VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Idrees Shaikh (1BM23CS117)

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 **Idrees Shaikh (1BM23CS117),** 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. 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.Seema Patil Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

GITHUB LINK:

https://github.com/IdreesShaikh15/JAVA-LAB-PROGRAMS

Lab Program 1:

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

ALGORITHM:

30/09/24

LAB PROGRAM - 1.

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 the quadratic formula. If the discriminant b^2-4ac is negative, display a message that there are no real solutions.

```
import java.util.Scanner;

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

System.out.println("Enter the value of a:");
double a = scanner.nextDouble();

System.out.println("Enter the value of b:");
double b = scanner.nextDouble();

System.out.println("Enter the value of c:");
double c = scanner.nextDouble();

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

if(d>0)
{
double r1 = (-b + Math.sqrt(d))/(2*a);
double r2 = (-b - Math.sqrt(d))/(2*a);
```

```
System.out.println("The equation has two real roo
System.out.println("Root1:" + r1);
System.out.println("Root2:" + r2);
}
else if (d == 0)
{
double root = -b/(2*a);
System.out.println("The equation has one real root
System.out.println("Root:" + root);
}
else
{
System.out.println("There are no real solutions
}
System.
Scanner.close();
}
}
```

CODE:

```
import java.util.Scanner;


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


        System.out.print("Enter the value of a: ");
        double a = scanner.nextDouble();


        System.out.print("Enter the value of b: ");
        double b = scanner.nextDouble();


        System.out.print("Enter the value of c: ");
        double c = scanner.nextDouble();


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


        if (d > 0) {
            double r1 = (-b + Math.sqrt(d)) / (2 * a);
            double r2 = (-b - Math.sqrt(d)) / (2 * a);
            System.out.println("The equation has two real roots:");
            System.out.println("Root 1: " + r1);
```

```
            System.out.println("Root 2: " + r2);
        } else if (d == 0) {
            double root = -b / (2 * a);
            System.out.println("The equation has one real root:");
            System.out.println("Root: " + root);
        } else {
            System.out.println("There are no real solutions.");
        }

        scanner.close();
    }
}
```

OUTPUT:

```
}
}
```

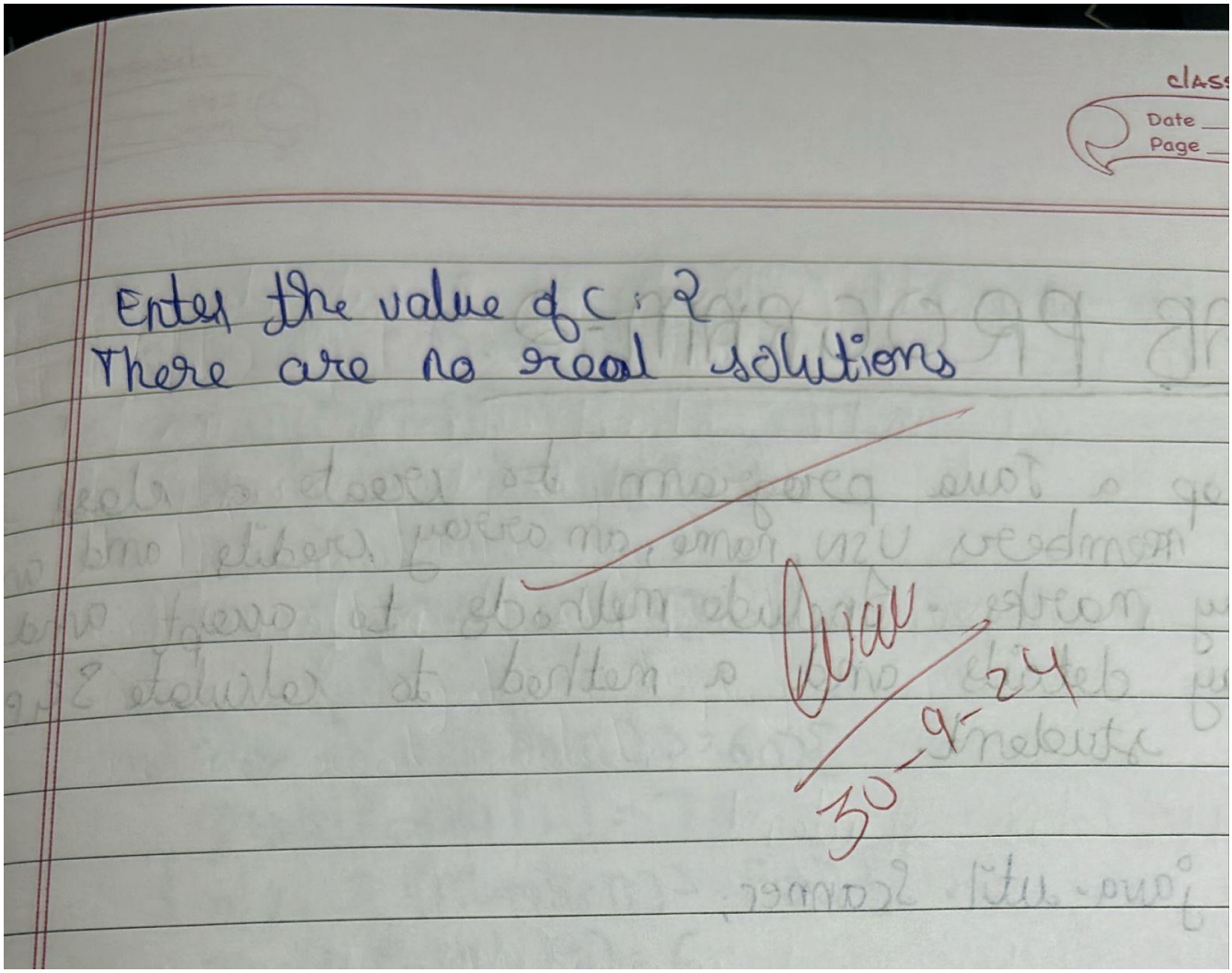
OUTPUT 1:

```
Enter the value of a: 3
Enter the value of b: 10
Enter the value of c: 2
The equation has two real roots:
Root1 : -0.21370035215310867
Root2 : -3.119632981180244
```

OUTPUT 2:

```
Enter the value of a: 5
Enter the value of b: 3
```

Enter the value of c : 2
There are no real solutions



```
D:\1BM23CS117>java QuadraticEquation
Enter the value of a: 7
Enter the value of b: 9
Enter the value of c: 6
There are no real solutions.

D:\1BM23CS117>java QuadraticEquation
Enter the value of a: 9
Enter the value of b: 3
Enter the value of c: 2
There are no real solutions.

D:\1BM23CS117>java QuadraticEquation
Enter the value of a: 3
Enter the value of b: 10
Enter the value of c: 2
The equation has two real roots:
Root 1: -0.21370035215310867
Root 2: -3.1196329811802244
```

LAB PROGRAM 2:

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

ALGORITHM:

Monday
07/10/24

LAB 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.

```
import java.util.Scanner;

class Stud_details{
int marks[]=new int[8];
int credits[]={3,3,3,3,3,3,3,3};
String usn,name;
int gradePoints[]=new int[8];
double sgpa;
Scanner sc=new Scanner(System.in);

void getDetails(){
System.out.print("Enter USN:");
usn=sc.next();
System.out.print("Enter name:");
name=sc.next();
System.out.printf("Enter marks for 8 subjects:");
for(int i=0;i<8;i++)
{
system.out.println("Marks of subject"+(i+1)+":");
marks[i]=sc.nextInt();
calculateGradePoint(i);
}
}

void calculateGradePoint(int i)
```

```
{
 if (marks[i]>=90){
  gradePoints[i]=10;
 } else if (marks[i]>=80){
  gradePoints[i]=9;
 } else if (marks[i]>=70){
  gradePoints[i]=8;
 } else if (marks[i]>=60){
  gradePoints[i]=7;
 } else if (marks[i]>=50){
  gradePoints[i]=6;
 } else if (marks[i]>=40){
  gradePoints[i]=5;
 } else {
  gradePoints[i]=0;
 }
 }

 void calculateSgpa()
 {
  int totalCredits=0;
  int totalMarks=0;
  for(int i=0;i<8;i++)
  {
   totalCredits += credits[i];
   totalMarks += credits[i]* gradePoints[i]
  }
  sgpa=(double) totalMarks/totalCredits;
 }
 void display()
 {
  System.out.println("\nUSN:"+usn);
  System.out.println("Name:"+name);
```

```
System.out.println("Marks of 8 subjects:");
for(int i=0; i<8; i++) {
System.out.println("Subject"+(i+1)+":"+marks[i]+
                   "Grade Point:"+gradePoints[i]+"\n");
}
System.out.println("SGPA"+String.format("%.2f",sgpa));
}
}
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 the details of the
                    student"+(j+1)+":");
s1[j].getDetails();
}
for(int j=0; j<3; j++)
{
s1[j].calculateSgpa();
}
for(int j=0; j<3; j++){
System.out.println("\nDetails of student"+
                   (j+1)+":");
s1[j].display();
}
}
}
```

OUTPUT:

USN:1B
Name:
Marks
Subje
Subje
Sub
Sub
Sub
Su
Su
Su
S
S

CODE:

```
import java.util.Scanner;


class Stud_details {


    int marks[] = new int[8];
    int credits[] = {3, 3, 3, 3, 3, 3, 3, 3};
    String usn, name;
    int gradePoints[] = new int[8];
    double sgpa;
    Scanner sc = new Scanner(System.in);


    void getDetails() {
        System.out.print("Enter USN: ");
        usn = sc.next();
        System.out.print("Enter Name: ");
        name = sc.next();
        System.out.println("Enter marks for 8 subjects:");
        for (int i = 0; i < 8; i++) {
            System.out.print("Marks of Subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
            calculateGradePoint(i);
        }
    }
```

```
void calculateGradePoint(int i) {
    if (marks[i] >= 90) {
        gradePoints[i] = 10;
    } else if (marks[i] >= 80) {
        gradePoints[i] = 9;
    } else if (marks[i] >= 70) {
        gradePoints[i] = 8;
    } else if (marks[i] >= 60) {
        gradePoints[i] = 7;
    } else if (marks[i] >= 50) {
        gradePoints[i] = 6;
    } else if (marks[i] >= 40) {
        gradePoints[i] = 5;
    } else {
        gradePoints[i] = 0;
    }
}

void calculatesgpa() {
    int totalCredits = 0;
    int totalMarks = 0;
    for (int i = 0; i < 8; i++) {
        totalCredits += credits[i];
        totalMarks += gradePoints[i] * credits[i];
    }
```

```
        sgpa = (double) totalMarks / totalCredits;
    }

    void display() {
        System.out.println("\nUSN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Marks of 8 subjects:");
        for (int i = 0; i < 8; i++) {
            System.out.println("Subject " + (i + 1) + ": " + marks[i] + " (Grade Point: " + gradePoints[i] + ")");
        }
        System.out.println("SGPA: " + String.format("%.2f", sgpa));
    }
}

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 the details for Student " + (j + 1) + ":");
            s1[j].getDetails();
        }
```

```
        for (int j = 0; j < 3; j++) {
            s1[j].calculatesgpa();
        }
        for (int j = 0; j < 3; j++) {
            System.out.println("\nDetails of Student " + (j + 1) + ":");
            s1[j].display();
        }
    }
}


}
```

OUTPUT:

OUTPUT:

USN: 1BM23CS117
Name: Idrees
Marks of 8 subjects:
Subject1: 89 (Grade Point: 9)
Subject2: 97 (Grade Point: 10)
Subject3: 93 (Grade Point: 10)
Subject4: 92 (Grade Point: 10)
Subject5: 91 (Grade Point: 10)
Subject6: 90 (Grade Point: 10)
Subject7: 88 (Grade Point: 9)
Subject8: 95 (Grade Point: 10)
SGPA: 9.75

16-10-24

```
USN: 1BM23CS500
Name: Virat
Marks of 8 subjects:
Subject 1: 56 (Grade Point: 6)
Subject 2: 78 (Grade Point: 8)
Subject 3: 99 (Grade Point: 10)
Subject 4: 78 (Grade Point: 8)
Subject 5: 55 (Grade Point: 6)
Subject 6: 78 (Grade Point: 8)
Subject 7: 99 (Grade Point: 10)
Subject 8: 89 (Grade Point: 9)
SGPA: 8.13
USN: 1BM23CS117
Name: Idrees
Marks of 8 subjects:
Subject 1: 89 (Grade Point: 9)
Subject 2: 97 (Grade Point: 10)
Subject 3: 93 (Grade Point: 10)
Subject 4: 92 (Grade Point: 10)
Subject 5: 91 (Grade Point: 10)
Subject 6: 90 (Grade Point: 10)
Subject 7: 88 (Grade Point: 9)
Subject 8: 95 (Grade Point: 10)
SGPA: 9.75
```

LAB PROGRAM 3:

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

ALGORITHM:

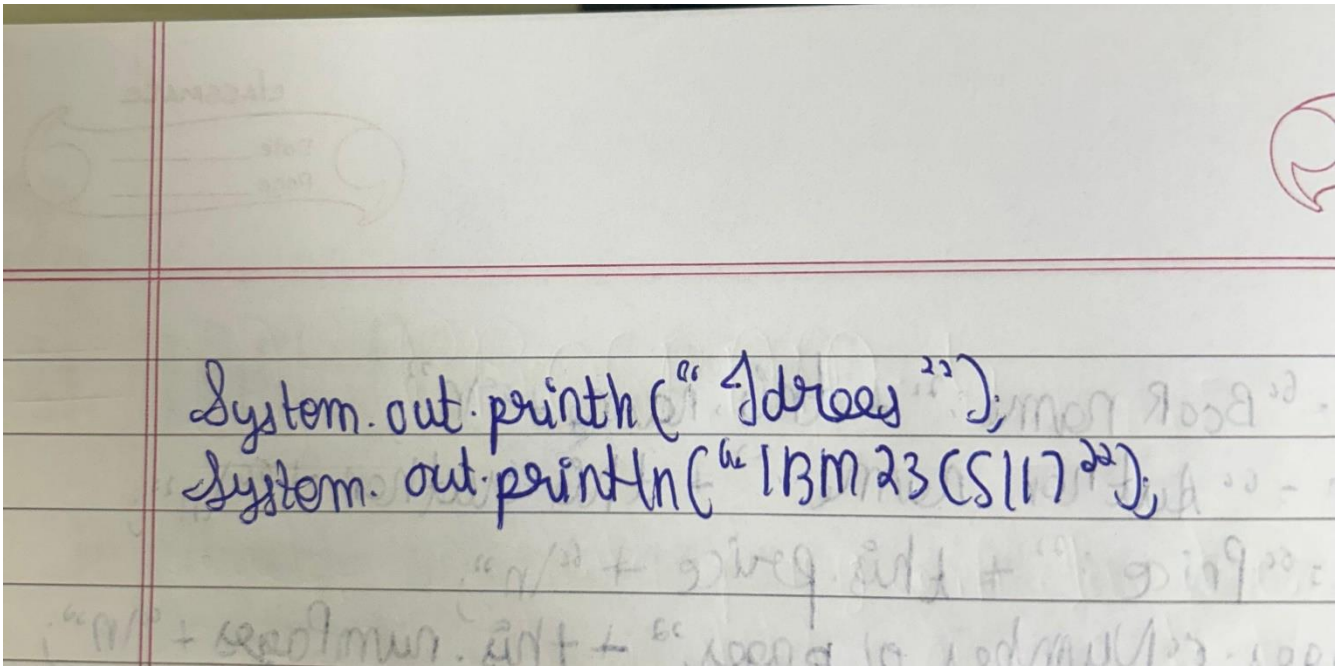
14/10/24
Monday

LAB PROGRAM-3:

CREATE A CLASS BOOK WHICH CONTAINS FOUR MEMBERS:

```
import java.util.Scanner;

class Book{
private String name;
private String author;
private int price;
private int numPages;

Book(String name, String author, int price, int numPages)
{

  this.name = name;
  this.author = author;
  this.price = price;
  this.numPages = numPages;
}

  public String toString(){
  String name, author, price, numPages;
```

```
name = "Book name: " + this.name + "\n";
author = "Author name: " + this.author + "\n";
price = "Price: " + this.price + "\n";
numPages: "Number of pages: " + this.numPages + "\n";
return name + author + price + numPages;
}}

public class Books{
public static void main (String args[]){
String name, author;
int price, numPages;
Scanner sc = new Scanner (System.in);
System.out.println("Enter number of books:");
int n = x.nextInt();
Book[] books = new Book[n];

for (int i=0; i<n; i++){
System.out.println("Enter the name of the " + (i+1) + book:");
name = sc.next();
System.out.println("Enter the author of the " + (i+1) + "
book:");

author = sc.next();
System.out.println("Enter the price of the:" + (i+1) + "
book:");

price = sc.nextInt();
System.out.println("Enter the number of pages of
the " + (i+1) + "book:");

numPages = sc.nextInt();

System.out.println("Result");
books(i) = new Book(name, author, price, numPage);
System.out.println(books[i]);}
```



CODE:

```
import java.util.Scanner;


class Book {
    private String name;
    private String author;
    private int price;
    private int numPages;


    Book(String name, String author, int price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
```

```
        this.numPages = numPages;
    }

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

public class Books {
    public static void main(String args[]) {
        String name, author;
        int price, numPages;
        Scanner sc = new Scanner(System.in);

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

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

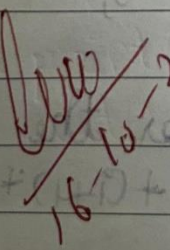
```
        for (int i = 0; i < n; i++) {
            System.out.println("Enter the name of the " + (i + 1) + " Book:");
            name = sc.nextLine();

            System.out.println("Enter the name of the author of the " + (i + 1) + " Book:");
            author = sc.nextLine();

            System.out.println("Enter the price of the " + (i + 1) + " Book:");
            price = sc.nextInt();

            System.out.println("Enter the number of pages of the " + (i + 1) + " Book:");
            numPages = sc.nextInt();

            sc.nextLine();

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

            System.out.println("Result:");
            System.out.println(books[i]);
        }
    }
}
```

OUTPUT:

:OUTPUT:

Enter the number of books: 1
Enter the name of 1 books: The Alchemist
Enter the author of 1 book: Idrees
Enter the price of 1 book: 450
Enter the number of book: 300

Result:
Book name : The Alchemist
Author name: Idrees
Price : 450
Number of pages: 300



```
C:\1BM23CS117>java Books
Enter the number of books: 2
Enter the name of the 1 Book:
The Gentlemen Of India
Enter the name of the author of the 1 Book:
Idrees
Enter the price of the 1 Book:
675
Enter the number of pages of the 1 Book:
432
Result:
Book name: The Gentlemen Of India
Author name: Idrees
Price: 675
Number of pages: 432

Enter the name of the 2 Book:
The Alchemist
Enter the name of the author of the 2 Book:
Paulo Coehlo
Enter the price of the 2 Book:
250
Enter the number of pages of the 2 Book:
400
Result:
Book name: The Alchemist
Author name: Paulo Coehlo
Price: 250
Number of pages: 400
```

LAB PROGRAM 4:

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

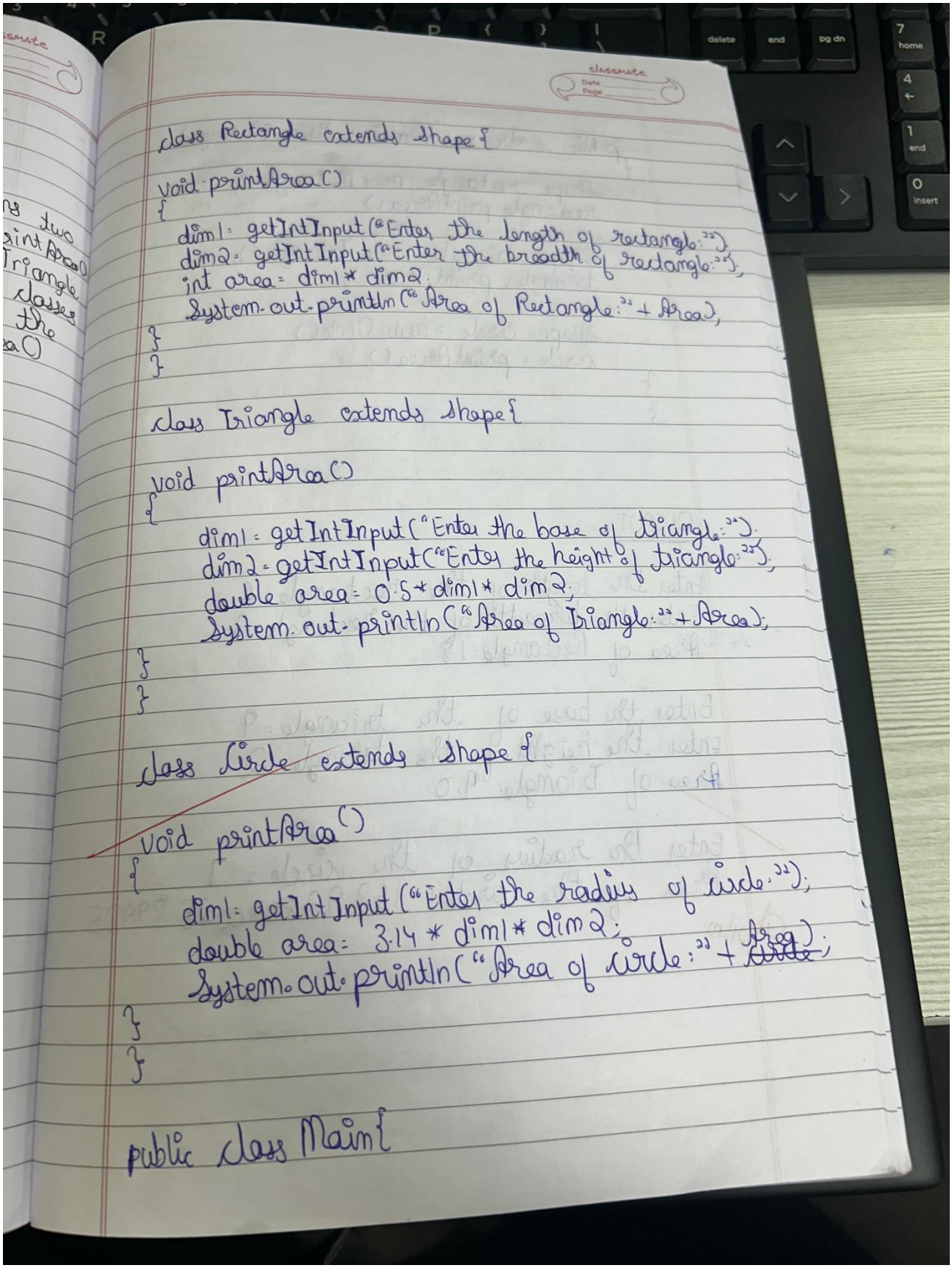
ALGORITHM:

Monday
21/10/24

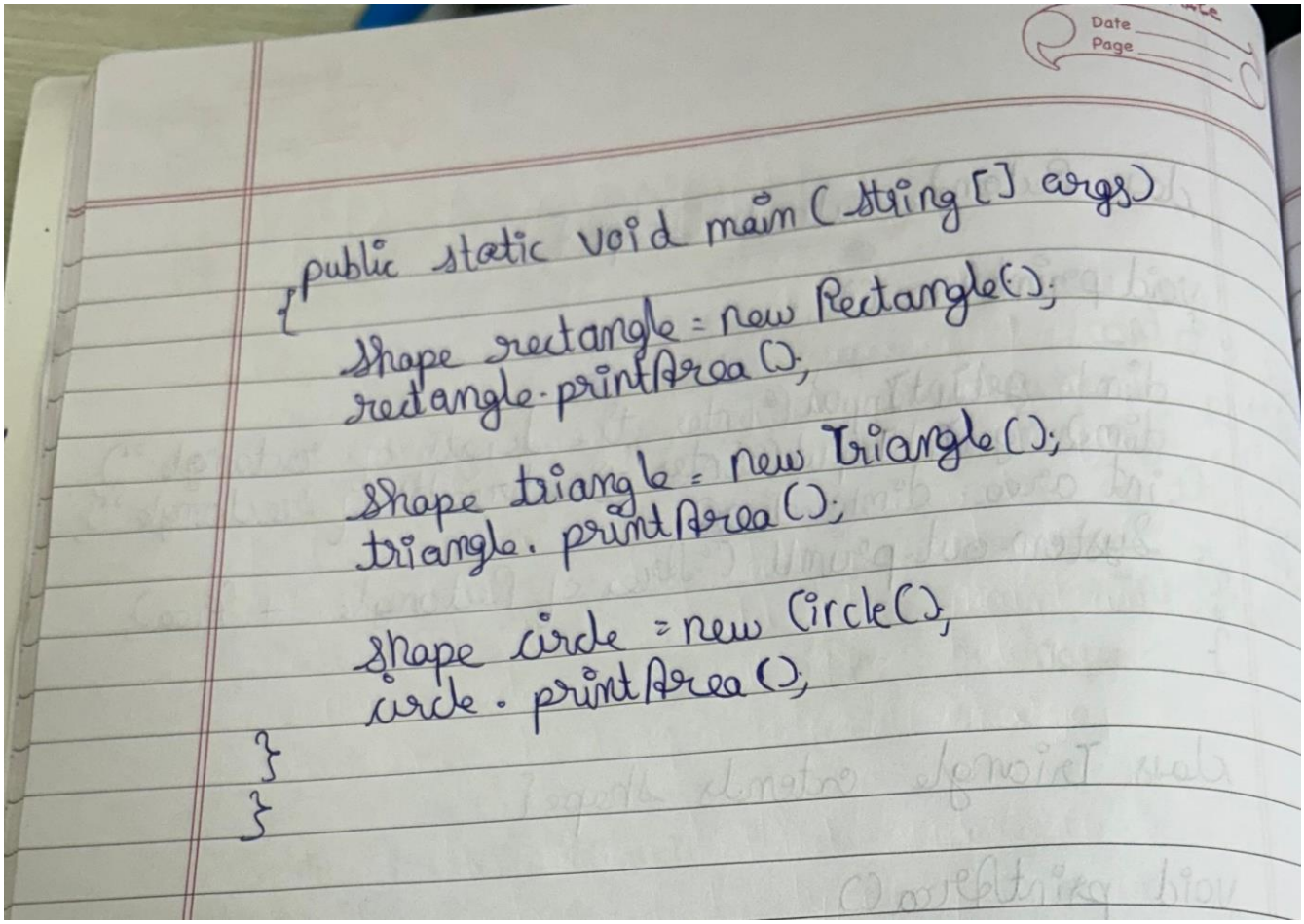
LAB PROGRAM-4:

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

```
import java.util.Scanner;

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

public int getIntInput (String prompt)
{
  System.out.print (prompt);
  return sc.nextInt ();
}

public double getDoubleInput (String prompt)
{
  System.out.print (prompt);
  return sc.nextDouble ();
}

abstract class Shape extends InputScanner
{
  int dim1, dim2;
  abstract void printArea ();
}
```



```
class Rectangle extends Shape{

void printArea()
{
    dim1 = getIntInput("Enter the length of rectangle:");
    dim2 = getIntInput("Enter the breadth of rectangle:");
    int area = dim1 * dim2;
    System.out.println("Area of Rectangle:" + Area);
}
}

class Triangle extends Shape{

void printArea()
{
    dim1 = getIntInput("Enter the base of triangle:");
    dim2 = getIntInput("Enter the height of triangle:");
    double area = 0.5 * dim1 * dim2;
    System.out.println("Area of Triangle:" + Area);
}
}

class Circle extends Shape{

void printArea()
{
    dim1 = getIntInput("Enter the radius of circle:");
    double area = 3.14 * dim1 * dim2;
    System.out.println("Area of circle:" + Area);
}
}

public class Main{
```



CODE:

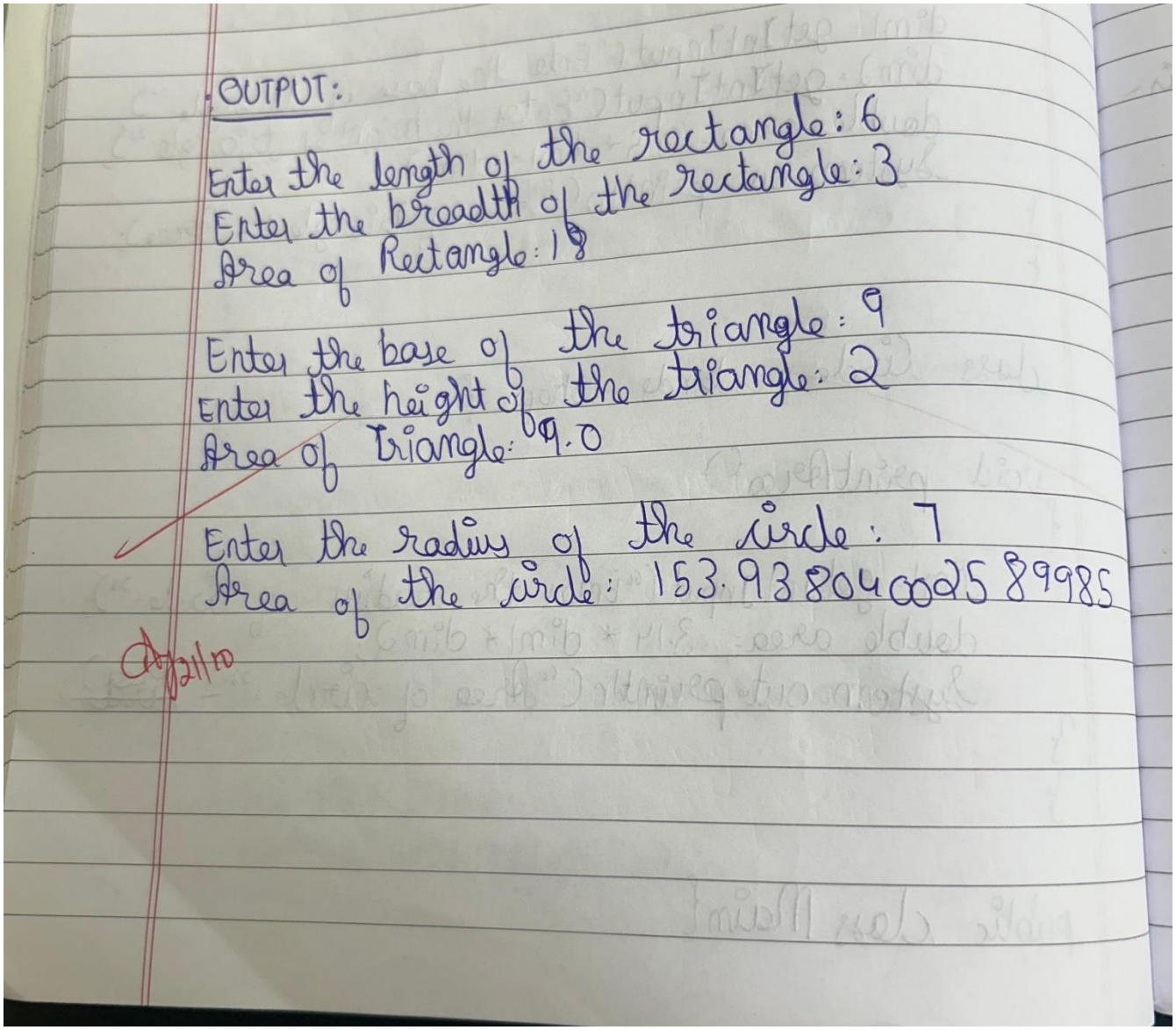
```
import java.util.Scanner;


class InputScanner {

    Scanner sc = new Scanner(System.in);


    public int getIntInput(String prompt) {

        System.out.print(prompt);
```

```
        return sc.nextInt();
    }

    public double getDoubleInput(String prompt) {
        System.out.print(prompt);
        return sc.nextDouble();
    }
}

abstract class Shape extends InputScanner {
    int dim1, dim2;

    abstract void printArea();
}

class Rectangle extends Shape {
    void printArea() {
        dim1 = getIntInput("Enter the length of the rectangle: ");
        dim2 = getIntInput("Enter the breadth of the rectangle: ");
        int area = dim1 * dim2;
        System.out.println("Area of Rectangle: " + area);
    }
}

class Triangle extends Shape {
```

```
    void printArea() {
        dim1 = getIntInput("Enter the base of the triangle: ");
        dim2 = getIntInput("Enter the height of the triangle: ");
        double area = 0.5 * dim1 * dim2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {
    void printArea() {
        dim1 = getIntInput("Enter the radius of the circle: ");
        double area = Math.PI * dim1 * dim1;
        System.out.println("Area of Circle: " + area);
    }
}

public class Main {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle();
        rectangle.printArea();

        Shape triangle = new Triangle();
        triangle.printArea();

        Shape circle = new Circle();
```

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

OUTPUT:



```
D:\1BM23CS117>java Main
Enter the length of the rectangle: 6
Enter the breadth of the rectangle: 3
Area of Rectangle: 18
Enter the base of the triangle: 9
Enter the height of the triangle: 2
Area of Triangle: 9.0
Enter the radius of the circle: 7
Area of Circle: 153.93804002589985
```

LAB 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

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

ALGORITHM:

LAB PROGRAM-5

Develop a java program to create a class that maintains two kinds of account for its customers one called 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 class Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

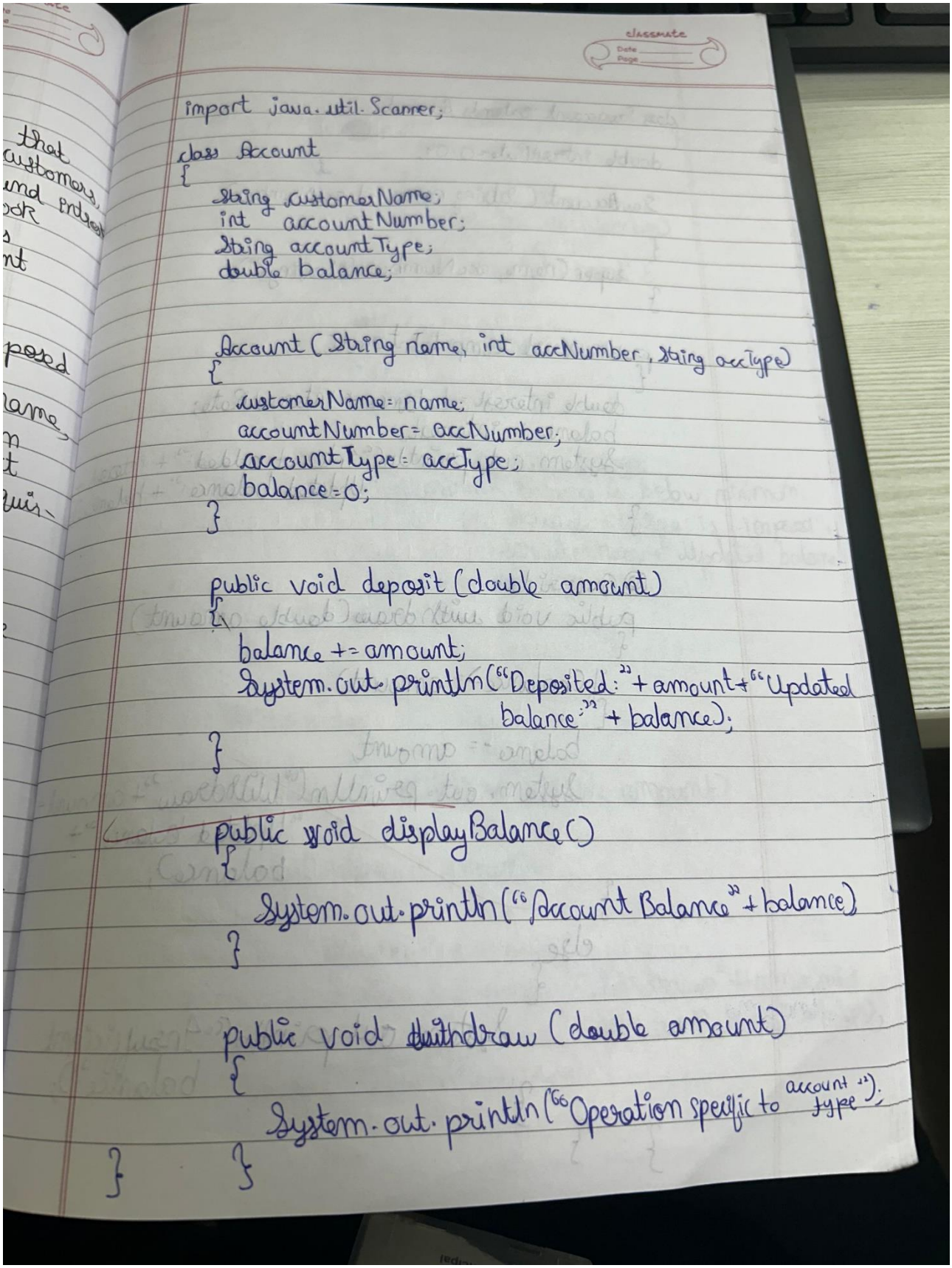
a) Accept deposit from customer and update the balance.

b) display the balance

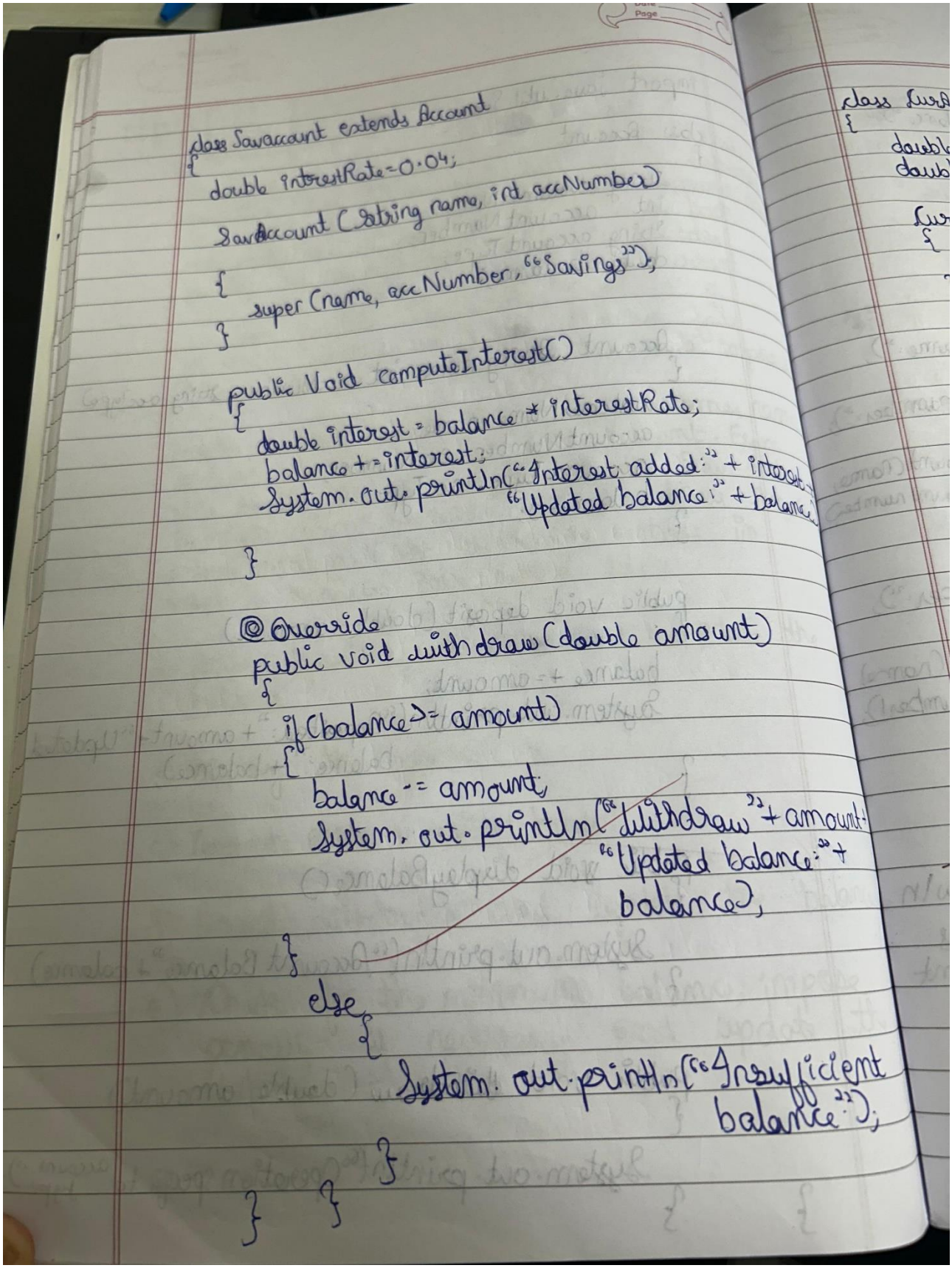
c) Compute and deposit interest.

d) Permit withdrawal and update the balance

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



```
import java.util.Scanner;

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

    Account(String name, int accNumber, String accType)
    {
        customerName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = 0;
    }

    public void deposit(double amount)
    {
        balance += amount;
        System.out.println("Deposited: " + amount + " Updated
                                balance: " + balance);
    }

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

    public void withdraw(double amount)
    {
        System.out.println("Operation specific to account type");
    }
}
```

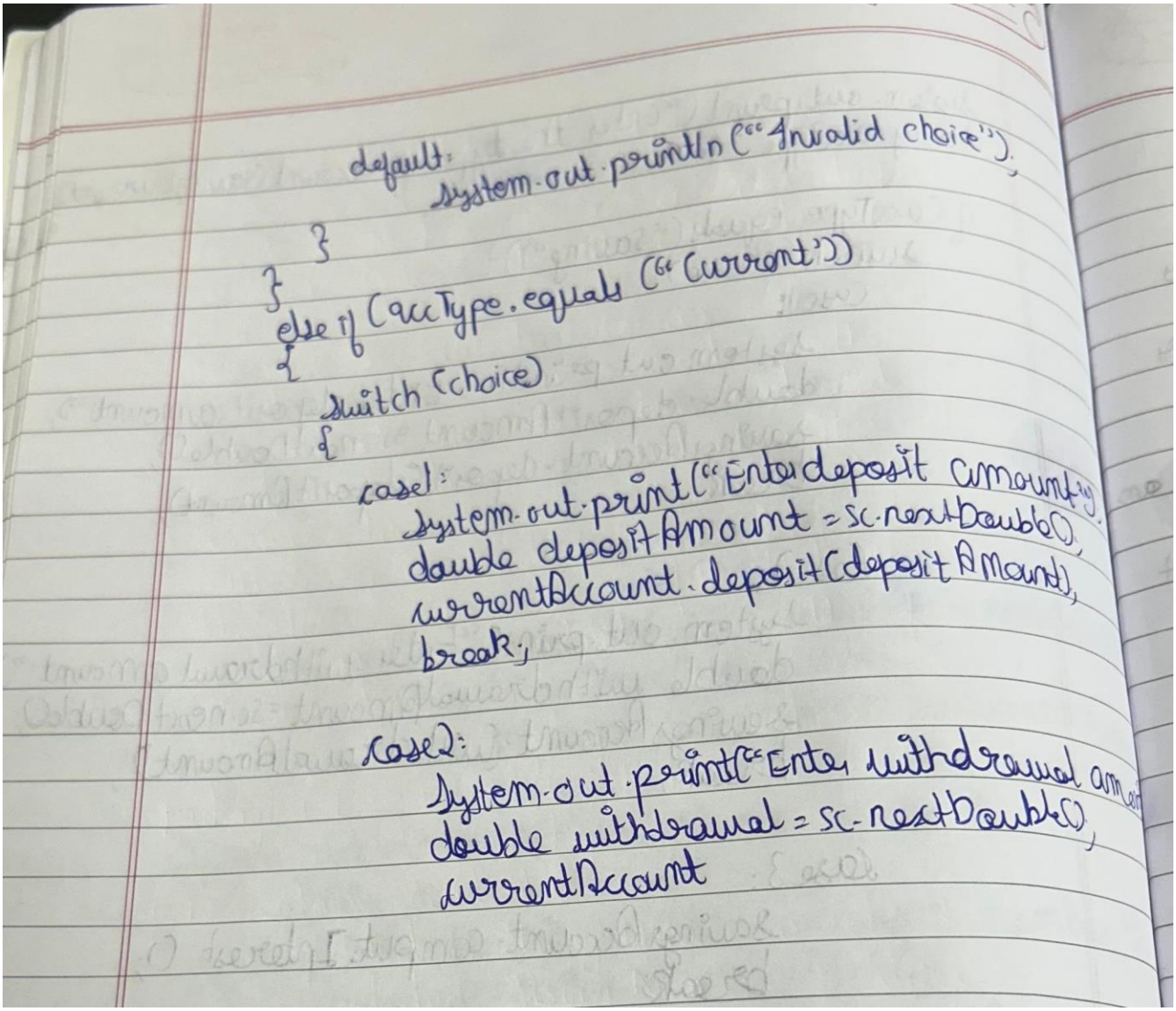


```
class SavAccount extends Account
{
double interestRate=0.04;
SavAccount (String name, int accNumber)
{
} super (name, accNumber, "Savings");

public Void computeInterest()
{
double interest = balance * interestRate;
balance += interest;
System.out.println("Interest added:" + interest+
                   "Updated balance:" + balance);
}

@Override
public void withdraw (double amount)
{
if (balance >= amount)
{
balance -= amount;
System.out.println("withdraw" + amount+
                   "Updated balance:" +
                   balance);
}
else
{
System.out.println("Insufficient
                   balance");
}
}
}
```

```
class CurrAccount extends Account
{
    double minBalance = 500.50;
    double serviceCharge = 50.0;

    CurrAccount (String name, int accNumber)
    {
        super(name, accNumber, "Current");
    }

    public void checkMinBalance()
    {
        if(balance < minBalance)
        {
            balance -= serviceCharge;
            System.out.println("Balance is below minimum.
                service charge is imposed" +
                serviceCharge + "Updated balance
                is" + balance);
        }
    }

    @Override

    public void withdraw(double amount)
    {
        if(balance >= amount)
        {
            balance -= amount;
            System.out.println("Withdrawn" + amount +
                "Updated Balance" + balance);
            checkMinBalance();
        }
        else:
```

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

public class Bank{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter customer name:");
        String name = sc.next();
        System.out.println("Enter account number:");
        int accountnumber = sc.nextInt();
        SavAccount savingsAccount = new SavAccount(name,
                                                   accountnumber);
System.out.println("Enter customer name");
        String name1 = sc.next();
        System.out.println("Enter account number:");
        int accountnumber1 = sc.nextInt();
        CurAccount currentAccount = new CurAccount(name1,
                                                   accountnumber1);


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

```
System.out.print("Enter the type of account(saving/current)");
String accType = sc.next();

if (accType.equals("saving")){
    switch(choice){
        case 1:
            System.out.println("Enter deposit amount:");
            double depositAmount = sc.nextDouble();
            savingsAccount.deposit(depositAmount);
            break;

        case 2:
            System.out.print("Enter withdrawal amount:");
            double withdrawalAmount = sc.nextDouble();
            savingsAccount.(withdrawalAmount);
            break;

        case 3:
            savingsAccount.computeInterest();
            break;

        case 4:
            System.out.println("Customer Name" + savingsAccount
                                .customerName);
            System.out.println("Account number" + savingsAc-
                                count.number);
            System.out.println("Type of account:" + saving
                                Account.accountType);

            savingAccount.displayBalance();
            break;

        case 5,
            system.exit(0);
            break;
```



CODE:

```
import java.util.Scanner;

class Account {

    String customerName;
```

```
int accountNumber;
String accountType;
double balance;

Account(String name, int accNumber, String accType) {
    customerName = name;
    accountNumber = accNumber;
    accountType = accType;
    balance = 0;
}

public void deposit(double amount) {
    balance += amount;
    System.out.println("Deposited: " + amount + ". Updated balance: " + balance);
}

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

public void withdraw(double amount) {
    System.out.println("This operation is specific to account type.");
}
```

```
}

class SavAccount extends Account {
    double interestRate = 0.04;

    SavAccount(String name, int accNumber) {
        super(name, accNumber, "Savings");
    }

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

    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance.");
        }
```

```
    }
}

class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;

    CurAccount(String name, int accNumber) {
        super(name, accNumber, "Current");
    }

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

    @Override
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
```

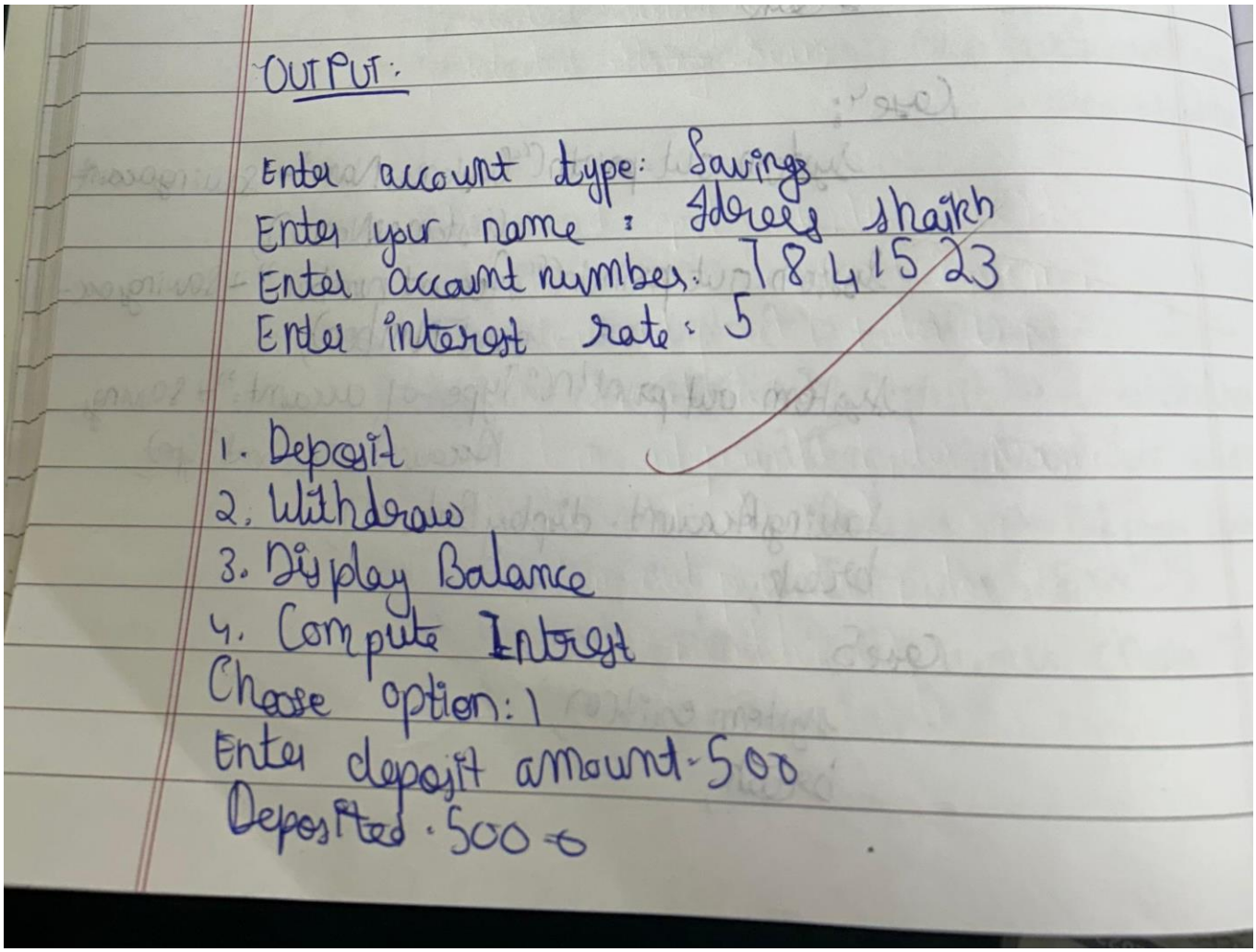
```
            checkMinBalance();
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

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

        System.out.println("Enter customer name:");
        String name = sc.next();
        System.out.println("Enter account number:");
        int accountNumber = sc.nextInt();
        SavAccount savingsAccount = new SavAccount(name, accountNumber);

        System.out.println("Enter customer name:");
        String name1 = sc.next();
        System.out.println("Enter account number:");
        int accountNumber1 = sc.nextInt();
        CurAccount currentAccount = new CurAccount(name1, accountNumber1);
```

```
        while (true) {
            System.out.println("\n-----MENU-----");
            System.out.println("1. Deposit\n2. Withdraw\n3. Compute Interest for Savings Account\n4. Display Account Details\n5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();

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

            if (accType.equals("saving")) {
                switch (choice) {
                    case 1:
                        System.out.print("Enter the deposit amount: ");
                        double depositAmount = sc.nextDouble();
                        savingsAccount.deposit(depositAmount);
                        break;
                    case 2:
                        System.out.print("Enter the withdrawal amount: ");
                        double withdrawalAmount = sc.nextDouble();
                        savingsAccount.withdraw(withdrawalAmount);
                        break;
                    case 3:
                        savingsAccount.computeInterest();
```

```
            break;
        case 4:
            System.out.println("Customer name: " + savingsAccount.customerName);
            System.out.println("Account number: " + savingsAccount.accountNumber);
            System.out.println("Type of Account: " + savingsAccount.accountType);
            savingsAccount.displayBalance();
            break;
        case 5:
            System.exit(0);
            break;
        default:
            System.out.println("Invalid choice.");
    }
} else if (accType.equals("current")) {
    switch (choice) {
        case 1:
            System.out.print("Enter the deposit amount: ");
            double depositAmount = sc.nextDouble();
            currentAccount.deposit(depositAmount);
            break;
        case 2:
            System.out.print("Enter the withdrawal amount: ");
            double withdrawalAmount = sc.nextDouble();
```

```
                currentAccount.withdraw(withdrawalAmount);
                break;
            case 3:
                System.out.println("Current accounts do not earn interest.");
                break;
            case 4:
                System.out.println("Customer name: " + currentAccount.customerName);
                System.out.println("Account number: " + currentAccount.accountNumber);
                System.out.println("Type of Account: " + currentAccount.accountType);
                currentAccount.displayBalance();
                break;
            case 5:
                System.exit(0);
                break;
            default:
                System.out.println("Invalid choice.");
        }
      } else {
        System.out.println("Invalid account type.");
      }
    }
  }
}
```

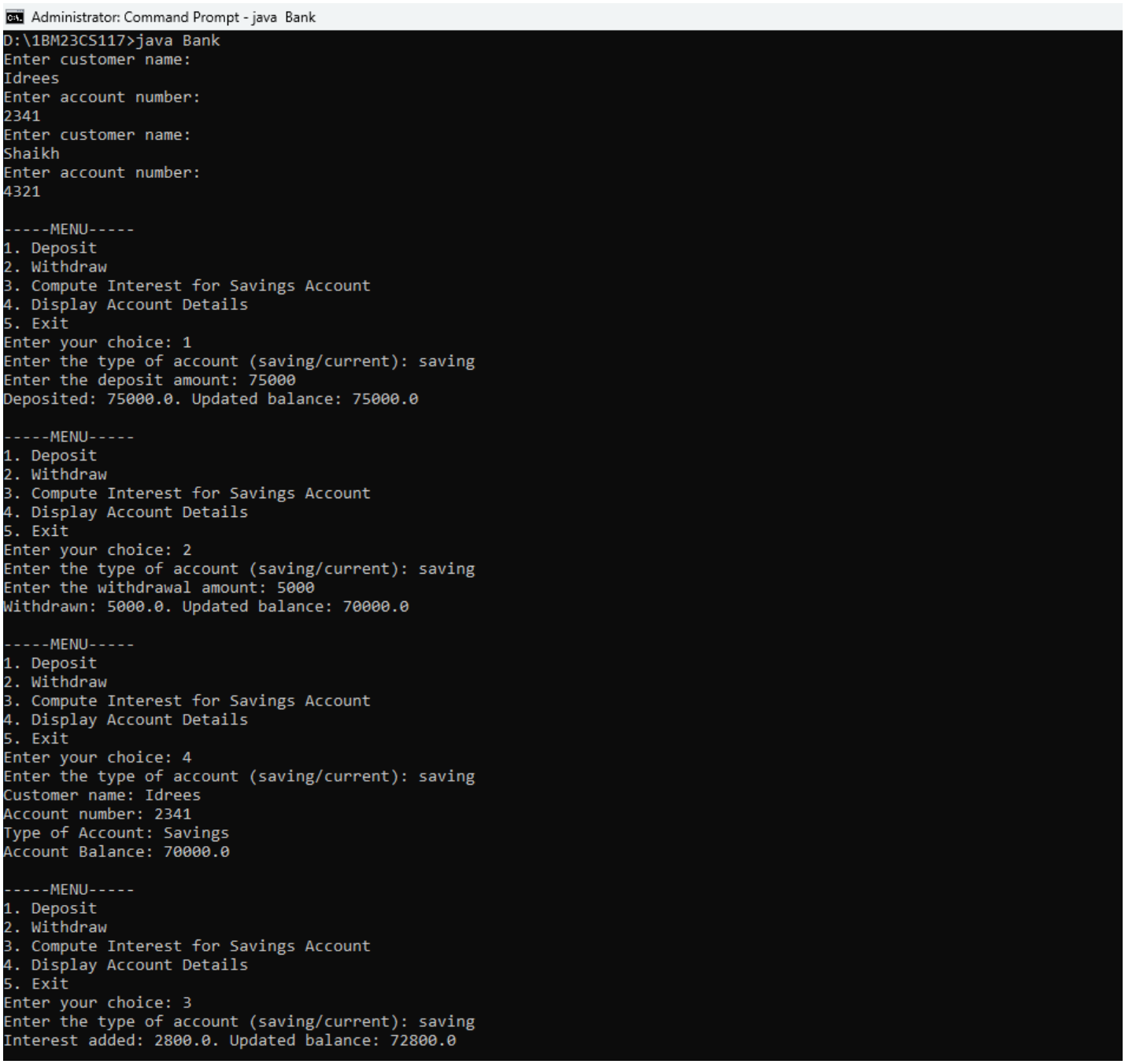
OUTPUT:


OUTPUT:
Enter account type: Savings
Enter your name : Idrees shaikh
Enter account number: 784523
Enter interest rate: 5
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
Choose option: 1
Enter deposit amount: 500
Deposited: 500.0

1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
Choose option
Enter withdraw amount: 50
withdrawn: 50

1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest
Choose option: 4
Interest deposited: 22.5

18/11/24


Administrator: Command Prompt - java Bank
D:\1BM23CS117>java Bank
Enter customer name:
Idrees
Enter account number:
2341
Enter customer name:
Shaikh
Enter account number:
4321
-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 1
Enter the type of account (saving/current): saving
Enter the deposit amount: 75000
Deposited: 75000.0. Updated balance: 75000.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 2
Enter the type of account (saving/current): saving
Enter the withdrawal amount: 5000
Withdrawn: 5000.0. Updated balance: 70000.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 4
Enter the type of account (saving/current): saving
Customer name: Idrees
Account number: 2341
Type of Account: Savings
Account Balance: 70000.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): saving
Interest added: 2800.0. Updated balance: 72800.0

```
-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): saving
Interest added: 200.0. Updated balance: 5200.0

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 4
Enter the type of account (saving/current): saving
Customer name: idrees
Account number: 23
Type of Account: Savings
Account Balance: 5200.0

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 2
Enter the type of account (saving/current): current
Enter the withdrawal amount: 5000
Insufficient balance.

-----MENU-----
1. Deposit
2. Withdraw
3. Compute Interest for Savings Account
4. Display Account Details
5. Exit
Enter your choice: 3
Enter the type of account (saving/current): current
Current accounts do not earn interest.
```

LAB PROGRAM 6:

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

ALGORITHM:

LAB PROGRAM-6:

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

```
package CIE

public class Student {
    protected String USN;
    protected String Name;
    protected int Sem;

public void inputStudentDetails() {
    Scanner s = new Scanner(System.in);
    System.out.print("Enter USN: ");
    USN = s.nextLine();
    System.out.print("Enter Name: ");
    Name = s.nextLine();
    System.out.print("Enter Sem: ");
    Sem = s.nextLine();
}
```

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

```
package CIE
public class Internals extends Student{
    private int[] internal marks;

    public void inputCIEmarks(){
        Scanner= new Scanner(System.in);
        System.out.println("Enter internal marks for 5 subjects:");
        for(int i=0; i<5; i++){
            System.out.println("Enter marks for subject"+(i+1)+":");
            marks[i]= s.nextInt();
        }
    }
}
```

```
package SEE;

import CIE.Student;
import java.util.Scanner;
public class Externals extends Students{
    private int[] external marks;

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

```
        System.out.println("Enter SEE marks for 5 subjects:");
        for(int i=0;i<5;i++){
            System.out.print("Enter SEE marks for subject:"
                    +(i+1)+":");
            marks[i]=s.nextInt();
        }
    }

    public void calculateFinalMarks(){
        for(int i=0;i<5;i++){
            finalMarks[i]=marks[i]+this.marks[i];
        }
    }

    public void displayFinalMarks(){
        displayStudentDetails();
        System.out.println("Final marks:");
        for(int i=0;i<5;i++){
            System.out.println("Subject:"+(i+1)+":"+
                    finalMarks[i]);
        }
    }
}

import SEE.CIE.Internals
import SEE.Externals

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

        System.out.print("Enter number of stu
        int n=s.nextInt();
        s.nextLine();
```

```
Externals[] students = new Externals[n];

for(int i=0; i<n; i++){
    students[i] = new Externals();
    System.out.println("\nEnter details for student" +
                       (i+1) + "i");

    students[i].inputStudentDetails();
    students[i].inputCIEmarks();
    students[i].inputSEEmarks();
    students[i].calculateFinalMarks();
}

System.out.println("\nDisplaying final marks for
                    all students");

for(int i=0; i<n; i++){
    students[i].displayFinalMarks();
}

s.close();
}
}
```

CODE:

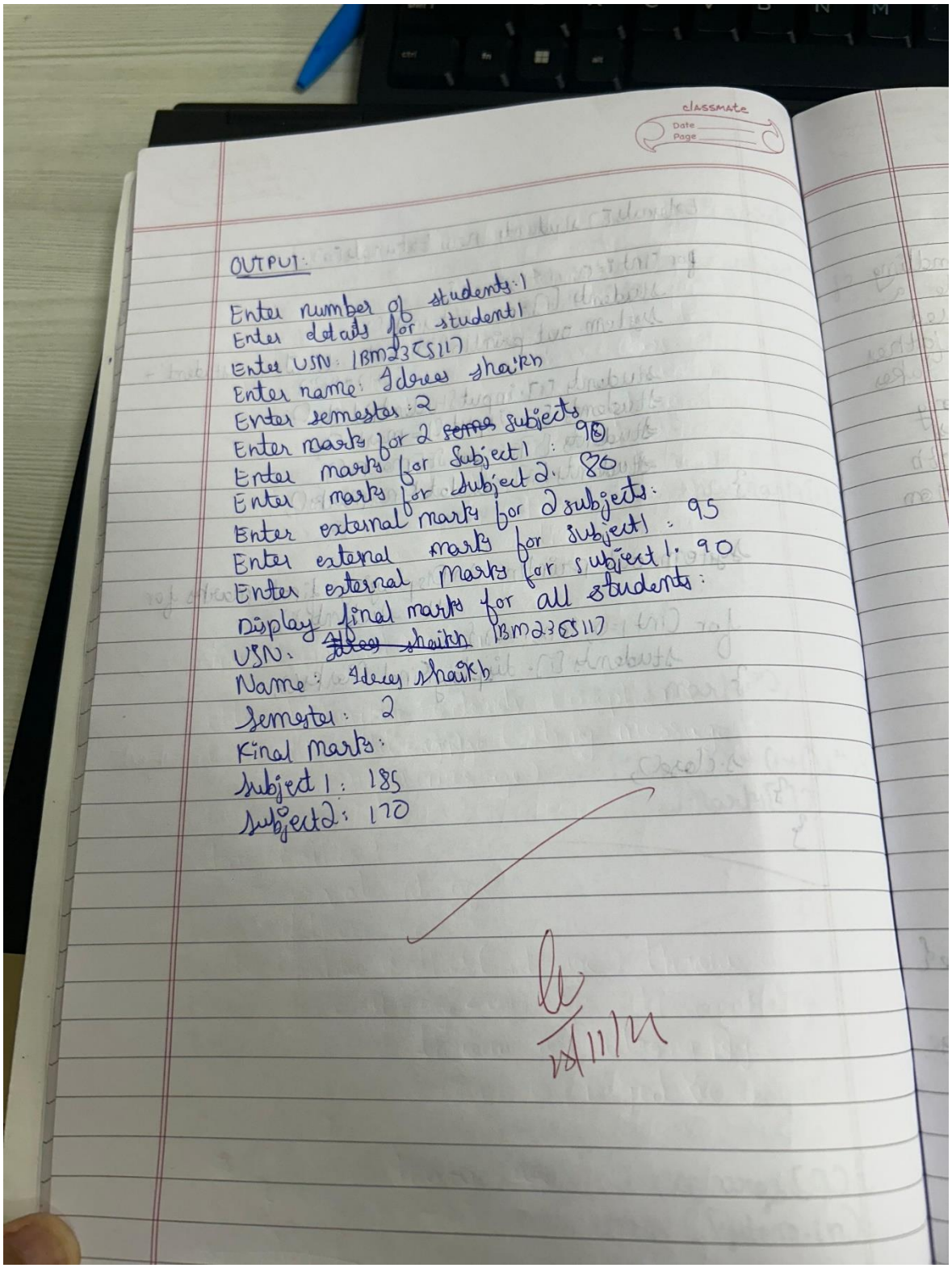
CIE

```
import java.util.Scanner;

public class Internals extends Student {

public int marks[] = new int[5];

 public void inputCIEmarks() {

 Scanner sc = new Scanner(System.in);

System.out.println("Enter marks for 5 subjects:");

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

            System.out.print("Enter marks for subject " + (i + 1) + ": ");

            marks[i] = sc.nextInt();

}

}

public void calculateFinalMarks() {

 }

       public void displayFinalMarks() {

        displayStudentDetails();

        System.out.println("CIE Final Marks:");

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

            System.out.println("Subject " + (i + 1) + ": " + marks[i]);

        }


      }

}
```

```
package SEE;

import CIE.Internals;

import CIE.Student;


import java.util.Scanner;


public class Externals extends Student {


   public int marks[] = new int[5];

   public int finalMarks[] = new int[5];



    public Externals() {

        marks = new int[5];

       finalMarks = new int[5];

    }


       public void inputSEEmarks() {


Scanner sc = new Scanner(System.in);



        System.out.println("Enter SEE marks for 5 subjects:");


                for (int i = 0; i < 5; i++) {
```

```
            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
        }
    }

    public void calculateFinalMarks() {

        for (int i = 0; i < 5; i++) {
                    finalMarks[i] = marks[i];         }
    }


    public void displayFinalMarks() {

        displayStudentDetails();

                System.out.println("Final Marks for 5 subjects:");

        for (int i = 0; i < 5; i++) {
            System.out.println("Subject " + (i + 1) + ": " + finalMarks[i]);  // Print final marks
        }
    }
}
```

OUTPUT:


classmate
Date
Page
OUTPUT:
Enter number of students: 1
Enter details for student1
Enter USN: 1BM23CS117
Enter name: Idrees shaikh
Enter semester: 2
Enter marks for 2 subjects
Enter marks for subject1: 90
Enter marks for subject2: 80
Enter external marks for 2 subjects:
Enter external marks for subject1: 95
Enter external marks for subject1: 90
Display final marks for all students:
USN: 1BM23CS117
Name: Idrees shaikh
Semester: 2
Final marks:
Subject1: 185
Subject2: 170
14/11/24

:

```
C:\Windows\System32\cmd.e X + v

Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\STUDENT\Desktop\1BM23CS117\LAB6>javac main.java

C:\Users\STUDENT\Desktop\1BM23CS117\LAB6>java main

Enter the number of students:
2

Enter details for Students here1:
Enter your usn here:
117
Enter your name here:
Idrees
Enter your semester here:
3
Enter your CIE marks for the 5 subjects here:
Subject1:9
Subject2:98
Subject3:99
Subject4:95
Subject5:92
CIE marks are as follows:

Subject1:9
Subject2:98
Subject3:99
Subject4:95
Subject5:92
Enter the 5 SEE Marks here:

Subject1:88

Subject2:89

Subject3:98

Subject4:94

Subject5:90

Enter details for Students here2:
Enter your usn here:
116
Enter your name here:
Harish
Enter your semester here:
3
Enter your CIE marks for the 5 subjects here:
Subject1:89
Subject2:99
Subject3:95
Subject4:1
Subject5:87
CIE marks are as follows:

Subject1:89
Subject2:99
Subject3:95
Subject4:1
Subject5:87
Enter the 5 SEE Marks here:

Subject1:99

Subject2:98

Subject3:95

Subject4:56

Subject5:32

Finalmarks of students:

Student1:
Name:Idrees
USN:117
Semester:3

The final marks of the 5 subjects are:

Subject1:48
Subject2:93
Subject3:98
Subject4:94
Subject5:91
Student2:
Name:Harish
USN:116
Semester:3

The final marks of the 5 subjects are:

Subject1:94
Subject2:98
Subject3:95
Subject4:28
Subject5:59
C:\Users\STUDENT\Desktop\1BM23CS117\LAB6>
```

LAB PROGRAM 7:

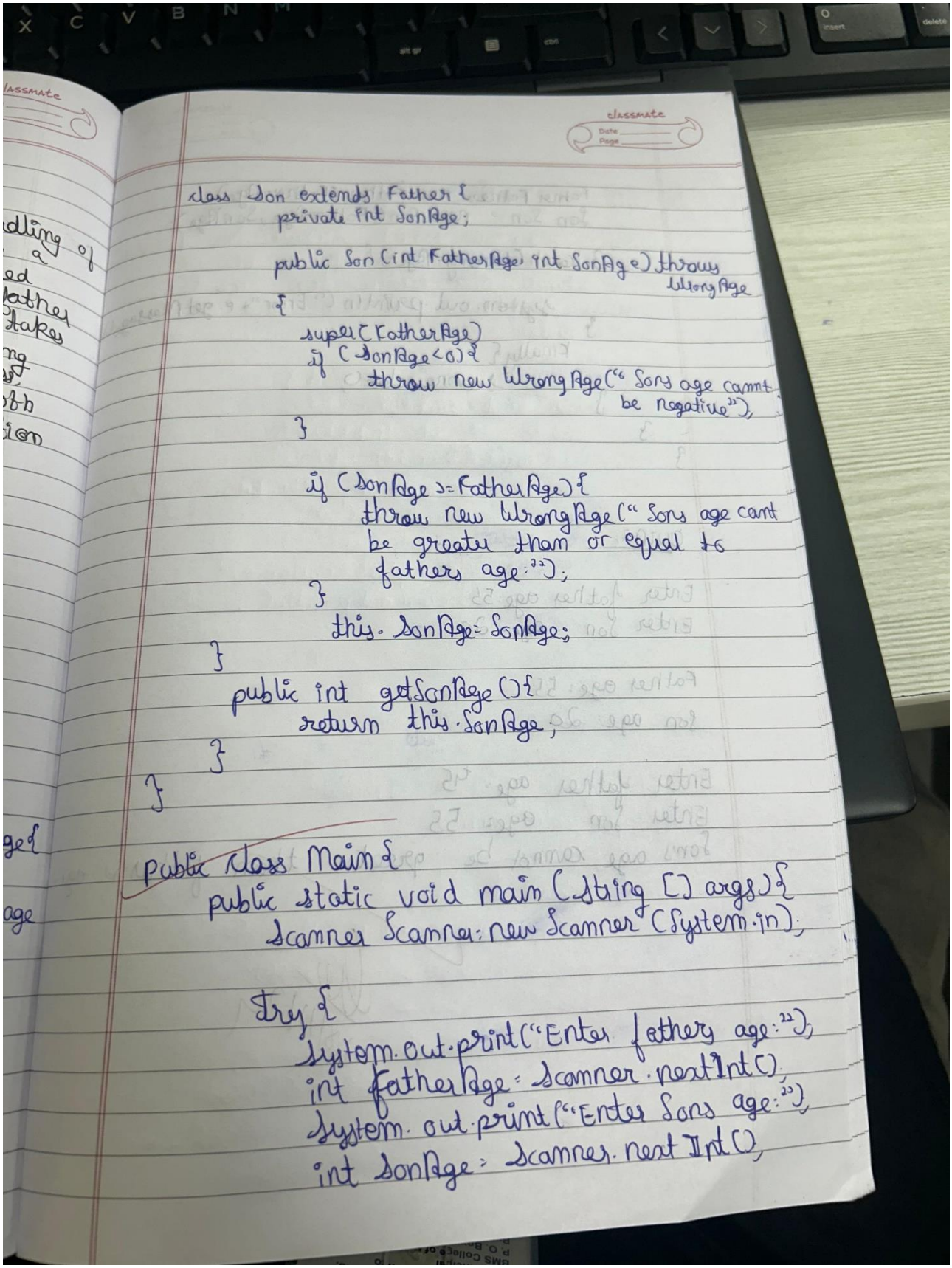
Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception Wrong Age () when the input age<0. In Son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

ALGORITHM:

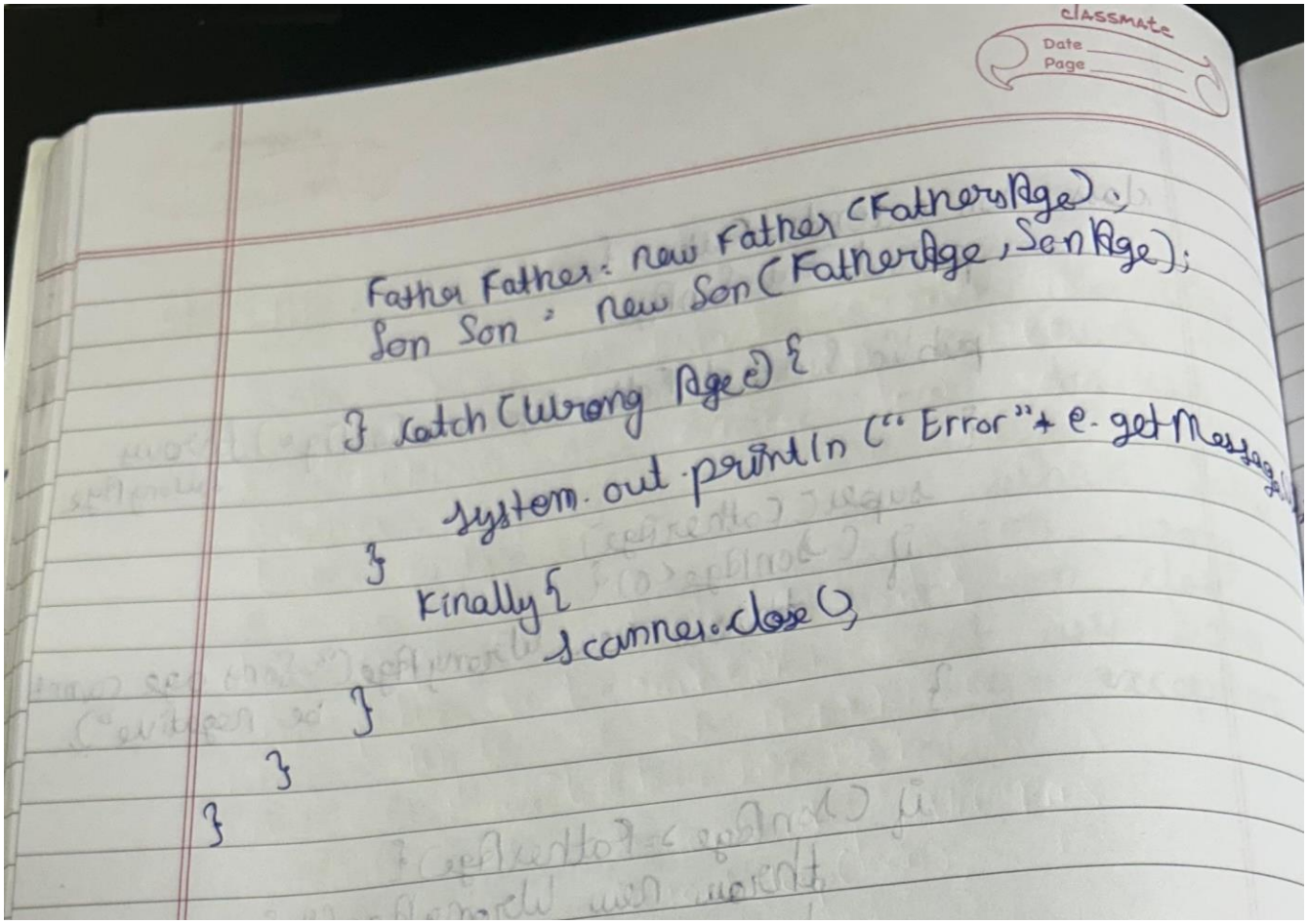
LAB PROGRAM-7

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

```
import java.util.Scanner;

class WrongAge extends Exception{
    public WrongAge(String message){
        super(message)

class Father{
    private int age;

    public Father(int age) throws WrongAge{
        if(age<0){
            throw new WrongAge("Father's age
            Cant be negative");
        }
        this.age = age;
    }
}

class Son{
    private int son;
    public int getAge(){
        return this.age
}   }
```



```
class Son extends Father {
    private int SonAge;

    public Son(int FatherAge, int SonAge) throws WrongAge
    {
        super(FatherAge)
        if (SonAge<0){
            throw new WrongAge("Sons age cannt be negative");
        }

        if (SonAge >= FatherAge){
            throw new WrongAge("Sons age cant be greater than or equal to fathers age.");
        }
        this.SonAge = SonAge;
    }
    public int getSonAge(){
        return this.SonAge;
    }
}

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

        try {
            System.out.print("Enter father age:");
            int FatherAge = Scanner.nextInt();
            System.out.print("Enter Sons age:");
            int SonAge = Scanner.nextInt();
```



CODE:

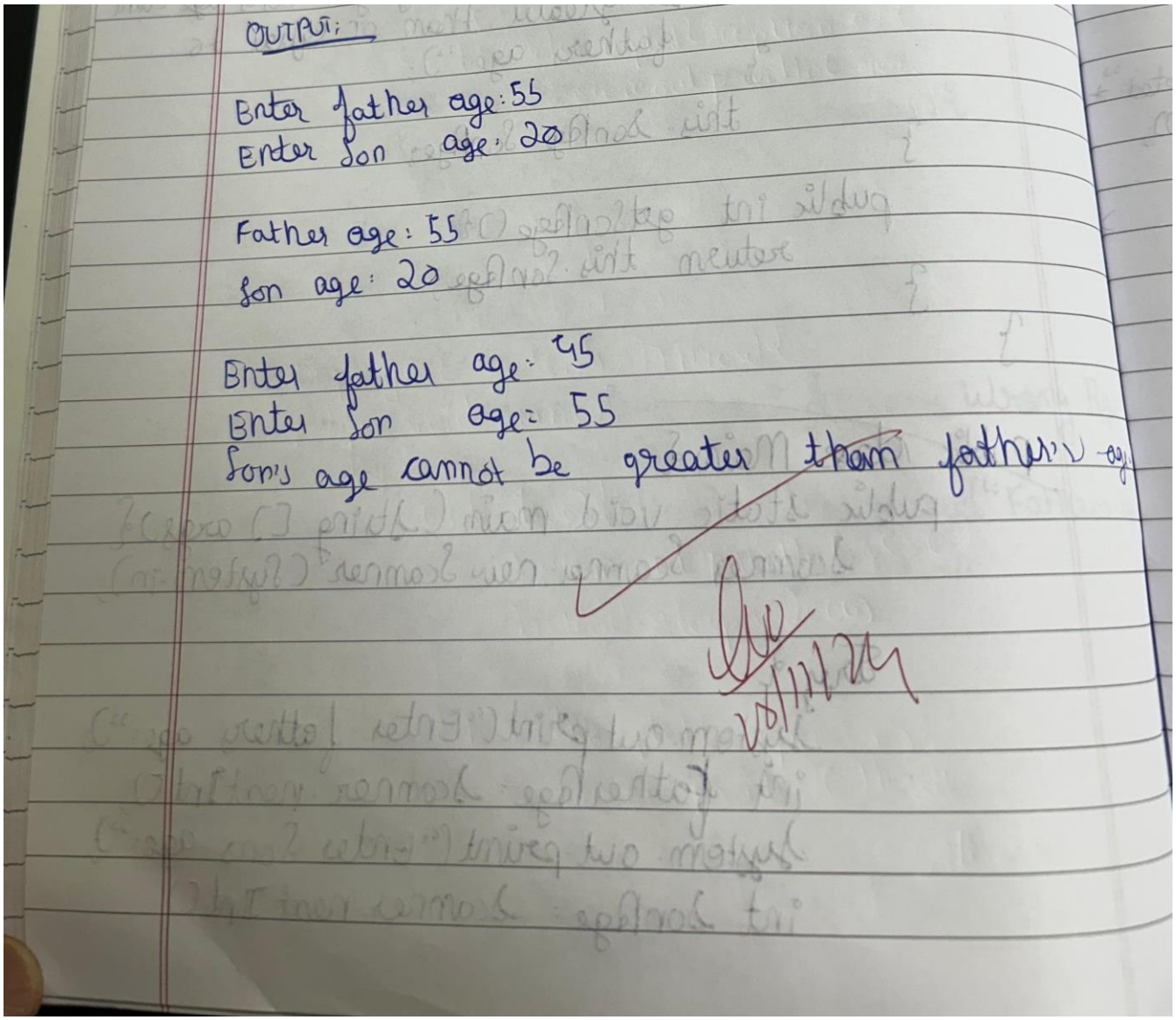
```
import java.util.Scanner;


class WrongAge extends Exception {
    public WrongAge() {
        super("Age Error");
    }

    public WrongAge(String message) {
```

```
        super(message);
    }
}

class Father {
    protected int fatherAge;

    public Father() throws WrongAge {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter Father's Age: ");
        fatherAge = s.nextInt();

        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }
}

class Son extends Father {
    private int sonAge;

    public Son() throws WrongAge {
        super();
        Scanner s = new Scanner(System.in);
        System.out.print("Enter Son's Age: ");
```

```
        sonAge = s.nextInt();

        if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to Father's age");
        }
    }

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

public class AgeValidation {
    public static void main(String[] args) {
        System.out.println("Idrees Shaikh 1BM23CS117");
        try {
            Son son = new Son();
            son.display();
        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
```

}

OUTPUT:

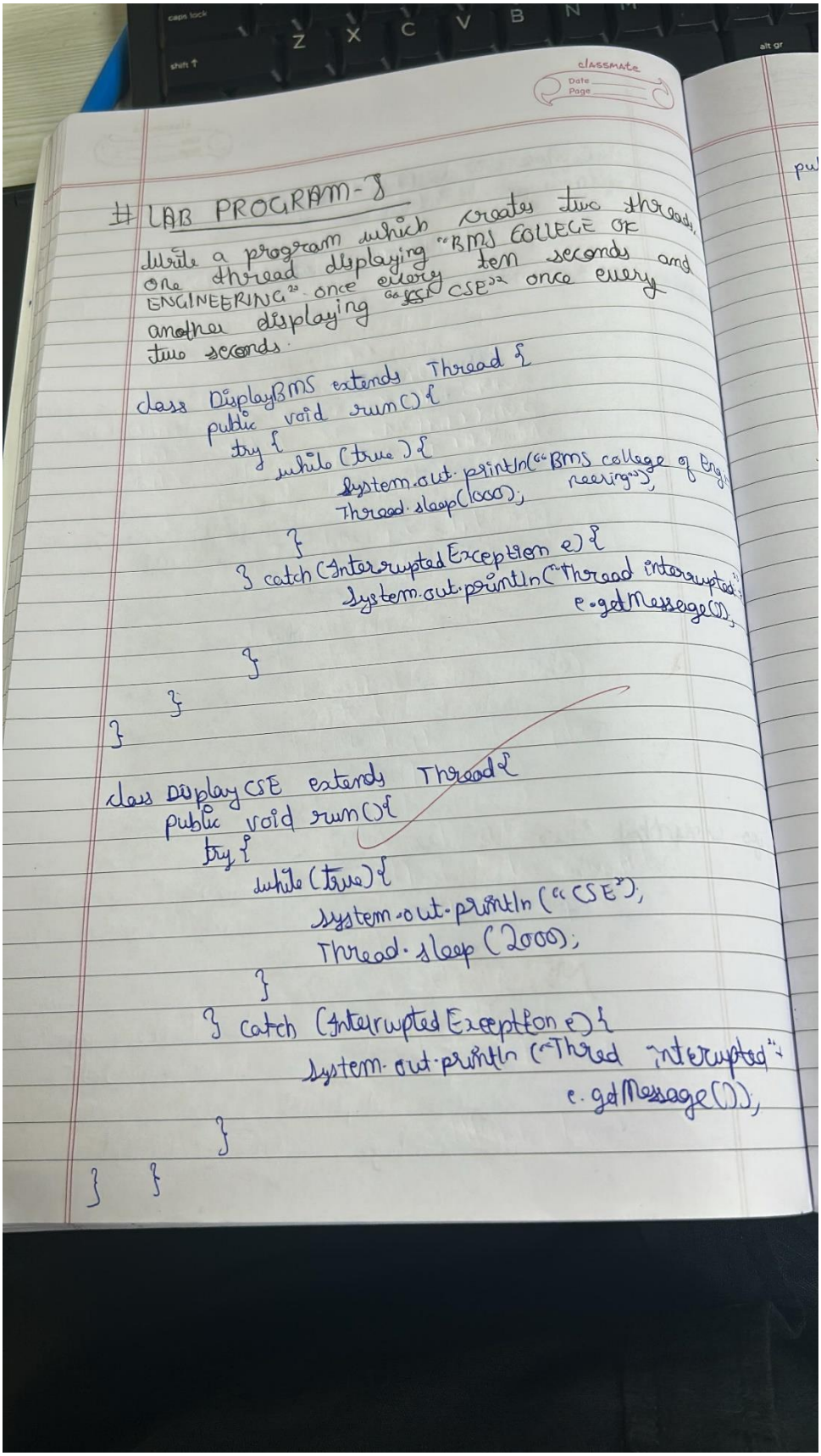

OUTPUT:
Enter father age: 55
Enter son age: 20
Father age: 55
son age: 20
Enter father age: 45
Enter son age: 55
Son's age cannot be greater than father's age
10/11/24

```
C:\1BM23CS117>java AgeValidation
Idrees Shaikh 1BM23CS117
Enter Father's Age: 56
Enter Son's Age: 20
Son's Age: 20

C:\1BM23CS117>java AgeValidation
Idrees Shaikh 1BM23CS117
Enter Father's Age: 23
Enter Son's Age: 45
Exception: Son's age cannot be greater than or equal to Father's age
```

LAB PROGRAM 8:

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

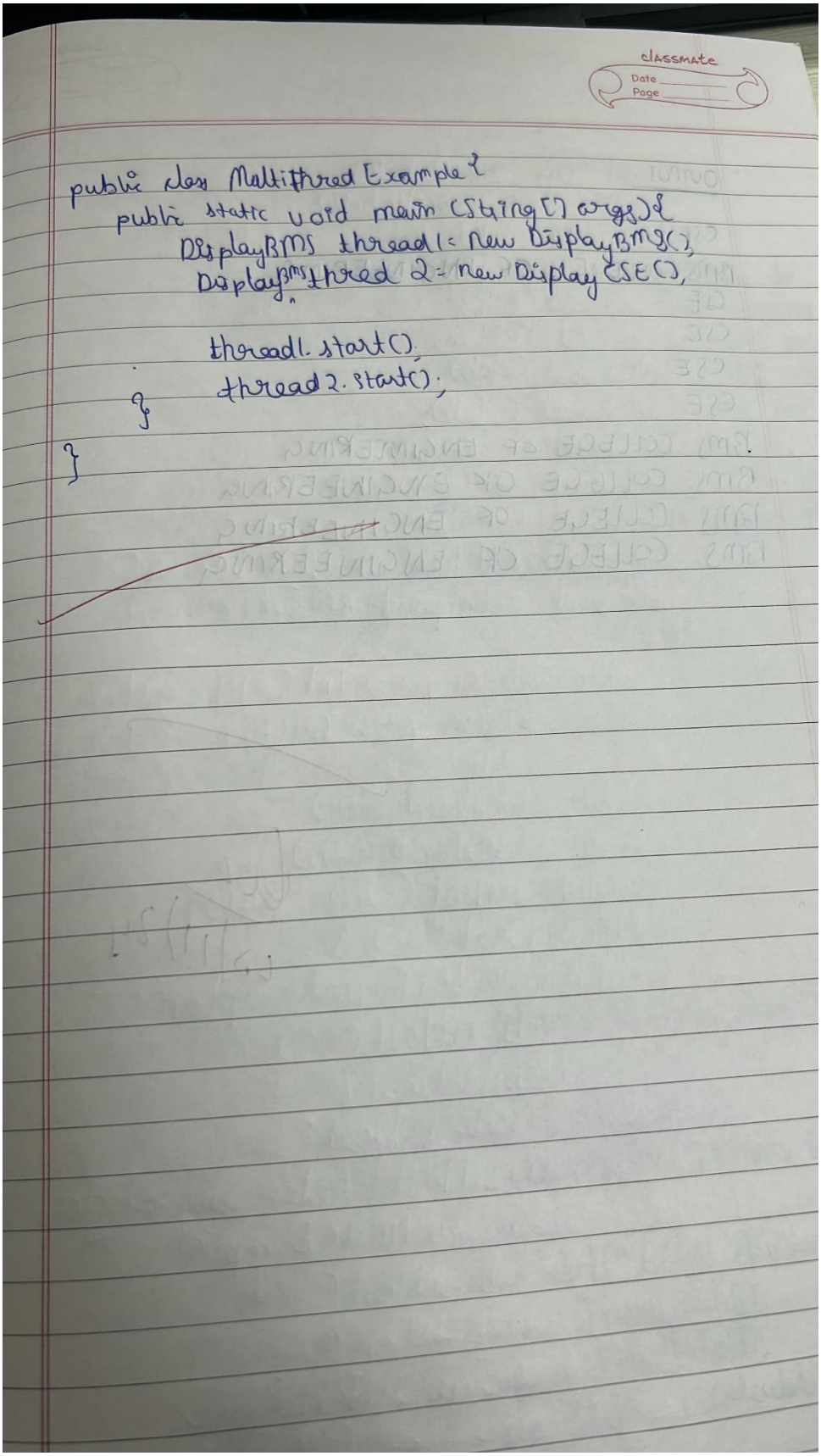
ALGORITHM:



LAB PROGRAM-7

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

class DisplayCSE extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted" + e.getMessage());
        }
    }
}
```



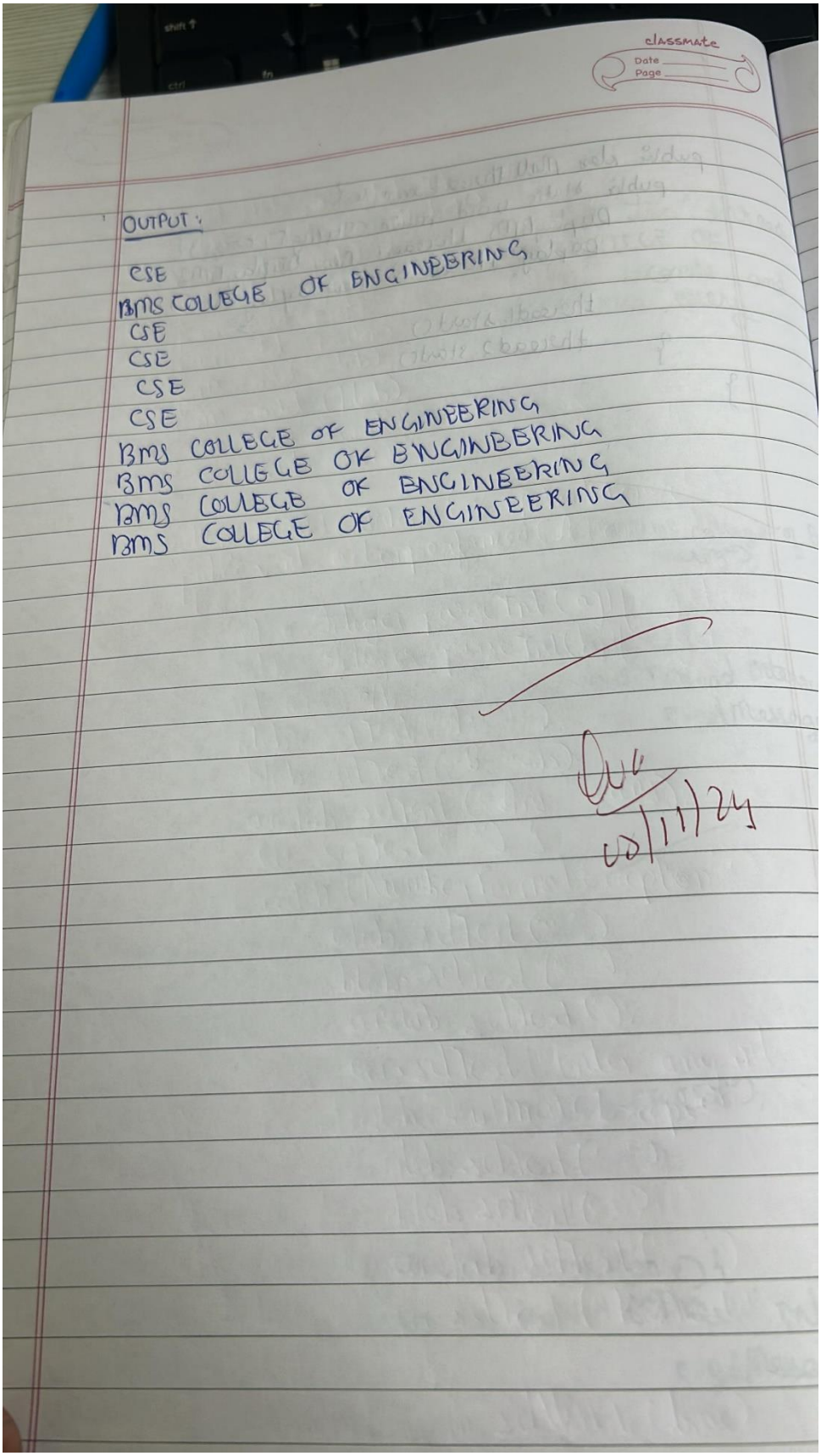
```
public class MultithreadExample {
    public static void main (String[] args) {
        DisplayBMS thread1 = new DisplayBMS();
        DisplayBMS thread2 = new DisplayCSE();

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

CODE:

```
public void run() {
        try {
            for(int i=1;i<=10;i++){
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000);
            }
        } catch (InterruptedException e) {
            System.out.println("CollegeThread interrupted: " + e.getMessage());
        }
    }
}



class CSEThread extends Thread {
    public void run() {
        try {
            for(int i=1;i<=10;i++){


                System.out.println("CSE");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            System.out.println("CSEThread interrupted: " + e.getMessage());
        }
```

```
    }
}



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

        CollegeThread collegeThread = new CollegeThread();
        CSEThread cseThread = new CSEThread();



        collegeThread.start();
        cseThread.start();
System.out.println("Idrees Shaikh");
    }
}
```

OUTPUT:



OUTPUT:

CSE
BMS COLLEGE OF ENGINEERING
CSE
CSE
CSE
CSE
BMS COLLEGE OF ENGINEERING
BMS COLLEGE OF ENGINEERING
BMS COLLEGE OF ENGINEERING
BMS COLLEGE OF ENGINEERING

02/11/24

```
C:\1BM23CS117>java MultiThreadExample
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
```

LAB PROGRAM 9:

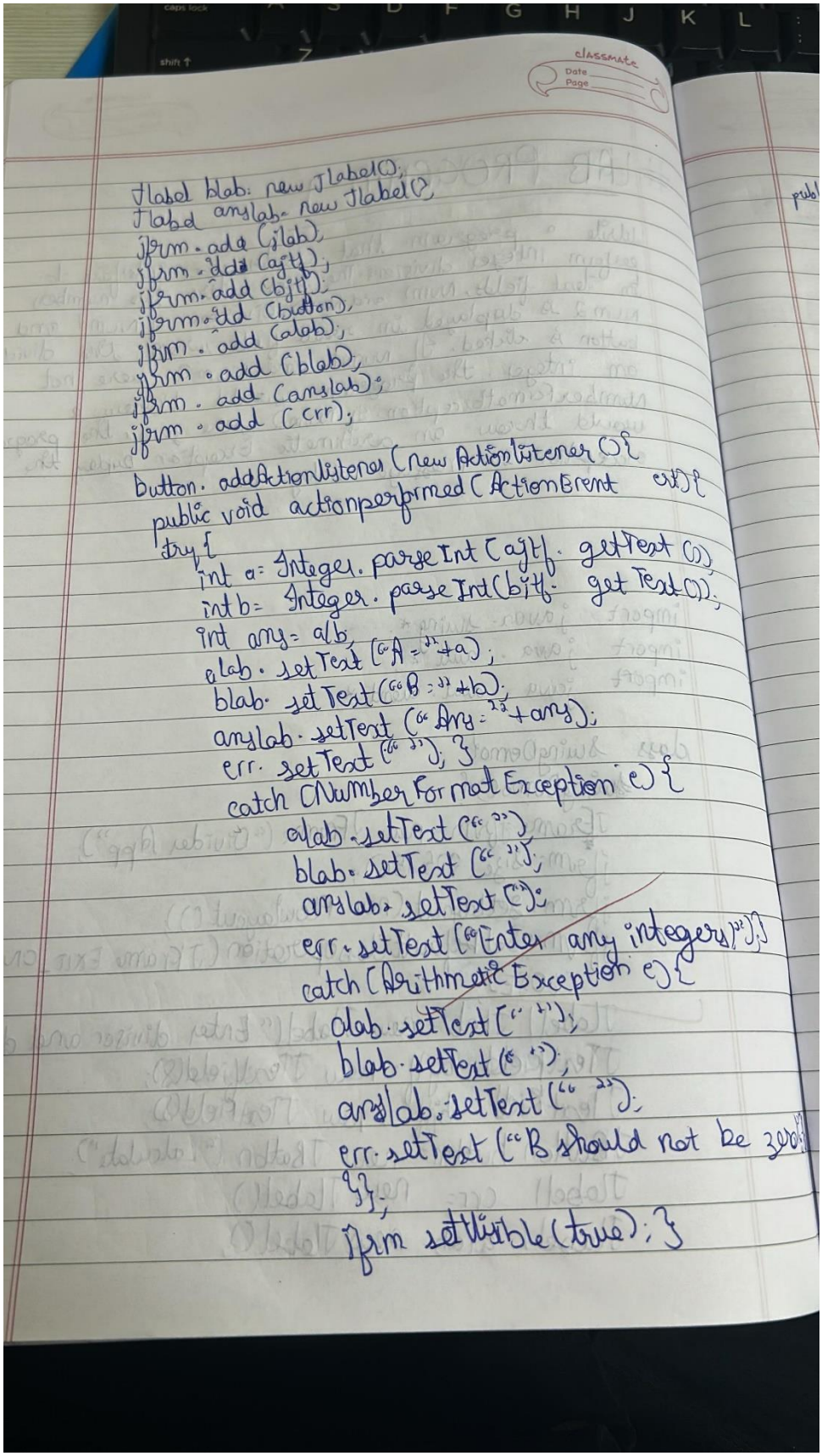
Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException.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

ALGORITHM:

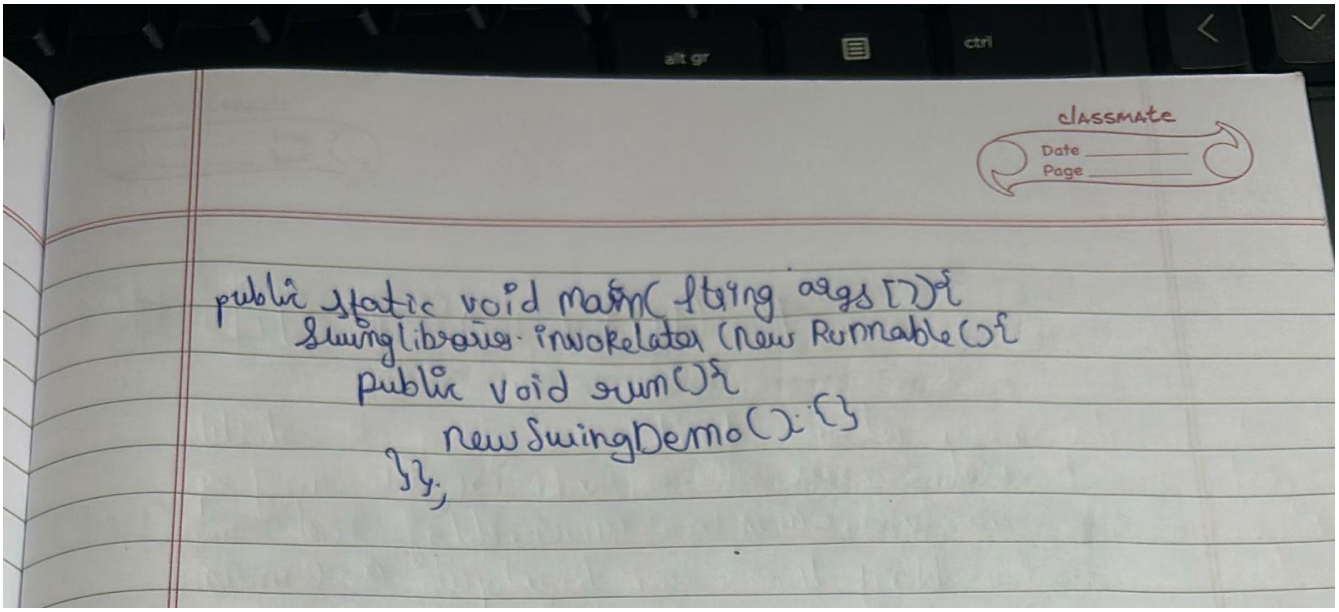
LAB PROGRAM-9

Write a program that creates user interface to perform integer divisions. The user enters two numbers in text fields, Num1 and Num2. Division of Num1 and Num2 is displayed in result field when the divide button is clicked. If Num1 and 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 message dialog box.

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

class SwingDemo{
    SwingDemo(){
        JFrame jfrm = new JFrame("Divider App");
        jfrm.size(275,150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_
            CLOSE);
        JLabel jlab = new JLabel("Enter divisor and divide
        JTextField aff = new JTextfield(8);
        JTextField bff = new JTextField(8);
        JButton button = new JButton("Calculate");
        JLabel err = new JLabel();
        JLabel alab = new JLabel();
```



```
JLabel blab= new JLabel();
JLabel anylab= new JLabel();
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anylab);
jfrm.add(err);

button.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
try{
int a= Integer.parseInt(ajtf.getText());
int b= Integer.parseInt(bjtf.getText());
int any= a/b;
alab.setText("A="+a);
blab.setText("B="+b);
anylab.setText("Any="+any);
err.setText("");}
catch(NumberFormatException e){
alab.setText("");
blab.setText("");
anylab.setText("");
err.setText("Enter any integers");}
catch(ArithmeticException e){
alab.setText("");
blab.setText("");
anylab.setText("");
err.setText("B should not be zero");
}});
jfrm.setVisible(true);}
```



CODE:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;


class SwingDemo {

    SwingDemo() {

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

        jfrm.setSize(300, 200);

        jfrm.setLayout(new FlowLayout());

        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);


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

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

```
JTextField bjtf = new JTextField(8);
ajtf.setToolTipText("Enter an integer as the dividend");
bjtf.setToolTipText("Enter an integer as the divider");

JButton button = new JButton("Calculate");
JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();
JLabel anslab = new JLabel();

jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(err);
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("Error: Please enter valid integers!");
            } catch (ArithmeticException e) {
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("Error: Divider (B) cannot be zero!");
            }
        }
    });

    jfrm.setVisible(true);
}

public static void main(String args[]) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new SwingDemo();
```

```
        }
      });
    }
}
```

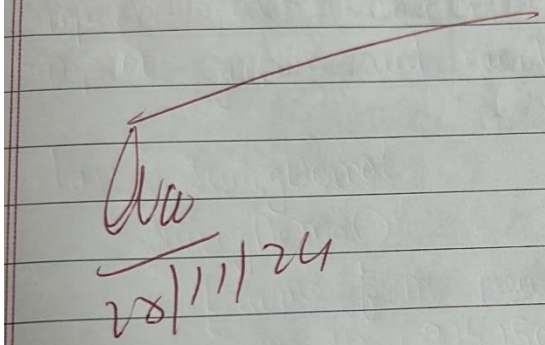
OUTPUT:

OUTPUT:

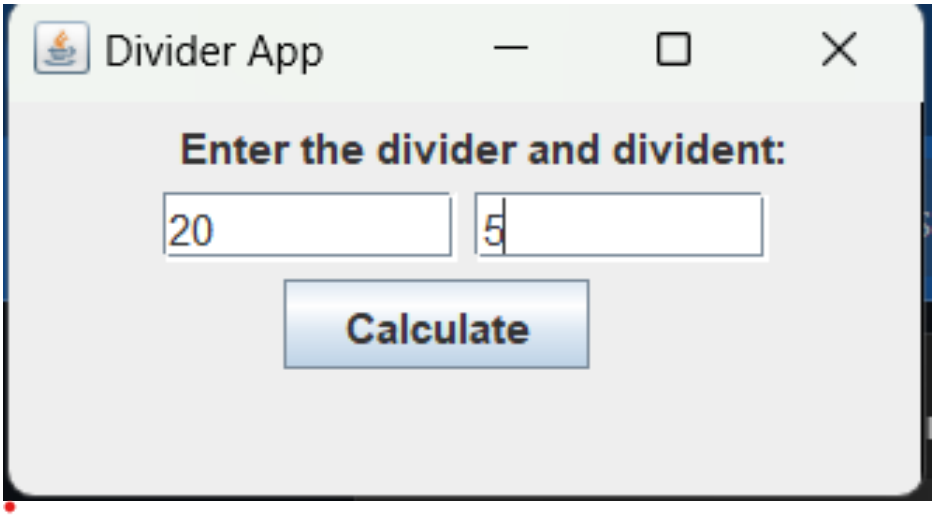
Divider app

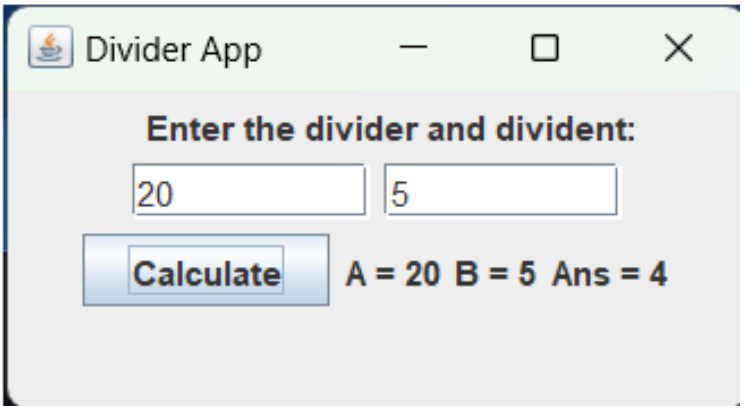
Enter divider and dividend
5
Calculate

A=5 B=5 Ans=1



28/11/24

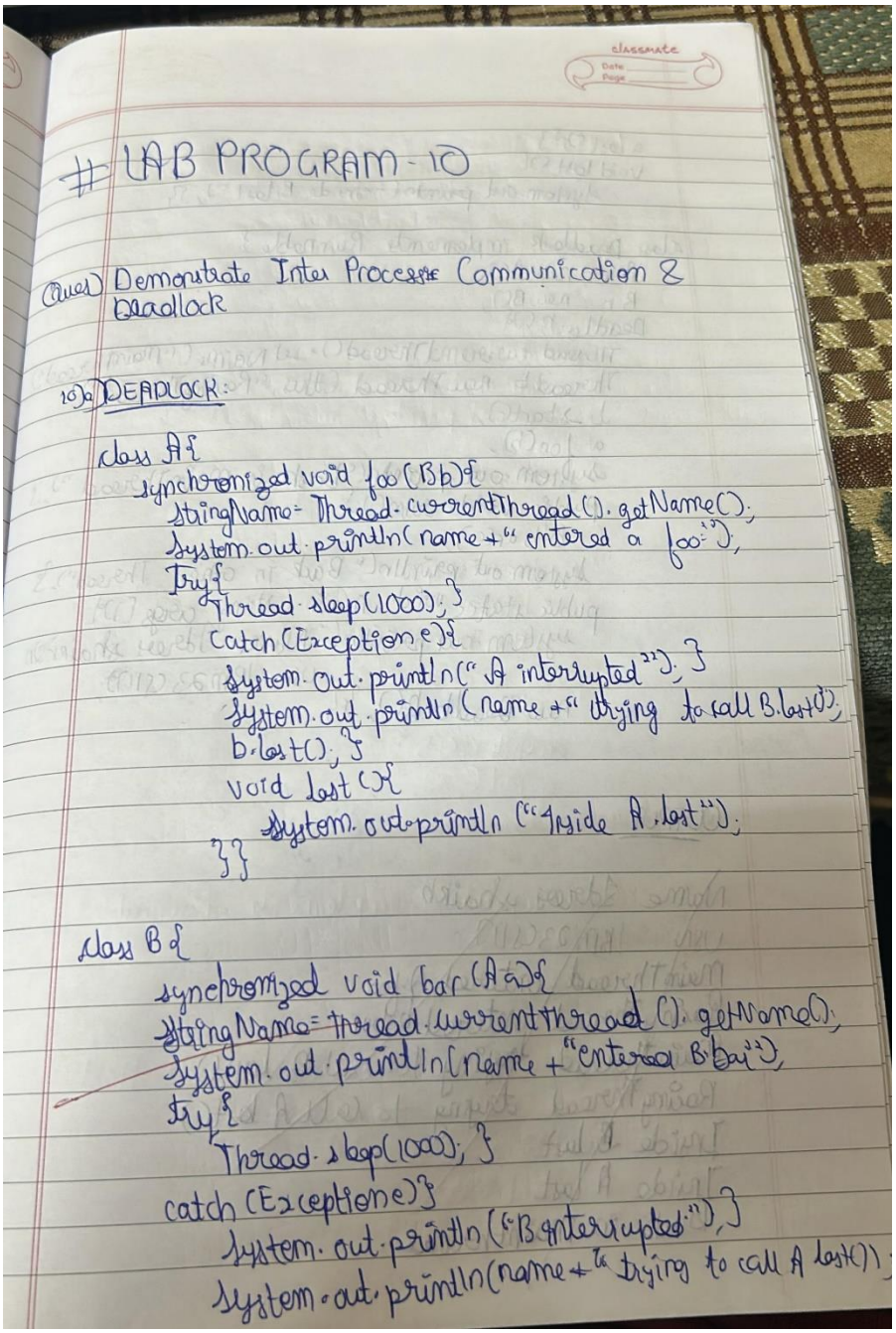

Divider App
Enter the divider and divident:
20
5
Calculate


Divider App
Enter the divider and divident:
20
5
Calculate
A = 20 B = 5 Ans = 4

LAB PROGRAM 10:

Demonstrate Inter process Communication and deadlock

ALGORITHM:

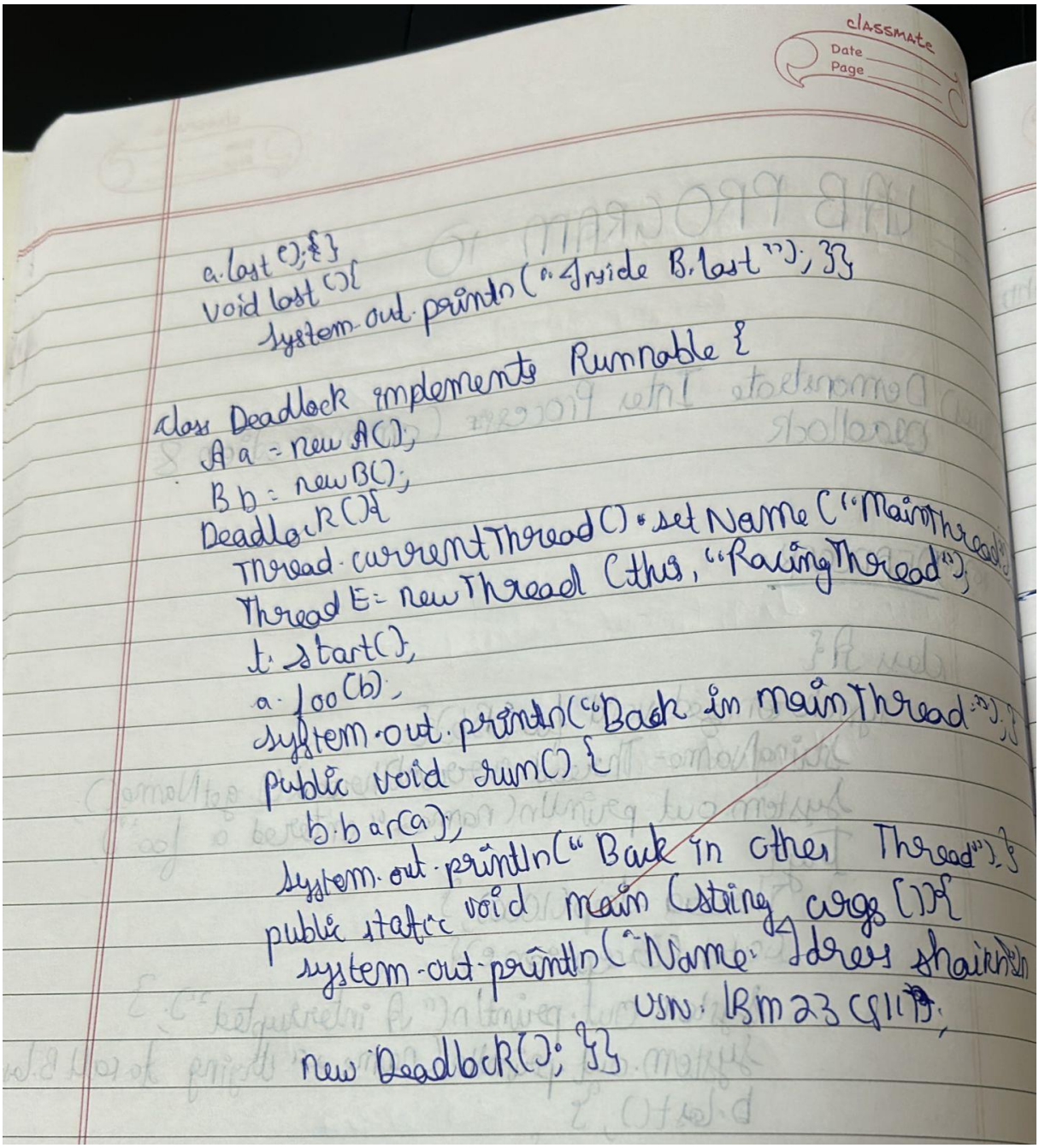


LAB PROGRAM - 10

Aim) Demonstrate Inter Process Communication & Deadlock

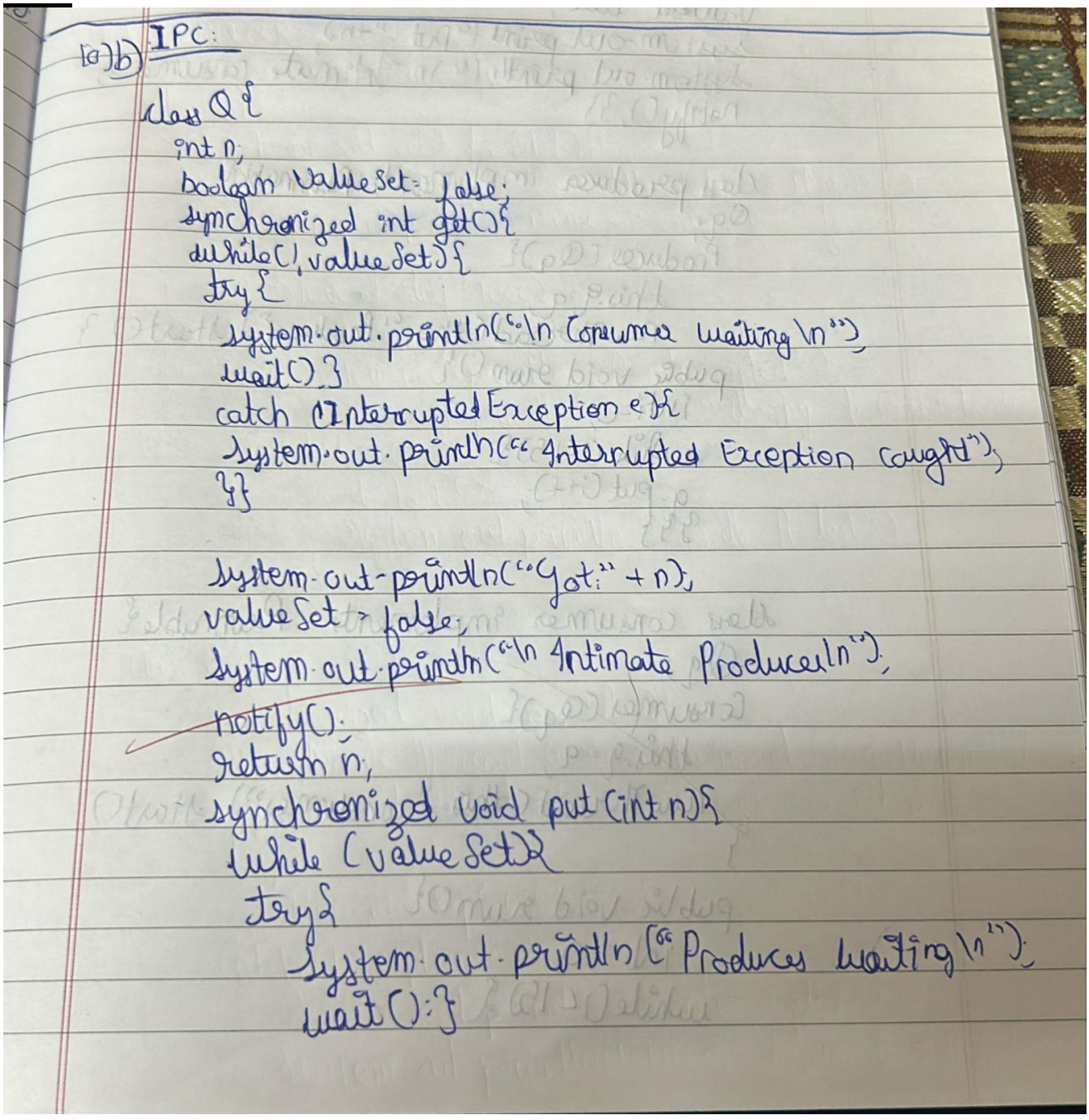
10a) DEADLOCK:

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

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

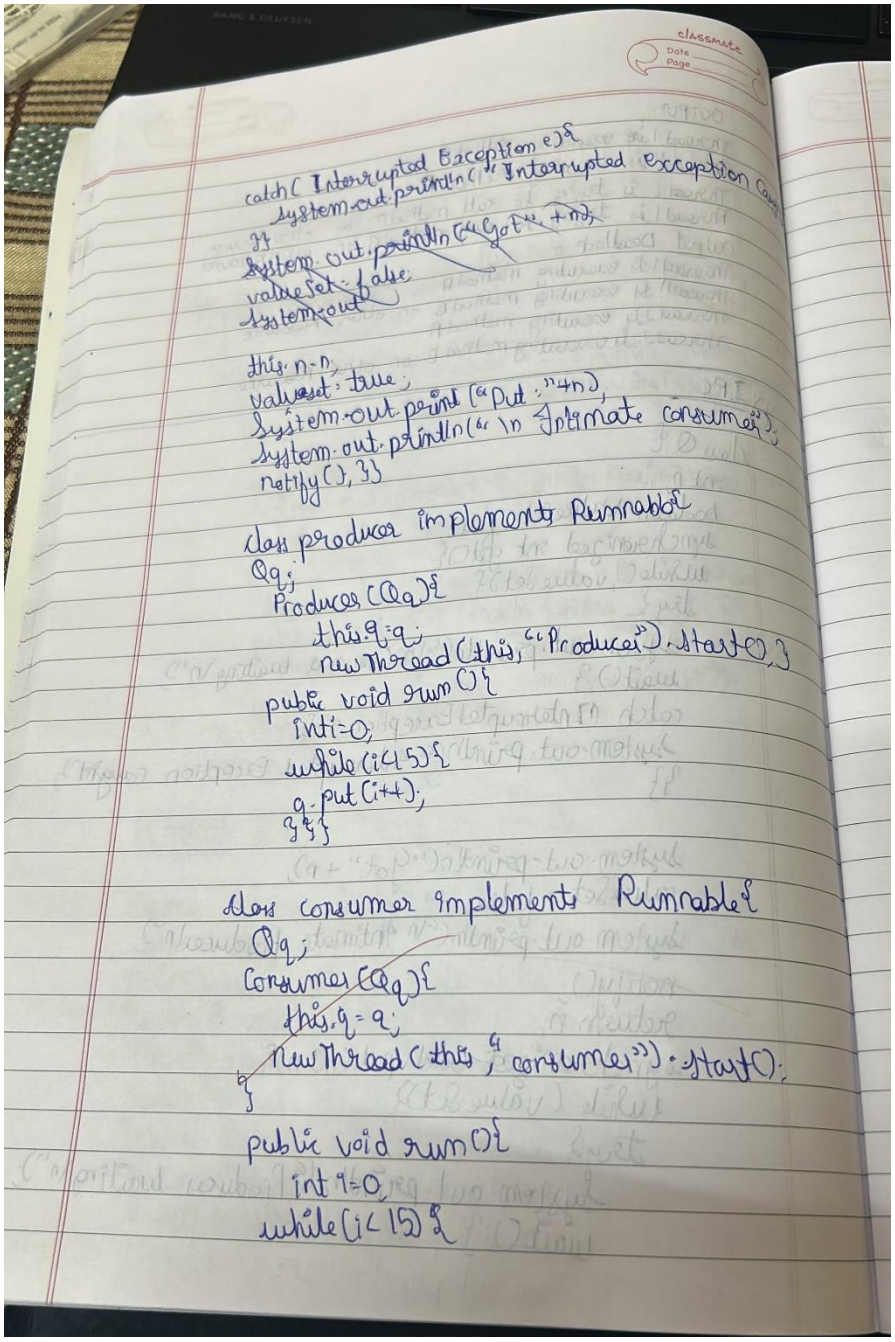


```
a.last();}}
void last(){
    System.out.println("Inside B.last");}}

class Deadlock implements Runnable {
A a = new A();
B b = new B();
Deadlock(){
Thread.currentThread().setName("MainThread");
Thread E= new Thread(this, "RacingThread");
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[]){
    System.out.println("Name: Adrees Shaikh
                        USN: 1BM23CS111");
    new Deadlock();}}
```

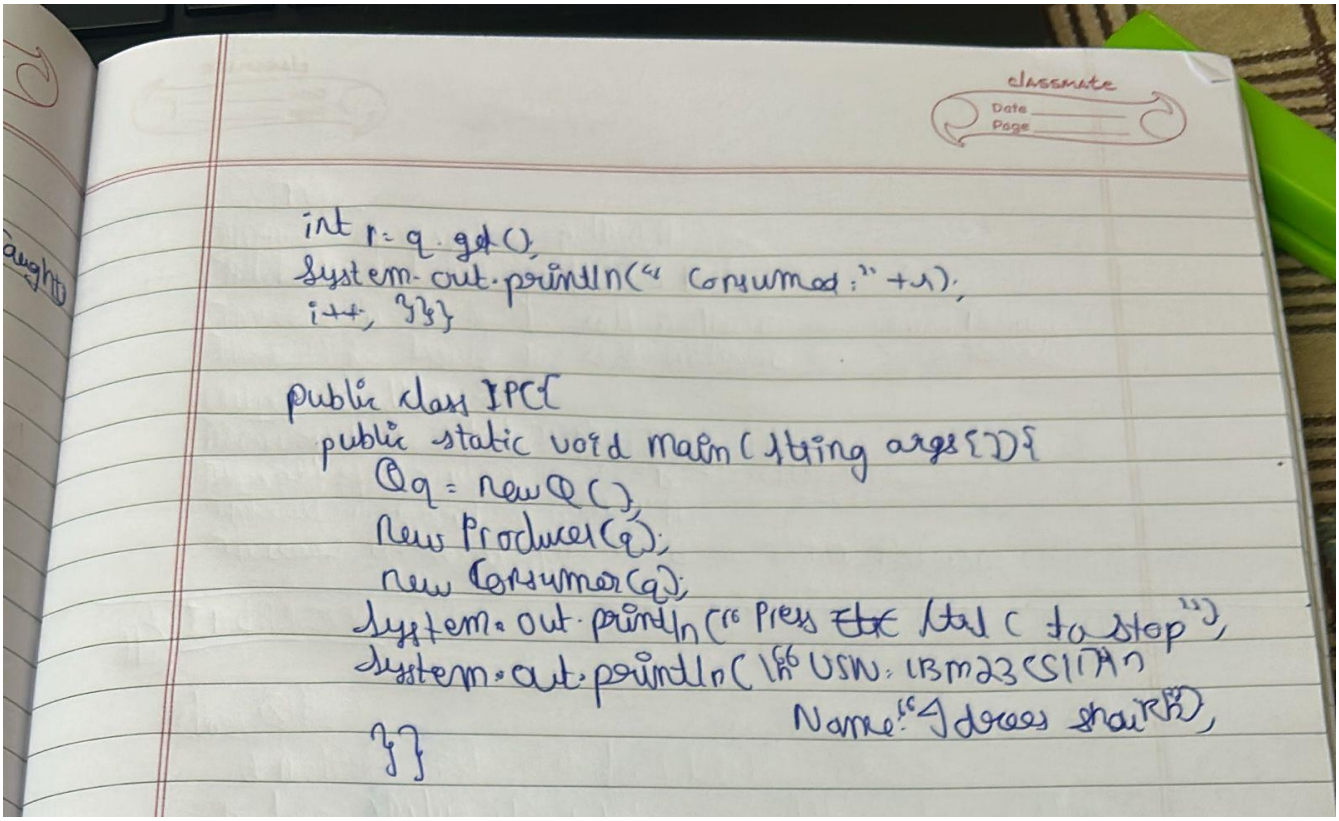
IPC:



```
10)b) IPC:
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(" Interrupted Exception caught");
   }}

   System.out.println("Got:" + n);
   valueSet = false;
   System.out.println("\n Intimate Producer\n");
   notify();
   return n;
  synchronized void put(int n){
   while (valueSet){
    try{
      System.out.println(" Producer waiting\n");
      wait();}
```



```
catch( InterruptedException e){
  System.out.println("Interrupted exception caught");
}
System.out.println("Got:" + n);
valueSet = false;
System.out
this.n=n;
valueSet = true;
System.out.print("Put: "+n);
System.out.println("\n Intimate consumer");
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<5){
  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){
```



CODE:

```
package Lab;


class A {

    synchronized void foo(B b) {

        String name = Thread.currentThread().getName();

        System.out.println(name + " entered A.foo");



        try {

            Thread.sleep(1000);

        } catch (InterruptedException e) {
```

```
            System.out.println("A Interrupted");
        }

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

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

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

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

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

```
    }

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

public 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");
    }

    @Override
    public void run() {
        b.bar(a);
```

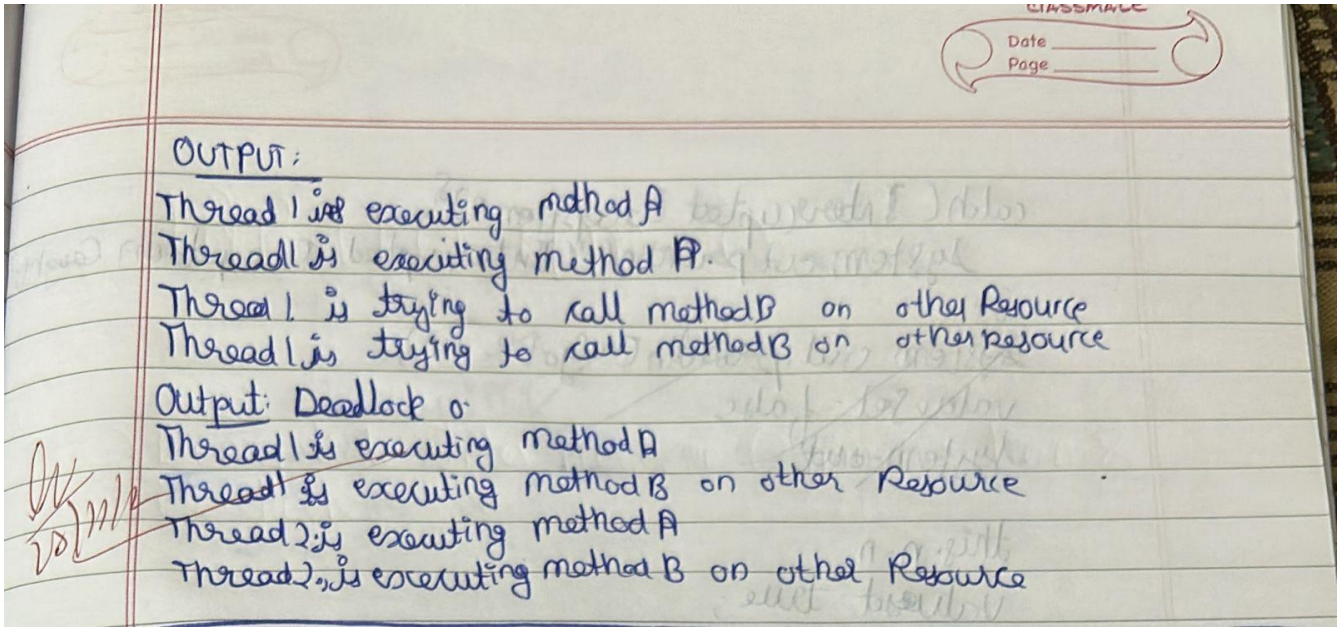
```
    }

    public static void main(String[] args) {
        new Deadlock();
    }
}
class Q {
    int n;
    boolean valueSet = false;

    synchronized int get() {

        while (!valueSet) {
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();  // Set interrupt flag
                return -1;
            }
        }

        System.out.println("Got: " + n);
        valueSet = false;
        notify();
```

```
        return n;
    }

    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting\n");
                wait();
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                return;
            }
        }

        this.n = n;
        valueSet = true;
        System.out.println("Put: " + 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();
            if (r != -1) {
```

```
            System.out.println("Consumed: " + r);
          }
          i++;
        }
    }
}

public class ProducerConsumerFix {
    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

OUTPUT:



OUTPUT:
Thread 1 is executing method A
Thread 1 is executing method A.
Thread 1 is trying to call method B on other Resource
Thread 1 is trying to call method B on other Resource

Output: Deadlock o.
Thread 1 is executing method A
Thread 1 is executing method B on other Resource
Thread 2 is executing method A
Thread 2 is executing method B on other Resource

OUTPUT:

USN : 1BM23CS117
Name: Idrees Sparkh
Put: 0
Intimate consumer
Producer waiting
Got: 0
Intimate producer
Put: 1

Put: 1
Get: 1
Put: 2
Got: 2
Put: 3
Got: 3
Put: 4
Got: 4
Put: 5
Got: 5

```
PS C:\Users\idree\OneDrive\Desktop\Lab>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\idree\AppData\Roami
ng\Code\User\workspaceStorage\b637fbb570e5bdbfb2f898faedee0abc\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'Lab.Deadlock'
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
Inside A.last
MainThread trying to call B.last()
Inside B.last
Back in main thread
PS C:\Users\idree\OneDrive\Desktop\Lab>
```

```
PS C:\Users\idree\OneDrive\Desktop\Lab2>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\idree\AppData\Roam
ing\Code\User\workspaceStorage\106993c802e8b11dfe28acf45a8522df\redhat.java\jdt_ws\Lab2_82c55a0c\bin' 'ProducerConsumerFix'
Press Control-C to stop.
Put: 0

Producer waiting

Got: 0
Put: 1

Producer waiting

Consumed: 0
Got: 1
Consumed: 1
Put: 2

Producer waiting

Got: 2
Consumed: 2
Put: 3

Producer waiting

Got: 3
Consumed: 3
Put: 4

Producer waiting

Got: 4
Consumed: 4
Put: 5

Producer waiting
```

```
Got: 5
Consumed: 5
Put: 6
Got: 6
Consumed: 6
Put: 7

Producer waiting

Got: 7
Consumed: 7
Put: 8

Producer waiting

Got: 8
Consumed: 8
Put: 9

Producer waiting

Got: 9
Consumed: 9
Put: 10

Producer waiting

Got: 10
Consumed: 10
Put: 11

Producer waiting

Got: 11
Consumed: 11
Put: 12

Producer waiting

Got: 12
Consumed: 12
```