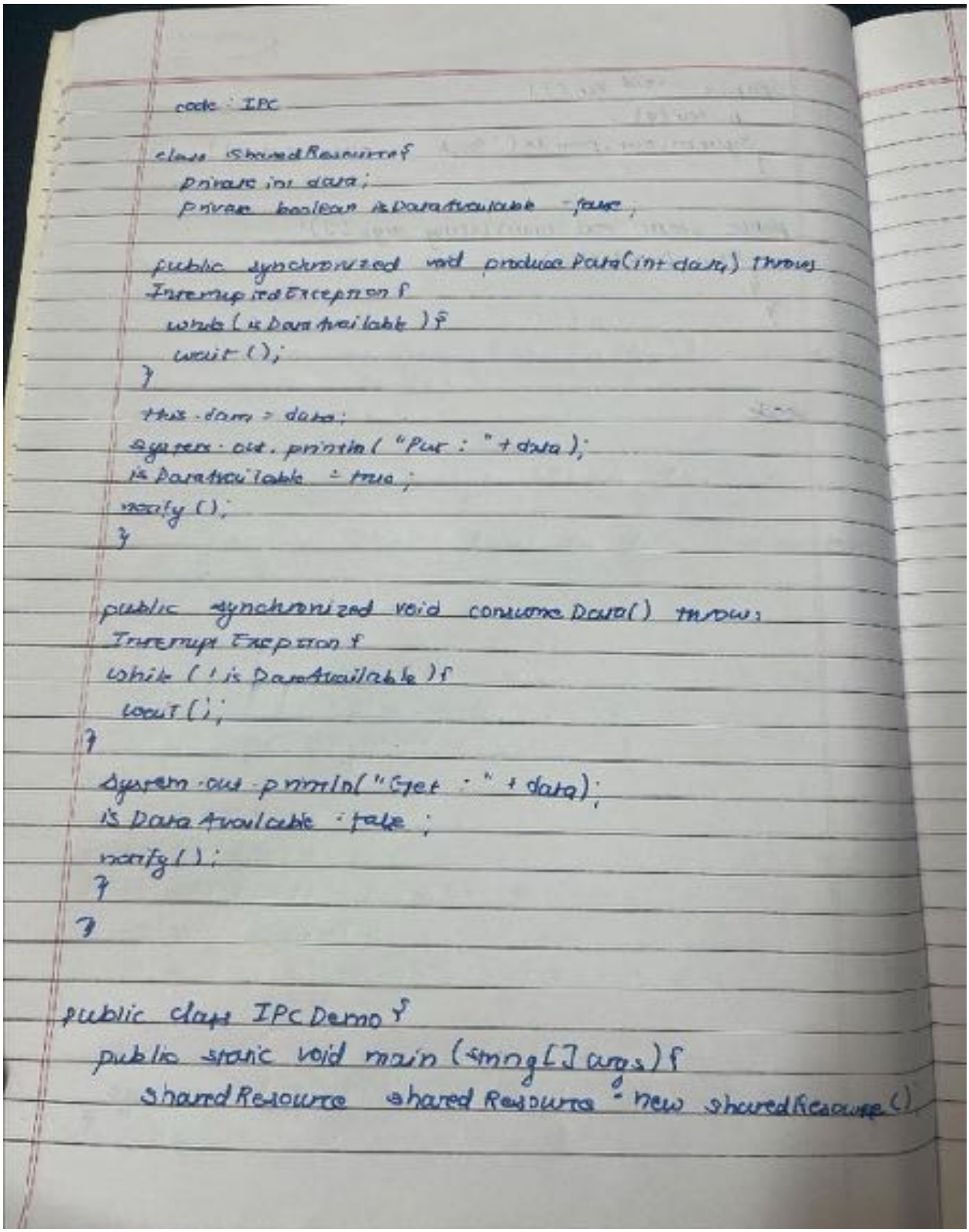
```
        Thread t = new Thread(this, "RacingThread");
        t.start();

        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:

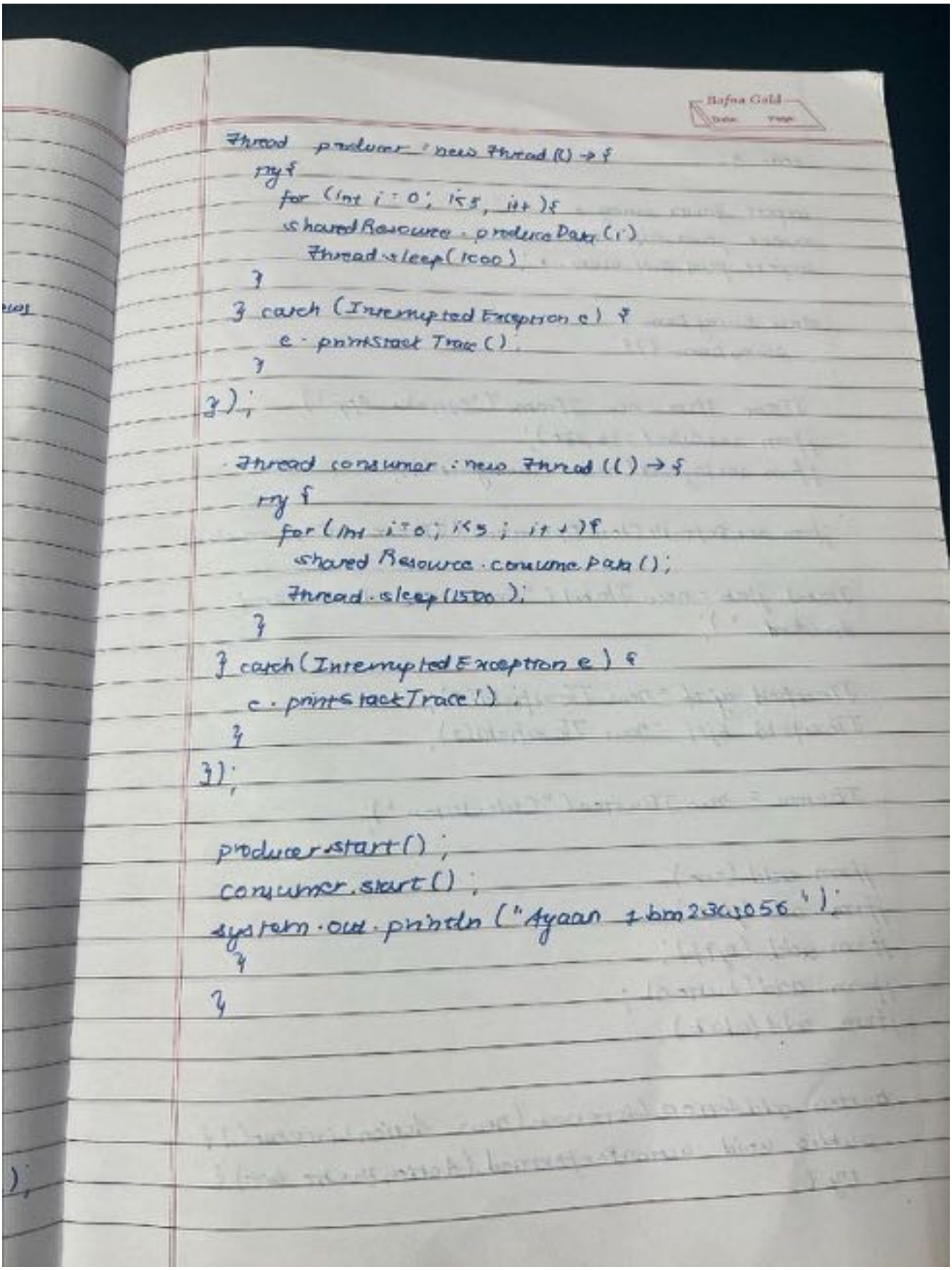



Program 10.2 :
Implement Inter-process Communication

Algorithm :



```
code : IPC

class SharedResource {
  private int data;
  private boolean isDataAvailable = false;

  public synchronized void produceData(int data) throws
  InterruptedException {
    while (isDataAvailable) {
      wait();
    }
    this.data = data;
    System.out.println("Put : " + data);
    isDataAvailable = true;
    notify();
  }

  public synchronized void consumeData() throws
  InterruptException {
    while (!isDataAvailable) {
      wait();
    }
    System.out.println("Get : " + data);
    isDataAvailable = false;
    notify();
  }
}

public class IPCDemo {
  public static void main(String[] args) {
    SharedResource sharedResource = new SharedResource();
```



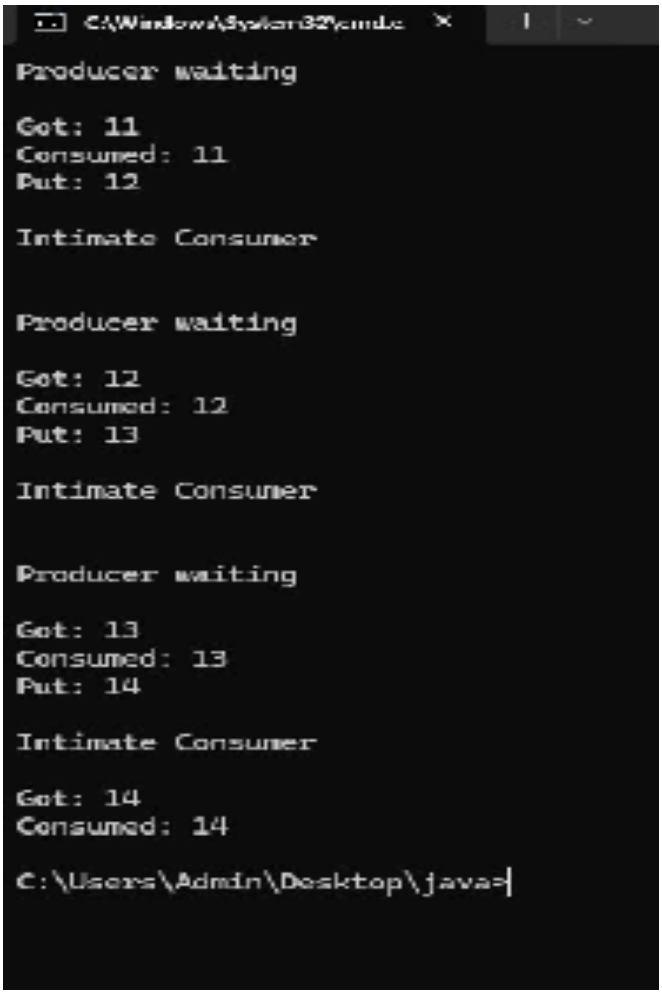
```
Thread producer = new Thread(() -> {
    try {
        for (int i = 0; i < 5; i++) {
            sharedResource.produceData(i);
            Thread.sleep(1000);
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
});

Thread consumer = new Thread(() -> {
    try {
        for (int i = 0; i < 5; i++) {
            sharedResource.consumeData();
            Thread.sleep(1500);
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
});

producer.start();
consumer.start();
System.out.println("Ayaan 1bm23cs056");
    }
}
```

Code :

```
class SharedResource {
   private int data;
   private boolean isDataAvailable = false;


   public synchronized void produceData(int data) throws InterruptedException {
      while (isDataAvailable) {
         wait();
      }
      this.data = data;
      System.out.println("Put: " + data);
      isDataAvailable = true;
      notify();
   }


   public synchronized void consumeData() throws InterruptedException {
      while (!isDataAvailable) {
         wait();
      }
      System.out.println("Get: " + data);
      isDataAvailable = false;
      notify();
   }
}

public class IPCDemo {
   public static void main(String[] args) {
      SharedResource sharedResource = new SharedResource();


      Thread producer = new Thread(() -> {
         try {
            for (int i = 0; i < 5; i++) {
               sharedResource.produceData(i);
               Thread.sleep(1000);
            }
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      });
```

```
        Thread consumer = new Thread(() -> {
            try {
                for (int i = 0; i < 5; i++) {
                    sharedResource.consumeData();
                    Thread.sleep(1500);
                }
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        });

        producer.start();
        consumer.start();
        System.out.println("Ayaan 1bm23cs056");
    }
}
```

OutPut:


C:\Windows\System32\cmd.e
Producer waiting
Got: 11
Consumed: 11
Put: 12
Intimate Consumer
Producer waiting
Got: 12
Consumed: 12
Put: 13
Intimate Consumer
Producer waiting
Got: 13
Consumed: 13
Put: 14
Intimate Consumer
Got: 14
Consumed: 14
C:\Users\Admin\Desktop\java>