VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SANJANA SRINIVAS(1BM23CS301)

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 **SANJANA SRINIVAS(1BM23CS301),** 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

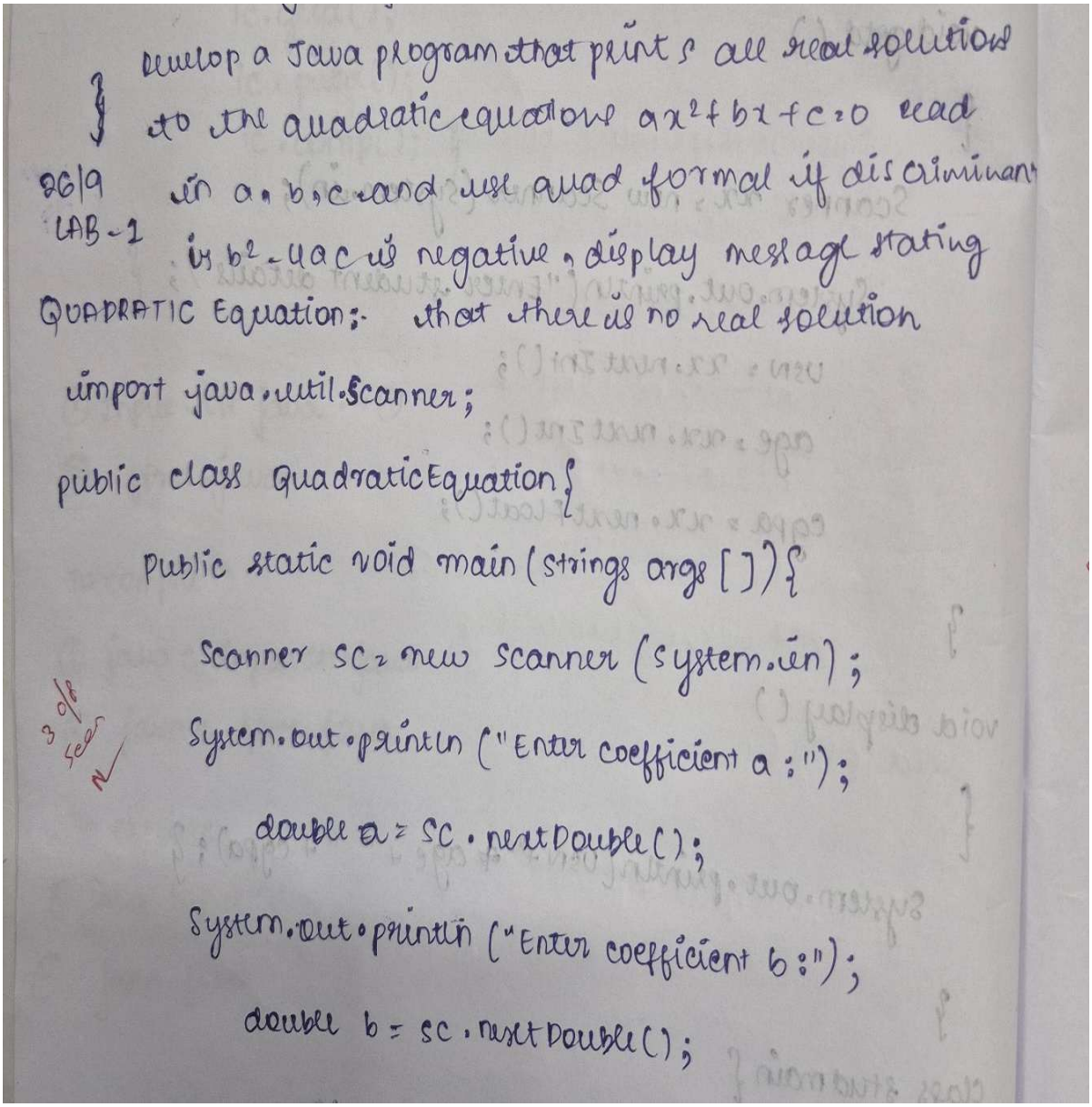
Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

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.



Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$ read in a, b, c and use quad formal if discriminant is b^2-4ac is negative, display message stating that there is no real solution

26/9
LAB-1

QUADRATIC Equation:-

```
import java.util.Scanner;

public class QuadraticEquation{

    Public static void main(strings args[]){

        Scanner sc= new Scanner (system.in);

        System.out.println ("Enter coefficient a :");

            double a= sc.nextDouble();

        System.out.println ("Enter coefficient b :");

            double b = sc.nextDouble();
```

3 0/8 see N

```
        System.out.println("Enter coefficient c:");
        double c = sc.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("Roots are real and distinct");
        System.out.println("Root 1 = " + root1);
        System.out.println("Root 2 = " + root2);
    }
    else if (discriminant == 0) {
            double root = -b / (2*a);
        System.out.println("Roots are real and equal");
        System.out.println("Root = " + root);
    }
    else {
        System.out.println("There are no real solutions");
    }
  }
}
```

Output :-

Enter the coefficient a :

1

Enter the coefficient b:-

-3

Enter the coefficient c:-

2

the equation has two real & distinct roots;-

Root 1 = 2.0

Root 2 = 1.0

Enter the coefficient a:

1

Enter the coefficient b:

-6

Enter the coefficient c:

9

Roots are real and equal

Root = 3.0

26/9/24

Enter the coefficient a:

1

Enter the coefficient b:-

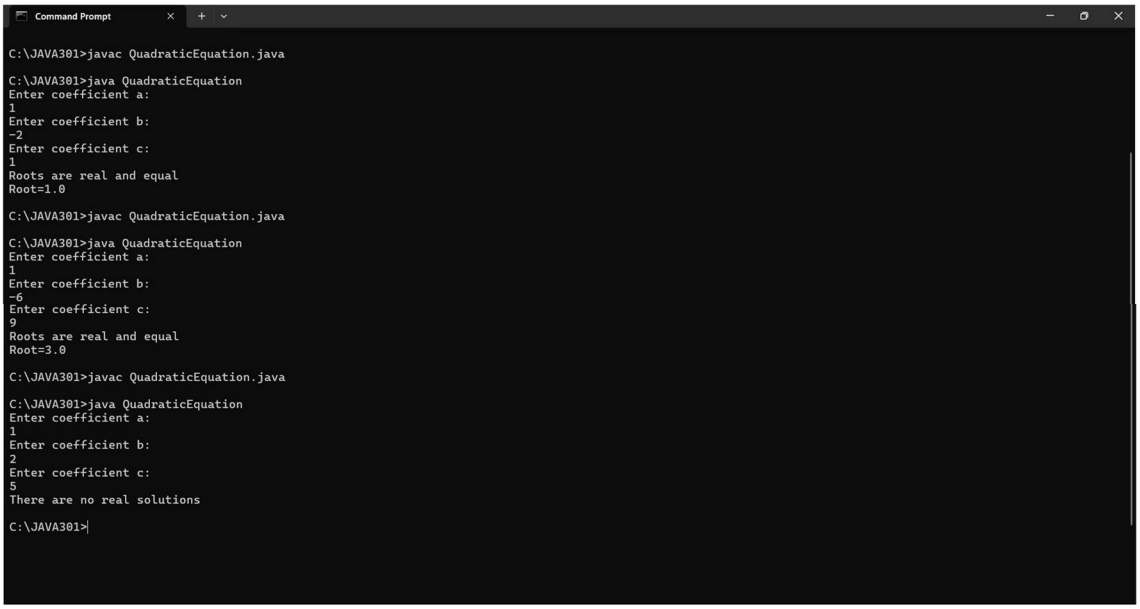
2

There are no real solutions.

Enter the coefficient c:

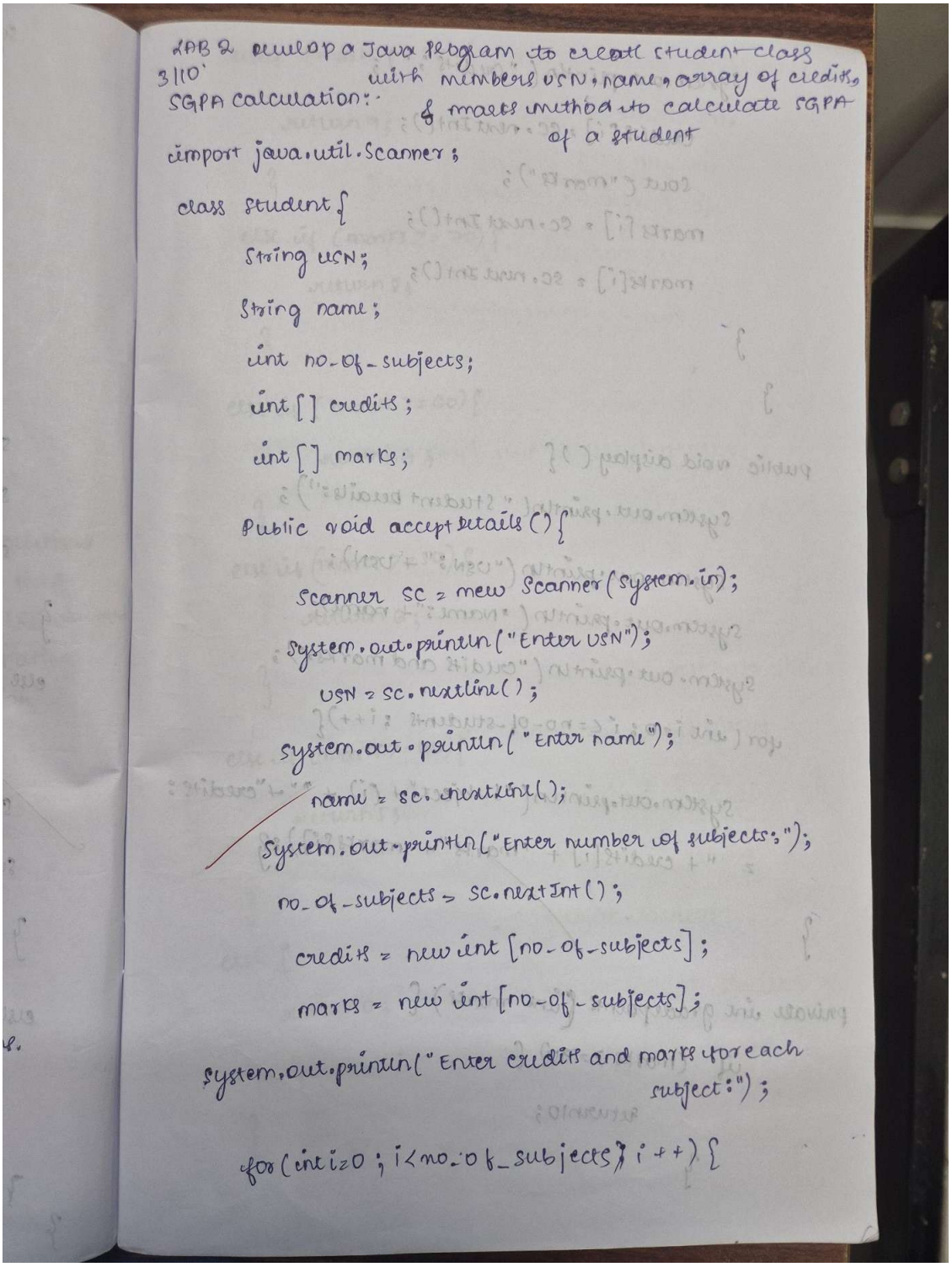
5

```
import java.util.Scanner;

class QuadraticEquation {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter coefficient a:");
        double a = sc.nextDouble();
        System.out.println("Enter coefficient b:");
        double b = sc.nextDouble();
        System.out.println("Enter coefficient c:");
        double c = sc.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("Roots are real and distinct");
            System.out.println("Root 1=" + root1);
            System.out.println("Root 2=" + root2);
        } else if (discriminant == 0) {
            double root = -b / (2 * a);
            System.out.println("Roots are real and equal");
            System.out.println("Root=" + root);
        } else {
            System.out.println("There are no real solutions");
        }
    }
}
```

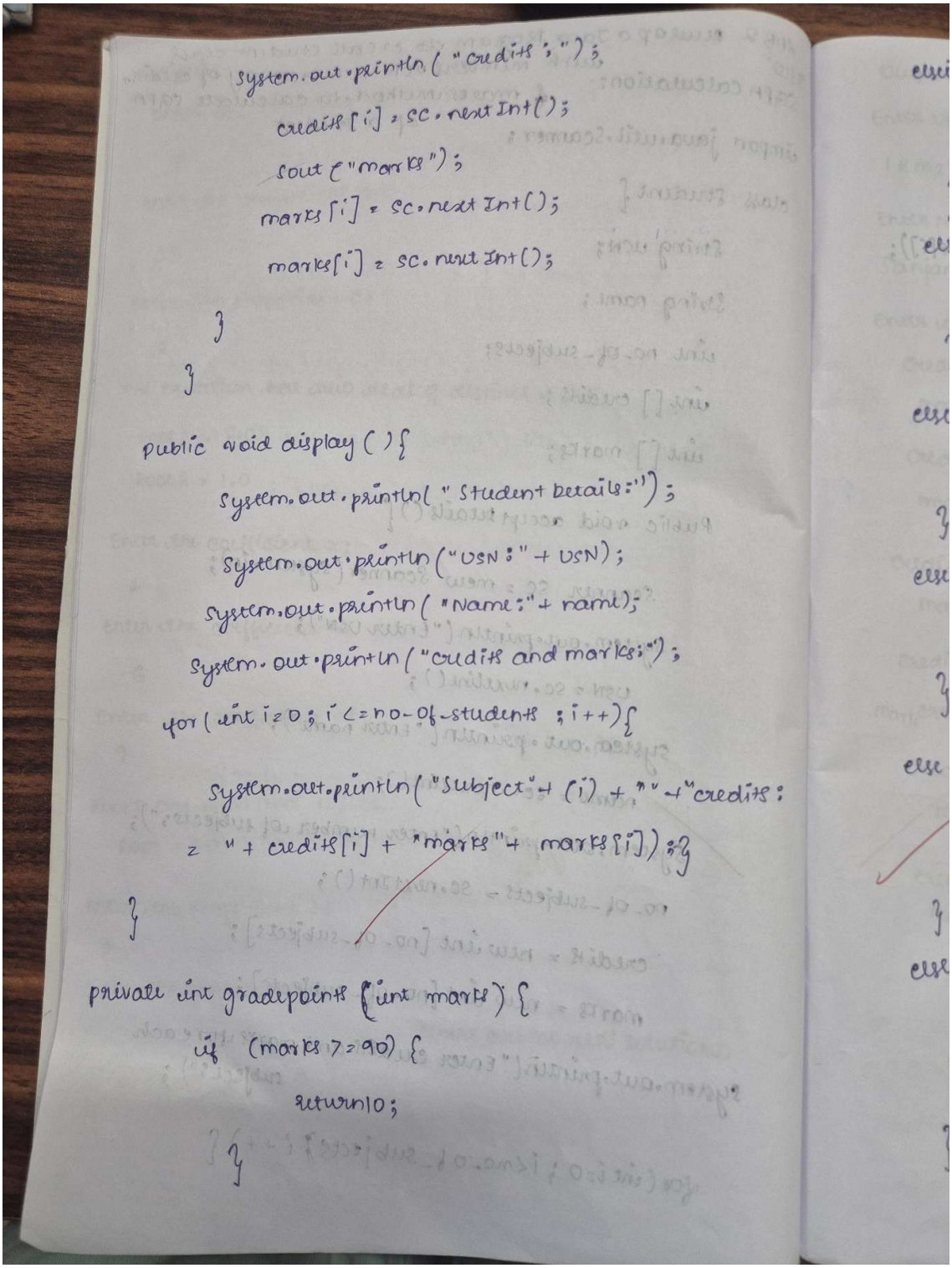

Command Prompt
C:\JAVA301>javac QuadraticEquation.java
C:\JAVA301>java QuadraticEquation
Enter coefficient a:
1
Enter coefficient b:
-2
Enter coefficient c:
1
Roots are real and equal
Root=1.0
C:\JAVA301>javac QuadraticEquation.java
C:\JAVA301>java QuadraticEquation
Enter coefficient a:
1
Enter coefficient b:
-6
Enter coefficient c:
9
Roots are real and equal
Root=3.0
C:\JAVA301>javac QuadraticEquation.java
C:\JAVA301>java QuadraticEquation
Enter coefficient a:
1
Enter coefficient b:
2
Enter coefficient c:
5
There are no real solutions
C:\JAVA301>

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.

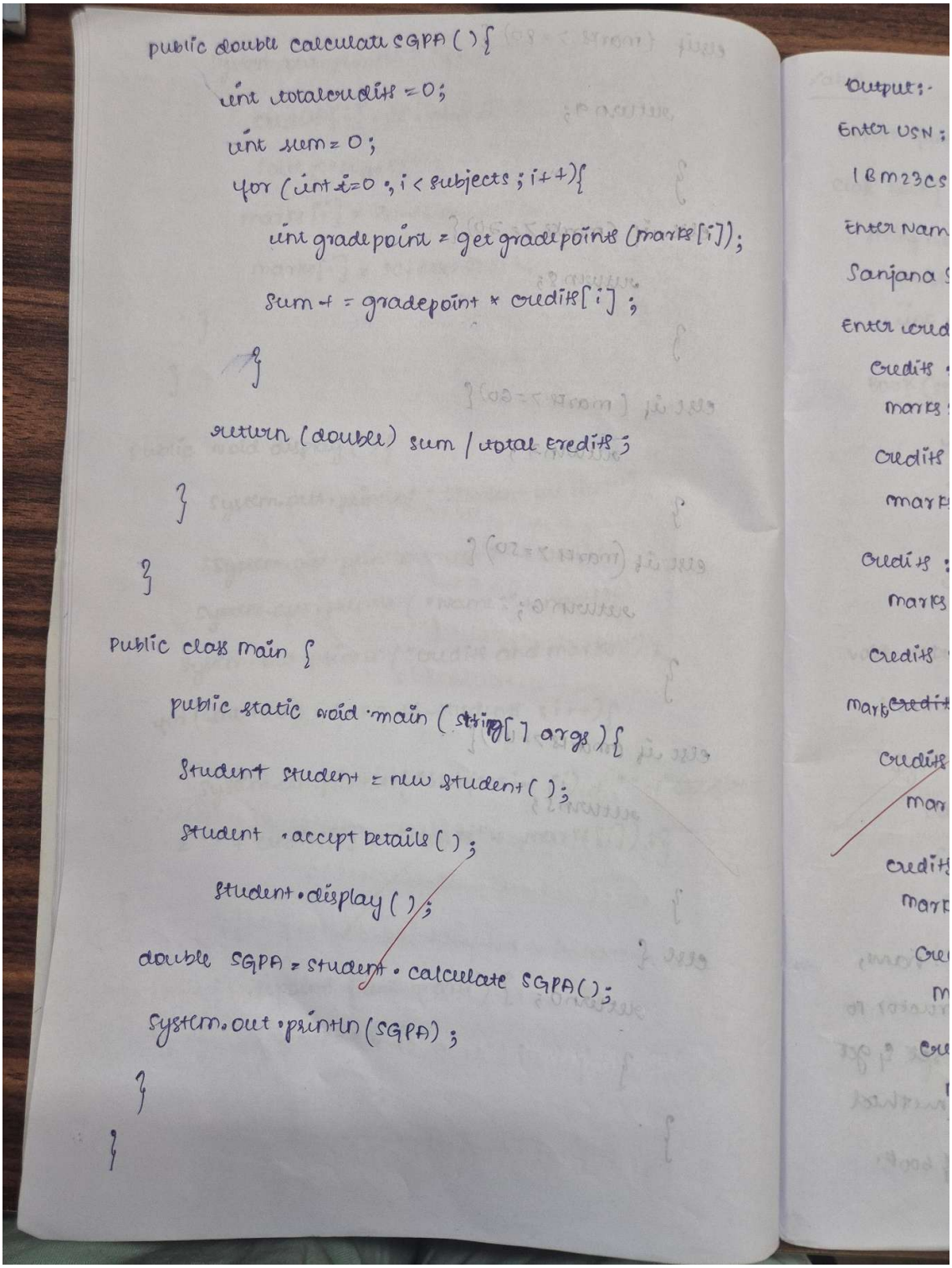


```
LAB 2 Develop a Java program to create student class
3|10                with members USN, name, array of credits,
SGPA calculation:-   & marks method to calculate SGPA
                                of a student
import java.util.Scanner;

class Student {
    String USN;
    String name;
    int no_of_subjects;
    int[] credits;
    int[] marks;
Public void acceptDetails() {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter USN");
      USN = sc.nextLine();
    System.out.println("Enter name");
      name = sc.nextLine();
    System.out.println("Enter number of subjects:");
    no_of_subjects = sc.nextInt();
      credits = new int[no_of_subjects];
      marks = new int[no_of_subjects];
System.out.println("Enter credits and marks for each
                                    subject:");

for(int i=0; i<no_of_subjects; i++) {
```



```
        System.out.println("credits:");
            credits[i] = sc.nextInt();
            sout("marks");
            marks[i] = sc.nextInt();
            marks[i] = sc.nextInt();
        }
    }

public void display(){
        System.out.println("Student details:");
        System.out.println("USN:" + USN);
        System.out.println("Name:" + name);
        System.out.println("credits and marks:");
    for(int i=0; i<=no_of_students ; i++){
        System.out.println("Subject" + (i) + "" + "credits:
    = " + credits[i] + "marks" + marks[i]);}
}

privale int gradepoints (int marks) {
        if (marks >= 90) {
            return 10;
        }
```

```
elseif (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 double calculateSGPA(){
    int totalcredits = 0;
    int sum = 0;
    for(int i=0; i<subjects; i++){
        int gradepoint = getgradepoints(marks[i]);
        sum += gradepoint * credits[i];
    }
    return (double) sum / totalcredits;
  }
}

Public class main {
    public static void main(String[] args){
    Student student = new Student();
    Student.acceptDetails();
        Student.display();
  double SGPA = student.calculateSGPA();
  System.out.println(SGPA);
  }
}
```

Output:-

Enter USN :

1BM23CS

Enter Nam

Sanjana

Enter cred

Credits

marks

Credits

mark

Credits

marks

Credits

Mark Credit

Credits

mar

Credit

mark

Cre

m

Cre

Output:-

Enter USN:

1BM23CS301

Enter Name:-

Sanjana Srinivas

Enter credits and marks for each subject:-

Credits : 4

marks : 96

Credits : 4

marks : 93

Credits : 3

marks : 89

Credits : 3

marks ~~Credits~~ : 86

Credits : 3

marks : 92

Credits : 1

marks : 95

Credits : 1

marks : 95

Credits : 1

marks = 95

Student Details

USN : 1BM23CS301

Name : Sanjana Srinivas

Credits and marks :-

Subject 1 : credits = 4 marks = 96

Subject 2 : credits = 4 marks = 93

subject 3 : credits = 3 marks = 89

Subject 4 : credits = 3 marks = 86

Subject 5 : credits = 3 marks = 92

Subject 6 : credit = 1 marks = 95

Subject 7 : credit = 1 marks = 95

Subject 8 : credit = 1 marks = 95

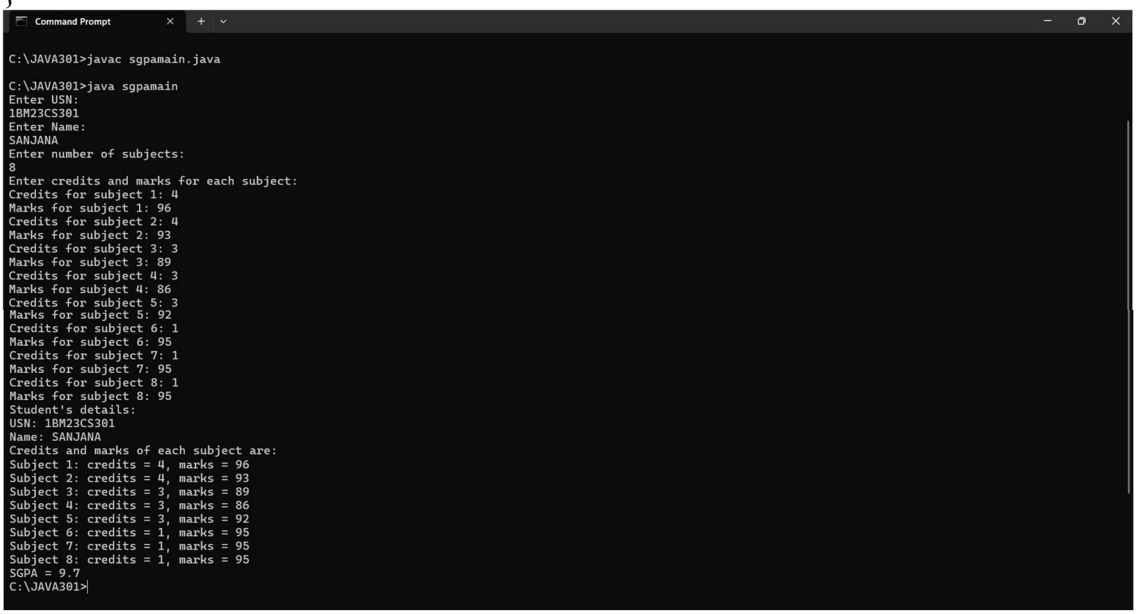
9.7

3/10/24

```
import java.util.Scanner;

class Student_SGPA {

    String usn;
    String name;
    int n;
    int[] credits;
    int[] marks;

    public void acceptDetails() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter USN:");
        usn = sc.next();
        System.out.println("Enter Name:");
        name = sc.next();
        System.out.println("Enter number of subjects:");
        n = sc.nextInt();
        credits = new int[n];
        marks = new int[n];
        System.out.println("Enter credits and marks for each subject:");
        for (int i = 0; i < n; i++) {
            System.out.print("Credits for subject " + (i + 1) + ": ");
            credits[i] = sc.nextInt();
            System.out.print("Marks for subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
        }
    }

    public void display() {
```

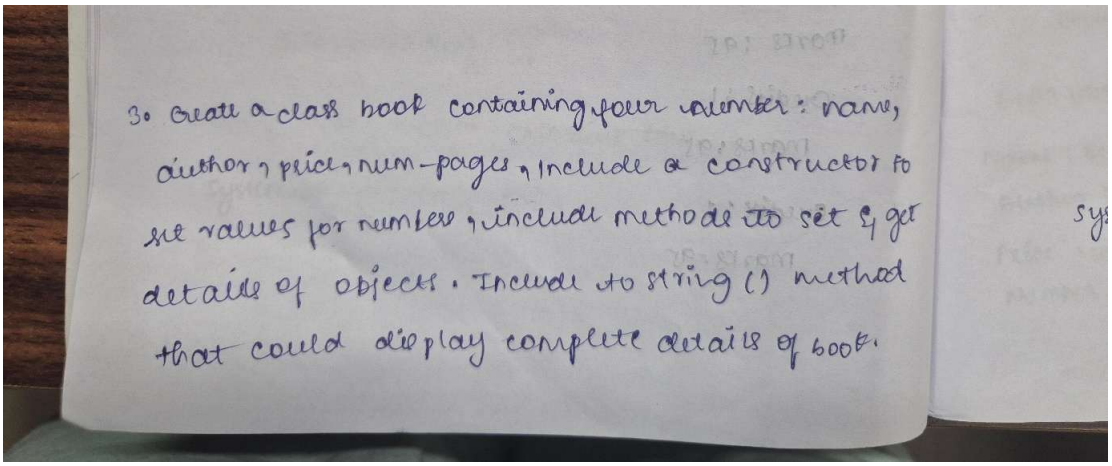
```
        System.out.println("Student's details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Credits and marks of each subject are:");
        for (int i = 0; i < n; i++) {
            System.out.println("Subject " + (i + 1) + ": credits = " + credits[i] + ", marks = " +
marks[i]);
        }
    }

    private int getGradePoint(int mark) {
        if (mark >= 90) {
            return 10;
        } else if (mark >= 80) {
            return 9;
        } else if (mark >= 70) {
            return 8;
        } else if (mark >= 60) {
            return 7;
        } else if (mark >= 50) {
            return 6;
        } else if (mark >= 40) {
            return 5;
        } else {
            return 0;
        }
    }

    public double calculateSGPA() {
        int totalCredits = 0;
        int sum = 0;
```

```
        for (int i = 0; i < n; i++) {
            int gradePoint = getGradePoint(marks[i]);
            sum += gradePoint * credits[i];
            totalCredits += credits[i];
        }
        return (double) sum / totalCredits;
    }
}

class sgpamain {

    public static void main(String[] args) {
        Student_SGPA student = new Student_SGPA();
        student.acceptDetails();
        student.display();
        double SGPA = student.calculateSGPA();
        System.out.printf("SGPA = " + SGPA);
    }
}
```



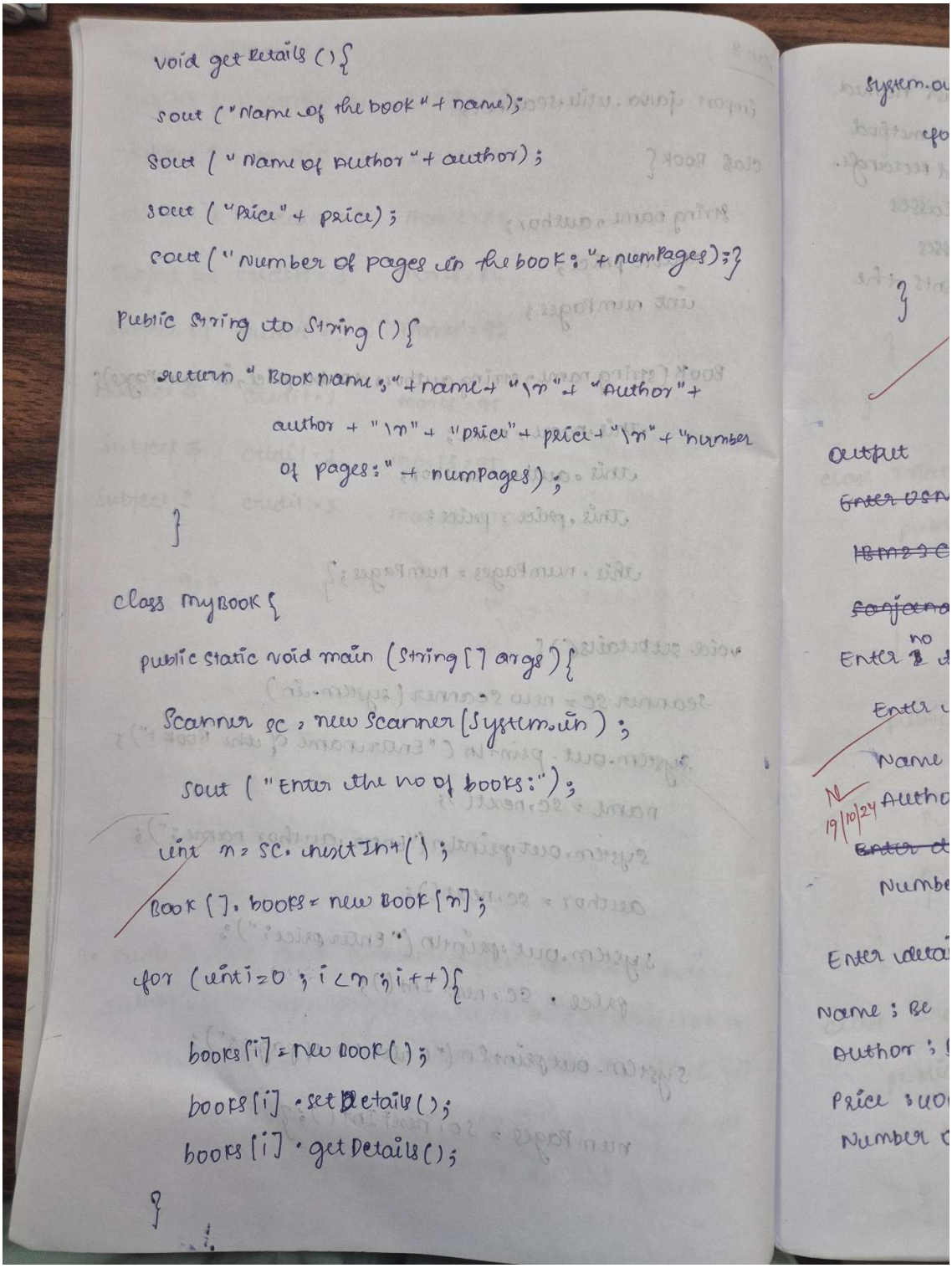
Program 3

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



Lab 3

```
import java.util.Scanner;

Class Book{

    String name, author;
    double price;
    int numPages;

    BOOK (string name, string author, double price, int numPages){
        this.name = name;
        this.author = author;
        This.price = price;
        this.numPages = numPages;}

void setDetails(){
    Scanner sc = new Scanner (System.in)
    System.out.println ("Enter name of the Book:");
        name = sc.next();
        System.out.println ("Enter author name:");
        author = sc.next();
        System.out.println ("Enter price:");
        price = sc.nextInt();
    System.out.println ("Enter no of pages:");
        numPages = sc.nextInt();}
```



```
void getDetails(){
sout("Name of the book"+name);
sout("Name of Author"+author);
sout("Price"+price);
sout("Number of pages in the book:"+numPages);}

Public String toString(){
  return "Book name:"+name+"\n"+"Author"+
         author+"\n"+"price"+price+"\n"+"number
         of pages:"+numPages);
}

class myBook{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    sout("Enter the no of books:");
    int n = sc.nextInt();
    Book[]. books = new Book[n];
    for(int i=0;i<n;i++){
      books[i] = new Book();
      books[i].setDetails();
      books[i].getDetails();
    }
```

Output

Enter no

Name

Author

Number

Enter details

Name : Be

Author :

Price :

Number

19/10/24

```
System.out.println("\n All Book Details");
for (Book Books : Books){
    Sout (Book);
}
}
```

Output

~~Enter USN & Name:~~

~~1BM23CS301~~

~~Sanjana~~

Enter no ~~subject~~ of Books : 1

Enter details for Book 1

Name :- Silent Patient

Author : Alexander

Price : 500

Number of pages = 400

19/10/24

Enter details Book 2:

Name : Be

Author : B

Price : 400

Number of pages : 200

Output:-

Book Name: Silent Patient

Author : Alexander

Price : 500

Pages : 400

Book Name : Be

Author : Ab

Price : 400

Page : 200

```
import java.util.Scanner
abstract class account{
    String customername,accountnumber;
    double balance;
    account(String customername,String accountnumber,double initialbalance){
        this.customername=customername;
        this.accountnumber=accountnumber;
        this.balance=initialbalance;
    }
    abstract void deposit(double amount);
    abstract void displaybalance();
    abstract void withdraw(double amount);
}

class savacc extends account{
    double interestrate;
    savingsaccount(String customername,String accountnumber,double initialbalance){
        super(customername,accountnumber,initialbalance);
        this.interestrate=interestrate;
    }
    void displaybalance(){
    System.out.println("savings balance:"+balance);
    }
    void deposit(double amount){
    balance+=amount;
    }

    void withdraw(double amount){
        if (amount<=balance){
            balance==amount;
```

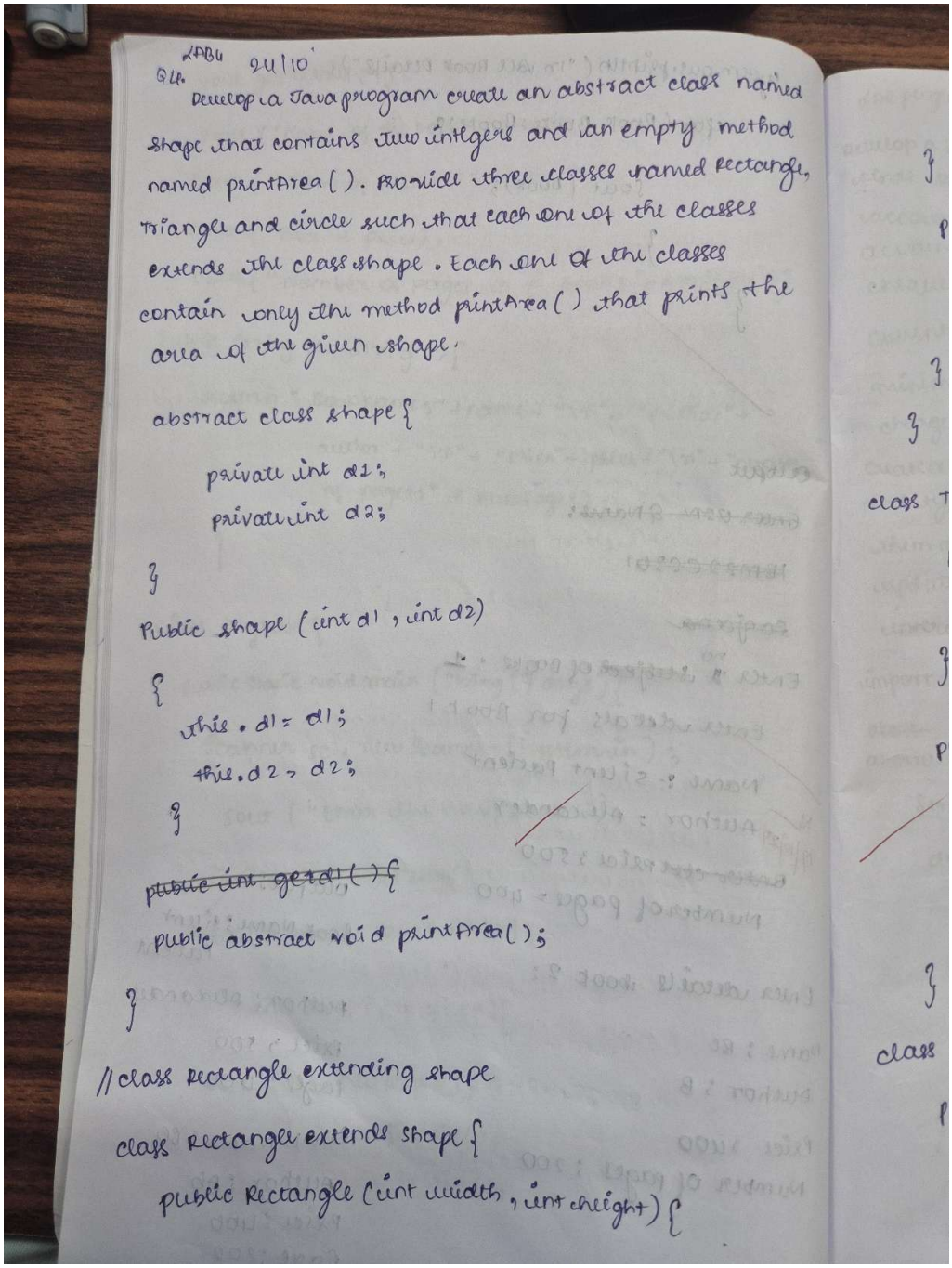
```
        }
    }

    void computeanddepositinterest(){
        balance+=balance.interestrate/100;
    }
}
class curacc extends account{
static final double min_balance=1000, service_charge=50;
curacc(String customername,String accountname,double initialbalance){
    super(customername,accountname,initialbalance);
    }

void deposit(double amount){
balance+=amount;
}

void displaybalance(){
System.out.println("current balance:"+balance);
}
void withdraw(double amount){
    if (amount<=balance){
        balance=amount;
        if (balance<min_balance)
            balance-=service_charge;
}}}

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

```
    System.out.println("enter account type (savings/current)");
    String type= sx.nextLine();
    System.out.println("enter account name");
    String name= sx.nextLine();
    System.out.println("enter account number");
    String number= sx.nextLine();
account account;
if (type.equals("savings")){
    System.out.println("initial balance and interest rate");
    account= new savacc(name,number,sx.nextDouble(),sx.nextDouble());
     }
else{
    System.out.println("initial  balance:");
    account= new curacc(name,number,sx.nextDouble());
}
while(true){
System.out.println("1. deposit 2.display balance 3.withdraw 4. interest 5.exit");
int choice=sx.nextInt();
switch(choice){
    case 1: account.deposit(sx.nextDouble());
        break;
    case 2: account.displaybalance();
        break;
    case 3: account.withdraw(sx.nextDouble());
        break;
    case 4: if(account instance of savacc)
        ((savacc) account).computeanddepositinterest();
        break;
    case 5: return;
}}}}
```

```
C:\JAVA301>javac bookdemo.java

C:\JAVA301>java bookdemo
Enter the number of books: 2
Enter Name:silent_patient
Enter Author:alexander
Enter Pages:300
Enter Price:500

Enter Name:anne_frank
Enter Author:anne_frank
Enter Pages:200
Enter Price:800

Name: silent_patient
Author: alexander
Pages: 300
Price: 500.0
Sanjana Srinivas 1BM23CS301
Name: anne_frank
Author: anne_frank
Pages: 200
Price: 800.0
Sanjana Srinivas 1BM23CS301

C:\JAVA301>
```

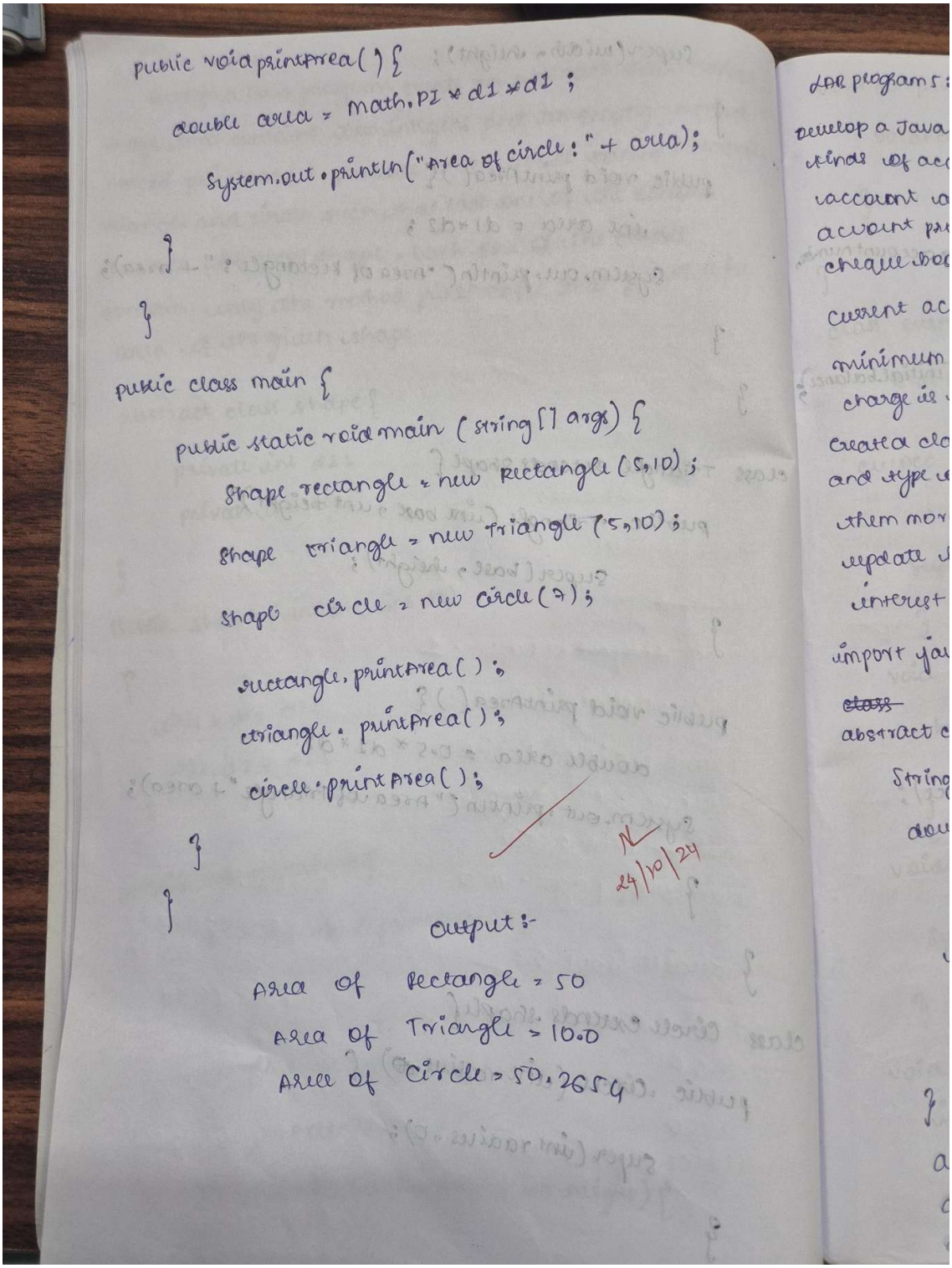
Program 4

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.



```
Q4. ADA 24/10
Develop a Java program create an abstract class named
shape that contains two intlgers 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.

abstract class shape {
    private int d1;
    private int d2;
}
Public shape (int d1 , int d2)
{
  this . d1 = d1;
  this.d2 = d2;
}
public int getd1( ){
public abstract void printArea( );
}
// class Rectangle extending shape
class Rectangle extends shape {
  public Rectangle (int width , int cheight) {
```

```
        super(width, height);
    }
    public void printArea(){
        int area = d1*d2;
        System.out.println("Area of Rectangle: " + area);
    }
}
class Triangle extends Shape{
    public Triangle(int base, int height){
        super(base, height);
    }
    public void printArea(){
        double area = 0.5 * d1 * d2;
        System.out.println("Area of Triangle" + area);
    }
}
class Circle extends Shape{
    public Circle(int radius){
        super(int radius, 0);
    }
```



```
public void printArea() {
    double area = Math.PI * d1 * d1;
    System.out.println("Area of circle: " + area);
}
}

public class main {
    public static void main (String[] args) {
        Shape rectangle = new Rectangle(5,10);
        Shape triangle = new Triangle(5,10);
        Shape circle = new Circle(7);

        rectangle.printArea();
        triangle.printArea();
        circle.printArea();
    }
}
```

24/10/24

output:-

Area of Rectangle = 50
Area of Triangle = 10.0
Area of Circle = 50.2654

```
import java.util.Scanner;

abstract class shape {
```

```
    int d1;
    int d2;

    public shape(int d1, int d2) {
        this.d1 = d1;
        this.d2 = d2;

    }

    public abstract void printarea();
}

class rectangle extends shape {

    public rectangle(int width, int height) {
        super(width, height);
    }

    public void printarea() {
        int area = d1 * d2;
        System.out.println("area of rectangle:" + area);

    }
}

class triangle extends shape {

    public triangle(int base, int height) {
        super(base, height);
```

```
    }

    public void printarea() {
        double area = 0.5 * d1 * d2;
        System.out.println("area of triangle:" + area);

    }
}

class circle extends shape {

    public circle(int radius) {

        super( radius,0);
    }
    public void printarea() {
        double area = Math.PI * d1 * d1;
        System.out.println("area of circle:" + area);

    }
}
class mainshape {

    public static void main(String[] args) {
        shape rectangle = new rectangle(5, 10);
        shape triangle = new triangle(4, 5);
        shape circle = new circle(4);
        rectangle.printarea();
        triangle.printarea();
        circle.printarea();
```

```
    }
}
```

```
C:\JAVA301>javac mainshape.java

C:\JAVA301>java mainshape
area of rectangle:50
area of triangle:10.0
area of circle:50.265482457436691

C:\JAVA301>
```

Program 5

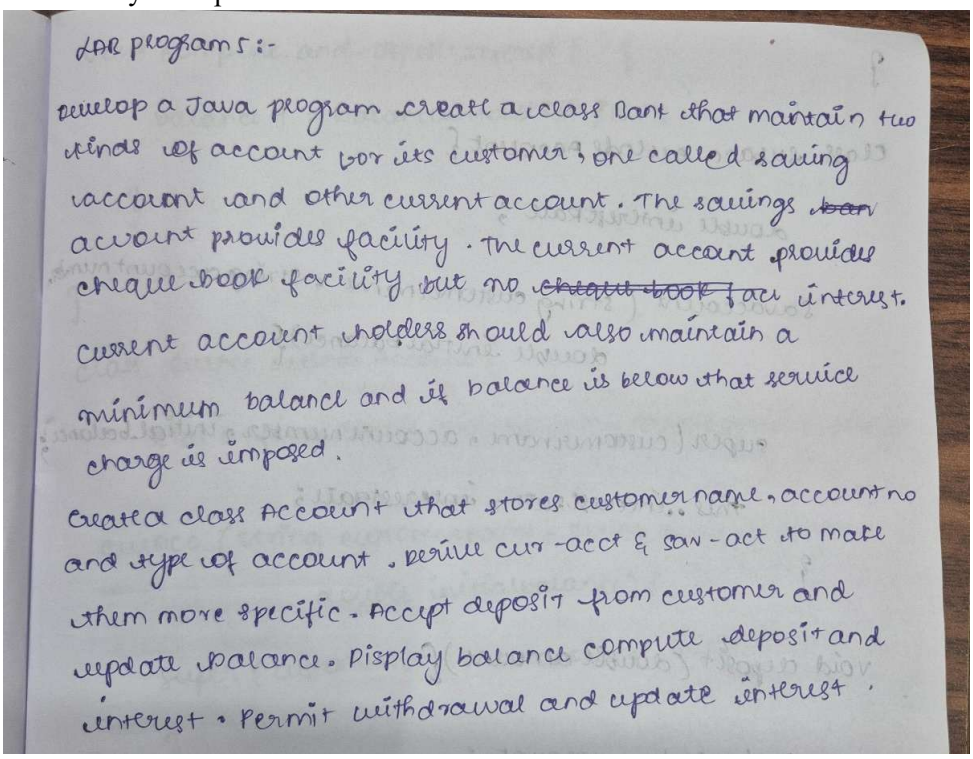
Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book 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

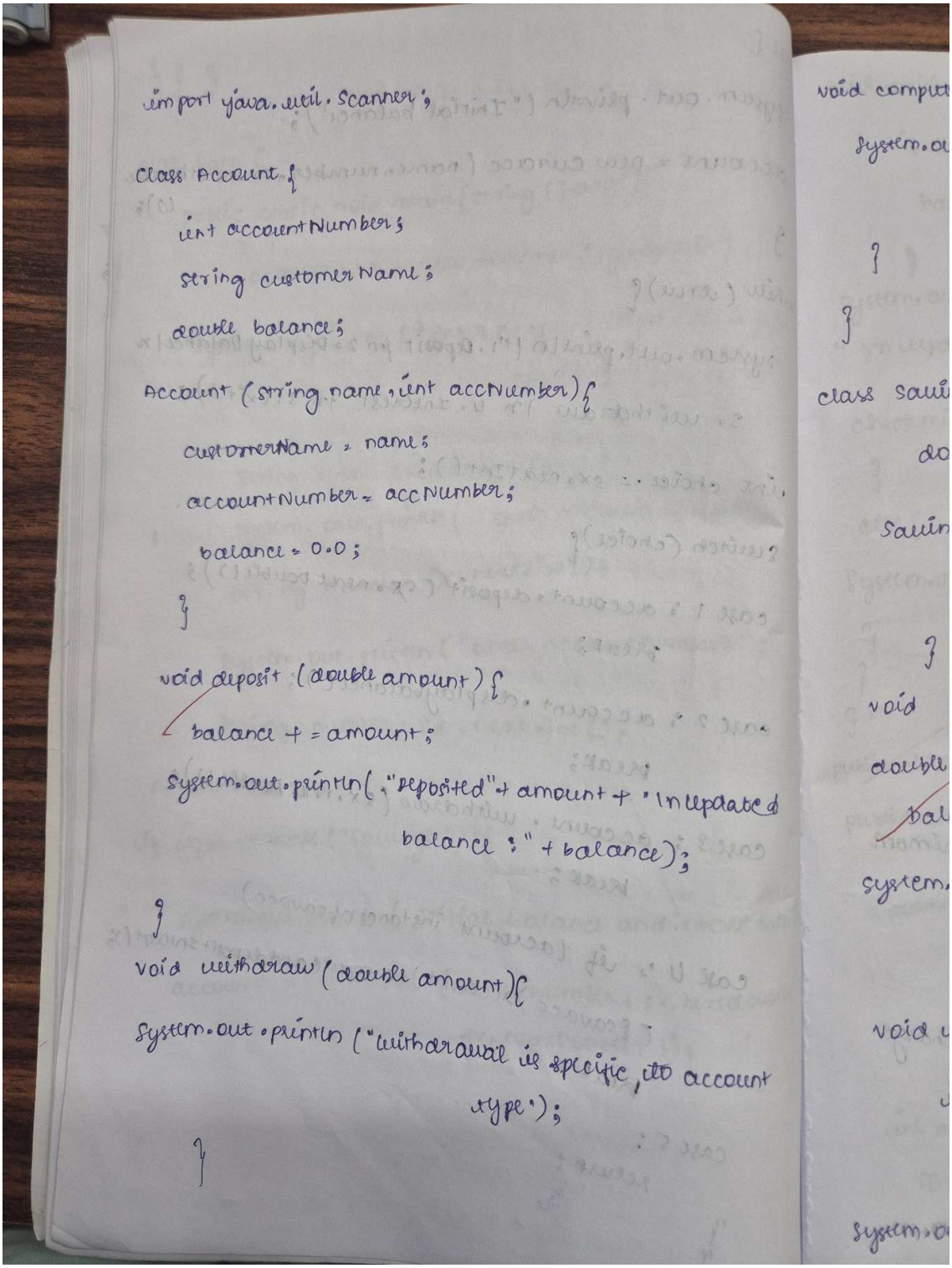
d) Permit withdrawal and update the balance Check for the minimum balance, impose penalty if necessary and update the balance.



DAA program 5 :-

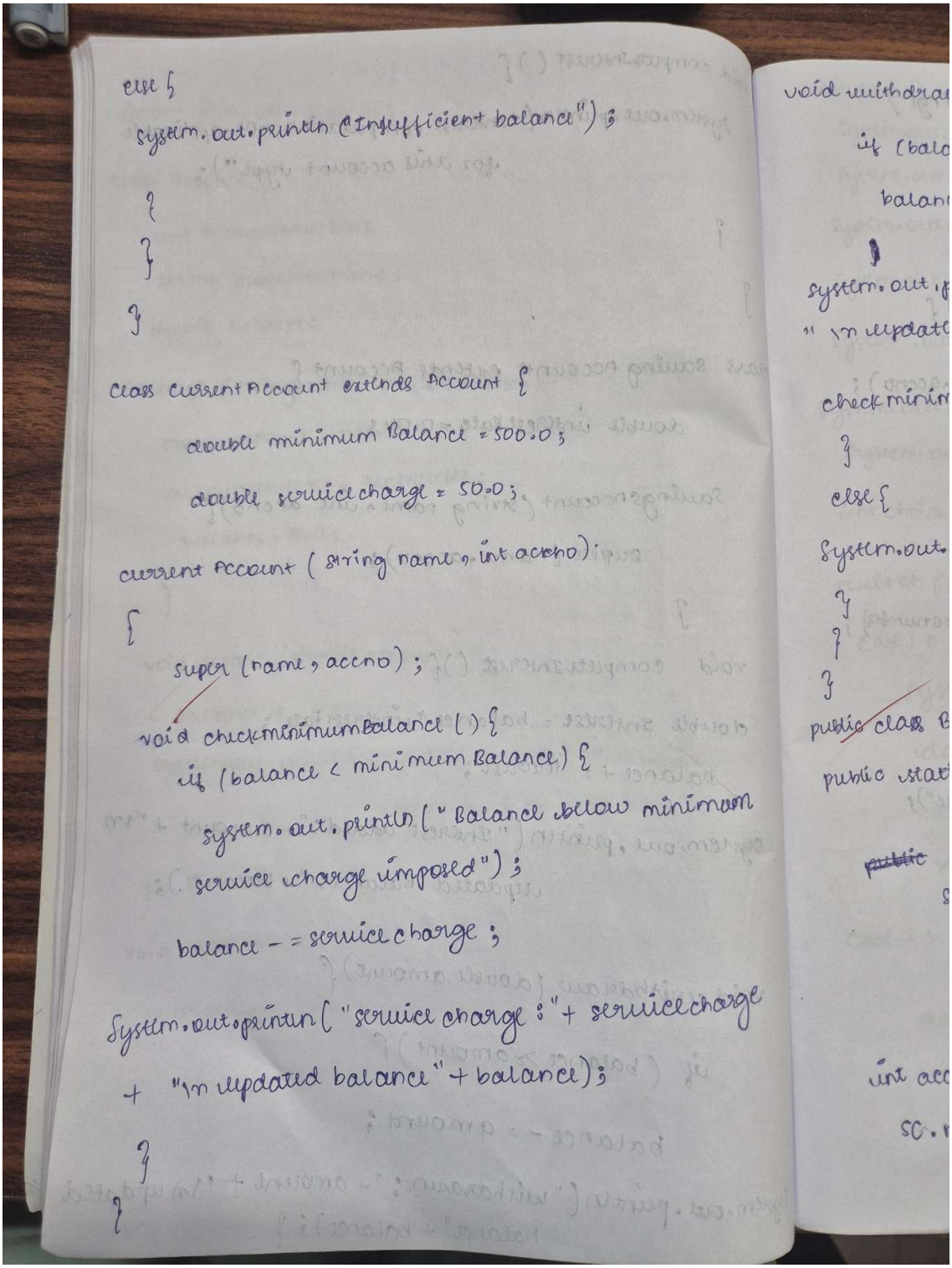
develop a Java program creat a class Bank that maintain two kinds of account for its customer, one called saving account and other current account. The savings ~~bank~~ account provides facility. The current account provides cheque book facility but no ~~chkbk book~~ facility interest. current account holders should also maintain a minimum balance and if balance is below that service charge is imposed.

Create a class Account that stores customer name, account no and type of account. Derive cur-acct & sav-act to make them more specific. Accept deposit from customer and update balance. Display balance compute deposit and interest. Permit withdrawal and update interest.

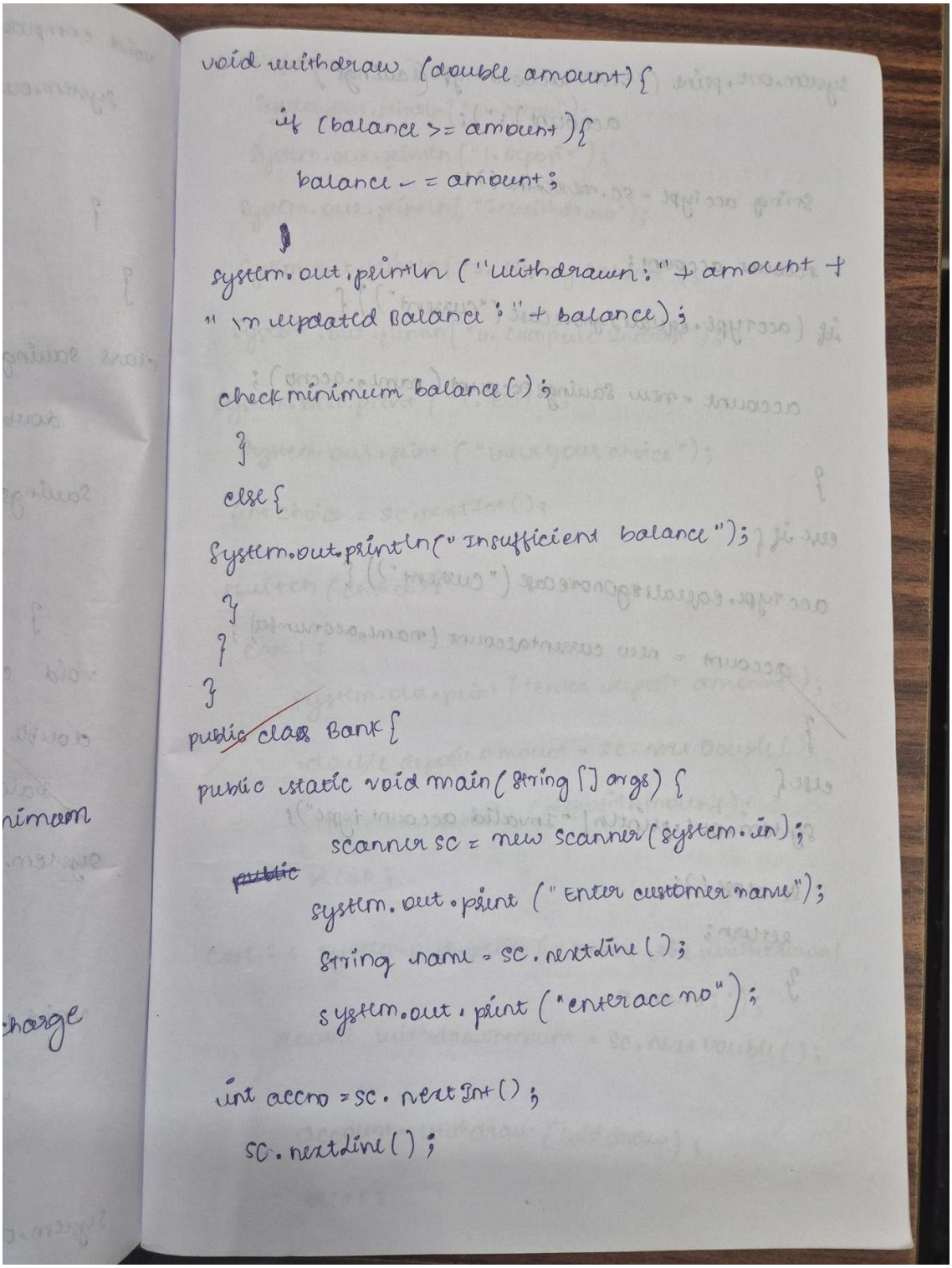


```
import java.util.Scanner;

Class Account {
    int accountNumber;
    String customerName;
    double balance;
Account (String name, int accNumber){
    customerName = name;
    accountNumber = accNumber;
    balance = 0.0;
}
void deposit (double amount) {
    balance += amount;
System.out.println("Deposited" + amount + "\nUpdated
                    balance :" + balance);
}
void withdraw (double amount){
System.out.println ("withdrawal is specific, to account
                    type");
}
```

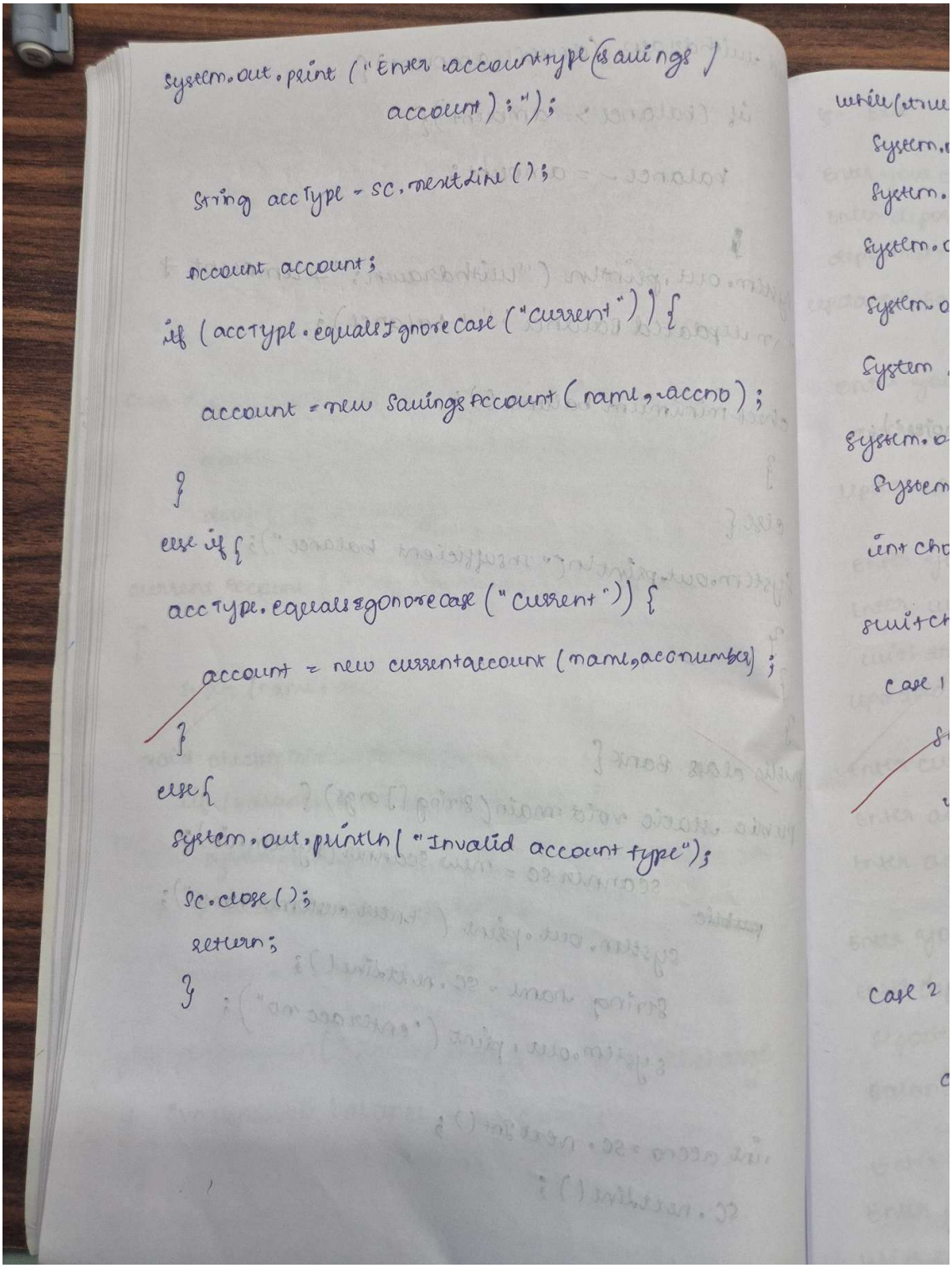
```
void computeInterest () {
    System.out.println ("Interest computation not applicable
                        for this account type");
    }
}

class SavingAccount extends Account {
    double interestRate = 0.04;

    SavingsAccount (String name, int accno) {
        super (name, accno);
    }
    void computeInterest () {
    double Interest = balance * interestRate;
    balance += interest;
System.out.println ("Interest added:" + account + "\n
                    updated Balance: " + balance);

void withdraw (double amount) {
    if (balance >= amount) {
        balance -= amount;
System.out.println ("withdrawn : " + amount + "\n updated
                    balance" + balance); }
```



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

class CurrentAccount extends Account {
     double minimumBalance = 500.0;
     double serviceCharge = 50.0;
CurrentAccount (String name, int accno);
{
    super (name, accno);
 void checkminimumBalance () {
    if (balance < minimumBalance) {
     system.out.println ("Balance below minimum
     service charge imposed");
     balance -= serviceCharge;
System.out.println ("service charge :" + serviceCharge
+ "\n updated balance" + balance);
 }
}
```

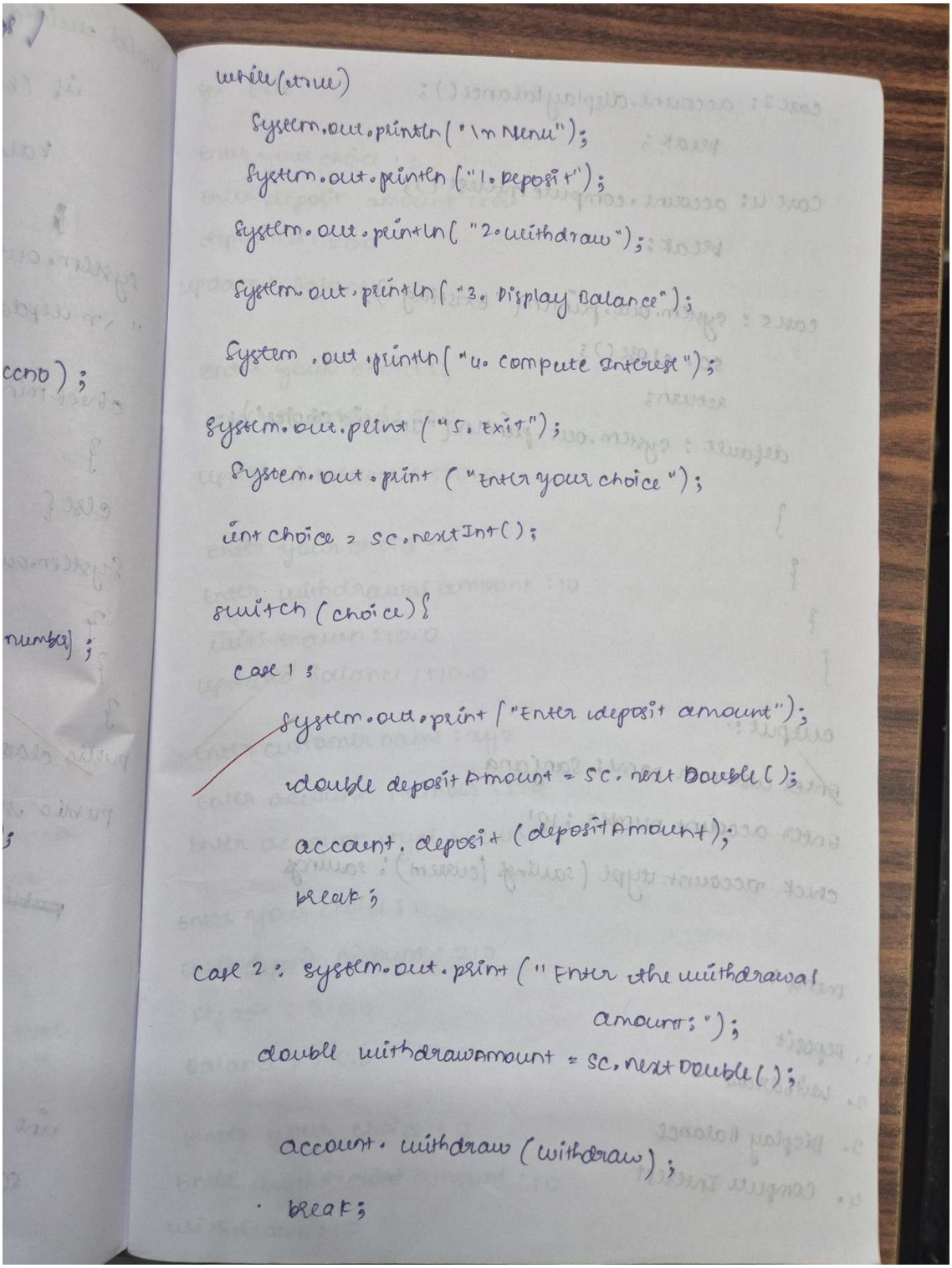


```
void withdraw (double amount){
    if (balance >= amount){
      balance -= amount;
    }
System.out.println ("withdrawn:" + amount +
" \n updated balance :" + balance);

checkminimum balance();
 }
 else{
System.out.println(" Insufficient balance");
 }
 }
}
public class Bank{
public static void main(String[] args) {
        scanner sc = new scanner(system.in);
public
        system.out.print ("Enter customer name");
        String name = sc.nextline();
        system.out.print ("enter acc no");

int accno = sc.nextInt();
 sc.nextline();
```

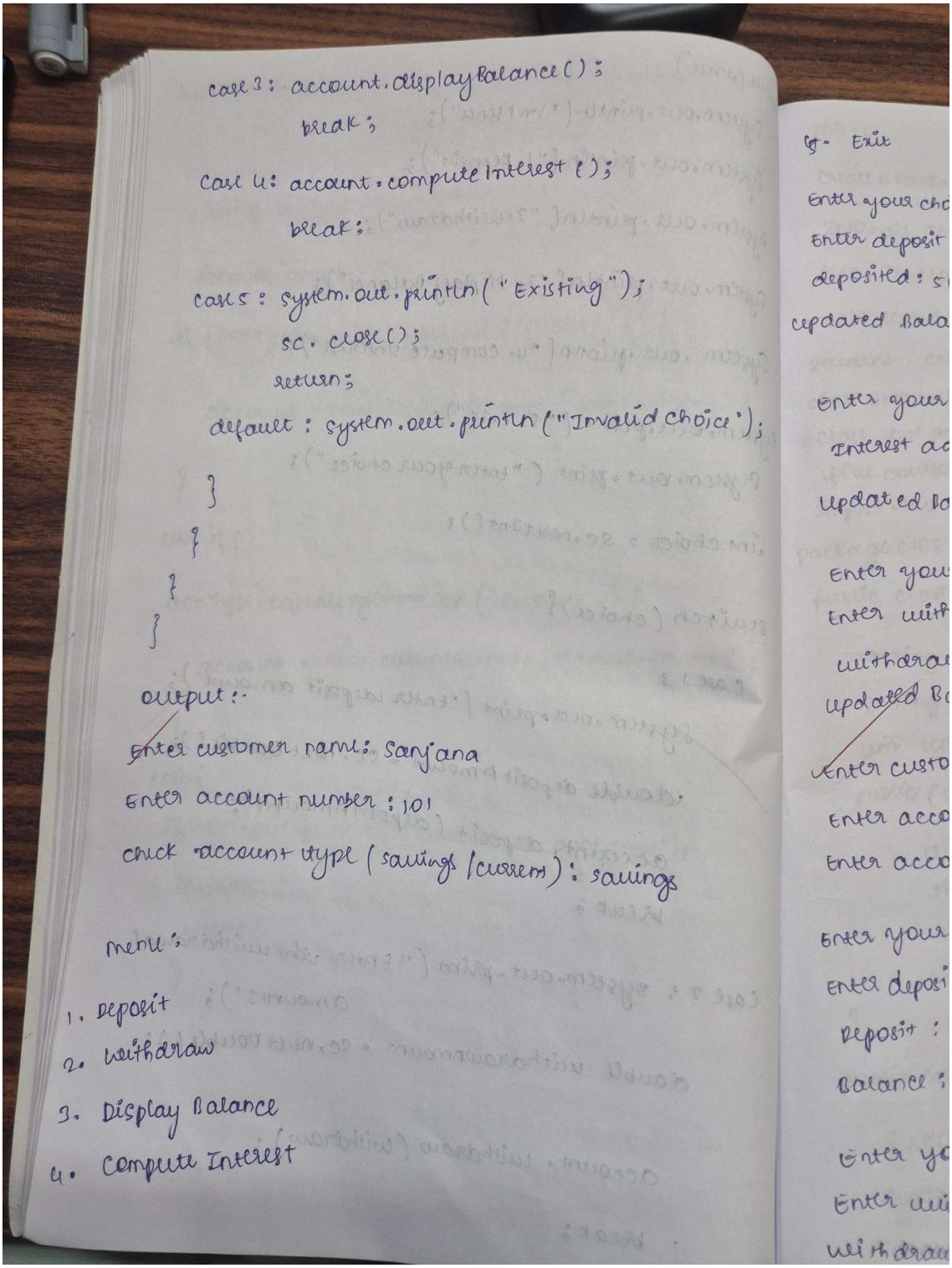


```
System.out.print ("Enter account type (savings / account):");

String acctype = sc.nextLine();

account account;

if (acctype.equalsIgnoreCase ("current")) {

    account = new SavingsAccount (name, accno);

}

else if {

acctype.equalsIgnorecase ("current")) {

    account = new currentaccount (name, accnumber);

}

else {

System.out.println ("Invalid account type");

sc.close();

return;

}
```

b



```
while(true)
    System.out.println("\n Menu");
    System.out.println("1. Deposit");
    System.out.println("2. withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. compute Interest");
System.out.println("5. Exit");
    System.out.print("Enter your choice");
    int choice = sc.nextInt();
switch(choice){
    case 1:
        System.out.print("Enter deposit amount");
        double depositAmount = sc.nextDouble();
        account.deposit(depositAmount);
        break;
case 2: System.out.print("Enter the withdrawal
                          amount:");
    double withdrawAmount = sc.nextDouble();
        account.withdraw(withdraw);
        break;
```



```
case 3: account.displayBalance();
        break;
case 4: account.computeInterest();
        break;
case 5: system.out.println("Existing");
        sc.close();
        return;
default : system.out.println("Invalid choice");
}
}
}
}
```

output :-

Enter customer name: Sanjana

Enter account number : 101

Chick account type (savings/current): savings

menu :

1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest

5. Exit

Enter your cho

Enter deposit

deposited : 5

updated Bala

Enter your

Interest ac

Updated Ba

Enter you

Enter with

withdrau

Updated B

Enter custo

Enter acco

Enter acco

Enter your

Enter depos

Deposit :

Balance :

Enter yo

Enter wi

withdrau

4. Exit

Enter your choice : 1
Enter deposit amount : 500
deposited : 500
updated Balance : 500

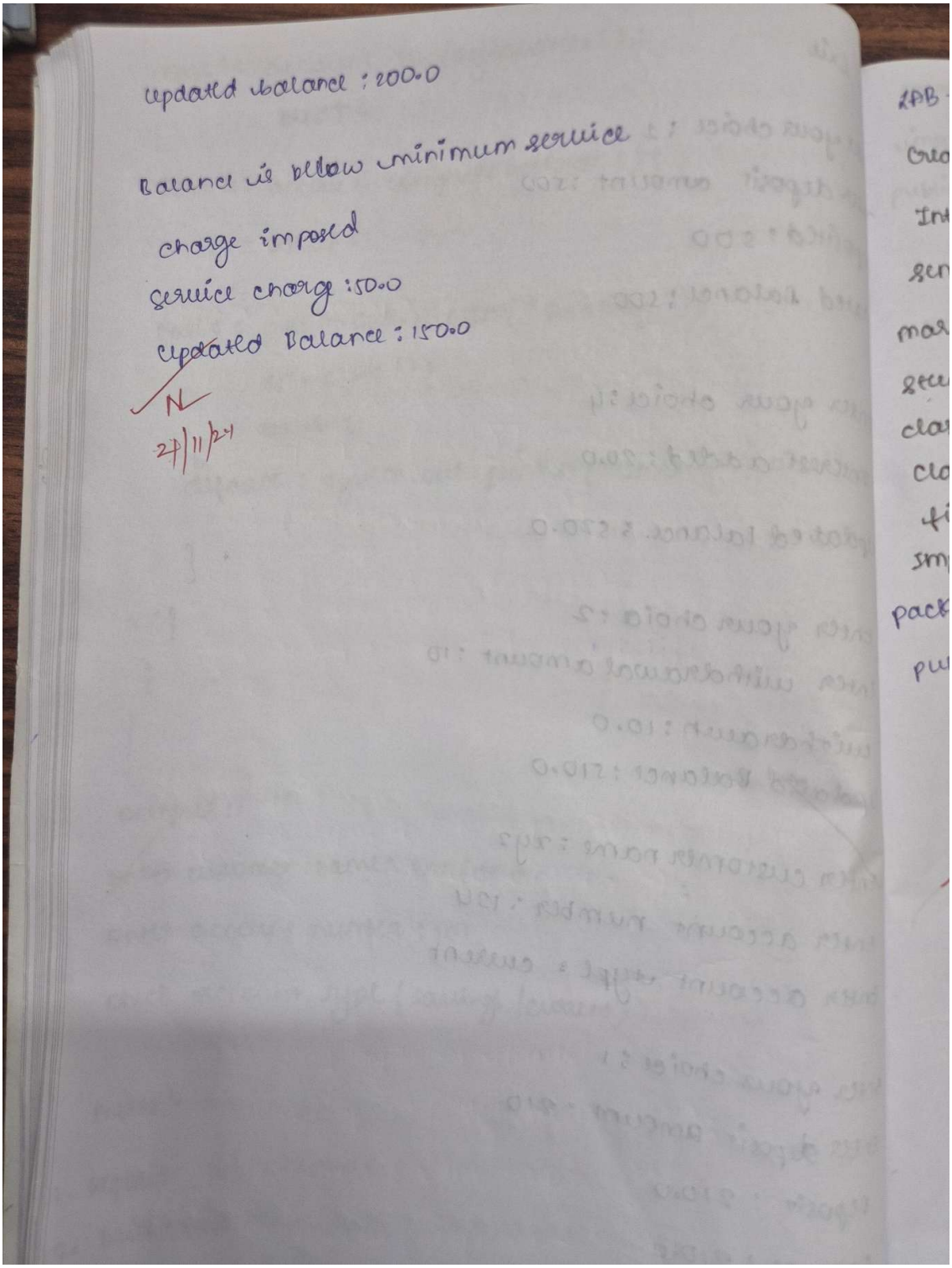
Enter your choice : 4
Interest added : 20.0
Updated Balance : 520.0

Enter your choice : 2
Enter withdrawal amount : 10
withdrawn : 10.0
updated Balance : 510.0

Enter customer name : xyz
Enter account number : 134
Enter account type : current

Enter your choice : 1
Enter deposit amount : 210
Deposit : 210.0
Balance : 210.0

Enter your choice : 2
Enter withdrawal amount : 10
withdrawn : 10



updated balance : 200.0

Balance is below minimum service

charge imposed

service charge : 50.0

updated Balance : 150.0

N

27/11/24

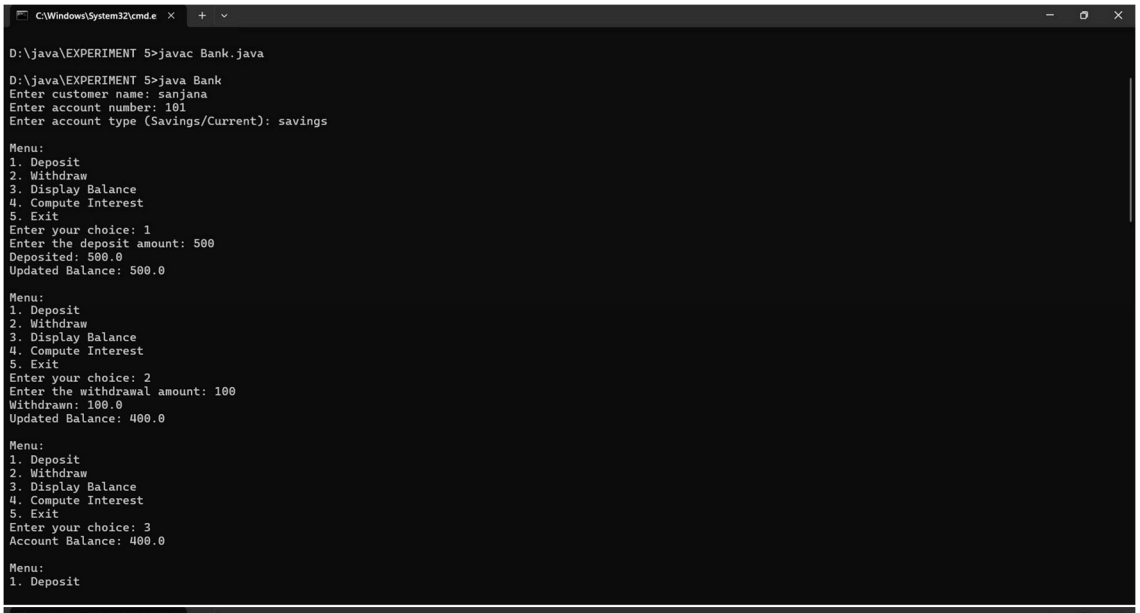
```
import java.util.Scanner;

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

    Account(String name, int accNumber) {
        customerName = name;
        accountNumber = accNumber;
        balance = 0.0;
    }

    void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + "\nUpdated Balance: " + balance);
    }

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

    void withdraw(double amount) {
        System.out.println("Withdrawal is specific to account type.");
    }

    void computeInterest() {
        System.out.println("Interest computation not applicable for this account type.");
    }
}
```

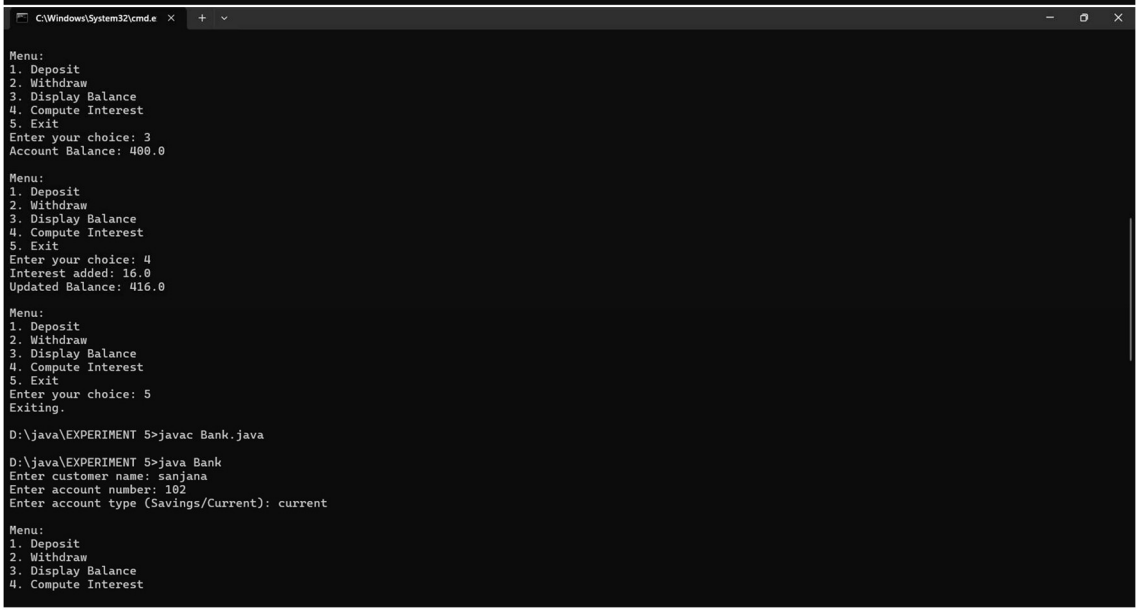
```
class SavingsAccount extends Account {
    double interestRate = 0.04;

    SavingsAccount(String name, int accNumber) {
        super(name, accNumber);
    }

    void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest added: " + interest + "\nUpdated Balance: " + balance);
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + "\nUpdated Balance: " + balance);
        } else {
            System.out.println("Insufficient  balance.");
        }
    }
}

class CurrentAccount extends Account {
    double minimumBalance = 500.0;
    double serviceCharge = 50.0;

    CurrentAccount(String name, int accNumber) {
        super(name, accNumber);
```

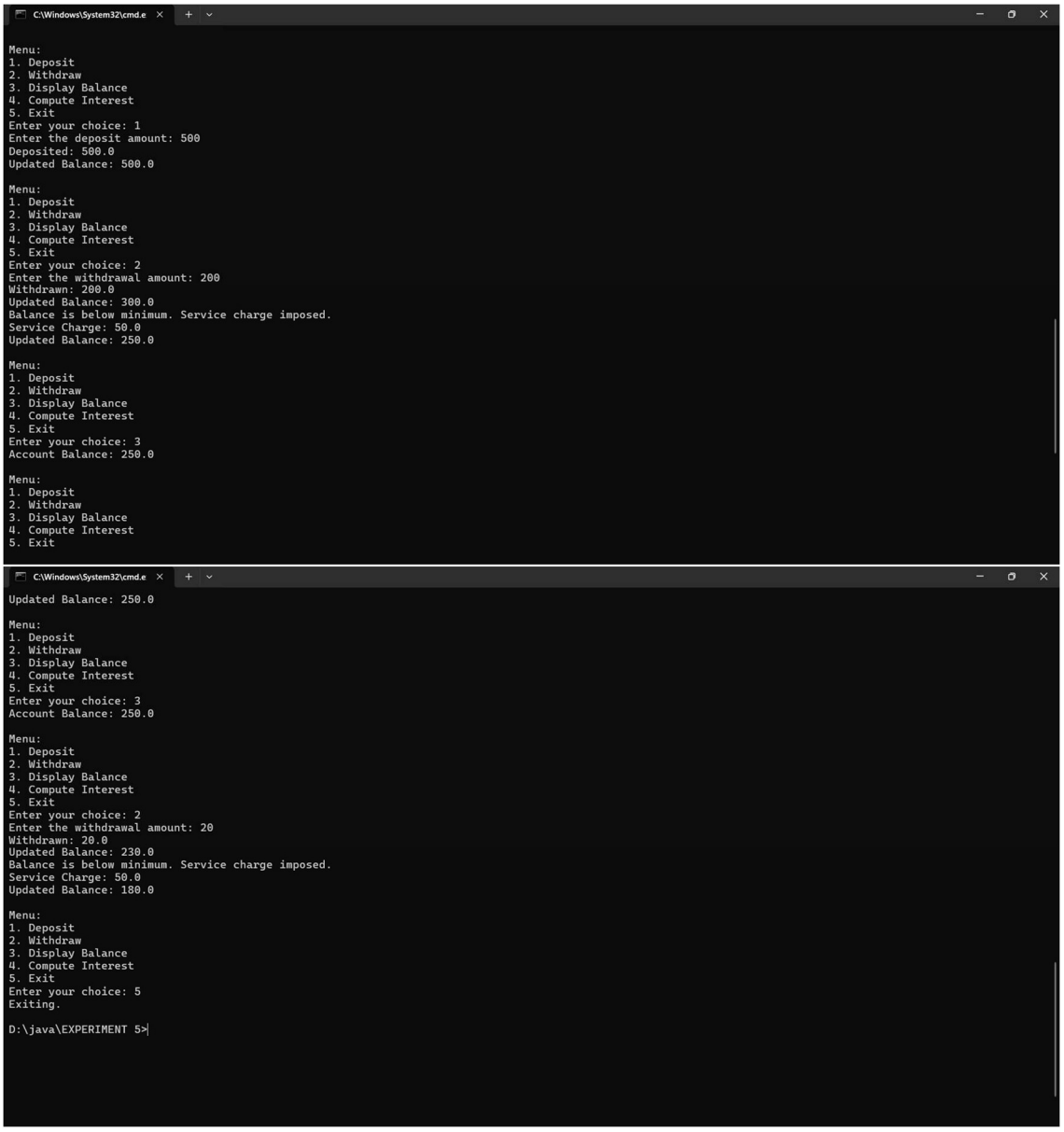
```
    }

    void checkMinimumBalance() {
        if (balance < minimumBalance) {
            System.out.println("Balance is below minimum. Service charge imposed.");
            balance -= serviceCharge;
            System.out.println("Service Charge: " + serviceCharge + "\nUpdated Balance: " +
balance);
        }
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + "\nUpdated Balance: " + balance);
            checkMinimumBalance();
        } else {
            System.out.println("Insufficient  balance.");
        }
    }
}

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

        System.out.print("Enter customer name: ");
        String name = sc.nextLine();
        System.out.print("Enter account number: ");
        int accNumber = sc.nextInt();
        sc.nextLine();
```

```
System.out.print("Enter account type (Savings/Current): ");
String accType = sc.nextLine();

Account account;
if (accType.equalsIgnoreCase("Savings")) {
    account = new SavingsAccount(name, accNumber);
} else if (accType.equalsIgnoreCase("Current")) {
    account = new CurrentAccount(name, accNumber);
} else {
    System.out.println("Invalid account type.");
    sc.close();
    return;
}

while (true) {
    System.out.println("\nMenu:");
    System.out.println("1. Deposit");
    System.out.println("2. Withdraw");
    System.out.println("3. Display Balance");
    System.out.println("4. Compute Interest");
    System.out.println("5. Exit");
    System.out.print("Enter your choice: ");
    int choice = sc.nextInt();

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

```
            case 2:
                System.out.print("Enter the withdrawal amount: ");
                double withdrawAmount = sc.nextDouble();
                account.withdraw(withdrawAmount);
                break;
            case 3:
                account.displayBalance();
                break;
            case 4:
                account.computeInterest();
                break;
            case 5:
                System.out.println("Exiting.");
                sc.close();
                return;
            default:
                System.out.println("Invalid  choice.");
        }
    }
  }
}
```


C:\Windows\System32\cmd.e
D:\java\EXPERIMENT 5>javac Bank.java
D:\java\EXPERIMENT 5>java Bank
Enter customer name: sanjana
Enter account number: 101
Enter account type (Savings/Current): savings
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 1
Enter the deposit amount: 500
Deposited: 500.0
Updated Balance: 500.0
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 2
Enter the withdrawal amount: 100
Withdrawn: 100.0
Updated Balance: 400.0
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 3
Account Balance: 400.0
Menu:
1. Deposit


C:\Windows\System32\cmd.e
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 3
Account Balance: 400.0
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 4
Interest added: 16.0
Updated Balance: 416.0
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
5. Exit
Enter your choice: 5
Exiting.
D:\java\EXPERIMENT 5>javac Bank.java
D:\java\EXPERIMENT 5>java Bank
Enter customer name: sanjana
Enter account number: 102
Enter account type (Savings/Current): current
Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest



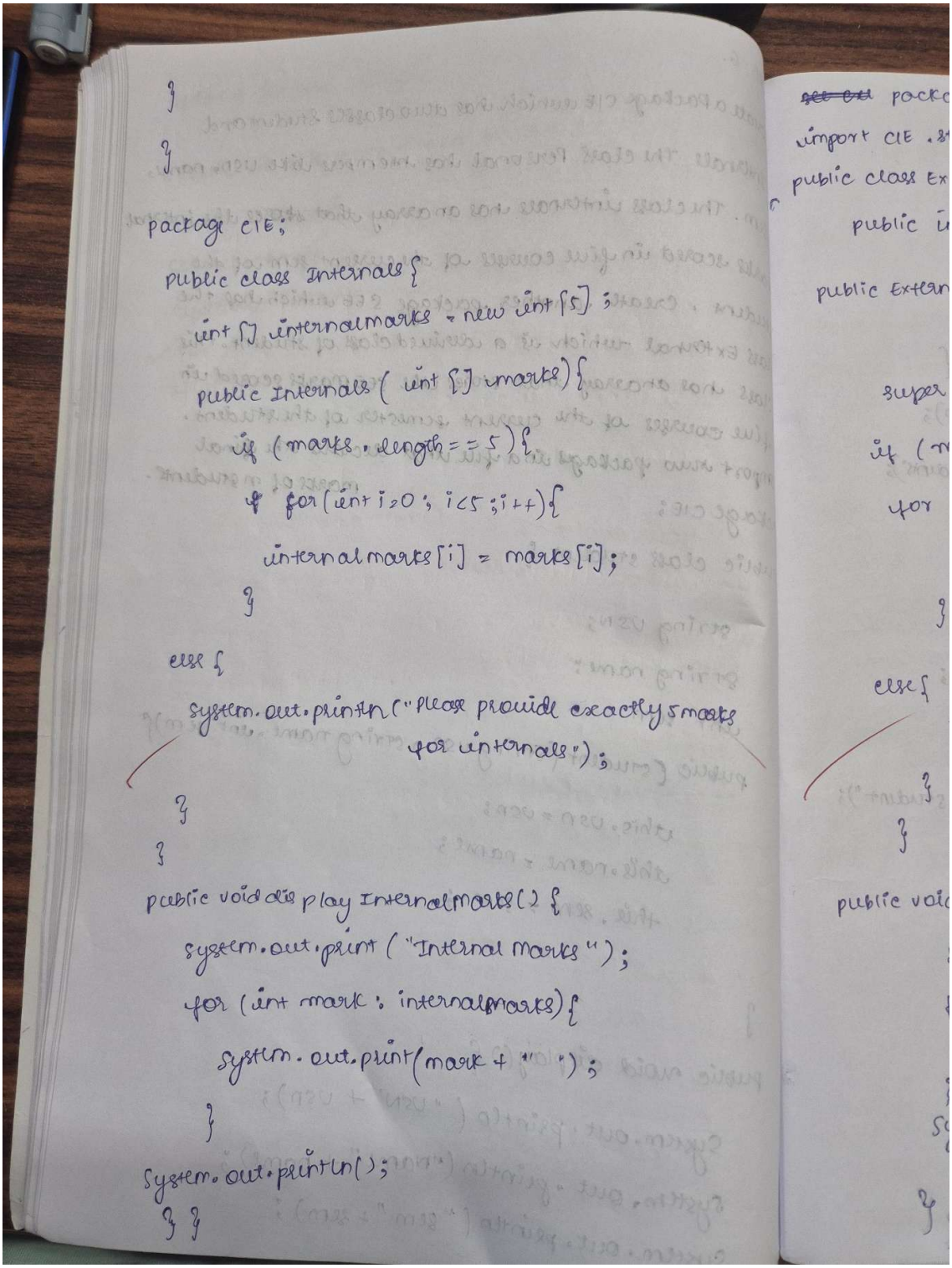
Program 6

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.

JAVA - 6.

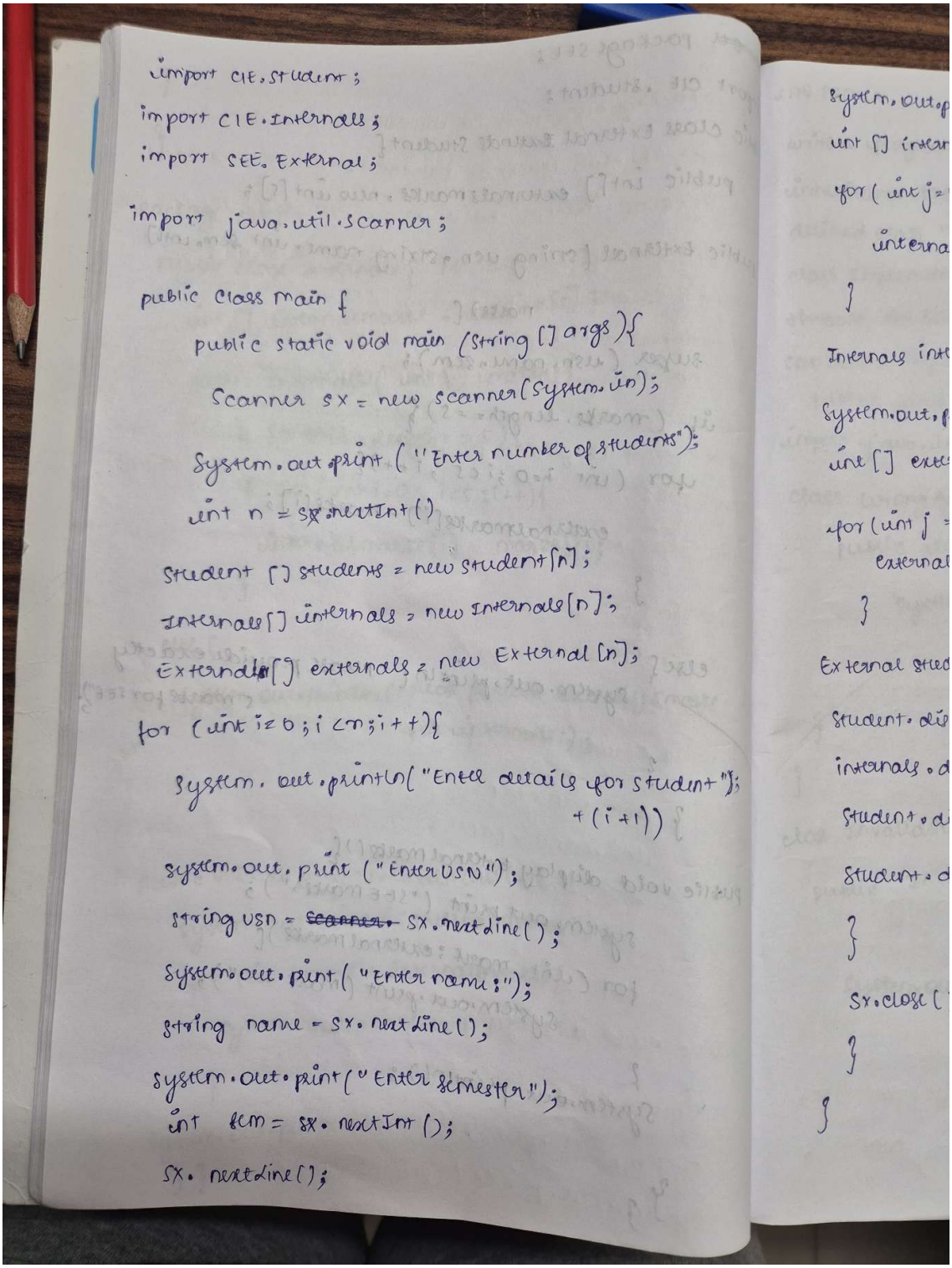
Create a Package CIE which has two classes Student and Internals. The class Personal has members like USN, name, sem. The class Internals has an array that stores the internal marks scored in five courses of the current sem of the student. Create another package SEE which has the class External which 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 two packages in a file that declares the final marks of n students.

```
package CIE;
public class student {
    string USN;
    string name;
    int sem
    public {student (string usn , string name , int sem){
        this.usn = usn;
        this.name = name;
        this.sem = sem ;
}
public void display() {
    System.out.println ( "USN" + usn);
    System.out.println ("Name" + name);
    System.out.println ("sem" + sem);
```



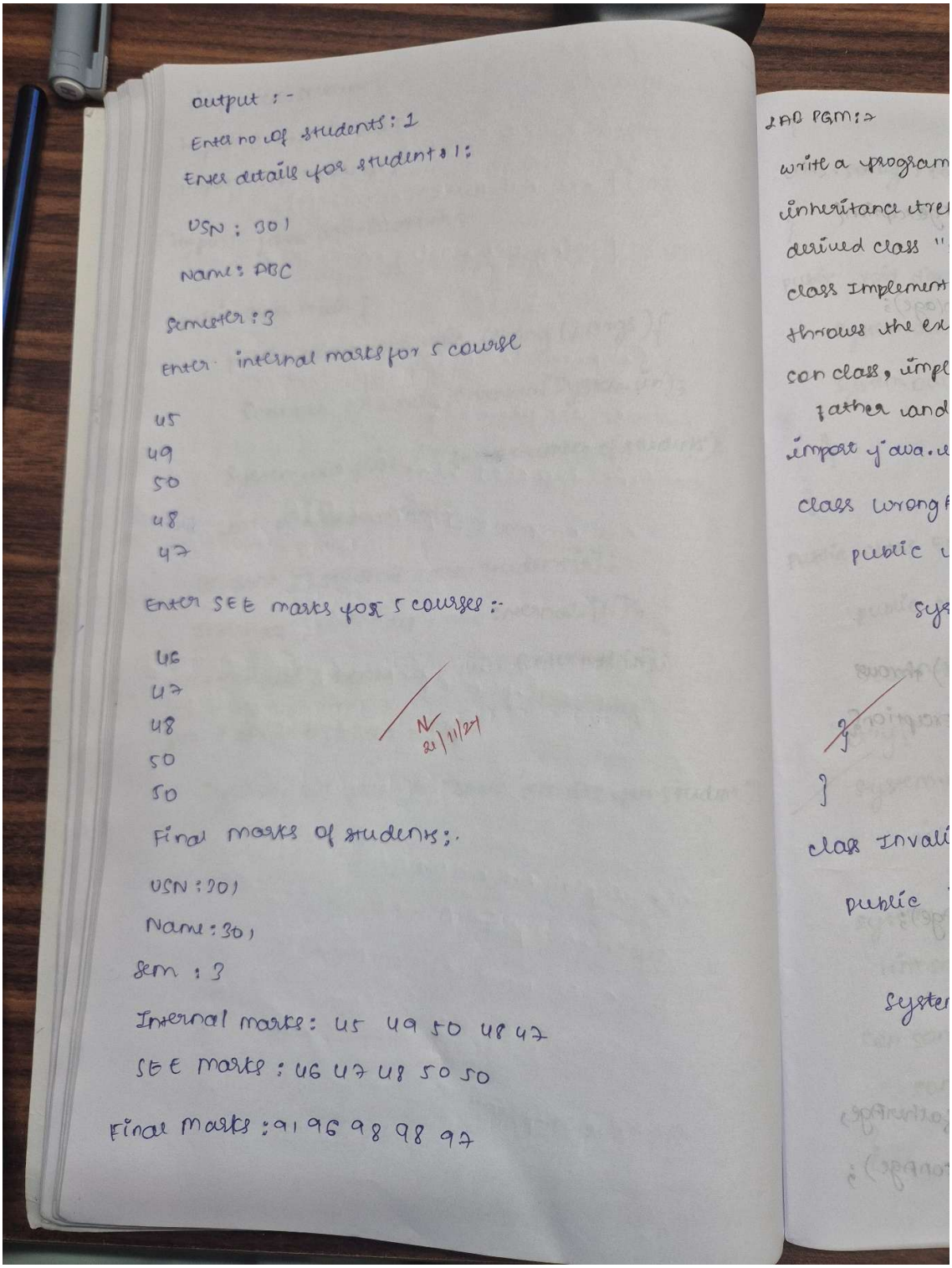
```
}
}

package CIE;

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

    public Internals(int[] marks) {
        if (marks.length == 5) {
            for(int i=0; i<5; i++){
                internalmarks[i] = marks[i];
            }
        else {
            System.out.println("Please provide exactly 5 marks
                                for internals");
        }
    }
}

public void displayInternalMarks() {
    System.out.print("Internal Marks");
    for (int mark : internalmarks){
        System.out.print(mark + " ");
    }
System.out.println();
}}
```

```
package SEE;
import CIE.Student;
public class External extends Student{
    public int[] externalsmarks = new int[5];

public External(String usn, String name, int sem, int[]
                marks){
    super(usn, name, sem);
    if(marks.length==5){
    for(int i=0;i<5;i++){
        externalmarks[i] = marks[i];
    }
    else{
        System.out.println("please provide exactly
                5 marks for SEE");
    }
}

public void displayExternalmarks(){
        System.out.print("SEE marks:");
        for(int mark:externalmarks){
            System.out.print(mark+" ");
        }
        System.out.println();

}}
```



```
import CIE.Student;
import CIE.Internals;
import SEE.External;

import java.util.Scanner;

public class main {
    public static void main (String [] args){
        Scanner sx = new scanner(System.in);
        System.out.print ("Enter number of students");
        int n = sx.nextInt()
    Student [] students = new Student[n];
    Internals[] internals = new Internals[n];
    External[] externals = new External [n];
for (int i=0;i<n;i++){
    System. out.println("Enter details for student"+
                                    +(i+1))
    system.out.print ("Enter USN");
    String USN = ~~Scanner.~~ sx.nextline();
    System.out.print("Enter name:");
    String name = sx.nextline();
    System.out.print("Enter semester");
    int sem = sx.nextInt();
    sx.nextline();
```

```
System.out.println("Enter internal marks for 5 courses:");
int[] internalMarks = new int[5]

for(int j=0;j<5;j++){

    internalMarks[j] = sx.nextInt();

}

Internals internals = new Internals(internalMarks);

System.out.println("Enter external marks for 5 courses:");
int[] externalMarks = new int[];

for(int j =0;j<5;j++){
    externalMarks[j] = sx.nextInt();
}

External student = new External(usn, sem, externalMarks);

student.displayStudentDetails();
internals.displayInternalMarks();
student.displayExternalMarks();
student.displayFinalMarks(internals);

}

sx.close();

}

}
```



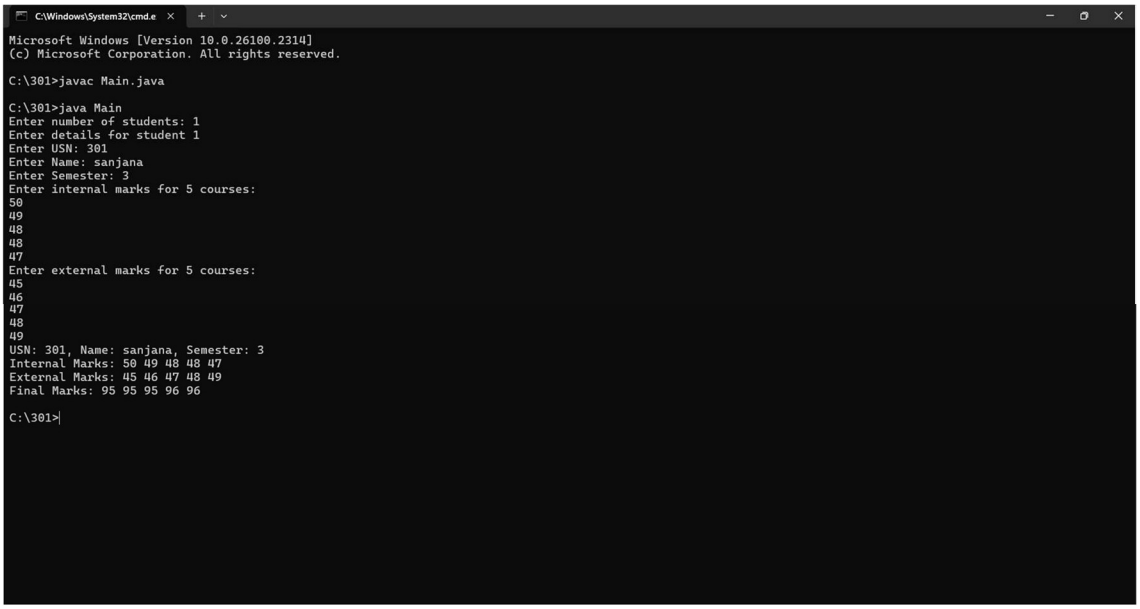

```
package CIE;

public class Internals {
    int[] internalMarks = new int[5]; // Array to store internal marks for 5 courses
```

```
    public Internals(int[] marks) {
        for (int i = 0; i < 5; i++) {
            internalMarks[i] = marks[i];
        }
    }

    public int[] getInternalMarks() {
        return internalMarks;
    }

    public void displayInternalMarks() {
        System.out.print("Internal Marks: ");
        for (int mark : internalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
    }
}

package CIE;

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

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

```
        this.sem = sem;
    }

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

import CIE.Student;
import CIE.Internals;

public class External extends Student {
    int[] externalMarks = new int[5]; // Array to store external marks for 5 courses

    public External(String usn, String name, int sem, int[] externalMarks) {
        super(usn, name, sem); // Call parent class constructor
        this.externalMarks = externalMarks;
    }

    public int[] getExternalMarks() {
        return externalMarks;
    }

    public void displayExternalMarks() {
        System.out.print("External Marks: ");
        for (int mark : externalMarks) {
            System.out.print(mark + " ");
        }
        System.out.println();
```

```
    }

    public void displayFinalMarks(Internals internals) {
        System.out.print("Final Marks: ");
        for (int i = 0; i < 5; i++) {
            int finalMark = internals.getInternalMarks()[i] + externalMarks[i]; // Calculate final
marks
            System.out.print(finalMark + " ");
        }
        System.out.println();
    }
}
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(); // Consume newline

        External[] students = new External[n];

        // Loop to input details for each student
        for (int i = 0; i < n; i++) {
            System.out.println("Enter 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();
            scanner.nextLine();

            System.out.println("Enter internal marks for 5 courses: ");
            int[] internalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = scanner.nextInt();
            }
            Internals internals = new Internals(internalMarks);

            System.out.println("Enter external marks for 5 courses: ");
            int[] externalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                externalMarks[j] = scanner.nextInt();
            }
            External student = new External(usn, name, sem, externalMarks);
            student.displayStudentDetails();
            internals.displayInternalMarks();
            student.displayExternalMarks();
            student.displayFinalMarks(internals);
        }
        scanner.close();
    }
}
```



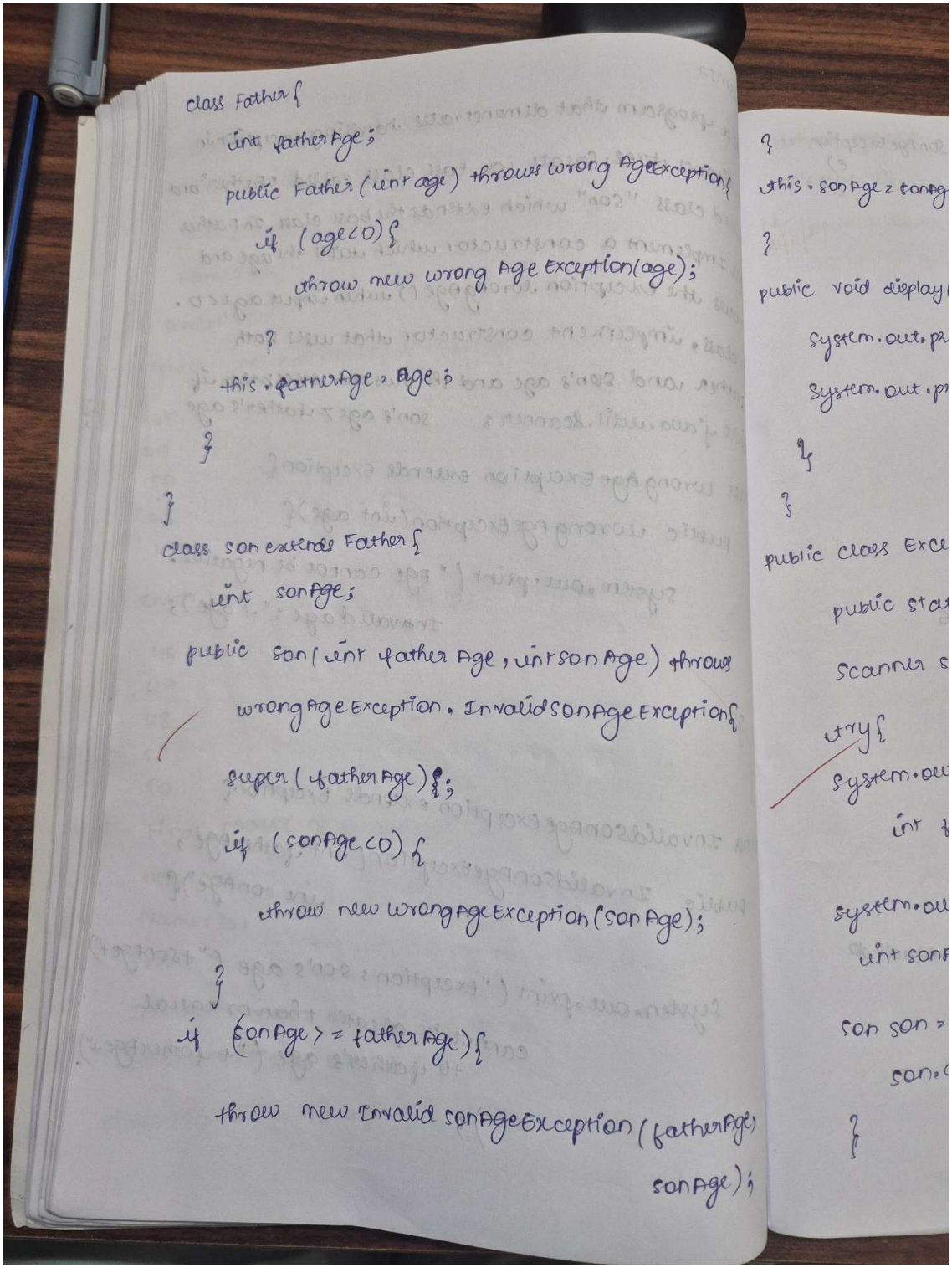
Program 7

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=father’s age.

2AD PGM:-

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

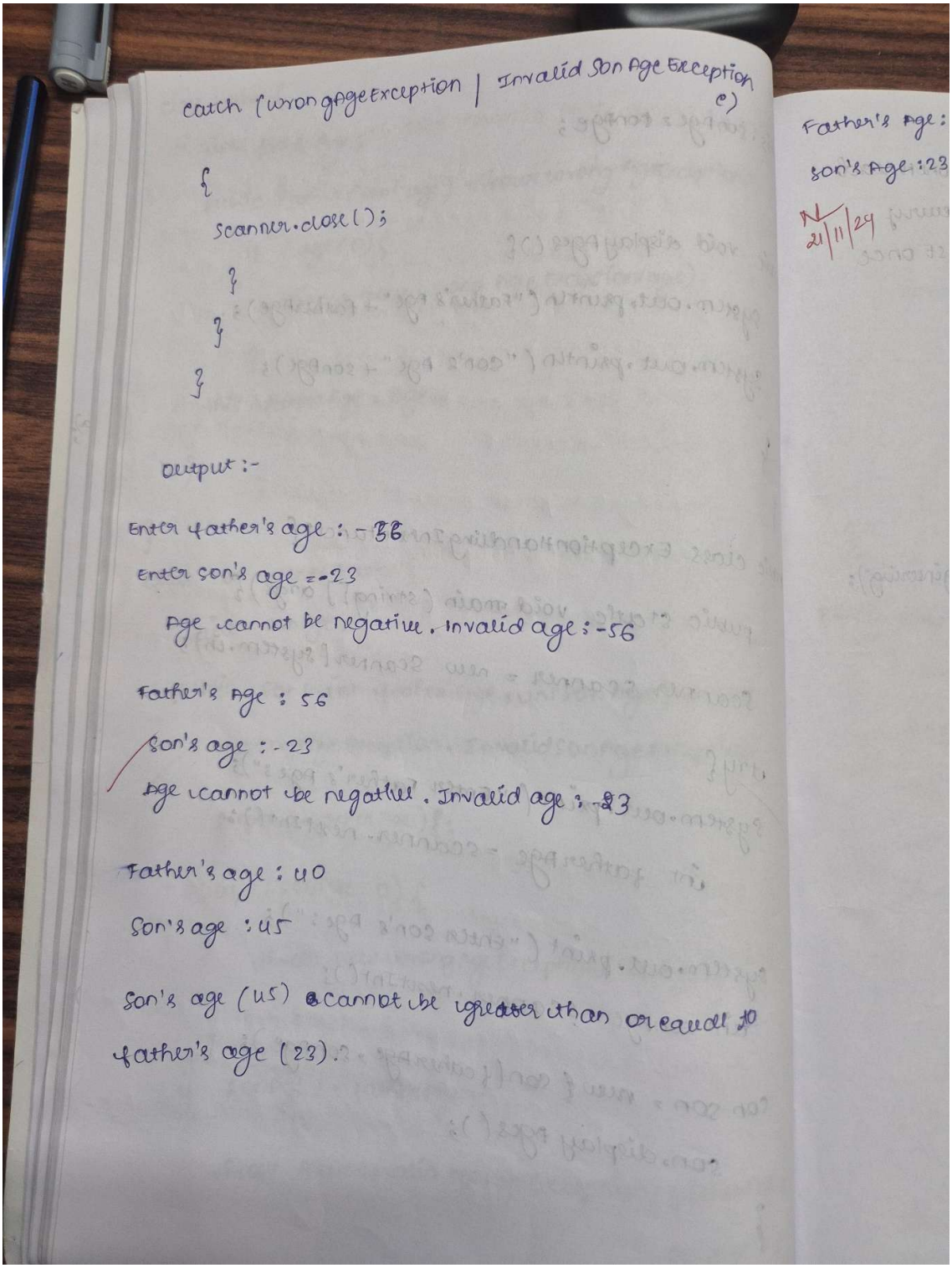
```
import java.util.Scanner;

class WrongAgeException extends Exception{
    public WrongAgeException(int age){
        System.out.print("Age cannot be negative.
                          Invalid age :" + age);
    }
}

class InvalidSonAgeException extends Exception{
    public InvalidSonAgeException(int fatherAge,
                                  int sonAge){
        System.out.print("Exception : Son's age ("+sonAge+")
                          cannot be greater than or equal
                          to father's age ("+fatherAge+").
                          ");
```



```
class Father{
    int fatherAge;
    public Father(int age) throws wrongAgeException{
        if (age<0){
            throw new wrongAgeException(age);
        }
        this.fatherAge = Age;
    }
}
class son extends Father{
    int sonAge;
    public son(int fatherAge, int sonAge) throws
        wrongAgeException, InvalidsonAgeExcption{
        super(fatherAge);
        if (sonAge<0){
            throw new wrongAgeException(sonAge);
        }
        if (sonAge>=fatherAge){
            throw new InvalidsonAgeException(fatherAge)
                sonAge);
```

```
}
this.sonAge = sonAg
}
public void display(
    System.out.pr
    System.out.pr
}
}
public class Exce
    public sta
    Scanner s
    try{
    System.ou
        int f
    system.ou
    int sonA
    son son =
        son.
    }
```

```
    }
    this.sonAge = sonAge;
    }
    public void displayAges(){
        System.out.println("Father's Age" + fatherAge);
        System.out.println("Son's Age" + sonAge);
    }
}
public class ExceptionHandlingInheritance2{
    public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    try{
    System.out.print("Enter Father's Age:");
        int fatherAge = scanner.nextInt();
    System.out.print("Enter Son's Age:");
    int sonAge = scanner.nextInt();
    Son son = new Son(fatherAge, sonAge);
        son.displayAges();
    }
```



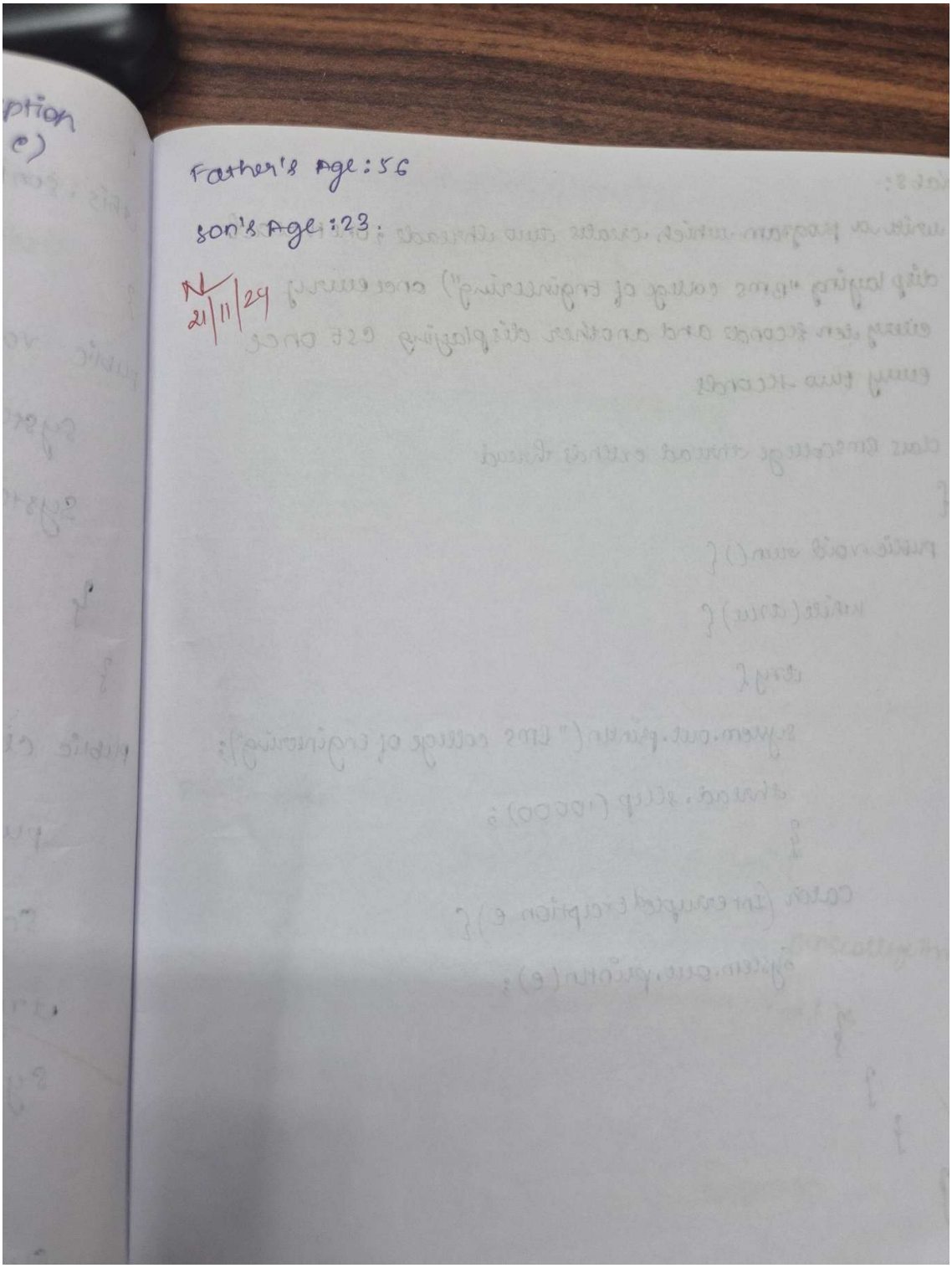
```
catch (WrongAgeException | InvalidSonAgeException e)
{
    scanner.close();
  }
 }
}
```

Output :-

```
Enter father's age : -56
Enter son's age : -23
  Age cannot be negative. Invalid age : -56
Father's Age : 56
Son's age : -23
  Age cannot be negative. Invalid age : -23
Father's age : 40
Son's age : 45
```

Son's age (45) cannot be greater than or equal to father's age (23).

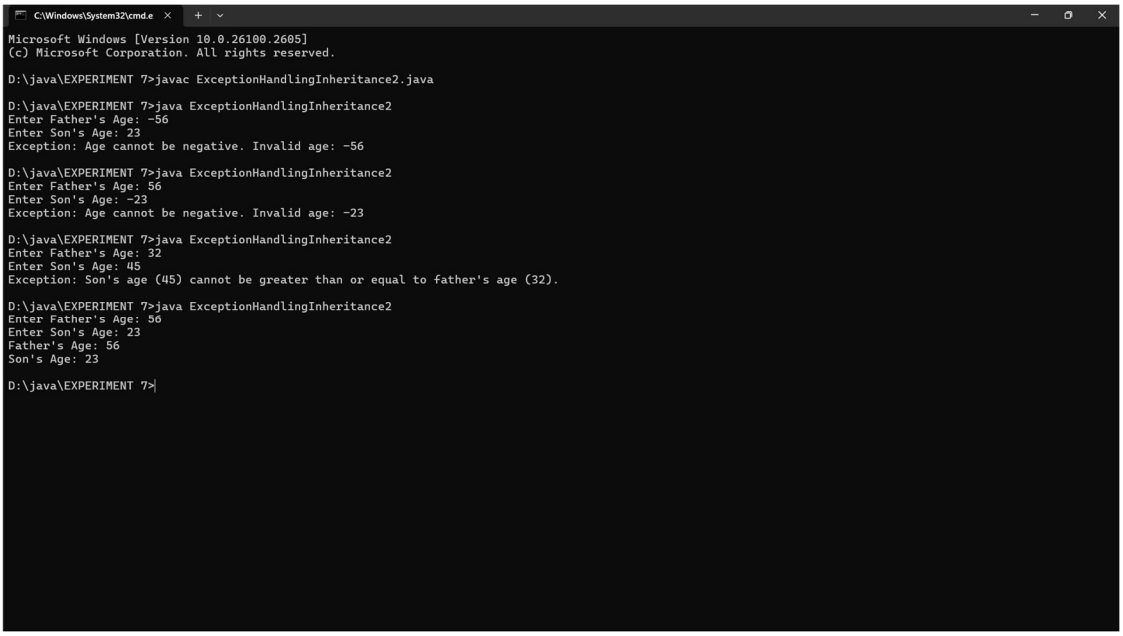
Father's Age :
Son's Age : 23
21/11/24




```
import java.util.Scanner;
class WrongAgeException extends Exception {
    public WrongAgeException(int age) {
        System.out.println("Exception: Age cannot be negative. Invalid age: " + age);
```

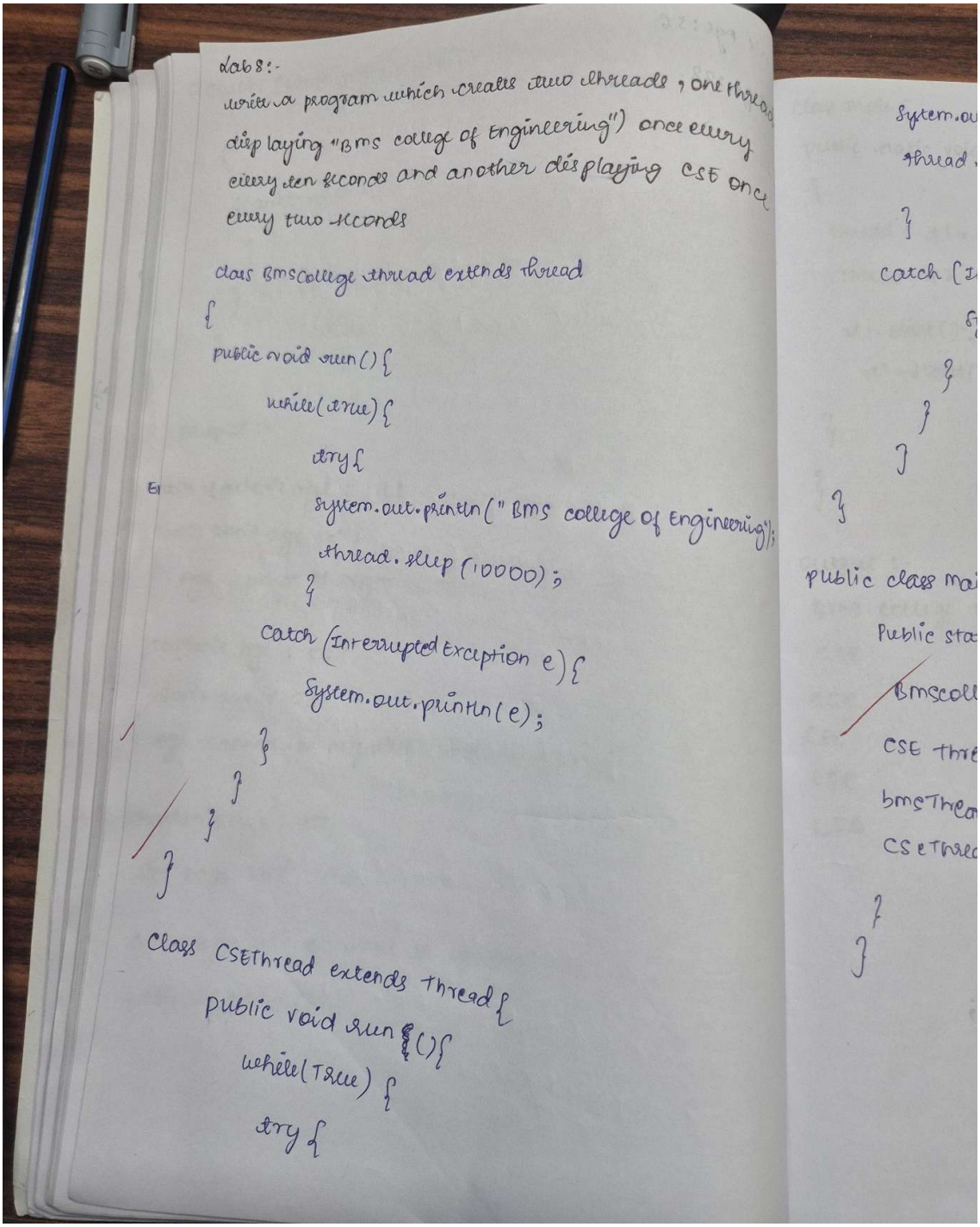
```
    }
}
class InvalidSonAgeException extends Exception {
    public InvalidSonAgeException(int fatherAge, int sonAge) {
        System.out.println("Exception: Son's age (" + sonAge + ") cannot be greater than or
equal to father's age (" + fatherAge + ").");
    }
}
class Father {
    int fatherAge;

    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException(age);
        }
        this.fatherAge = age;
    }
}
class Son extends Father {
    int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAgeException,
InvalidSonAgeException {
        super(fatherAge);
        if (sonAge < 0) {
            throw new WrongAgeException(sonAge);
        }
        if (sonAge >= fatherAge) {
            throw new InvalidSonAgeException(fatherAge, sonAge);
        }
        this.sonAge = sonAge;
```

```
    }

    public void displayAges() {
        System.out.println("Father's Age: " + fatherAge);
        System.out.println("Son's Age: " + sonAge);
    }
}
public class ExceptionHandlingInheritance2 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        try {
            System.out.print("Enter Father's Age: ");
            int fatherAge = scanner.nextInt();

            System.out.print("Enter Son's Age: ");
            int sonAge = scanner.nextInt();

            Son son = new Son(fatherAge, sonAge);
            son.displayAges();

        } catch (WrongAgeException | InvalidSonAgeException e) {

        } finally {
            scanner.close();
        }
    }
}
```



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

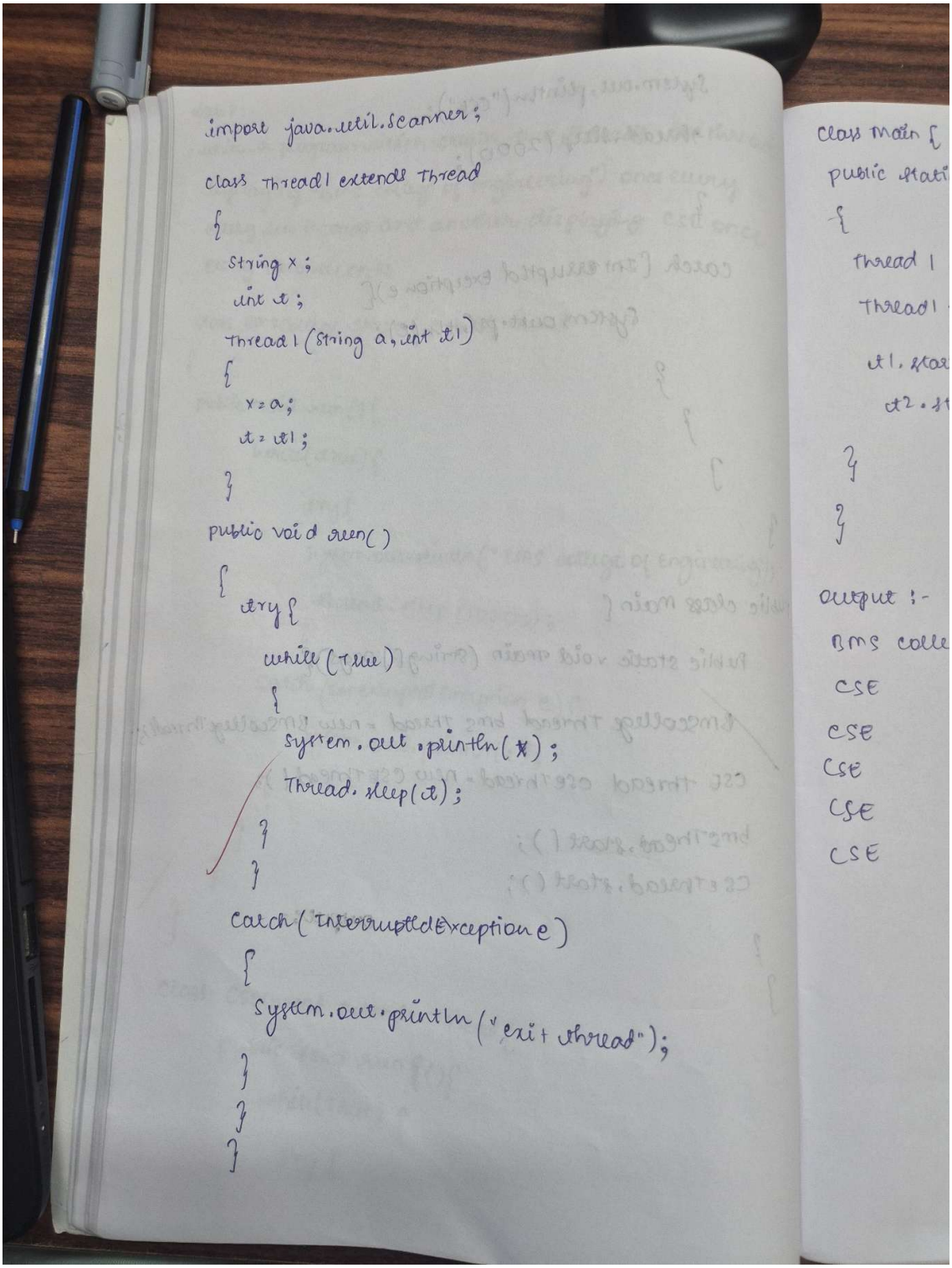


Lab 8:-
write a program which creates two threads, one thread displaying "BMS college of Engineering" once every ten seconds and another displaying CSE once every two seconds

```
class BmsCollege thread extends thread
{
public void run(){
    while(true){
        try{
        System.out.println("BMS college of Engineering");
        thread.sleep(10000);
        }
    Catch (InterruptedException e){
        System.out.println(e);
    }
  }
}
}

Class CSEThread extends Thread{
    public void run(){
        while(True){
            try{
```

```
            System.out.println("CSE");
            Thread.sleep(2000);
        }
        catch (InterruptedException e){
            System.out.println(e);
        }
    }
}

public class Main{
    Public static void main (String[] args){
        BMSCollegeThread bmsThread = new BMSCollegeThread();
        CSEThread cseThread = new CSEThread();
        bmsThread.start();
        cseThread.start();
    }
}
```

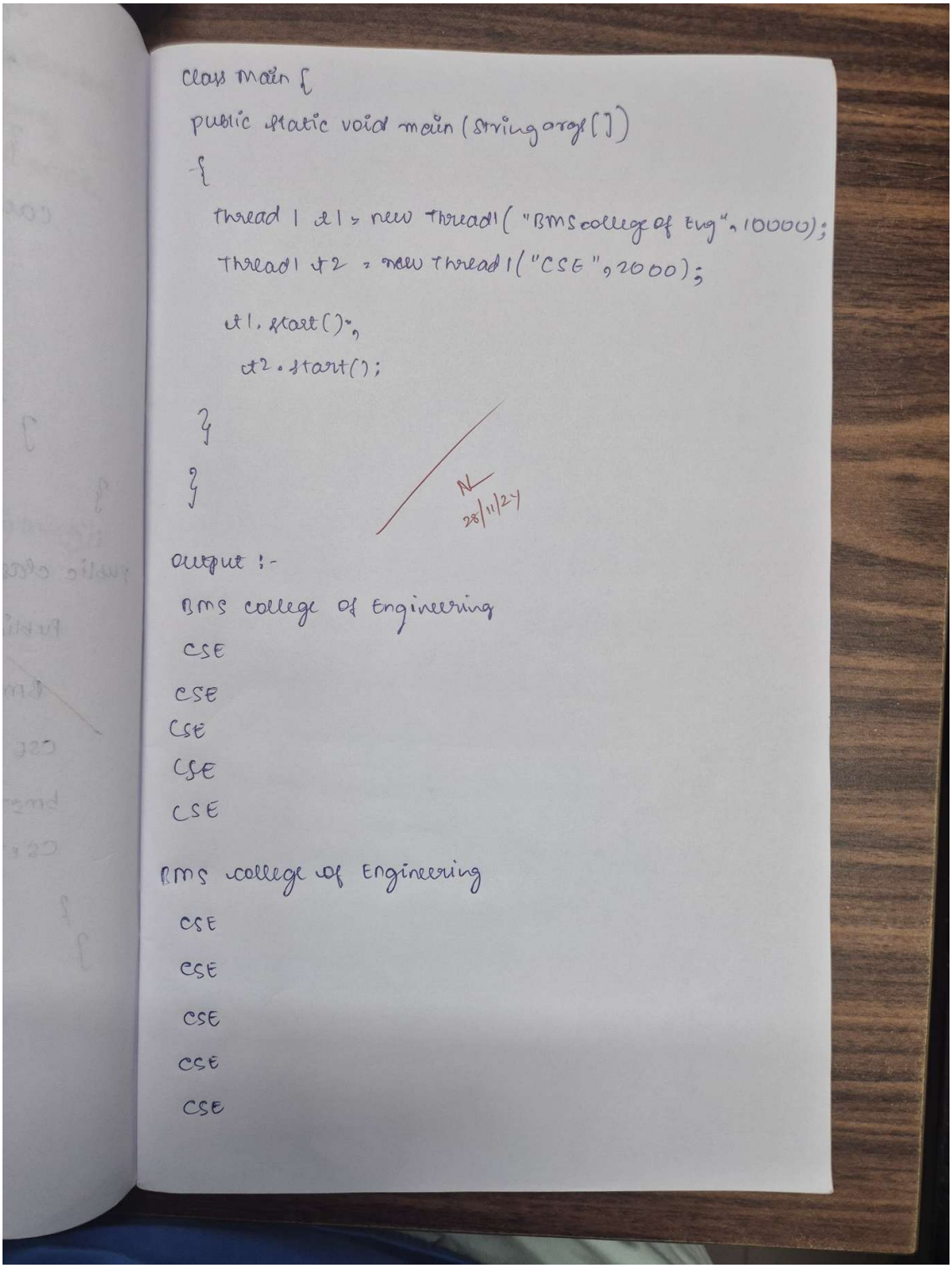
Output:-



```
import java.util.Scanner;
class Thread1 extends Thread
{
  String x;
  int t;
  Thread1 (String a, int t1)
  {
    x = a;
    t = t1;
  }
public void run()
  {
    try{
      while (true)
       {
        System.out.println(x);
        Thread.sleep(t);
       }
      }
    catch(InterruptedException e)
     {
      System.out.println("exit thread");
     }
    }
  }
```

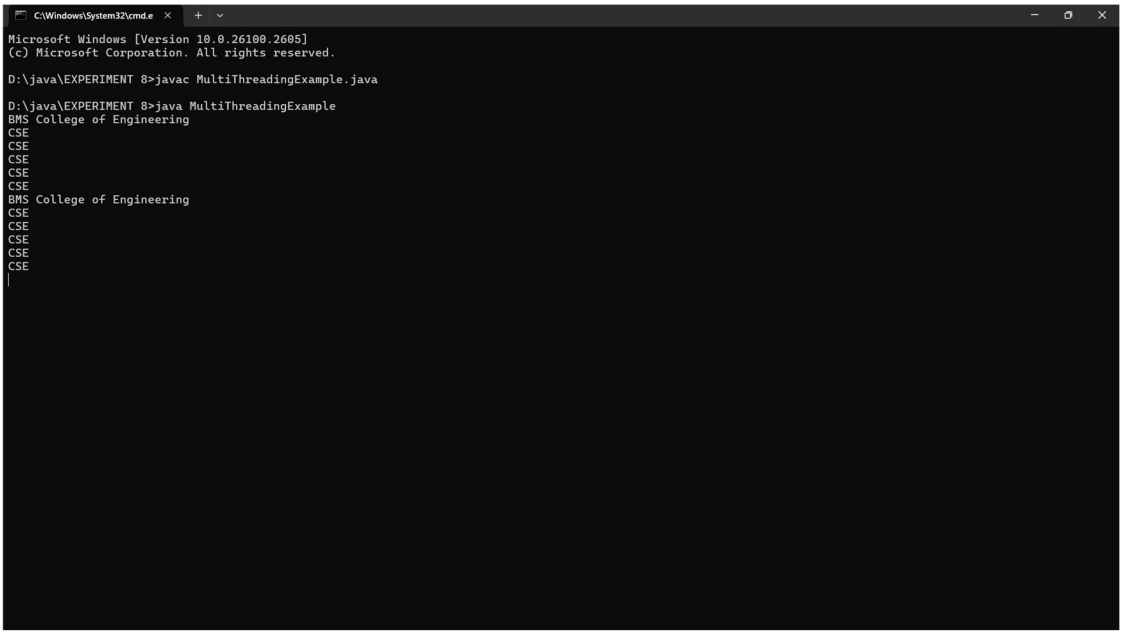
```
class main {
public stati
{
  Thread 1
  Thread1
  t1.star
  t2.s
}
}
```

```
output :-
BMS colle
CSE
CSE
CSE
CSE
CSE
```



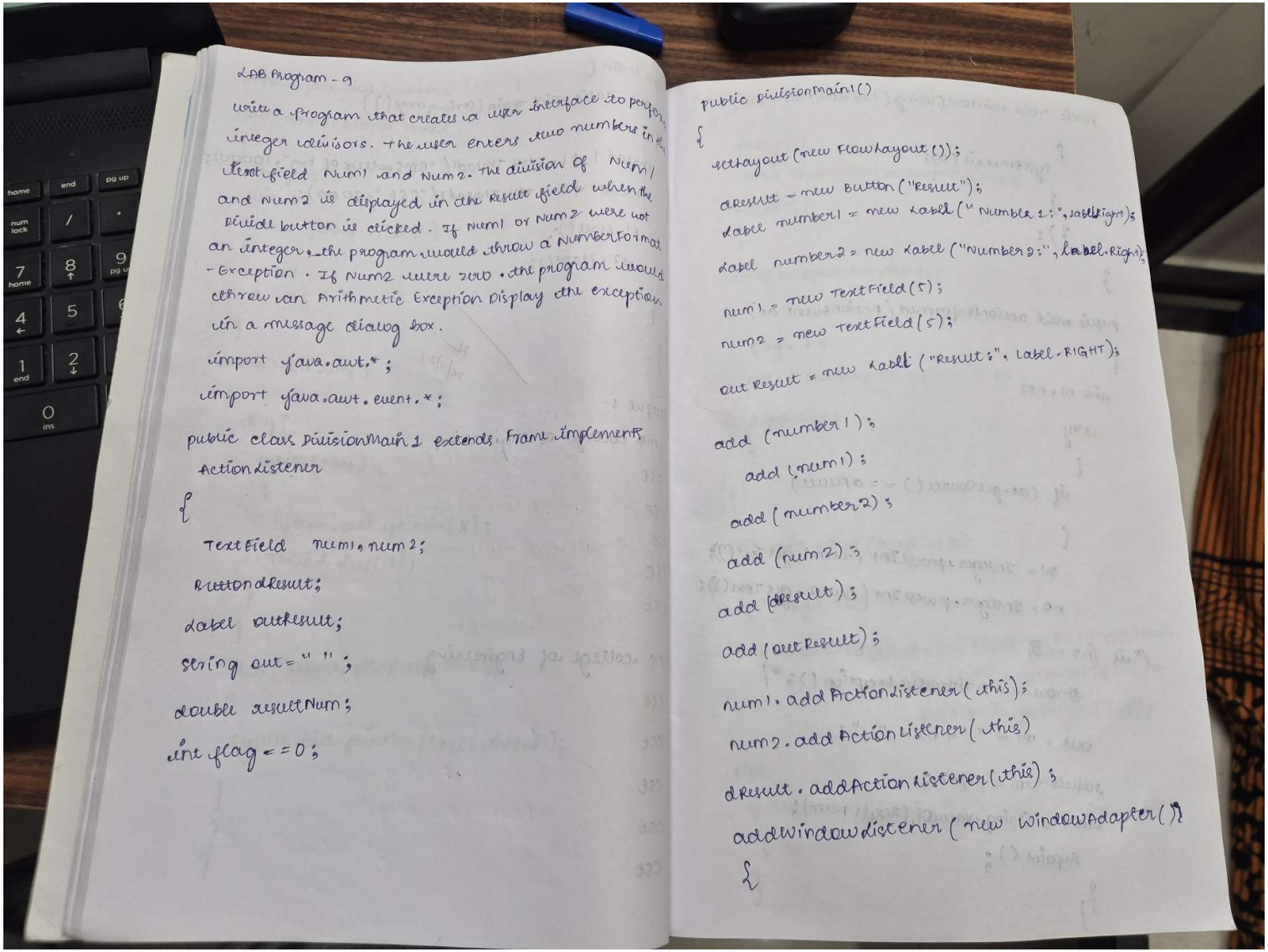

```
class DisplayMessage extends Thread {
    private String message;
    private int interval;
```

```
    public DisplayMessage(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }

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

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

        thread1.start();
        thread2.start();
    }
}
```



PROGRAM-9

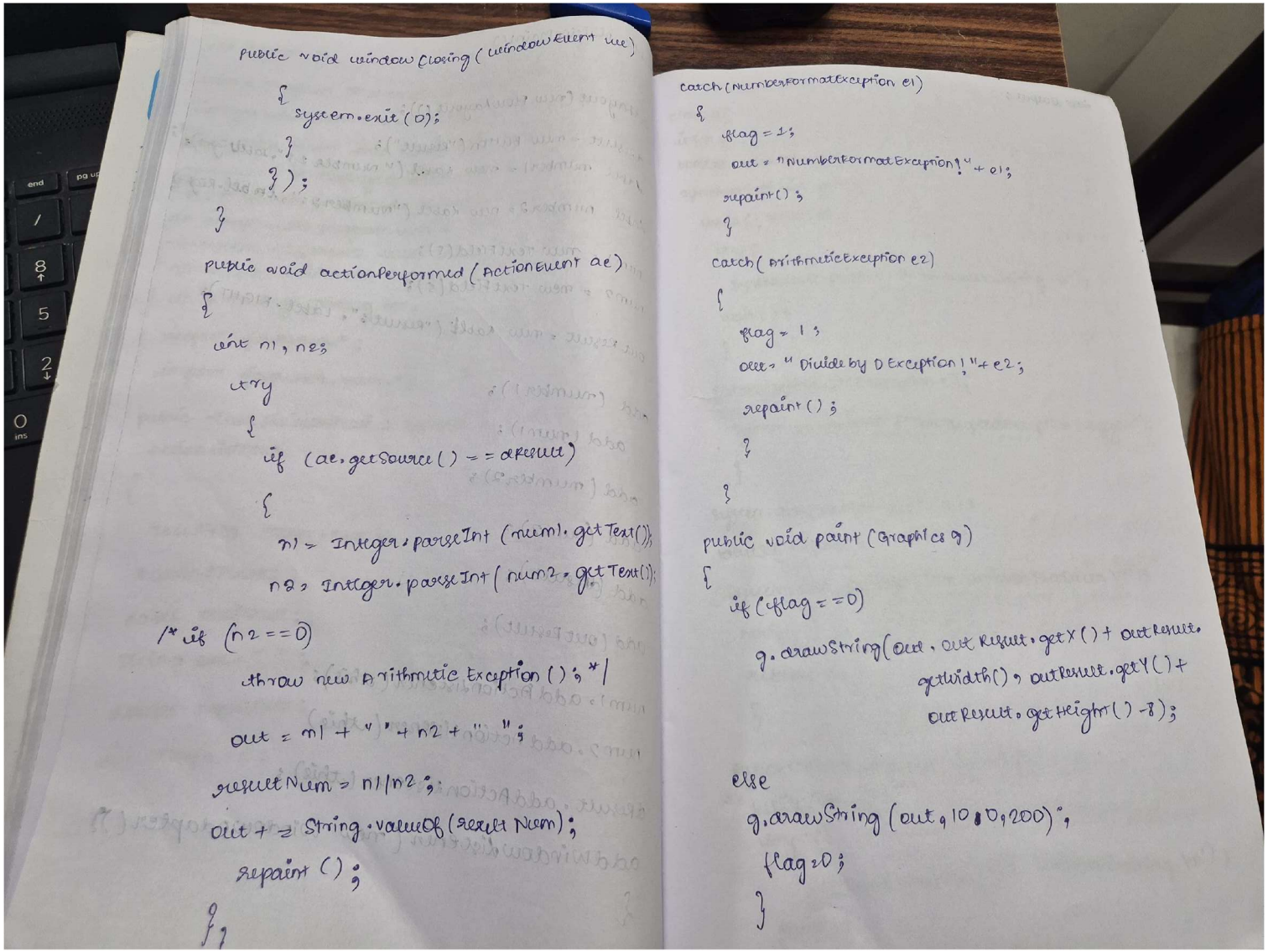
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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



LAB Program - 9

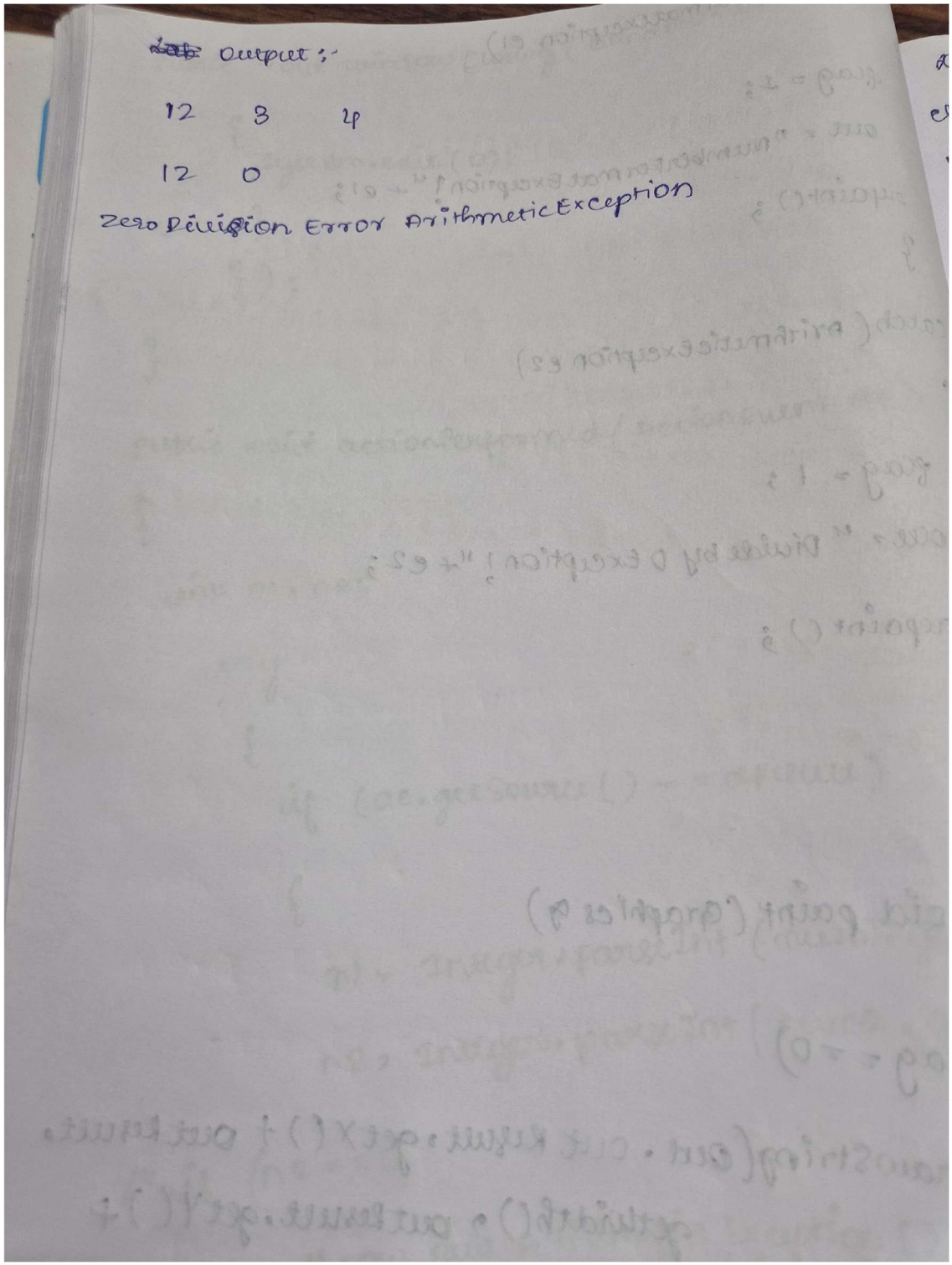
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the textfield Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormat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

```
import java.awt.*;
import java.awt.event.*;
public class DivisionMain1 extends Frame implements
 ActionListener
{
  TextField num1, num2;
 Button dResult;
 Label outResult;
 String out = " ";
 double resultNum;
 int flag = = 0;
public DivisionMain1()
{
 setLayout (new FlowLayout());
 dResult = new Button ("Result");
 Label number1 = new Label (" Number 1:", Label.Right);
 Label number2 = new Label ("Number 2:", Label.Right);
 num1 = new TextField(5);
 num2 = new TextField(5);
 outResult = new Label ("Result:", Label.RIGHT);
 add (number1);
   add (num1);
 add (number2);
  add (num2);
 add (dResult);
 add (outResult);
 num1.addActionListener(this);
 num2.addActionListener(this)
 dResult.addActionListener(this);
 addWindowListener (new WindowAdapter()
  {
```



```
public void windowClosing(WindowEvent we)
    {
      System.exit(0);
    }
    });
}

public void actionPerformed(ActionEvent ae)
{
  int n1, n2;
   try
    {
      if (ae.getSource() == aResult)
      {
        n1 = Integer.parseInt(num1.getText());
        n2 = Integer.parseInt(num2.getText());
/* if (n2 == 0)
      throw new ArithmeticException(); */
      out = n1 + " " + n2 + " ";
      resultNum = n1/n2;
      out += String.valueOf(resultNum);
       repaint();
},
```

```
catch(NumberFormatException e1)
{
  flag = 1;
  out = "NumberFormatException!" + e1;
  repaint();
}
catch(ArithmeticException e2)
{
  flag = 1;
  out = "Divide by 0 Exception!" + e2;
  repaint();
}
}
public void paint(Graphics g)
{
  if (flag == 0)
    g.drawString(out, outResult.getX() + outResult.
                 getWidth(), outResult.getY() +
                 outResult.getHeight() - 8);

  else
    g.drawString(out, 10, 0, 200);
     flag = 0;
}
```




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

public class DivisionMain1 extends Frame implements ActionListener {
```

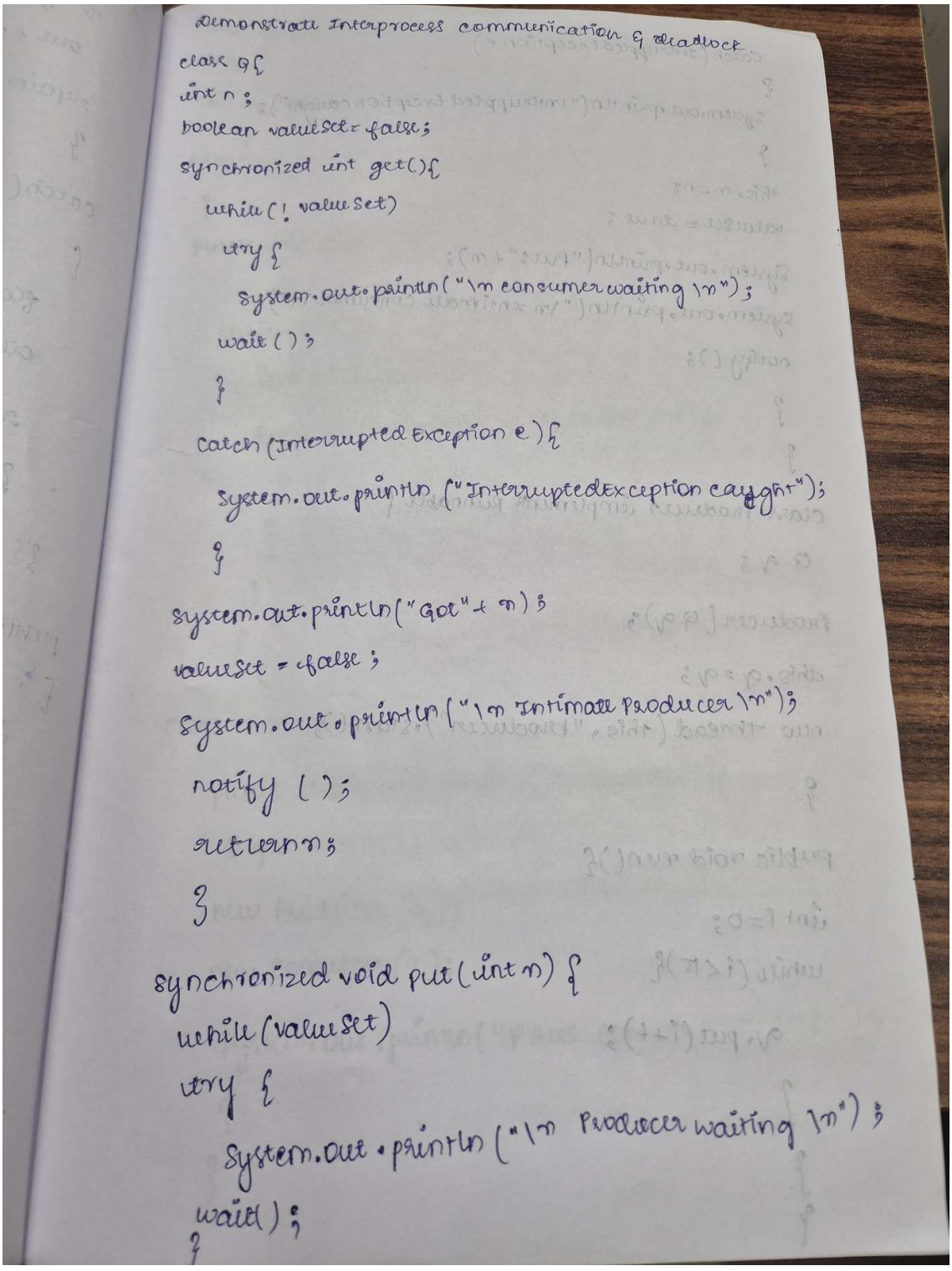
```
TextField num1, num2;
Button dResult;
Label outResult;
String out = "";
double resultNum;
int flag = 0;

public DivisionMain1() {
    setLayout(new FlowLayout());

    dResult = new Button("RESULT");
    Label number1 = new Label("Number 1:", Label.RIGHT);
    Label number2 = new Label("Number 2:", Label.RIGHT);
    num1 = new TextField(5);
    num2 = new TextField(5);

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

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

    num1.addActionListener(this);
    num2.addActionListener(this);
    dResult.addActionListener(this);

    addWindowListener(new WindowAdapter() {
```

```
        public void windowClosing(WindowEvent we) {
            System.exit(0);
        }
    });
}

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

            if (n2 == 0)
                throw new ArithmeticException();

            out = n1 + " " + n2 + " ";
            resultNum = n1 / n2;
            out += String.valueOf(resultNum);
            repaint();
        }
    } catch (NumberFormatException e1) {
        flag = 1;
        out = "Number Format Exception! " + e1;
        repaint();
    } catch (ArithmeticException e2) {
        flag = 1;
        out = "Divide by 0 Exception! " + e2;
        repaint();
    }
```

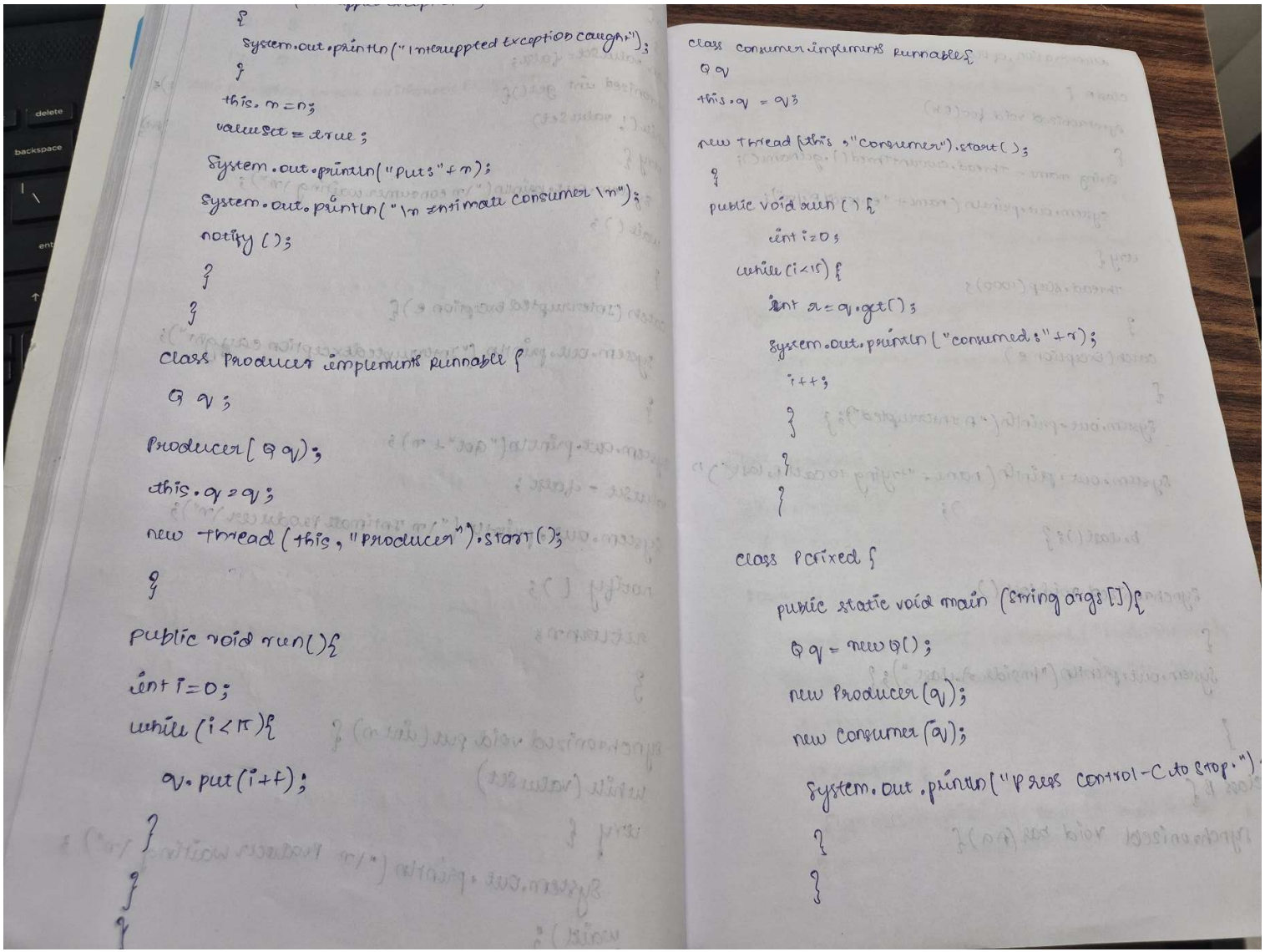
```
    }

    public void paint(Graphics g) {
        if (flag == 0) {
            g.drawString(out, outResult.getX() + outResult.getWidth(), outResult.getY() +
outResult.getHeight() - 8);
        } else {
            g.drawString(out, 100, 200);
        }
        flag = 0;
    }

    public static void main(String[] args) {
        DivisionMain1 frame = new DivisionMain1();
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```

PROGRAM -10

Demonstrate interprocess communication and deadlock

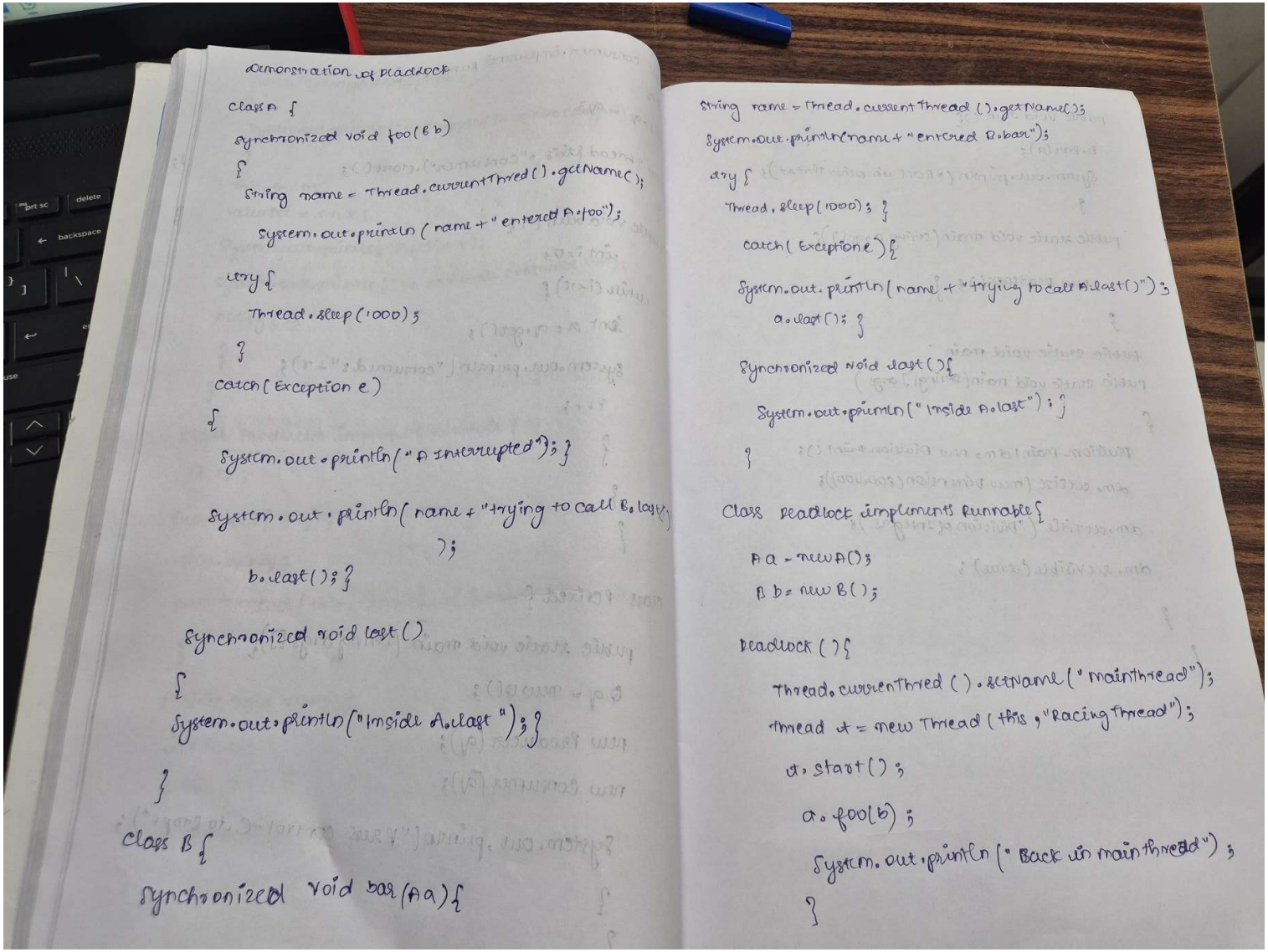


```
Demonstrate Interprocess communication & deadlock
class Q{
int n;
boolean valueSet = false;
synchronized int get(){
  while(! valueSet)
   try {
    System.out.println("\n consumer waiting \n");
   wait();
   }
   catch (InterruptedException e){
    System.out.println("InterruptedException caught");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



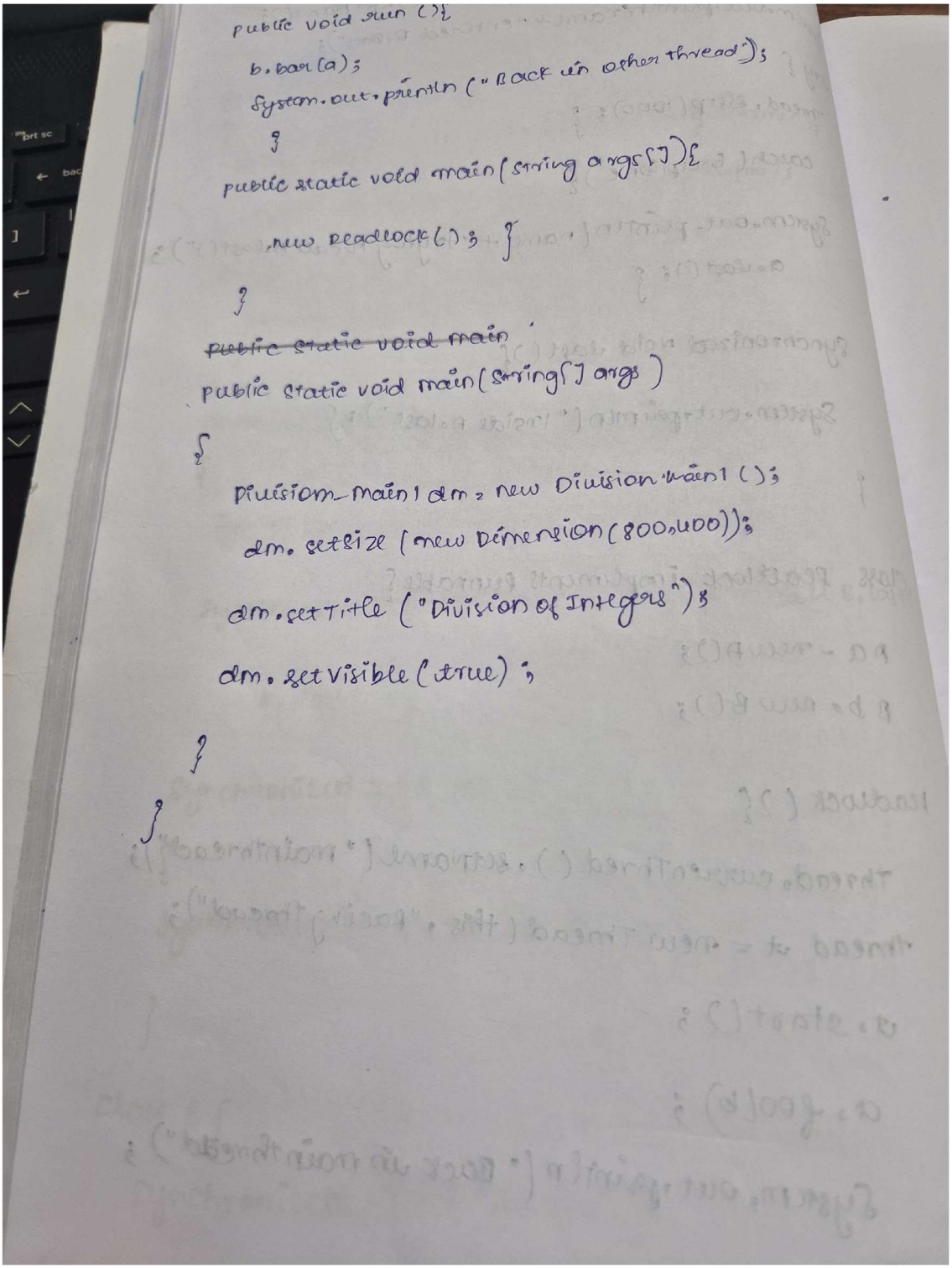
```
        System.out.println("Interrupted exception caught");
        }
        this.n=n;
        valueSet = true;
        System.out.println("Put:"+n);
        System.out.println("\n intimate consumer \n");
        notify();
        }
    }
class Producer implements Runnable {
    Q q;
Producer(Q 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
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 Fcfixed {
    public static void main(String args[]){
    Q q = new Q();
    new Producer(q);
    new Consumer(q);
    System.out.println("Press control-C to stop.");
    }
}
```



Demonstration of Deadlock

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

System.out.println( name + "trying to call B.last()
                        );
   b.last();}

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

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



```
public void run (){
  b.bar(a);
  System.out.println ("Back in other thread");
  }
public static void main(String args[]){
  new ReadLock(); }
}
public static void main
public static void main(String[] args )
{
  Division_main1 dm = new Division.main1 ();
  dm.setSize (new Dimension(800,400));
  dm.setTitle ("Division of Integers");
  dm.setVisible (true);
}
}
```

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

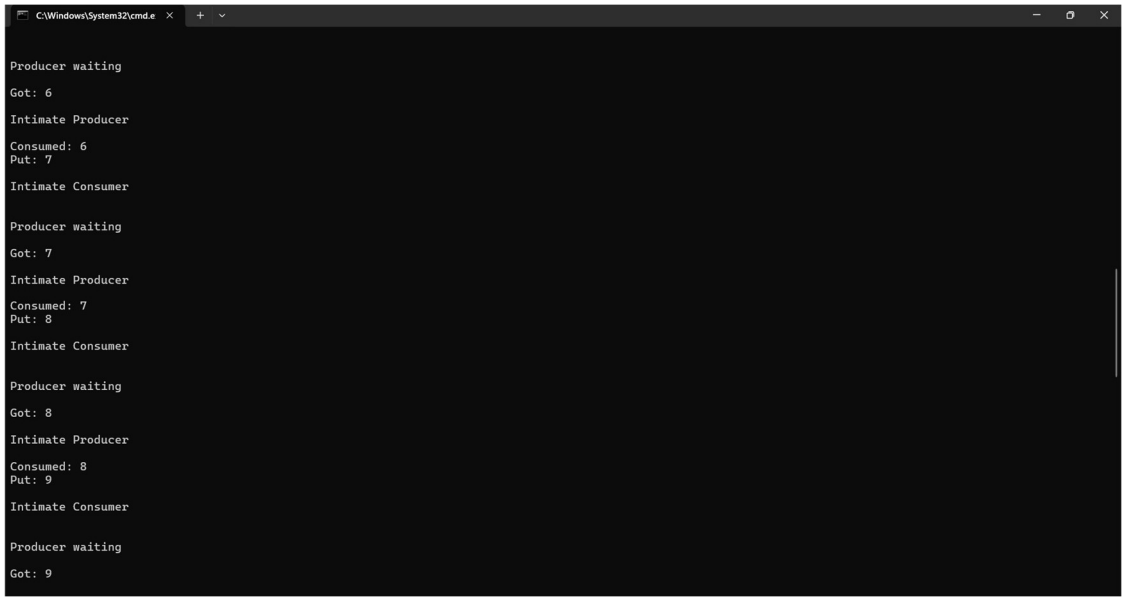
```
synchronized int get() {
    while (!valueSet) {
        try {
            System.out.println("\nConsumer waiting\n");
            wait();
        } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
        }
    }
    System.out.println("Got: " + n);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

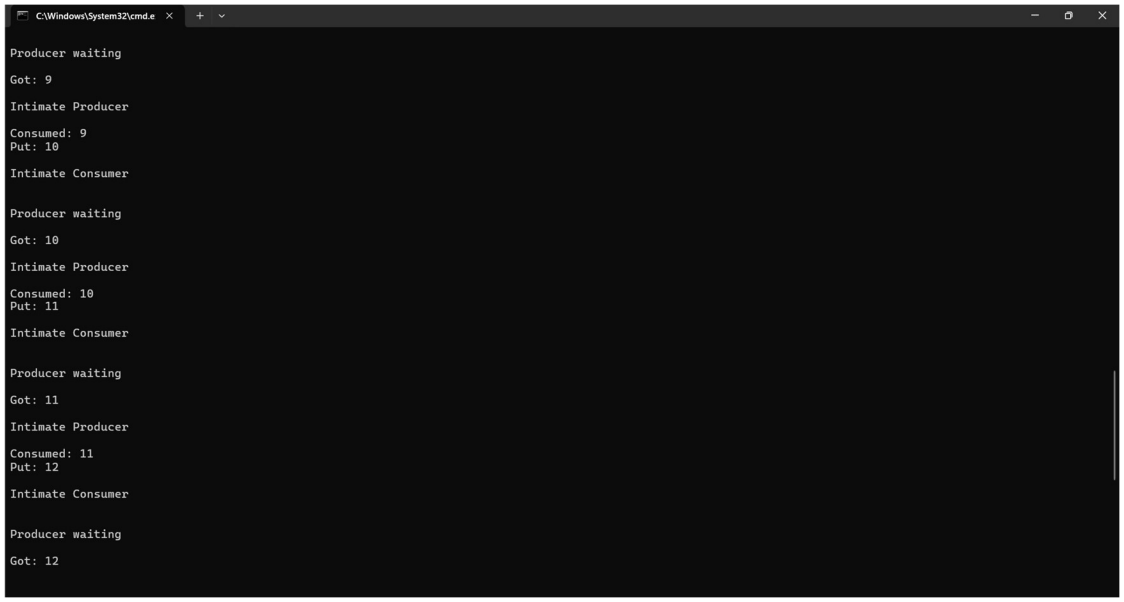
```
    }
}

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) {
```

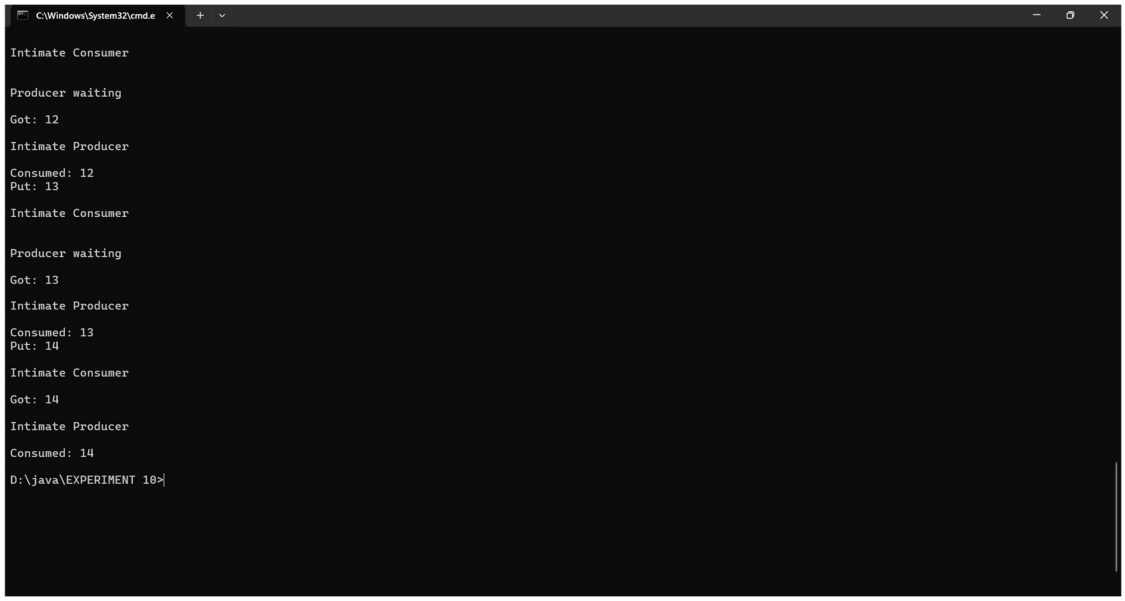
```
            int r = q.get();
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

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.");
    }
}

Demonstrate deadlock
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.foo");

        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("A Interrupted");
        }

        System.out.println(name + " trying to call B.last()");
        b.last();
```

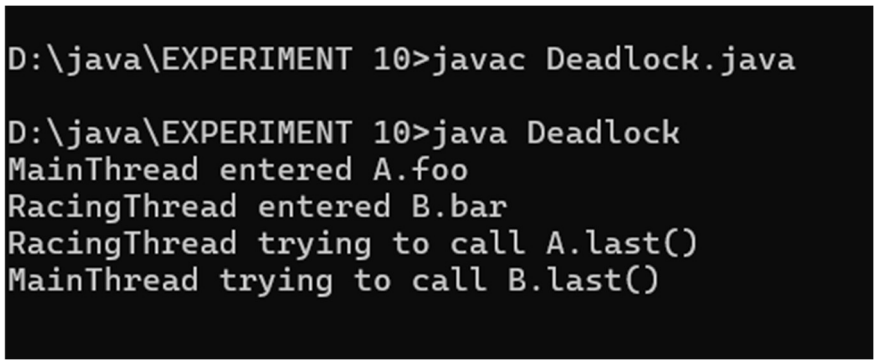
```
    }

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

class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");

        try {
            Thread.sleep(1000);
        } catch (Exception e) {
            System.out.println("B Interrupted");
        }

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

    synchronized void last() {
        System.out.println("Inside B.last");
    }
}

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

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

        a.foo(b); // get lock on A in this thread
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // get lock on B in other thread
        System.out.println("Back in other thread");
    }

    public static void main(String args[]) {
        new Deadlock();
    }
}
```


C:\Windows\System32\cmd.e
Producer waiting
Got: 6
Intimate Producer
Consumed: 6
Put: 7
Intimate Consumer
Producer waiting
Got: 7
Intimate Producer
Consumed: 7
Put: 8
Intimate Consumer
Producer waiting
Got: 8
Intimate Producer
Consumed: 8
Put: 9
Intimate Consumer
Producer waiting
Got: 9


C:\Windows\System32\cmd.e
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


C:\Windows\System32\cmd.e
Intimate Consumer
Producer waiting
Got: 12
Intimate Producer
Consumed: 12
Put: 13
Intimate Consumer
Producer waiting
Got: 13
Intimate Producer
Consumed: 13
Put: 14
Intimate Consumer
Got: 14
Intimate Producer
Consumed: 14
D:\java\EXPERIMENT 10>


D:\java\EXPERIMENT 10>javac Deadlock.java
D:\java\EXPERIMENT 10>java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()