VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SPURTHI REDDY P (1BM23CS338)

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 **SPURTHI REDDY P (1BM23CS338),** 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

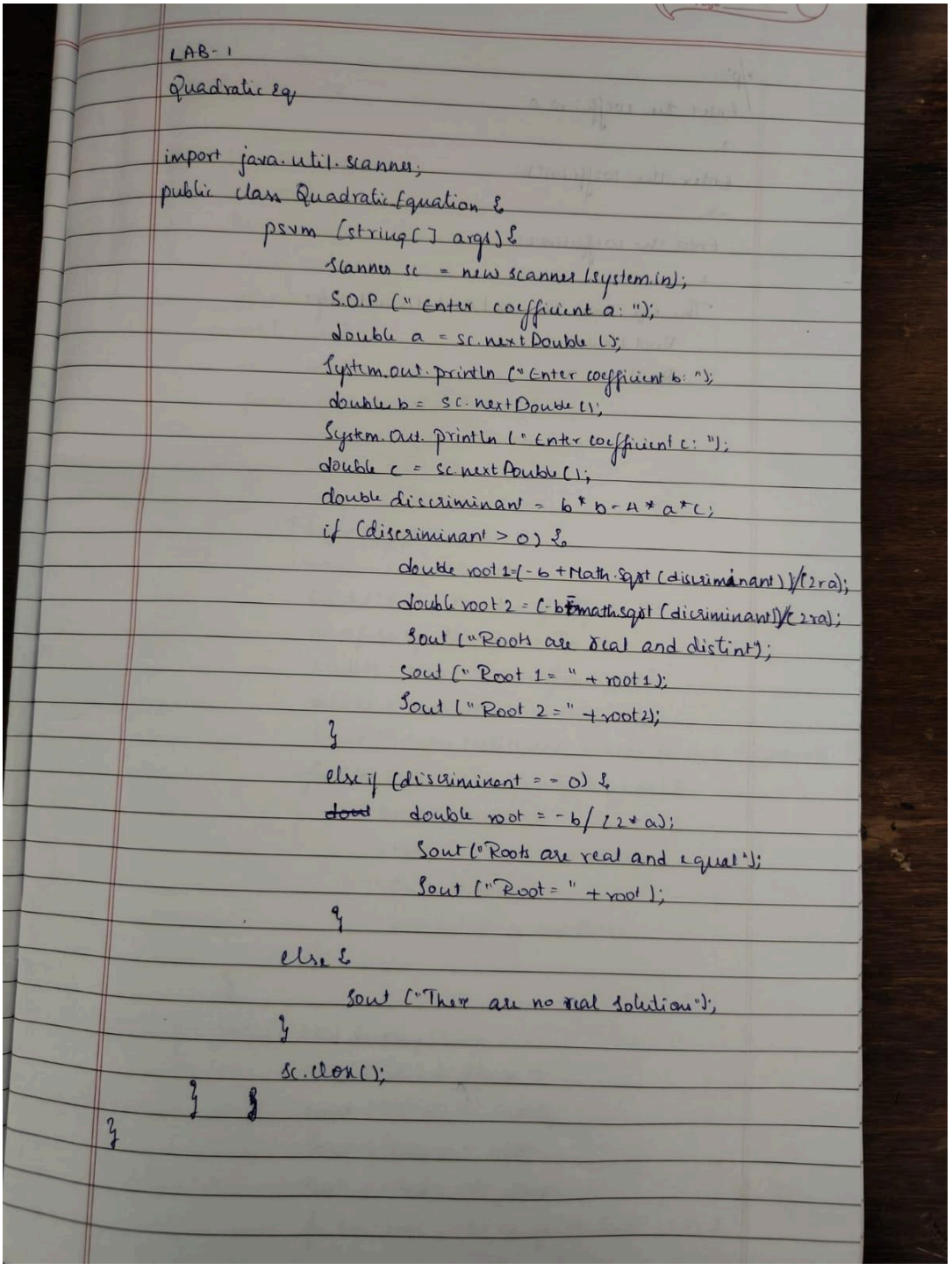
Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

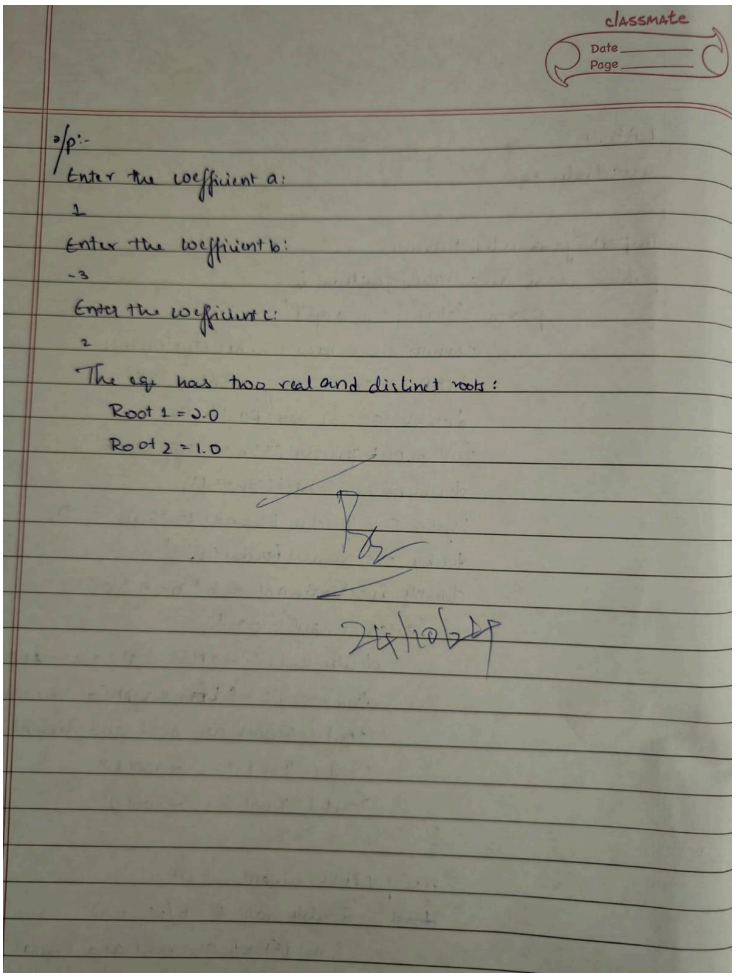
INDEX

LABORATORY PROGRAM - 1

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.



```
LAB-1
Quadratic Eq

import java.util.Scanner;
public class Quadratic_Equation {
        psvm [String[] args]{
            Scanner sc = new Scanner(System.in);
            S.O.P (" 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);
                Sout (" Roots are real and distinct");
                Sout (" Root 1 = " + root1);
                Sout (" Root 2 = " + root2);
            }
            else if (discriminant == 0) {
                double root = -b/(2*a);
                Sout (" Roots are real and equal");
                Sout (" Root = " + root);
            }
            else {
                Sout (" There are no real solution");
            }
            sc.close();
        }
}
```



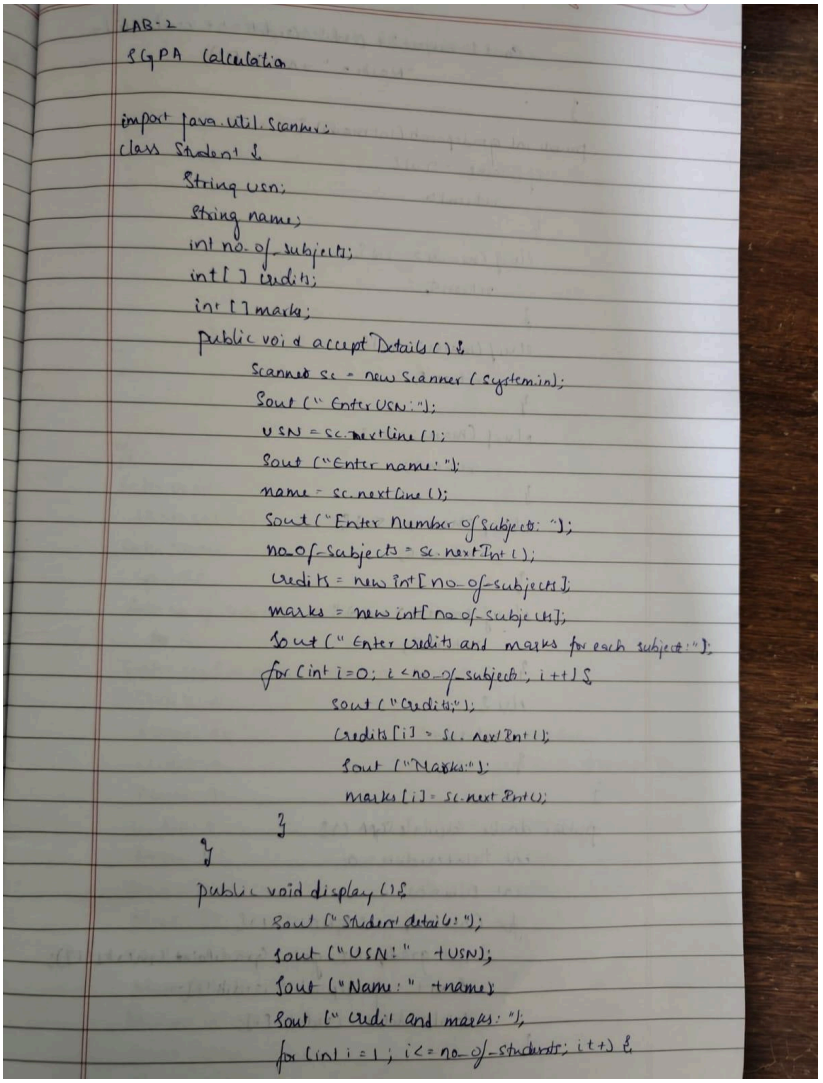

```
import java.util.Scanner;
public class quadratic_equation {
    public static void main (String[] args){
        Scanner sc = new Scanner (System.in);
        System.out.println("enter coefficient a:");
        double a = sc.nextDouble();
        System.out.println("enter coefficient b:");
        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 ("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");
        }
        sc.close();
    }
}
```

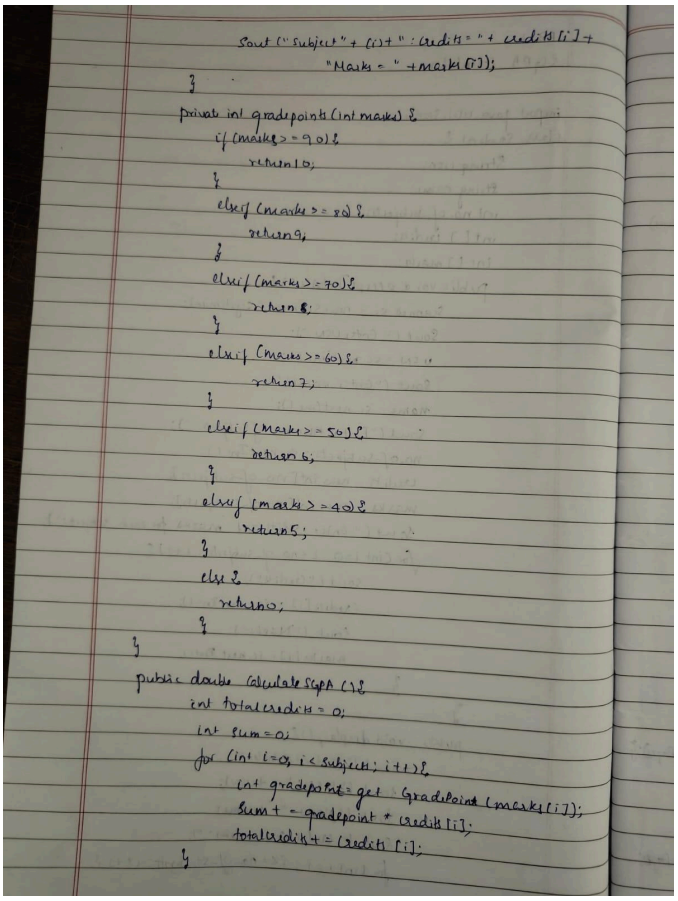
```
Microsoft Windows [Version 10.0.22631.4391]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Spandana>d:

D:\>cd Java_Lab_Programs

D:\Java_Lab_Programs>set path="C:\Program Files\Java\jdk-23\bin"

D:\Java_Lab_Programs>javac QuadraticEquation.java

D:\Java_Lab_Programs>java QuadraticEquation
Enter coefficient a:
1
Enter coefficient b:
-3
Enter coefficient c:
2
Roots are real and distinct
Root 1=2.0
Root 2=1.0

D:\Java_Lab_Programs>java QuadraticEquation.java
Enter coefficient a:
1
Enter coefficient b:
-6
Enter coefficient c:
9
Roots are real and equal
Root=3.0

D:\Java_Lab_Programs>java QuadraticEquation
Enter coefficient a:
1
Enter coefficient b:
2
Enter coefficient c:
5
There are no real solutions

D:\Java_Lab_Programs>
```

LABORATORY PROGRAM - 2

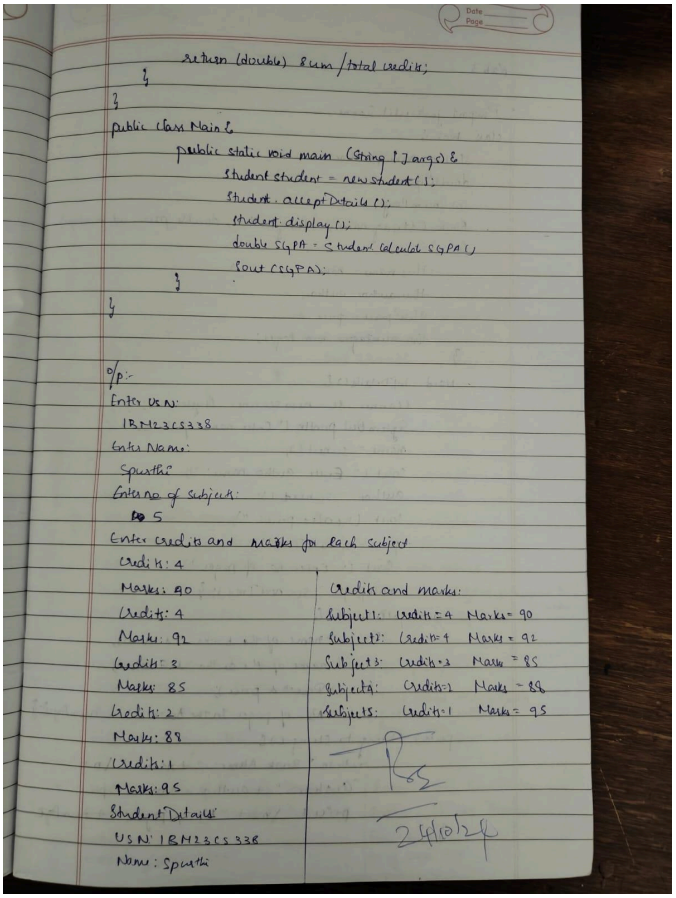
2. Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.



```
LAB: 2
SGPA Calculation

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();
            System.out.println("Marks:");
            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=1; i<=no_of_students; i++) {
```



```
        Sout ("Subject" + (i+1) + ": Credit = " + credits[i] +
                "Marks = " + marks[i]);
    }

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

    public double calculateSGPA () {
        int totalCredit = 0;
        int sum = 0;
        for (int i=0; i < subjects; i++) {
            int gradepoint = getGradePoint (marks[i]);
            sum += gradepoint * credits[i];
            totalCredit += 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();
        Sout ("SGPA");
    }
}
```

o/p:

```
Enter USN:
1BM23CS338
Enter Name:
Spurthi
Enter no of subjects:
5
Enter credits and marks for each subject
Credit: 4
Marks: 90
Credits: 4
Marks: 92
Credit: 3
Marks: 85
Credit: 2
Marks: 88
Credit: 1
Marks: 95
Student Details
USN: 1BM23CS338
Name: Spurthi
```

Credits and marks:

Subject1: Credit = 4 Marks = 90
Subject2: Credit = 4 Marks = 92
Subject3: Credit = 3 Marks = 85
Subject4: Credit = 2 Marks = 88
Subject5: Credit = 1 Marks = 95

24/10/24

```
import java.util.Scanner;
class student {
    String usn;
    String name;
    int no_of_subjects;
    int[] credits;
    int[] marks;
    public void accept_details(){
        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 no 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 of each subject:");
        for(int i=0;i<no_of_subjects; i++)
        {
            System.out.println("credits:");
            credits[i]=sc.nextInt();
            System.out.println("marks:");
            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("credid and marks:");
        for (int i=0;i<no_of_subjects; i++)
        {
                                        System.out.println("subject:"+(i+1)+"    credits:"
+credits[i]+"marks:"+marks[i]);
        }
    }
    public int gradepoint (int marks)
    {
        if (marks>= 90)
        {
            return 10;
        }
        else if (marks>= 80)
        {
            return 9;
        }
        else if (marks>= 70)
        {
            return 8;
        }
        else if (marks>= 60)
        {
            return 7;
        }
```

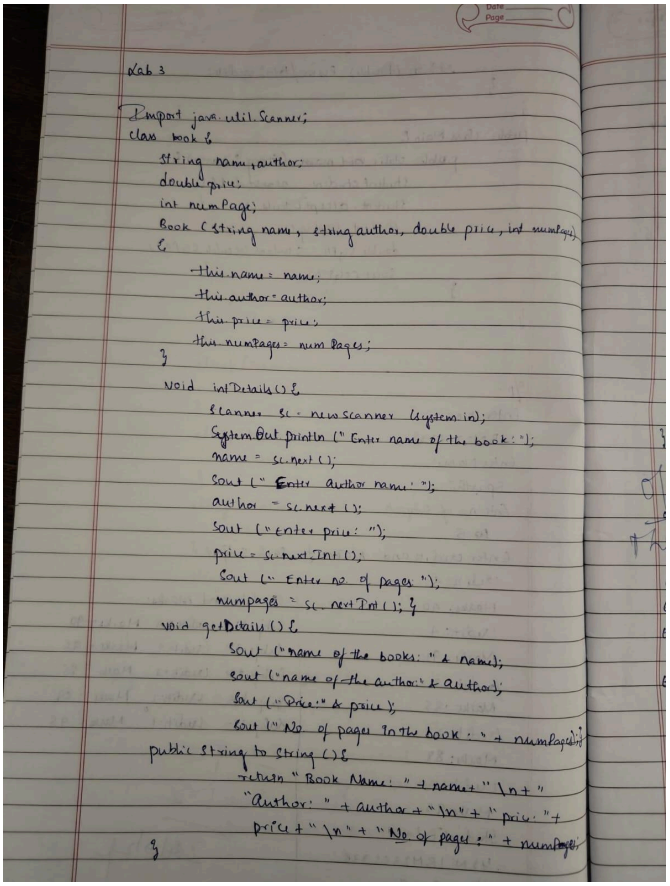
```
        else if (marks>= 50)
        {
            return 6;
        }
        else if (marks>= 40)
        {
            return 5;
        }
        else
        {
            return 0;
        }
    }
    public double calculate_sgpa()
    {
        int total_credits =0;
        int sum =0;
        for (int i =0; i< no_of_subjects; i++)
        {
            int grade_point = gradepoint (marks[i]);
            sum+= grade_point * credits[i];
            total_credits+=credits[i];
        }
        return (double)sum/total_credits;
    }
}
public class Main{
    public static void main(String[] args)
```

```
    {
        student student = new student();
        student.accept_details();
        student.display();
        double sgpa= student.calculate_sgpa();
        System.out.println("sgpa");
    }
}
```

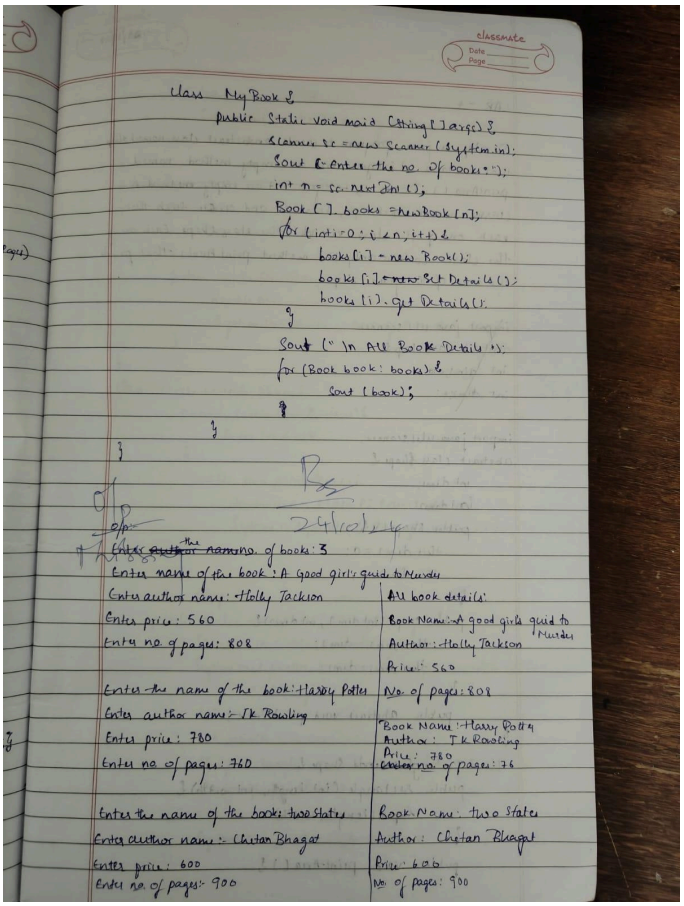
```
D:\Java_Lab_Programs>java Main.java
Enter USN:
1BM23CS337
Enter Name:
Spandana
Enter number of subjects:
8
Enter credits and marks for each subject:
Credits for subject 1: 4
Marks for subject 1: 95
Credits for subject 2: 4
Marks for subject 2: 94
Credits for subject 3: 3
Marks for subject 3: 88
Credits for subject 4: 3
Marks for subject 4: 83
Credits for subject 5: 3
Marks for subject 5: 91
Credits for subject 6: 1
Marks for subject 6: 95
Credits for subject 7: 1
Marks for subject 7: 95
Credits for subject 8: 1
Marks for subject 8: 97
Student's details:
USN: 1BM23CS337
Name: Spandana
Credits and marks of each subject are:
Subject 1: credits = 4, marks = 95
Subject 2: credits = 4, marks = 94
Subject 3: credits = 3, marks = 88
Subject 4: credits = 3, marks = 83
Subject 5: credits = 3, marks = 91
Subject 6: credits = 1, marks = 95
Subject 7: credits = 1, marks = 95
Subject 8: credits = 1, marks = 97
SGPA = 9.7
D:\Java_Lab_Programs>
```

LABORATORY PROGRAM - 3

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 numPage;
    Book (String name, String author, double price, int numPages)
    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }
    void intDetails() {
        Scanner sc = new Scanner (System.in);
        System.out.println ("Enter name of the book: ");
        name = sc.next();
        Sout (" Enter author name: ");
        author = sc.next();
        Sout ("Enter price: ");
        price = sc.nextInt();
        Sout (" Enter no of pages ");
        numpages = sc.nextInt(); }
    void getDetails() {
        Sout ("name of the books: " + name);
        Sout ("name of the author" + author);
        Sout ("Price:" + price);
        Sout ("No of pages in the book: " + numPages);
public String toString() {
        return "Book Name: " + name + "\n" +
        "Author: " + author + "\n" + "price: " +
        price + "\n" + "No of pages: " + numPages;
}
```




```
import java.util.Scanner;
class Book{
   String name, author;
   double price;
   int numpage;
   Book(String name , String author, double price, int numpage)
   {
      this.name =name;
      this.author =author;
      this.price =price;
      this.numpage =numpage;
   }
   Book(){
      this.name="";
```

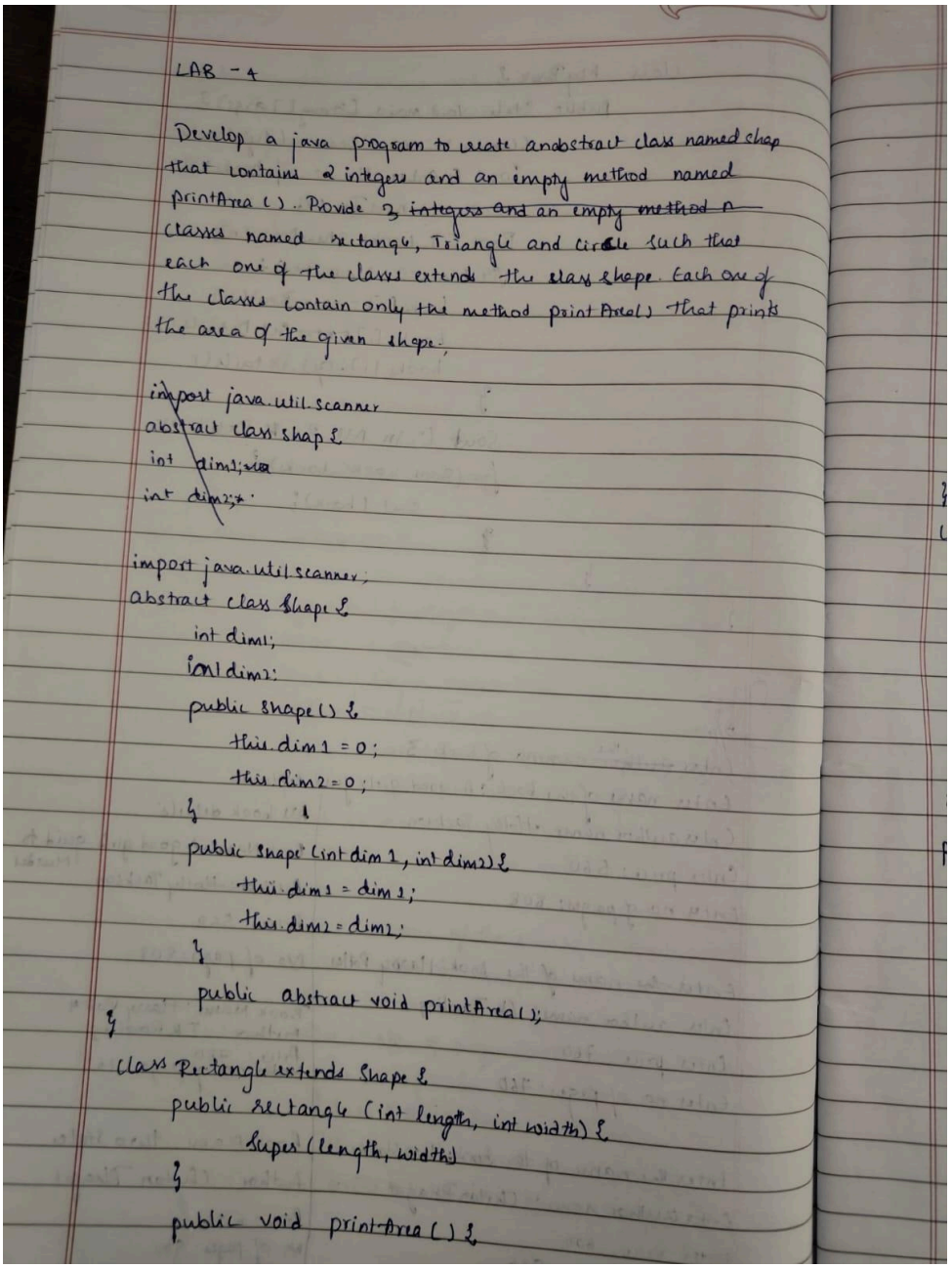
```
        this.author ="";
        this.price =0.0;
        this.numpage =0;
    }
    void int_details(){
        Scanner sc = new Scanner(System.in);
        System.out.println("enter name of the book:");
        name = sc.next();
        System.out.println("enter author name:");
        author = sc.next();
        System.out.println("enter price:");
        price= sc.nextDouble();
        System.out.println("enter no of pages:");
        numpage =sc.nextInt();
    }
    void get_details(){
        System.out.println("name of the book:" +name);
        System.out.println("name of the author :" +author);
        System.out.println("price:"+price);
        System.out.println("no of pages in the book:"+numpage);
    }
    public String toString(){
          return"book name:" + name+ "\n "+ "author:" +author+ "\n"+ "price:"
+price+ "\n"+ "no of pages:" +numpage;
    }
}
class myBook{
    public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("enter 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].int_details();
           books[i].get_details();
        }
        System.out.println("all book details:" );
        for(Book book: books)
        {
           System.out.println(book);
        }
    }
}
```

```
D:\Java_Lab_Programs>javac MyBook.java

D:\Java_Lab_Programs>java MyBook
Enter the number of books:
2
Enter the name of the book:
Harry Potter
Enter author name:
J K Rowling
Enter price:
1500
Enter number of pages:
1200
Name of the book:Harry Potter
Name of the author:J K Rowling
Price:1500.0
Number of pages in the book:1200
Enter the name of the book:
Silent Patient
Enter author name:
Alex Michaelides
Enter price:
350
Enter number of pages:
287
Name of the book:Silent Patient
Name of the author:Alex Michaelides
Price:350.0
Number of pages in the book:287
All book details:
Book Name:Harry Potter
Author:J K Rowling
Price:1500.0
Number of pages:1200
Book Name:Silent Patient
Author:Alex Michaelides
Price:350.0
Number of pages:287

D:\Java_Lab_Programs>
```

LABORATORY PROGRAM - 4

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.

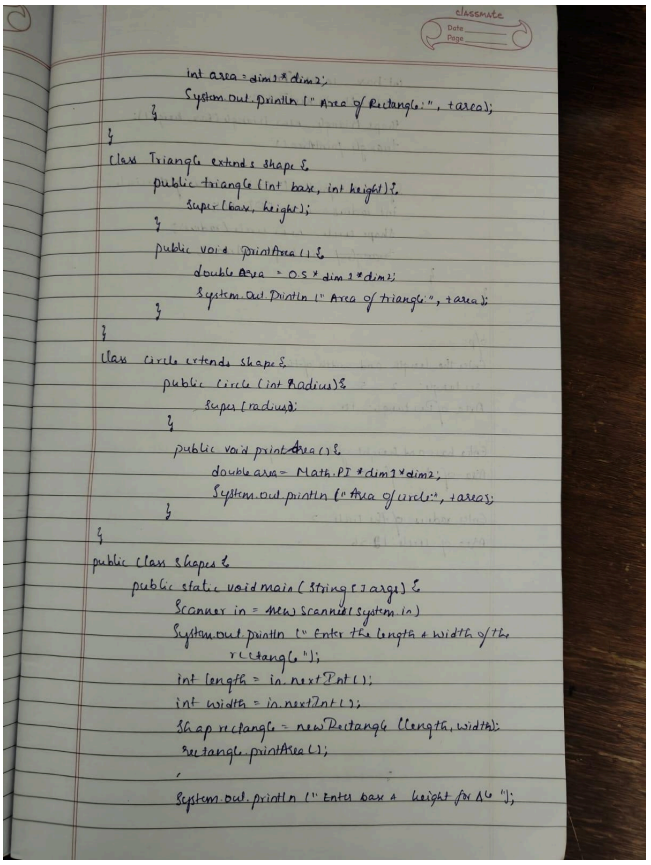


LAB - 4

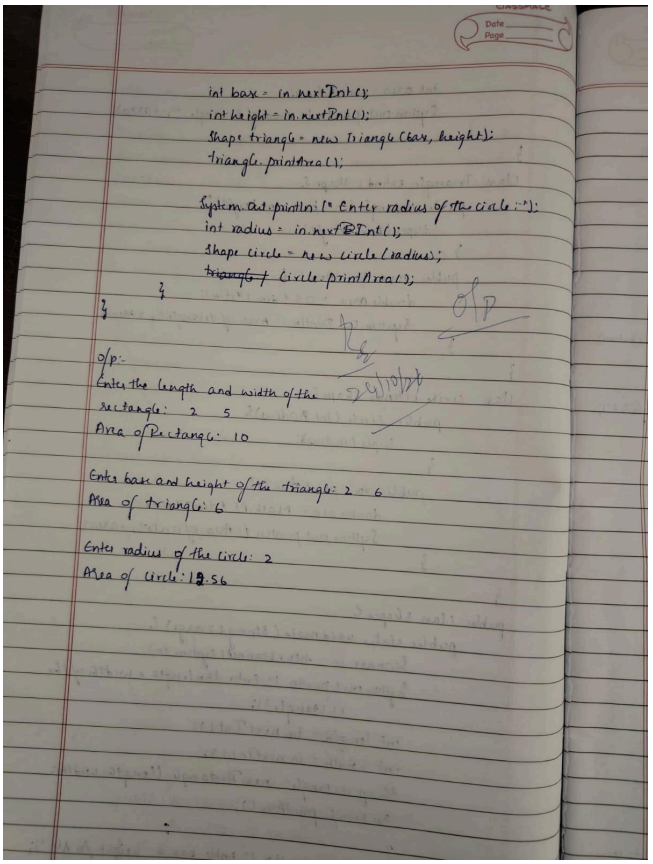
Develop a java program to create an abstract class named shap that contains 2 integers and an empty method named printArea ( ). Provide 3 ~~integers and an empty method n~~ classes named Rectangle, Triangle and Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

```
import java.util.scanner
abstract class shap{
int dim1;
int dim2;
```

```
import java.util.scanner;
abstract class Shape{
    int dim1;
    int dim2;
    public Shape(){
        this.dim1 = 0;
        this.dim2 = 0;
    }
    public Shape(int dim1, int dim2){
        this.dim1 = dim1;
        this.dim2 = dim2;
    }
    public abstract void printArea();
}
class Rectangle extends Shape{
    public Rectangle(int length, int width){
        Super(length, width)
    }
    public void printArea(){
```



```
        int area = dim1 * dim2;
        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 * dim1 * dim2;
        System.out.println("Area of triangle:", +area);
    }
}
class Circle extends Shape{
    public Circle(int Radius){
        super(radius);
    }
    public void printArea() {
        double area = Math.PI * dim1 * dim2;
        System.out.println("Area of circle:", +area);
    }
}
public class Shapes {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in)
        System.out.println("Enter the length & width of the
            rectangle");
        int length = in.nextInt();
        int width = in.nextInt();
        Shape rectangle = new Rectangle(length, width);
        rectangle.printArea();

        System.out.println("Enter base & height for Δle");
```



```
        int base = in.nextInt();
        int height = in.nextInt();
        Shape triangle = new Triangle(base, height);
        triangle.printArea();

        System.out.println("Enter radius of the circle:");
        int radius = in.nextInt();
        Shape circle = new Circle(radius);
        triangle = circle.printArea();
    }
}
```

o/p

o/p:

```
Enter the length and width of the
rectangle: 2  5
Area of Rectangle: 10

Enter base and height of the triangle: 2  6
Area of triangle: 6

Enter radius of the circle: 2
Area of circle: 12.56
```

```
import java.util.Scanner;

abstract class Shape {
    int dim1;
    int dim2;


    Shape(int dim1, int dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }


    abstract void printArea();
}

class Rectangle extends Shape {

    Rectangle(int length, int width) {
        super(length, width);
    }

    @Override
    void printArea() {
        int area = dim1 * dim2;
        System.out.println("Area of Rectangle: " + area);
    }
}
```

```
class Triangle extends Shape {

    Triangle(int base, int height) {
        super(base, height);
    }

    @Override
    void printArea() {
        double area = 0.5 * dim1 * dim2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape {

    Circle(int radius) {
        super(radius, 0);
    }

    @Override
    void printArea() {
        double area = Math.PI * dim1 * dim1;
        System.out.println("Area of Circle: " + area);
    }
}

public class Shapes1 {
```

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



        System.out.println("Enter length and width for Rectangle:");
        int length = sc.nextInt();
        int width = sc.nextInt();
        Rectangle a1 = new Rectangle(length, width);
        a1.printArea();



        System.out.println("Enter base and height for Triangle:");
        int base = sc.nextInt();
        int height = sc.nextInt();
        Triangle a2 = new Triangle(base, height);
        a2.printArea();



        System.out.println("Enter radius for Circle:");
        int radius = sc.nextInt();
        Circle a3 = new Circle(radius);
        a3.printArea();
    }
}
```

```
D:\Java_Lab_Programs>javac Shapes.java

D:\Java_Lab_Programs>java Shapes
Enter length and width for Rectangle:
4
2
Area of Rectangle: 8
Enter base and height for Triangle:
2
4
Area of Triangle: 4.0
Enter radius for Circle:
3
Area of Circle: 28.274333882308138

D:\Java_Lab_Programs>
```

LABORATORY PROGRAM - 5

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:

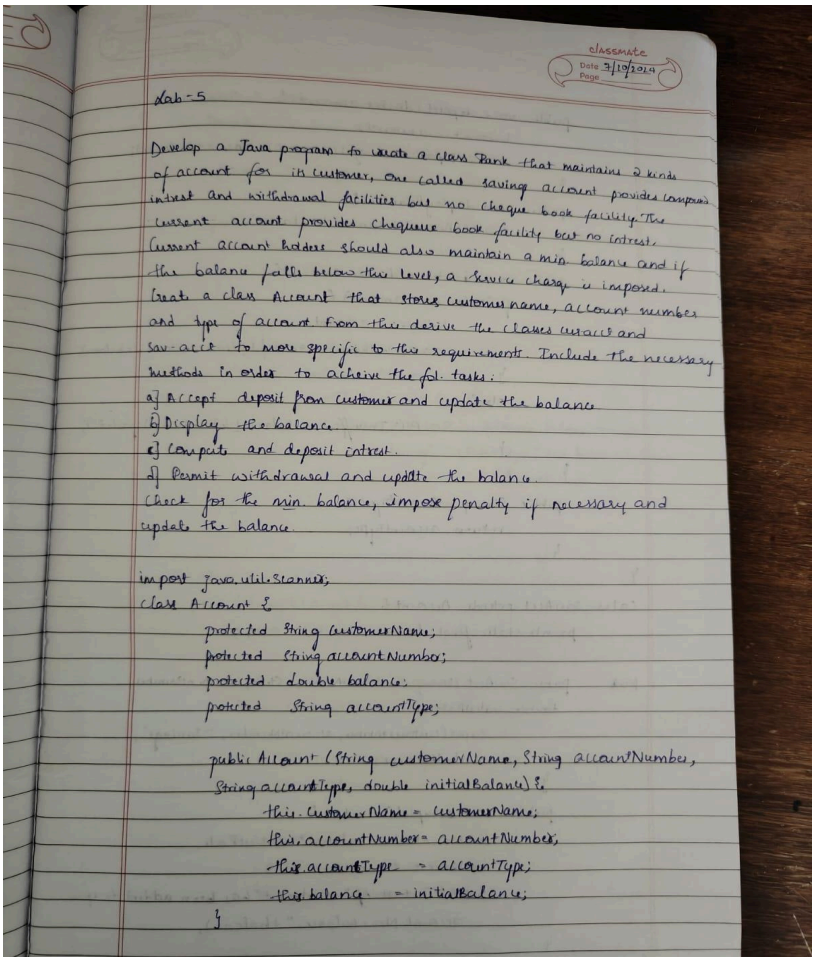
   a) Accept deposit from customer and update the balance.

   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.

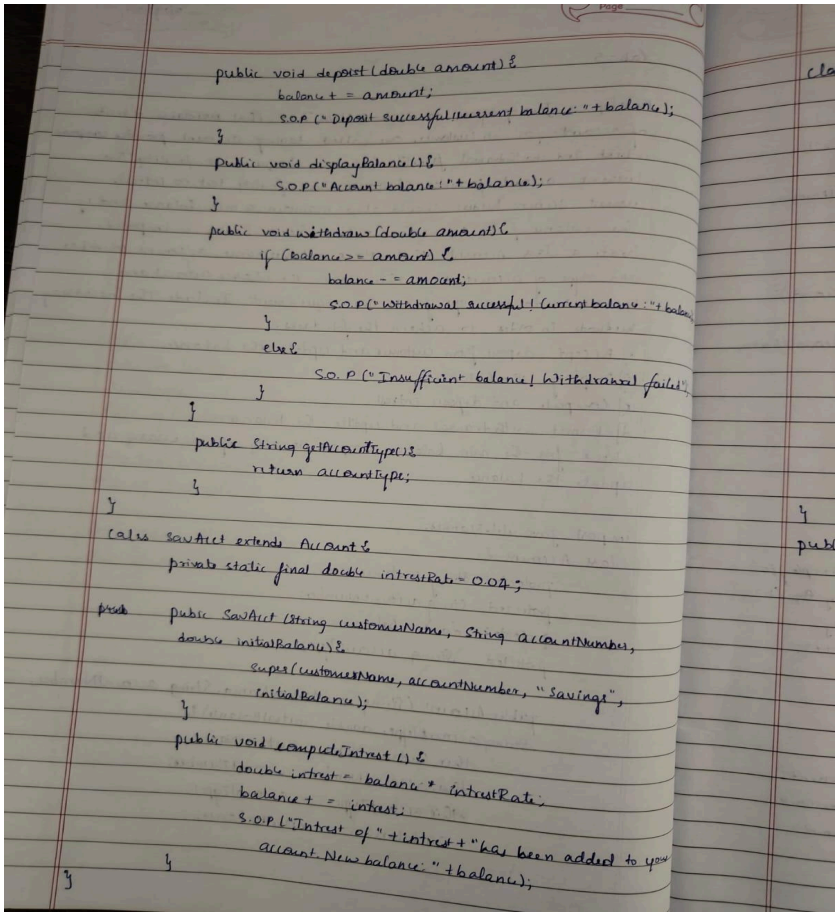


Lab-5

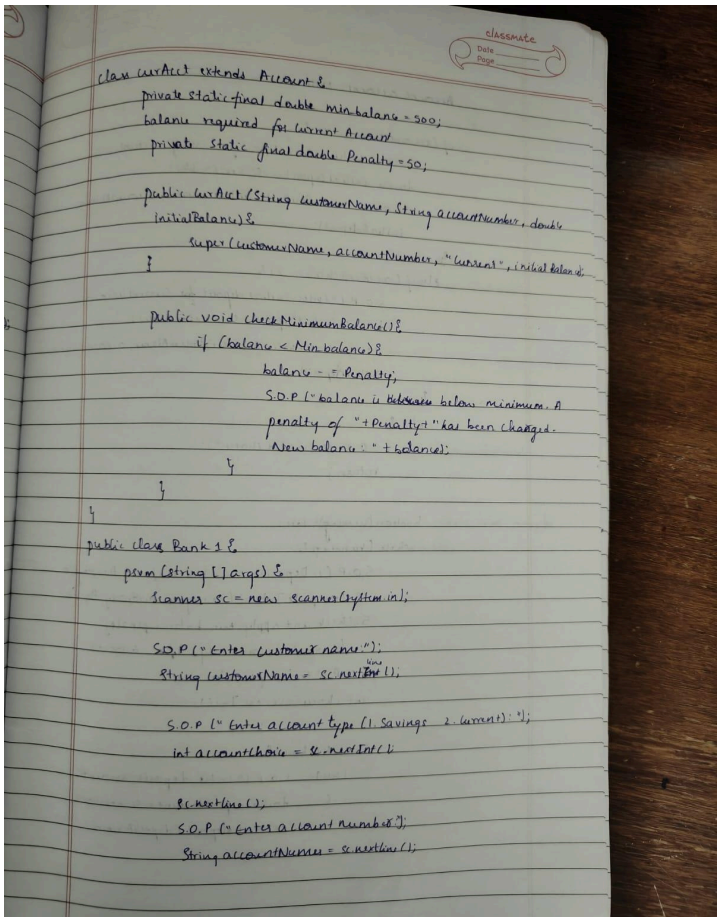
Develop a Java program to create a class Bank that maintains 2 kinds of account for its customer, one called saving account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a min. 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 curAcct and savAcct to make specific to the requirements. Include the necessary methods in order to achieve the fol. 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 min. balance, impose penalty if necessary and update the balance

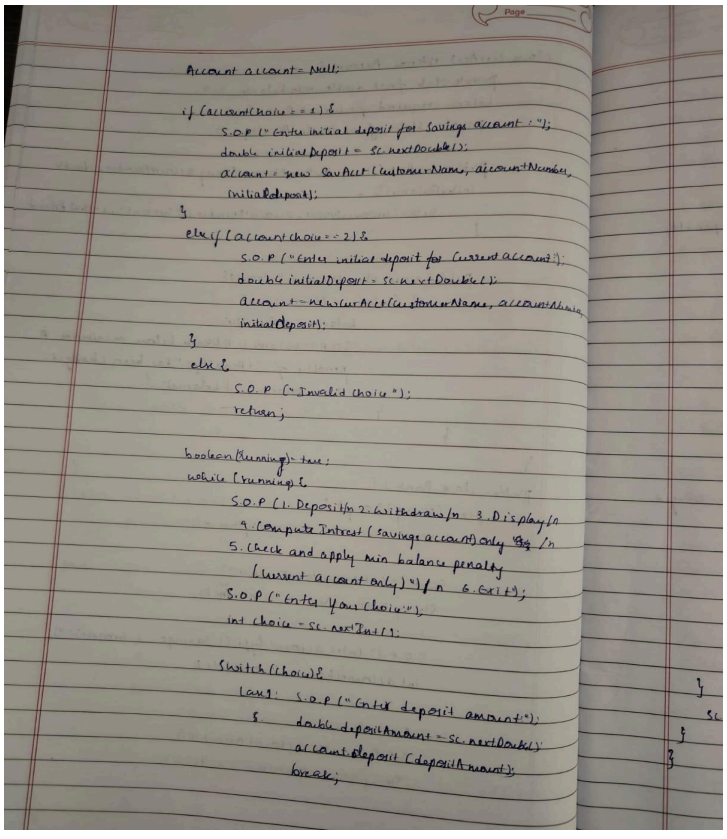
```
import java.util.Scanner;
class Account {
    protected String customerName;
    protected String accountNumber;
    protected double balance;
    protected String accountType;

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



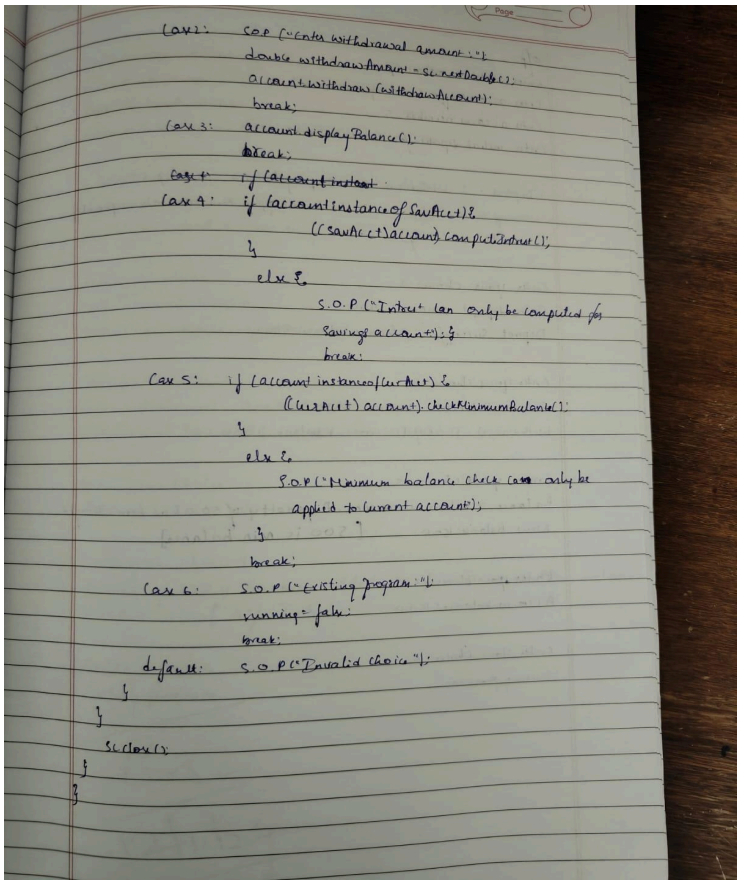
```
public void deposit(double amount) {
    balance += amount;
    S.O.P("Deposit successful! Current balance: " + balance);
}
public void displayBalance() {
    S.O.P("Account balance: " + balance);
}
public void withdraw(double amount) {
    if (balance >= amount) {
        balance -= amount;
        S.O.P("Withdrawal successful! Current balance: " + balance
    }
    else {
        S.O.P("Insufficient balance! Withdrawal failed");
    }
}
public String getAccountType() {
    return accountType;
}
}
class savAcct extends Account {
    private static final double interestRate = 0.07;

    public SavAcct(String customerName, String accountNumber,
    double initialBalance) {
        super(customerName, accountNumber, "Savings",
        initialBalance);
    }
    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        S.O.P("Interest of " + interest + " has been added to your
        account. New balance: " + balance);
    }
}
```



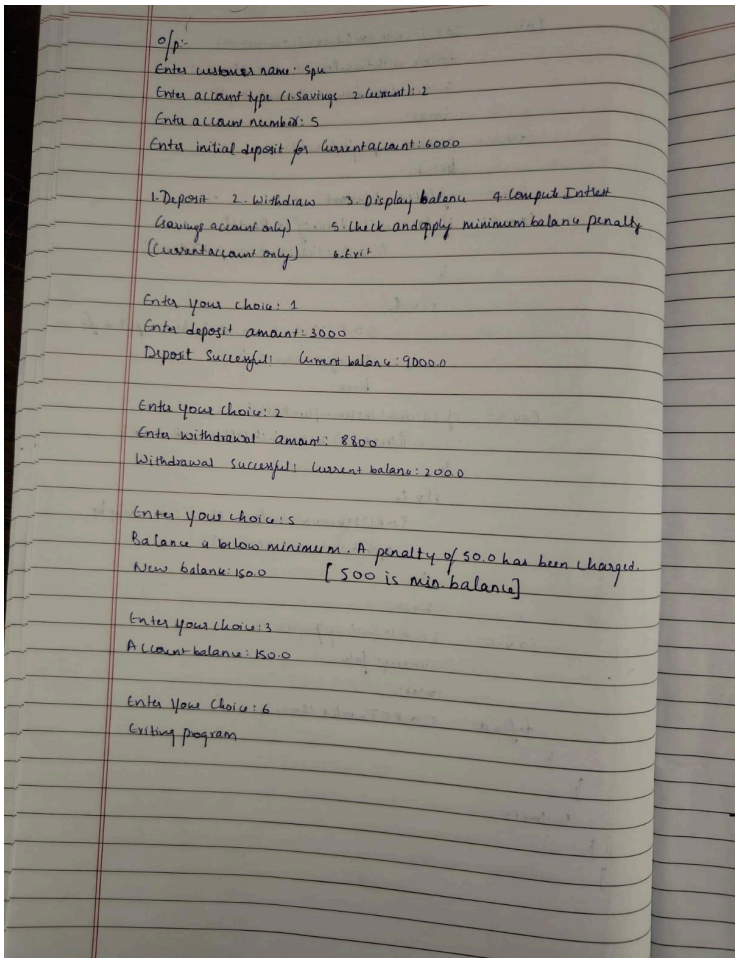
```
class curAcct extends Account {
    private static final double minBalance = 500;
    balance required for current Account
    private static final double Penalty = 50;

    public curAcct (String customerName, String accountNumber, double
    initialBalance) {
        super (customerName, accountNumber, "current", initial balance);
    }

    public void checkMinimumBalance() {
        if (balance < MinBalance) {
            balance -= Penalty;
            S.O.P ("balance is below minimum. A
            penalty of "+Penalty+" has been charged.
            New balance: "+balance);
        }
    }
}

public class Bank1 {
    psvm (String [] args) {
    Scanner sc = new Scanner(System.in);

    S.O.P ("Enter customer name");
    String customerName = sc.nextLine();

    S.O.P ("Enter account type (1. Savings 2. Current): ");
    int accountChoice = sc.nextInt();

    sc.nextLine();
    S.O.P ("Enter account number");
    String accountNumber = sc.nextLine();
```



```
Account account = null;

if (accountChoice == 1) {
    S.O.P ("Enter initial deposit for savings account: ");
    double initialDeposit = sc.nextDouble();
    account = new SavAcct (customerName, accountNumber,
    initialDeposit);
}
else if (accountChoice == 2) {
    S.O.P ("Enter initial deposit for Current account");
    double initialDeposit = sc.nextDouble();
    account = new curAcct (customerName, accountNumber,
    initialDeposit);
}
else {
    S.O.P ("Invalid choice");
    return;

boolean Running = true;
while (running) {
    S.O.P (1. Deposit\n 2. Withdraw\n 3. Display\n
    4. Compute Intrest (savings account only)\n
    5. Check and apply min balance penalty
    (current account only)\n 6. Exit");
    S.O.P ("Enter your choice:");
    int choice = sc.nextInt();

    switch (choice) {
        case 1: S.O.P ("Enter deposit amount:");
        double depositAmount = sc.nextDouble();
        account.deposit (depositAmount);
        break;
```



```
case 2: S.o.p ("Enter withdrawal amount: ");
        double withdrawAmount = sc.nextDouble();
        account.withdraw (withdrawAccount);
        break;
case 3: account.displayBalance();
        break;
case 4: if (account instanceof
case 4: if (account instanceof SavAcct) {
            ((SavAcct)account).computeInterest();
        }
        else {
            S.o.P ("Interest can only be computed for
            Savings account"); }
            break;
case 5: if (account instanceof CurAcct) {
            ((CurAcct) account).checkMinimumBalance();
        }
        else {
            S.o.P ("Minimum balance check can only be
            applied to Current account");
        }
        break;
case 6: S.o.P ("Existing program");
        running = false;
        break;
default: S.o.P ("Invalid choice");
    }
}
sc.close();
}
}
```



o/p:

```
Enter customer name: spu
Enter account type (1.Savings 2.Current): 2
Enter account number: 5
Enter initial deposit for Current account: 6000

1.Deposit 2.Withdraw 3.Display balance 4.Compute Interest
(Savings account only) 5.Check and apply minimum balance penalty
(Current account only) 6.Exit

Enter your choice: 1
Enter deposit amount: 3000
Deposit Successful! Current balance: 9000.0

Enter your choice: 2
Enter withdrawal amount: 8800
Withdrawal Successful! Current balance: 200.0

Enter your choice: 5
Balance is below minimum. A penalty of 50.0 has been charged.
New balance: 150.0      [500 is min. balance]

Enter your choice: 3
Account balance: 150.0

Enter your choice: 6
Exiting program
```

```
import java.util.Scanner;


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


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


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


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

   public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal successful! Current balance: " + balance);
        } else {
            System.out.println("Insufficient balance! Withdrawal failed.");
        }
    }


    public String getAccountType() {
        return accountType;
    }
}
```

```
class SavAcct extends Account {
    private static final double interestRate = 0.04;


    public SavAcct(String customerName, String accountNumber, double initialBalance)
{
        super(customerName, accountNumber, "Savings", initialBalance);
    }


    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest of " + interest + " has been added to your account.
New balance: " + balance);
    }
}


class CurAcct extends Account {
    private static final double MIN_BALANCE = 500;
    private static final double PENALTY = 50;
    public CurAcct(String customerName, String accountNumber, double initialBalance)
{
        super(customerName, accountNumber, "Current", initialBalance);
    }


    public void checkMinimumBalance() {
        if (balance < MIN_BALANCE) {
            balance -= PENALTY;
            System.out.println("Balance is below minimum. A penalty of " + PENALTY +
" has been charged. New balance: " + balance);
        }
    }
}

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


        System.out.print("Enter customer name: ");
```

```
        String customerName = scanner.nextLine();

        System.out.print("Enter account type (1 for Savings, 2 for Current): ");
        int accountChoice = scanner.nextInt();

        scanner.nextLine();
        System.out.print("Enter account number: ");
        String accountNumber = scanner.nextLine();

        Account account = null;

        if (accountChoice == 1) {
            System.out.print("Enter initial deposit for Savings account: ");
            double initialDeposit = scanner.nextDouble();
account = new SavAcct(customerName, accountNumber, initialDeposit);
 } else if (accountChoice == 2) {
            System.out.print("Enter initial deposit for Current account: ");
            double initialDeposit = scanner.nextDouble();
            account = new CurAcct(customerName, accountNumber, initialDeposit);
        } else {
            System.out.println("Invalid choice! Exiting program.");
            return;
        }


        boolean running = true;
        while (running) {
            System.out.println("\nBank Operations Menu:");
            System.out.println("1. Deposit \n 2.withdraw \n 3.Display Balance 4.  Compute
Interest (Savings account only) 5.       Check and apply minimum balance penalty
(Current account only) 6.Exit ");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

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

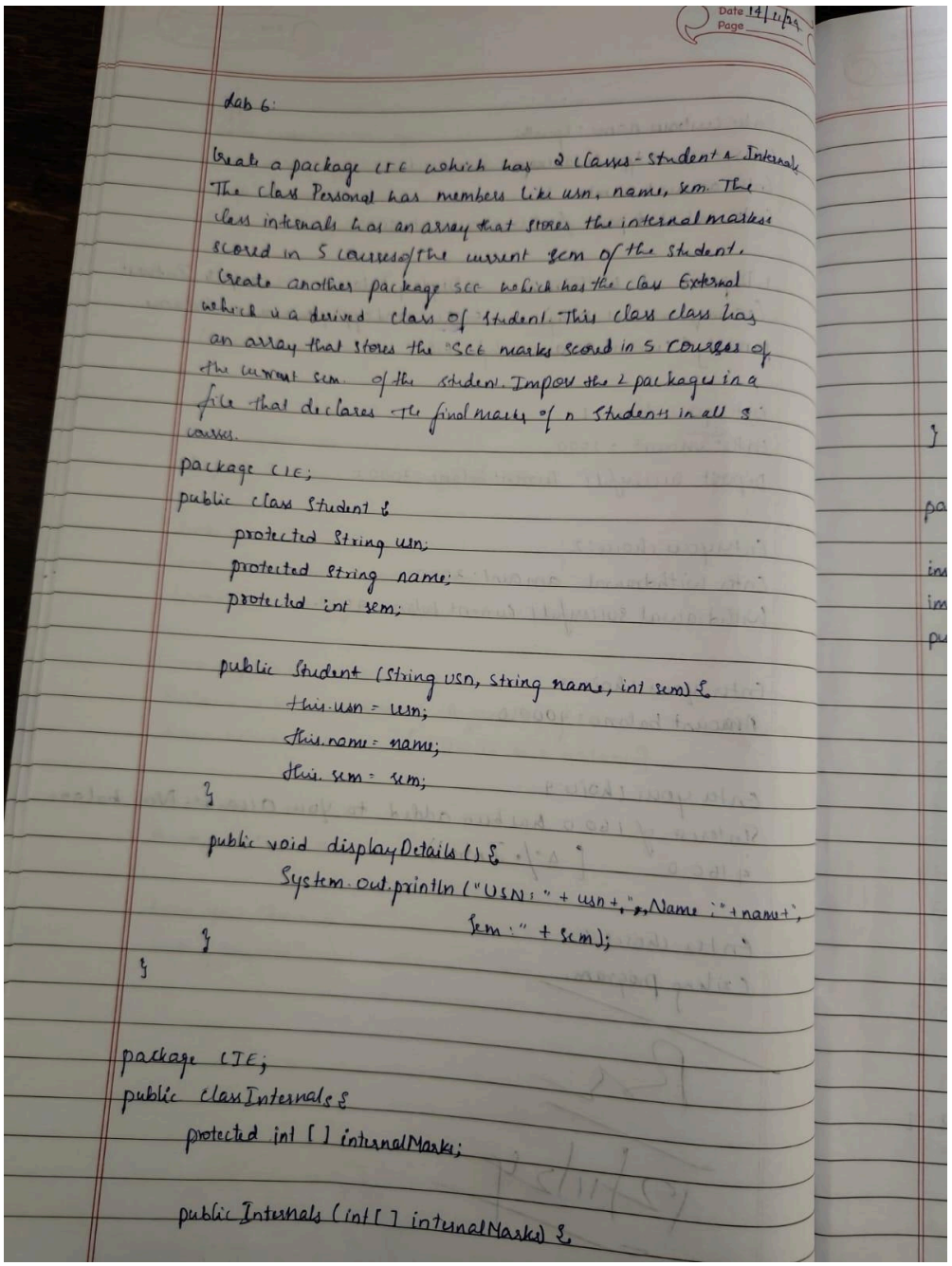
```
                break;
            case 3:
                account.displayBalance();
                break;
            case 4:
                if (account instanceof SavAcct) {
                    ((SavAcct) account).computeInterest();
                } else {
                    System.out.println("Interest can only be computed for Savings
account.");
                }
                break;
            case 5:
                if (account instanceof CurAcct) {
                    ((CurAcct) account).checkMinimumBalance();
                } else {
                    System.out.println("Minimum balance check can only be applied to
Current account.");
                }
                break;
            case 6:
                System.out.println("Exiting program.");
                running = false;
                break;
            default:
                System.out.println("Invalid choice! Please select a valid option.");
        }
    }

    scanner.close();
  }
}
```

```
D:\Java_Lab_Programs>javac Bank.java

D:\Java_Lab_Programs>java Bank
Enter customer name: Shasha
Enter account type (1 for Savings, 2 for Current): 1
Enter account number: SBI123456789
Enter initial deposit for Savings account: 6000

Bank Operations Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest (Savings account only)
5. Check and apply minimum balance penalty (Current account only)
6. Exit
Enter your choice: 4
Interest of 240.0 has been added to your account. New balance: 6240.0

Bank Operations Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest (Savings account only)
5. Check and apply minimum balance penalty (Current account only)
6. Exit
Enter your choice: 3
Account balance: 6240.0

Bank Operations Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest (Savings account only)
5. Check and apply minimum balance penalty (Current account only)
6. Exit
Enter your choice: 2
Enter withdrawal amount: 1000
Withdrawal successful! Current balance: 5240.0
```

```
Bank Operations Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest (Savings account only)
5. Check and apply minimum balance penalty (Current account only)
6. Exit
Enter your choice: 3
Account balance: 5240.0

Bank Operations Menu:
1. Deposit
2. Withdraw
3. Display Balance
4. Compute Interest (Savings account only)
5. Check and apply minimum balance penalty (Current account only)
6. Exit
Enter your choice: 6
Exiting program.

D:\Java_Lab_Programs>
```

LABORATORY PROGRAM - 6

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.

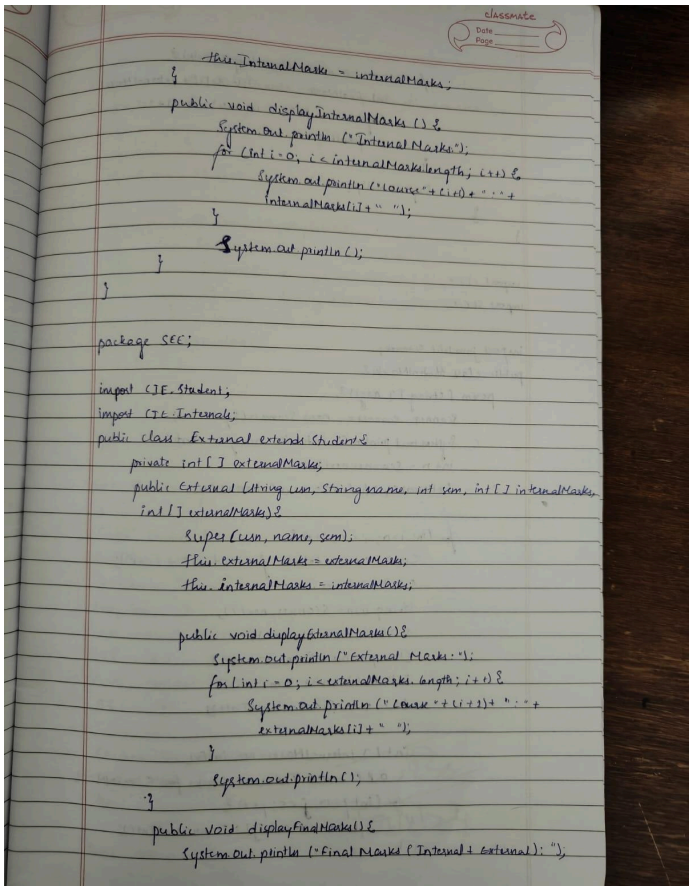


Lab 6:

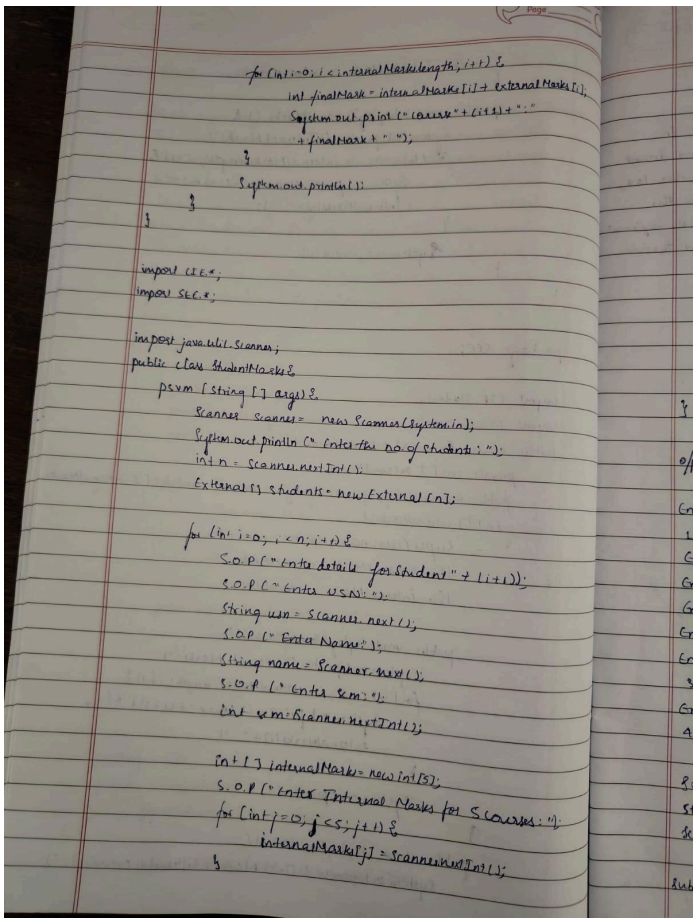
Create a package CIE which has 2 classes - Student & Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in 5 courses of the current sem of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in 5 courses of the current sem of the student. Import the 2 packages in a file that declares the final marks of n students in all 5 courses.

```
package CIE;
public class Student {
    protected String usn;
    protected String name;
    protected int sem;

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

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

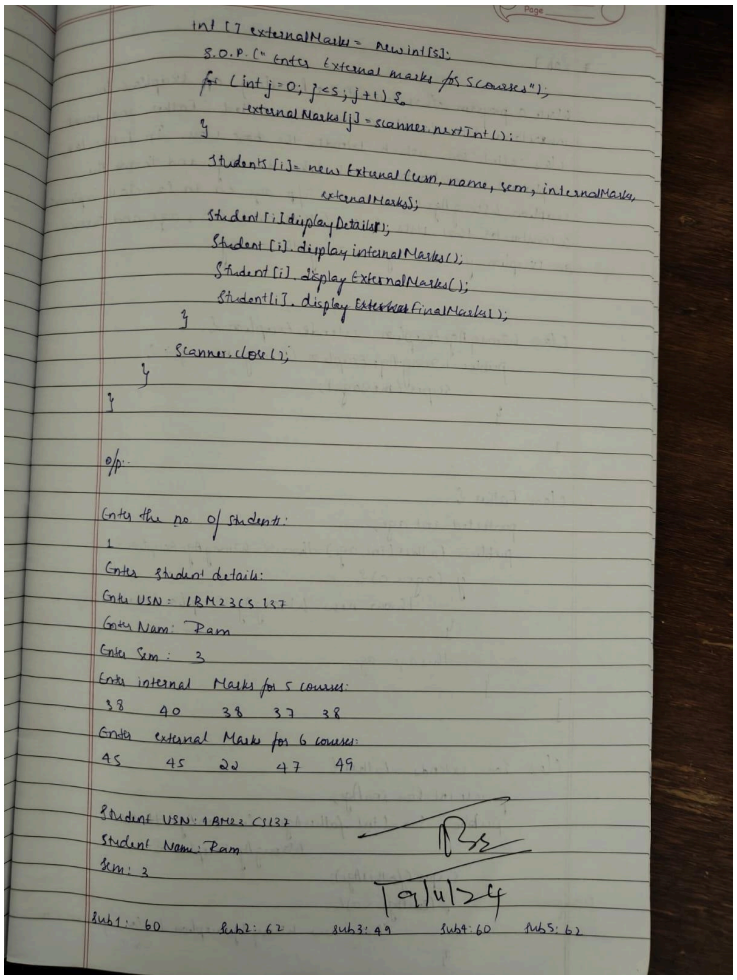
```
package CIE;
public class Internals {
    protected int [ ] internalMarks;

    public Internals (int [ ] internalMarks) {
```



```
        this.InternalMarks = internalMarks;
    }

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

package SEE;

import CIE.Student;
import CIE.Internals;
public class External extends Student {
    private int[] externalMarks;
    public External (String usn, String name, int sem, int[] internalMarks,
    int[] externalMarks) {
        super (usn, name, sem);
        this.externalMarks = externalMarks;
        this.internalMarks = internalMarks;

    public void displayExternalMarks() {
        System.out.println ("External Marks:");
        for (int i = 0; i < externalMarks.length; i++) {
            System.out.println ("Course" + (i+1) + ": " +
            externalMarks[i] + " ");
        }
        System.out.println ();
    }
    public void displayFinalMarks() {
        System.out.println ("Final Marks (Internal + External):");
```



```
        for (int i=0; i < internalMarks.length; i++) {
            int finalMark = internalMarks[i] + externalMarks[i];
            System.out.print ("Course" + (i+1) + ": "
            + finalMark + " ");
        }
        System.out.println ();
    }
}

import CIE.*;
import SEE.*;

import java.util.Scanner;
public class StudentMarks {
    psvm (String [] args) {
        Scanner scanner = new Scanner (System.in);
        System.out.println ("Enter the no of students:");
        int n = scanner.nextInt ();
        External[] students = new External[n];

        for (int i=0; i < n; i++) {
            S.O.P ("Enter details for Student" + (i+1));
            S.O.P ("Enter USN:");
            String usn = Scanner.next ();
            S.O.P ("Enter Name");
            String name = Scanner.next ();
            S.O.P ("Enter sem:");
            int sem = Scanner.nextInt ();

            int[] internalMarks = new int[5];
            S.O.P ("Enter Internal Marks for 5 courses:");
            for (int j=0; j<5; j++) {
                internalMarks[j] = Scanner.nextInt ();
            }
```




```
package CIE;
import java.util.Scanner;
public class Internal {
	public int marksCie[] = new int[5];
	public void getMarks() {
		for(int i=0;i<5;i++) {
			Scanner sc = new Scanner(System.in);
			System.out.println("Enter CIE marks in subject "+(i+1));
			marksCie[i]=sc.nextInt();
		}
	}
	public int returnCieMarks(int i) {
		return marksCie[i];
	}
}
```

```
package SEE;
import CIE.Student;
```

```
import CIE.Internal;
import java.util.Scanner;
public class External extends Student {
	int marksSee[] = new int[5];
	public void getMarks() {
		for(int i=0;i<5;i++) {
			Scanner sc = new Scanner(System.in);
			System.out.println("Enter SEE marks in subject "+(i+1));
			marksSee[i]=sc.nextInt();
		}
	}
	public void calcTotalMarks(Internal i1) {
		for(int i=0;i<5;i++) {
			System.out.println("Subject "+(i+1)+":
"+(i1.returnCieMarks(i)+(marksSee[i]/2)));
		}
		System.out.println();

	}
}


package CIE;
import java.util.Scanner;
public class Student {
	String usn;
	String name;
	int sem;
	public void getd() {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter student USN");
		usn = sc.nextLine();
		System.out.println("Enter student name");
		name = sc.nextLine();
		System.out.println("Enter semester");
		sem = sc.nextInt();
	}
	public void display() {
		System.out.println();
		System.out.println("Student USN: "+usn);
		System.out.println("Student name: "+name);
		System.out.println("Semester: "+sem);
		System.out.println();
	}
```

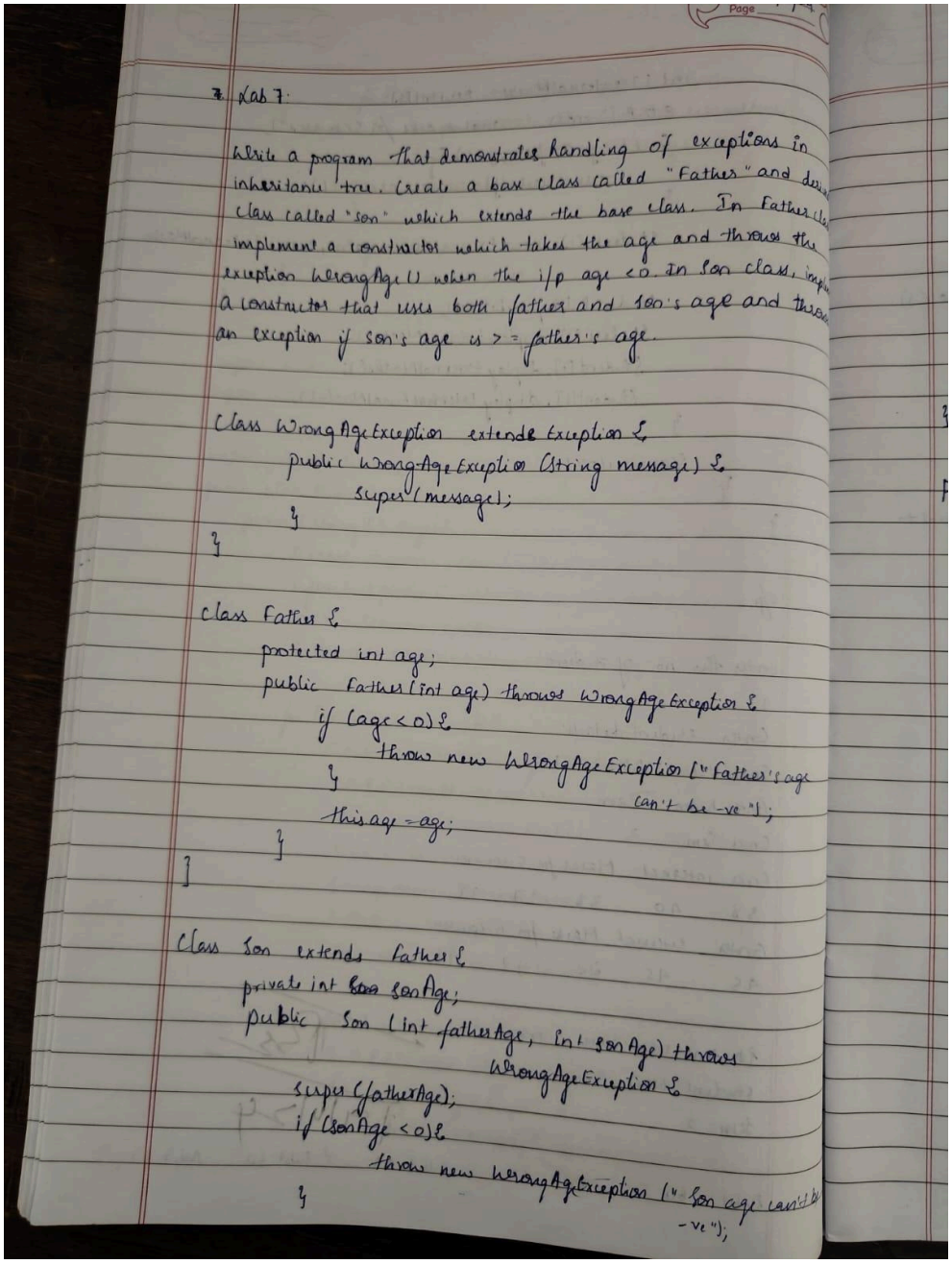
```
}


import CIE.Student;
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;
public class Main {
	public static void main(String args[]) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the number of students whose details you want
to enter");
		int n = sc.nextInt();
		Internals[] i1 = new Internals[n];
		Externals[] e1 = new Externals[n];
		for(int i=0;i<n;i++) {
			System.out.println("Student "+(i+1)+" details:");
			e1[i] = new Externals();
			i1[i] = new Internals();
			e1[i].getd();
			i1[i].getMarks();
			e1[i].getMarks();
		}
		for(int i=0;i<n;i++) {
			e1[i].display();
			e1[i].calcTotalMarks(i1[i]);
		}

	}
}
```

```
D:\package>java Main
Enter the number of students whose details you want to enter
2
Student 1 details:
Enter student USN
1BM23CS330
Enter student name
Sinchana Hemanth
Enter semester
3
Enter CIE marks in subject 1
45
Enter CIE marks in subject 2
49
Enter CIE marks in subject 3
48
Enter CIE marks in subject 4
40
Enter CIE marks in subject 5
50
Enter SEE marks in subject 1
95
Enter SEE marks in subject 2
98
Enter SEE marks in subject 3
90
Enter SEE marks in subject 4
100
Enter SEE marks in subject 5
100
Student 2 details:
Enter student USN
1BM23CS317
Enter student name
Shreya Raj
Enter semester
3
Enter CIE marks in subject 1
48
Enter CIE marks in subject 2
40
Enter CIE marks in subject 3
50
Enter CIE marks in subject 4
45
Enter CIE marks in subject 5
42
Enter SEE marks in subject 1
100
Enter SEE marks in subject 2
90
Enter SEE marks in subject 3
96
Enter SEE marks in subject 4
92
Enter SEE marks in subject 5
90
```

```
Student USN: 1BM23CS330
Student name: Sinchana Hemanth
Semester: 3

Subject 1: 92
Subject 2: 98
Subject 3: 93
Subject 4: 90
Subject 5: 100


Student USN: 1BM23CS317
Student name: Shreya Raj
Semester: 3

Subject 1: 98
Subject 2: 85
Subject 3: 98
Subject 4: 91
Subject 5: 87


D:\package>
```

LABORATORY PROGRAM - 7

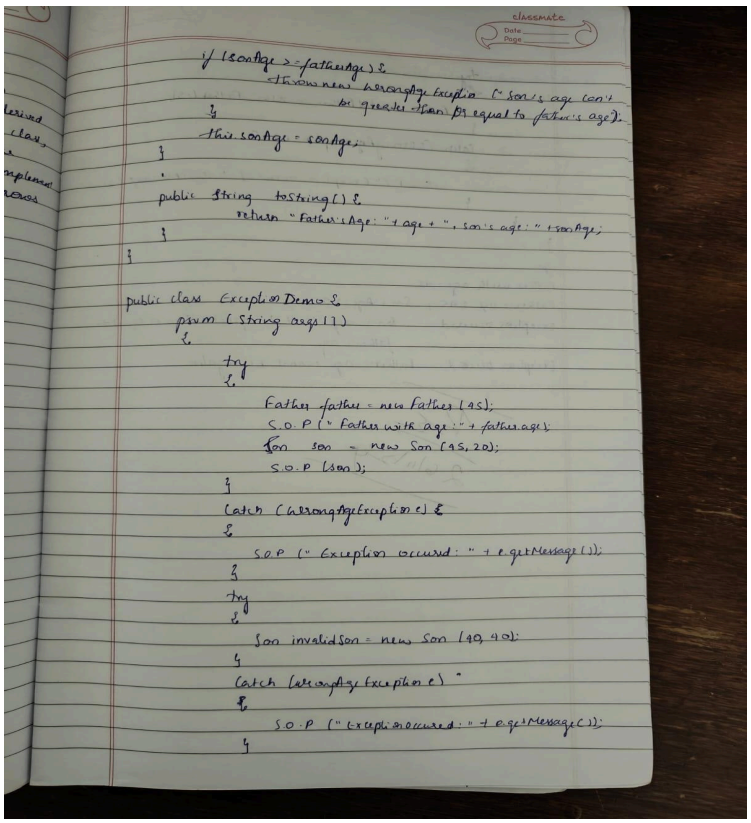
7. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age<0. In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



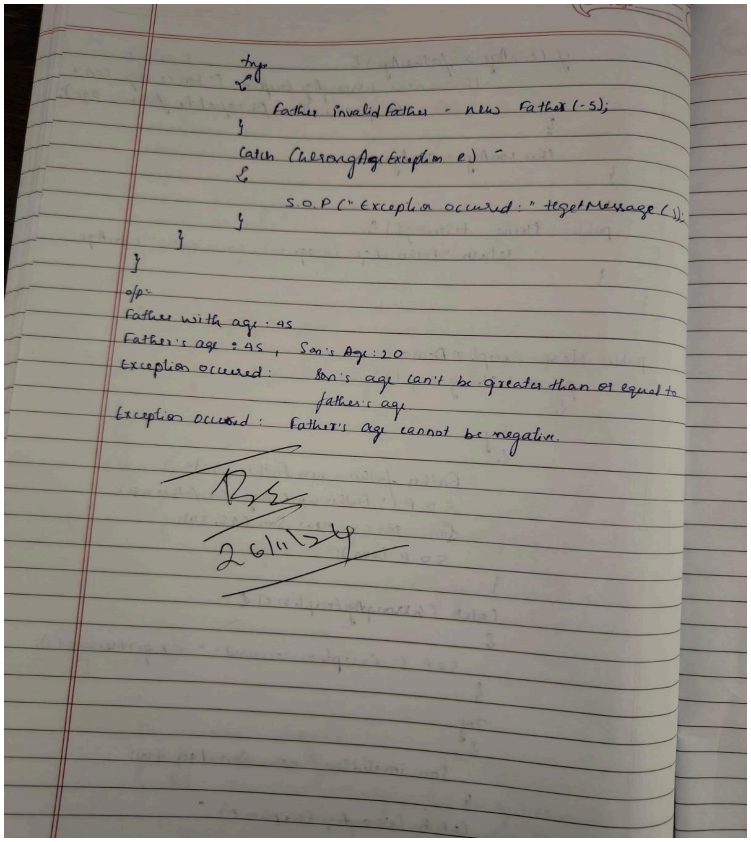
Lab 7:

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 i/p age < 0. In Son class, implement a constructor that uses both father and son's age and throws an exception if son's age is >= father's age.

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

class Father {
    protected int age;
    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Father's age can't be -ve");
        }
        this.age = age;
    }
}

class Son extends Father {
    private int sonAge;
    public Son(int fatherAge, int sonAge) throws WrongAgeException {
        super(fatherAge);
        if (sonAge < 0) {
            throw new WrongAgeException("Son age can't be -ve");
```



```
if (sonAge >= fatherAge) {
        throw new WrongAgeException ("Son's age can't
                be greater than or equal to father's age");
    }
    this.sonAge = sonAge;
}

public String toString() {
    return "Father's Age: " + age + ", Son's age: " + sonAge;
}
}

public class ExceptionDemo {
    psvm (String args[])
    {
        try
        {
            Father father = new Father(45);
            S.O.P("Father with age: " + father.age);
            Son son = new Son(45, 20);
            S.O.P(son);
        }
        Catch (WrongAgeException e) {
        {
            S.O.P ("Exception occured: " + e.getMessage());
        }
        try
        {
            Son invalidSon = new Son(40, 40);
        }
        Catch (WrongAgeException e)
        {
            S.O.P ("Exception occured: " + e.getMessage());
        }
```



```
        try
        {
            Father invalidFather = new Father(-5);
        }
        Catch (WrongAgeException e)
        {
            S.O.P("Exception occured: " + e.getMessage());
        }
    }
}
```

o/p:

Father with age: 45
Father's age: 45, Son's Age: 20
Exception occured: Son's age can't be greater than or equal to father's age
Exception occured: Father's age cannot be negative.

26/11/24

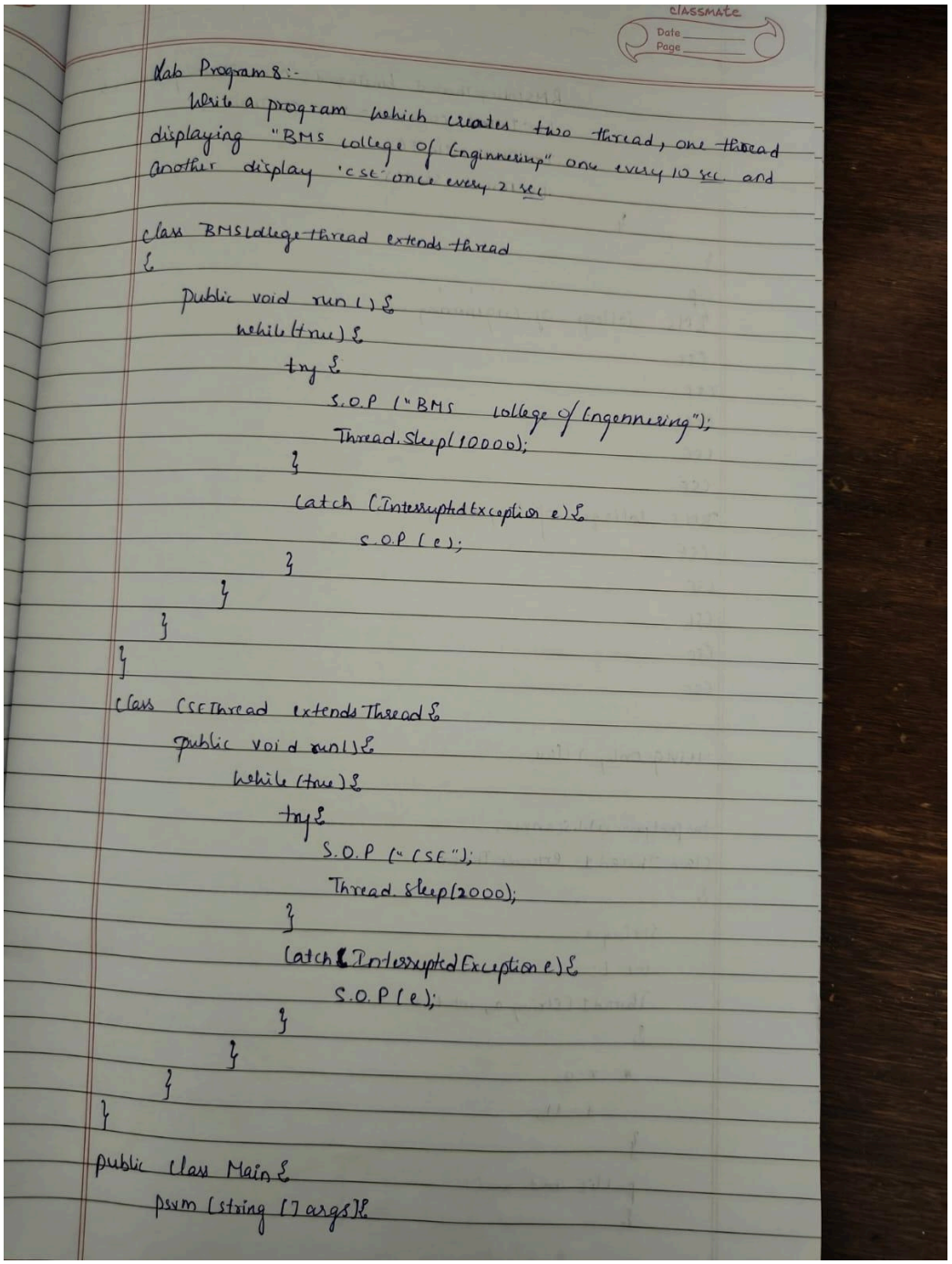
```
class WrongAgeException extends Exception {
    public WrongAgeException(String message) {
        super(message);
    }
}

class Father {
    protected int age;

    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Father's age cannot be negative.");
        }
        this.age = age;
    }
}

class Son extends Father {
    private int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAgeException {
        super(fatherAge);
        if (sonAge < 0) {
            throw new WrongAgeException("Son's age cannot be negative.");
        }
        if (sonAge >= fatherAge) {
            throw new WrongAgeException("Son's age cannot be greater than or equal to
Father's age.");
        }
        this.sonAge = sonAge;
    }

    @Override
    public String toString() {
        return "Father's Age: " + age + ", Son's Age: " + sonAge;
    }
}

public class ExceptionInheritanceDemo {
    public static void main(String[] args) {
        try {
            Father father = new Father(45);
            System.out.println("Father created with age: " + father.age);
```

```
            Son son = new Son(45, 20);
            System.out.println(son);
        } catch (WrongAgeException e) {
            System.err.println("Exception occurred: " + e.getMessage());
        }

        try {
            Son invalidSon = new Son(40, 40);
        } catch (WrongAgeException e) {
            System.err.println("Exception occurred: " + e.getMessage());
        }

        try {
            Father invalidFather = new Father(-5);
        } catch (WrongAgeException e) {
            System.err.println("Exception occurred: " + e.getMessage());
        }
    }
}
```

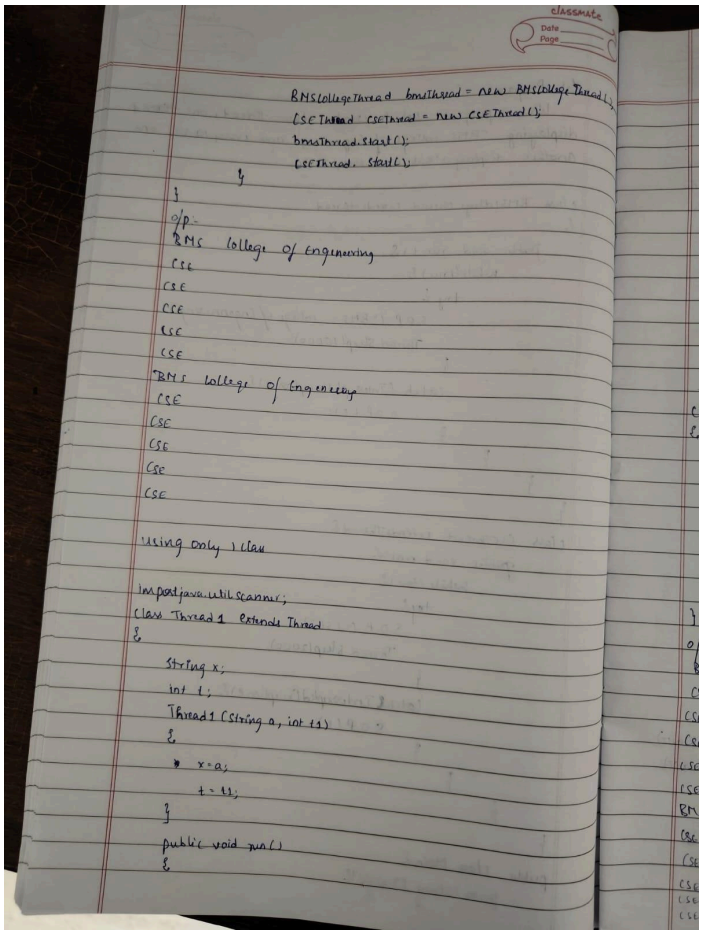
```
D:\Java_Lab_Programs>javac ExceptionInheritanceDemo.java

D:\Java_Lab_Programs>java ExceptionInheritanceDemo
Father created with age: 45
Father's Age: 45, Son's Age: 20
Exception occurred: Son's age cannot be greater than or equal to Father's age.
Exception occurred: Father's age cannot be negative.
```

LABORATORY PROGRAM - 8

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 Program 8:-
  Write a program which creates two thread, one thread
displaying "BMS college of Engineering" one every 10 sec. and
another display 'CSE' once every 2 sec.

class BMSCollegeThread extends Thread
{
   public void run(){
      while(true){
         try{
            S.O.P("BMS College of Engineering");
            Thread.sleep(10000);
         }
         catch(InterruptedException e){
            S.O.P(e);
         }
      }
   }
}
class CSEThread extends Thread{
   public void run(){
      while(true){
         try{
            S.O.P("CSE");
            Thread.sleep(2000);
         }
         catch(InterruptedException e){
            S.O.P(e);
         }
      }
   }
}
public class Main{
   psvm(String[] args){
```



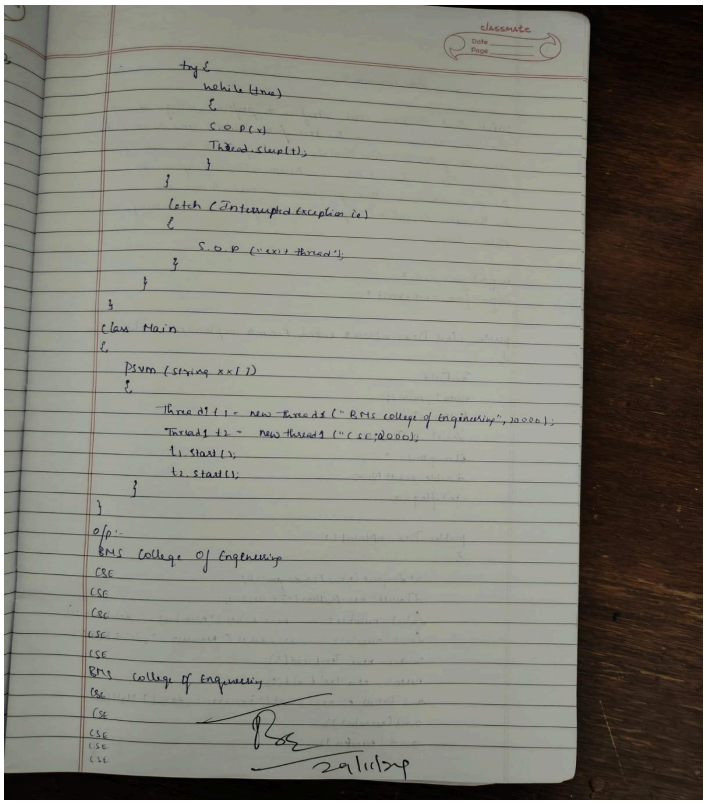
```
BMSCollegeThread bmsThread = NEW BMSCollegeThread();
CSEThread csEThread = NEW CSEThread();
bmsThread.start();
csEThread.start();
        }
}
```

o/p:-

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
```

using only 1 class

```
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)
        {
        S.O.P(x);
        Thread.sleep(t);
        }
    }
    catch(InterruptedException e)
    {
        S.O.P("exit thread");
    }
}
}
class Main
{
    Psvm(String xx[])
    {
        Thread1 t1 = new Thread1("BMS college of Engineering", 10000);
        Thread1 t2 = new Thread1("CSE",2000);
        t1.start();
        t2.start();
    }
}
```

o/p:-

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
```

29/11/24

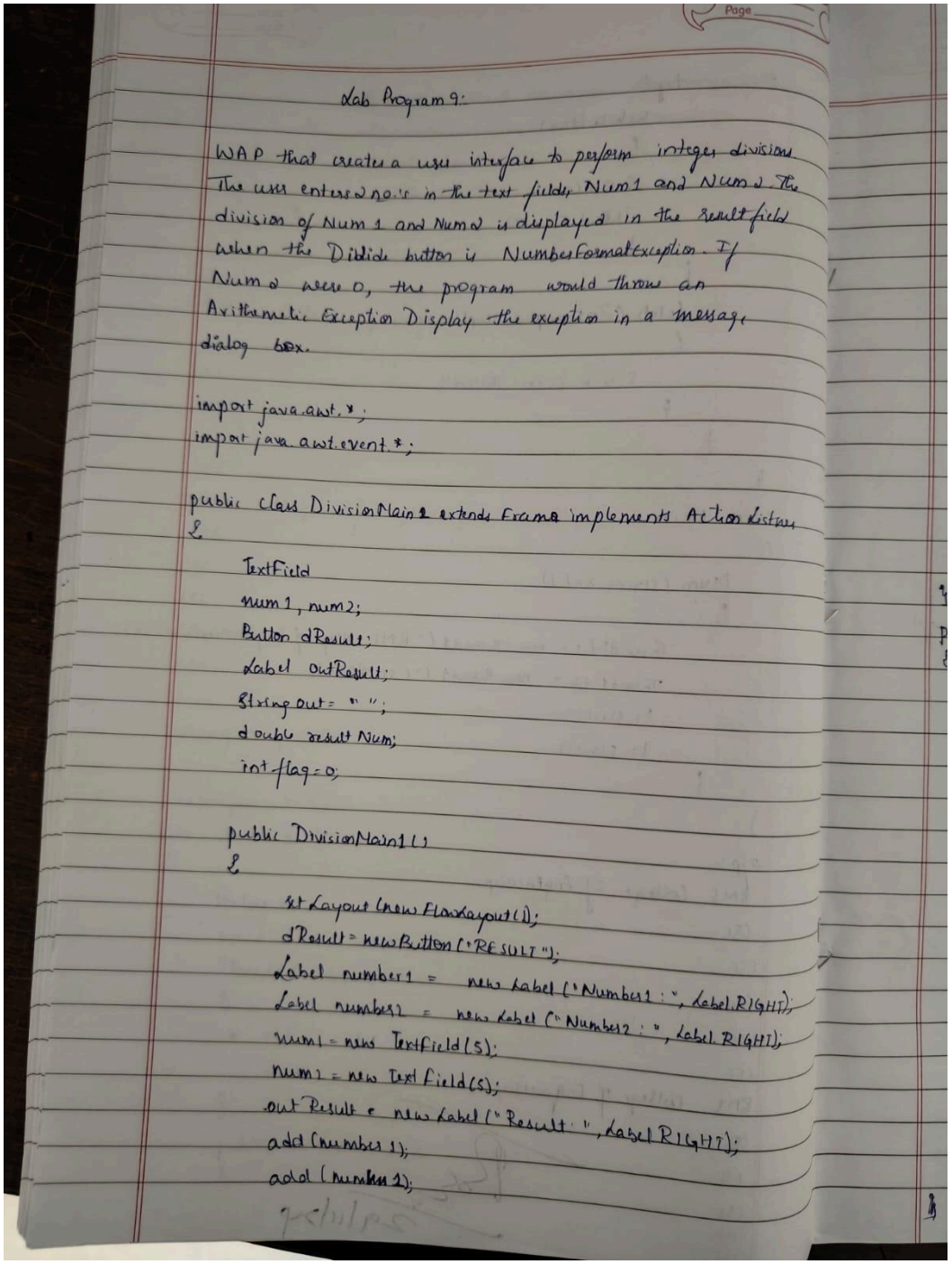
```
class BMSCollegeThread extends Thread {
    public void run() {
        while (true) {
            try {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000);
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

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();
    }
```

}

```
D:\Java_Lab_Programs>javac ThreadDemo.java

D:\Java_Lab_Programs>java ThreadDemo
BMS college of engineering
CSE
CSE
CSE
CSE
CSE
BMS college of engineering
CSE
CSE
CSE
CSE
CSE
BMS college of engineering
CSE
CSE
CSE
CSE
```

LABORATORY PROGRAM - 9

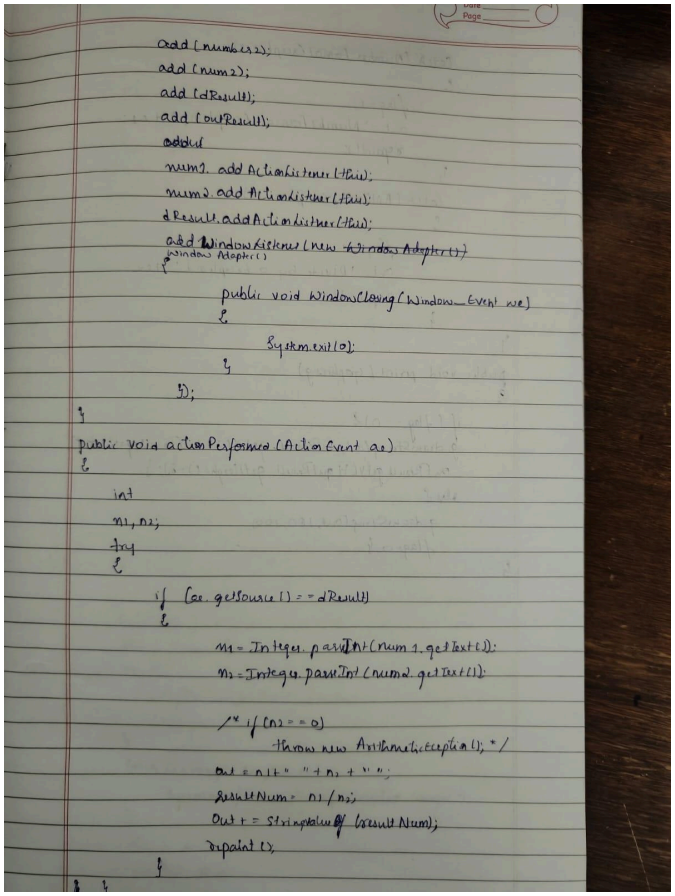
9. Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw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



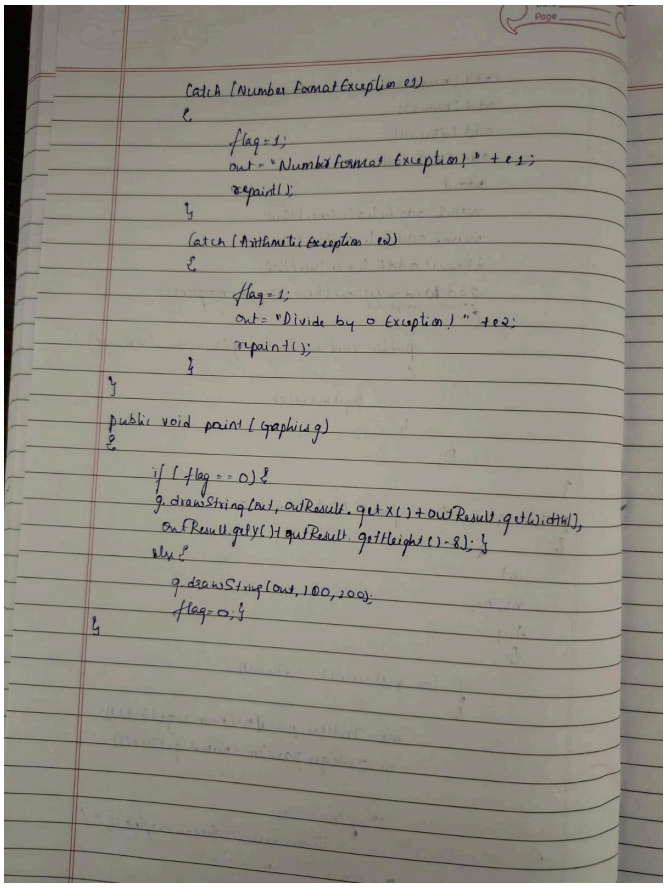
Lab Program 9:

WAP that creates a user interface to perform integer divisions The user enters two no.'s in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is NumberFormatException. If Num2 were 0, the program would throw an Arithmetic Exception Display the exception in a message dialog box.

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

public class DivisionMain2 extends Frame implements ActionListener
{
    TextField
    num1, num2;
    Button dResult;
    Label outResult;
    String out = " ";
    double result Num;
    int flag = 0;

    public DivisionMain1()
    {
        setLayout(new FlowLayout());
        dResult = new Button("RESULT");
        Label number1 = new Label("Number1 :", Label.RIGHT);
        Label number2 = new Label("Number2 :", Label.RIGHT);
        num1 = new TextField(5);
        num2 = new TextField(5);
        outResult = new Label("Result :", Label.RIGHT);
        add(number1);
        add(number2);
```



```
add(numbers2);
add(num2);
add(dResult);
add(outResult);
addWi
num2.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();
        }
```

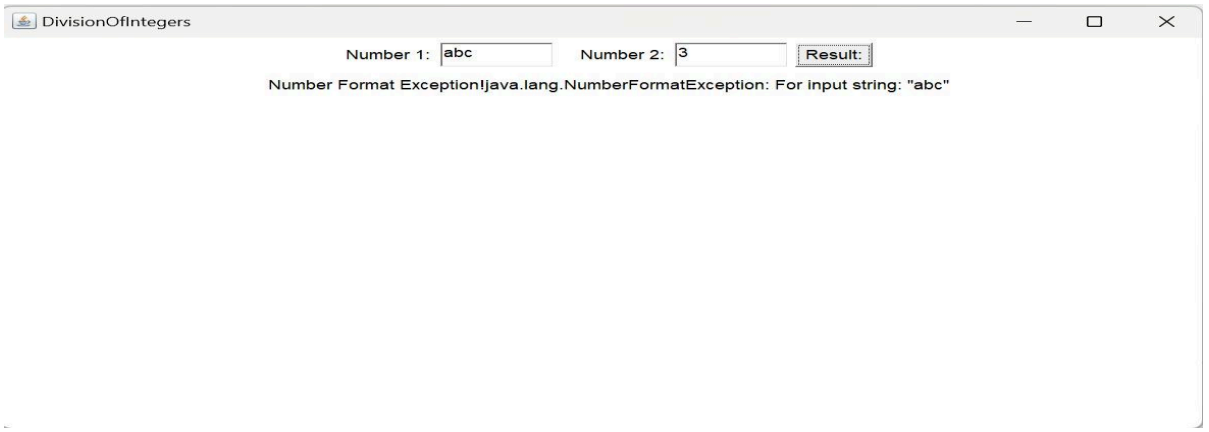


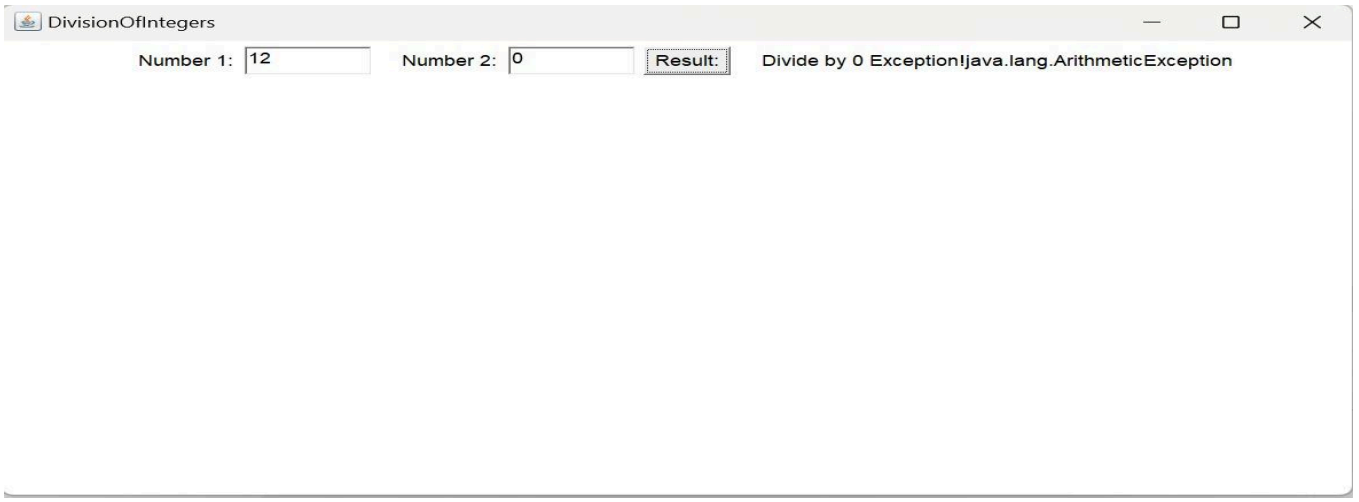
```
Catch(NumberFormatException e1)
{
    flag=1;
    out="NumberFormatException"+e1;
    repaint();
}
Catch(ArithmeticException e2)
{
    flag=1;
    out="Divide by 0 Exception"+e2;
    repaint();
}
}
public void paint(Graphics g)
{
    if(flag==0){
    g.drawString(out,outResult.getX()+outResult.getWidth(),
    outResult.getY()+outResult.getHeight()-8);}
    else{
    g.drawString(out,100,200);
    flag=0;}
}
```

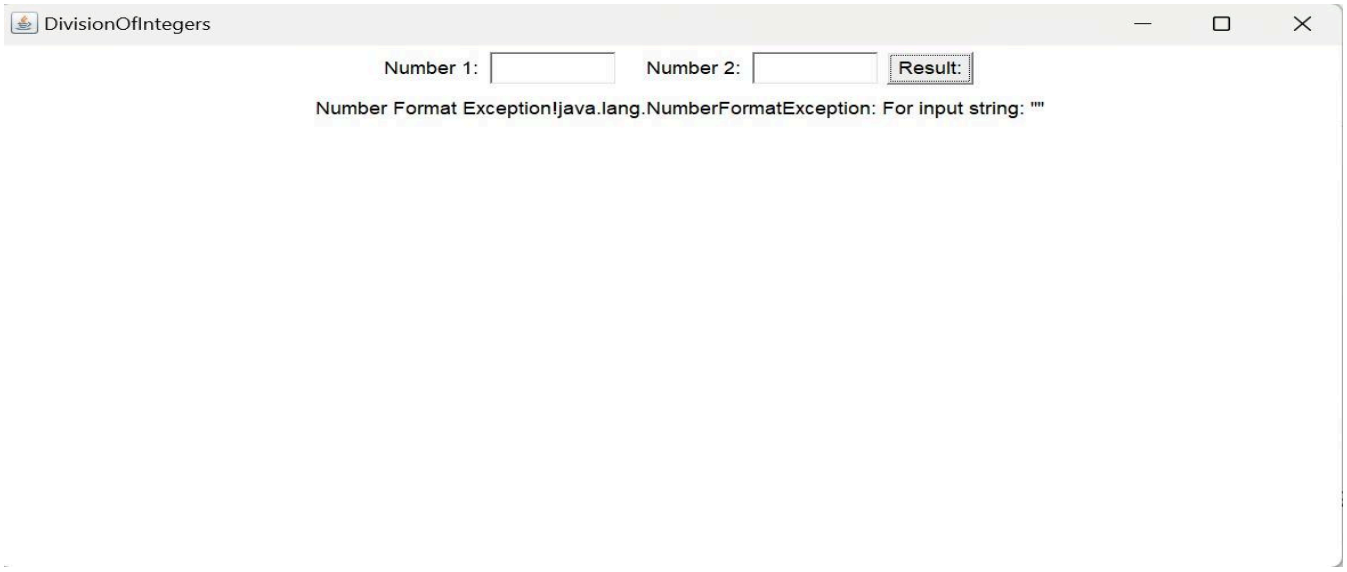
```
import java.awt.*;
import
java.awt.event.*;

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
```

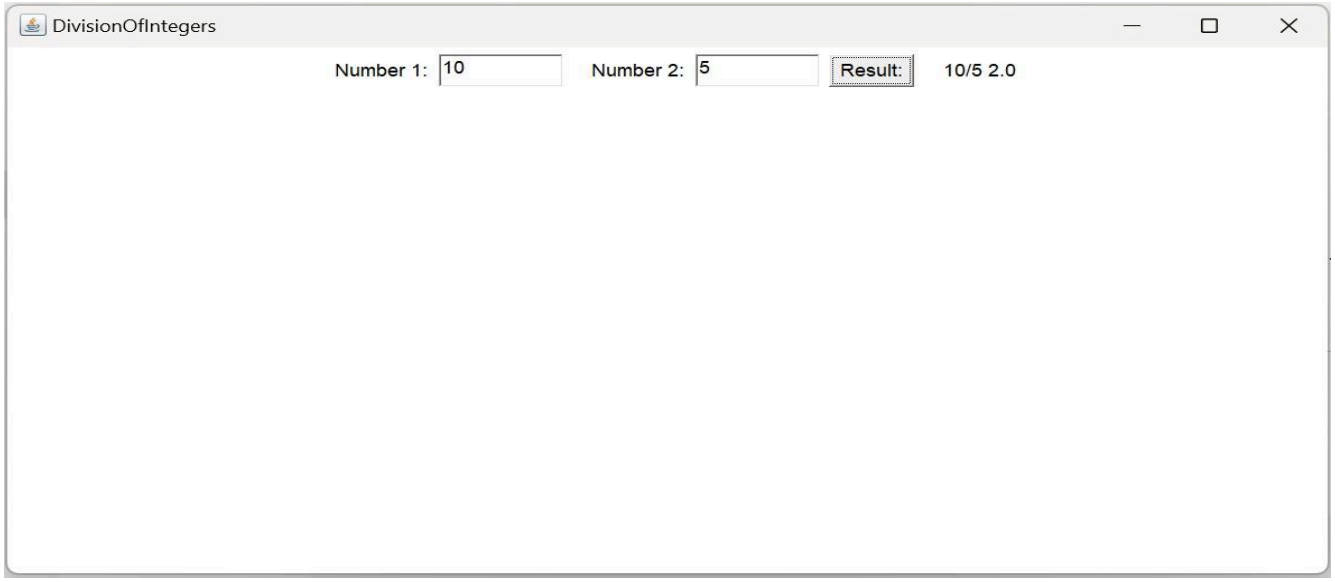
```
        );

        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(resultN
                        um); 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;
}
```



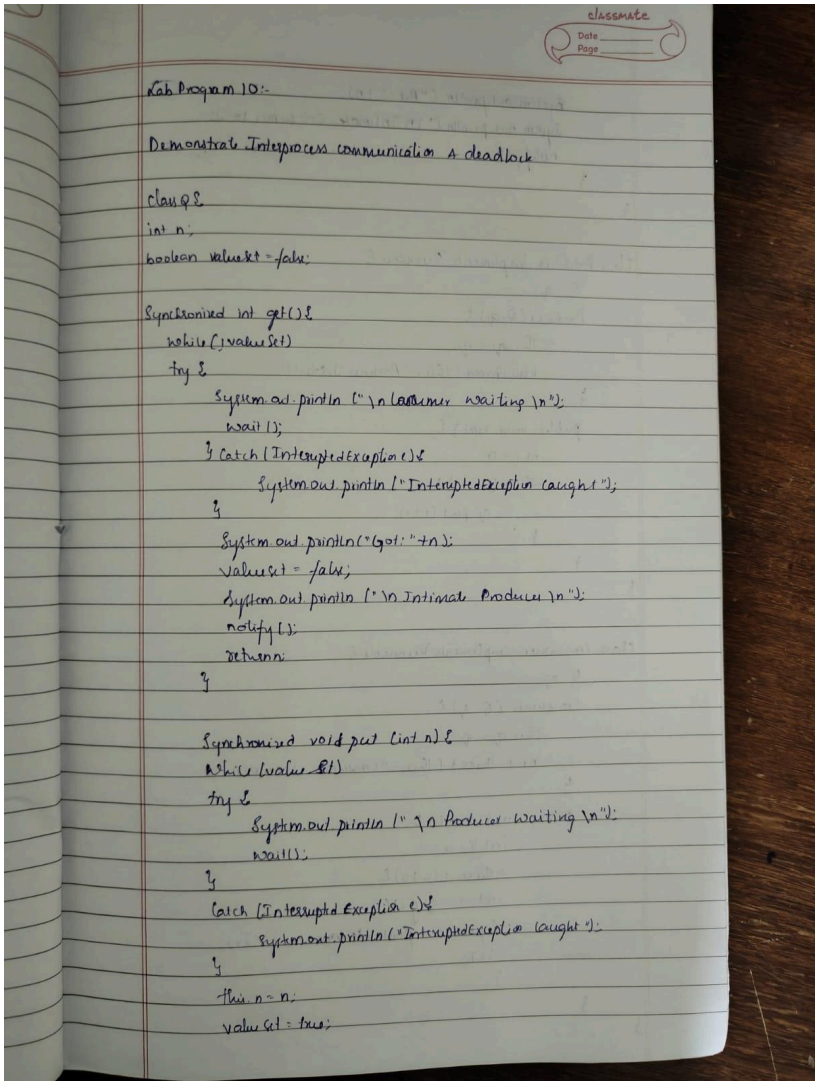

DivisionOfIntegers
Number 1: 12
Number 2: 0
Result:
Divide by 0 Exception!java.lang.ArithmeticException


DivisionOfIntegers
Number 1:
Number 2:
Result:
Number Format Exception!java.lang.NumberFormatException: For input string: ""

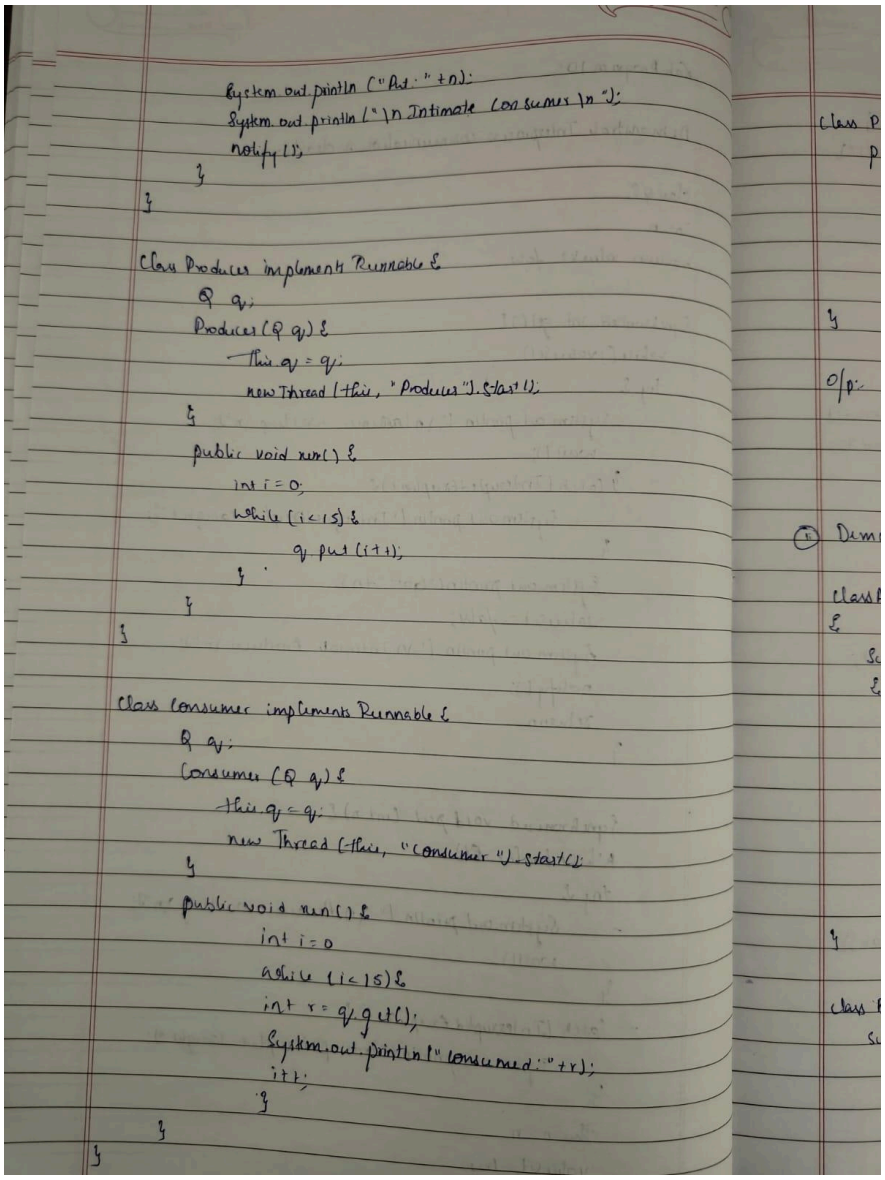

DivisionOfIntegers
Number 1: 10
Number 2: 5
Result:
10/5 2.0

LABORATORY PROGRAM - 10

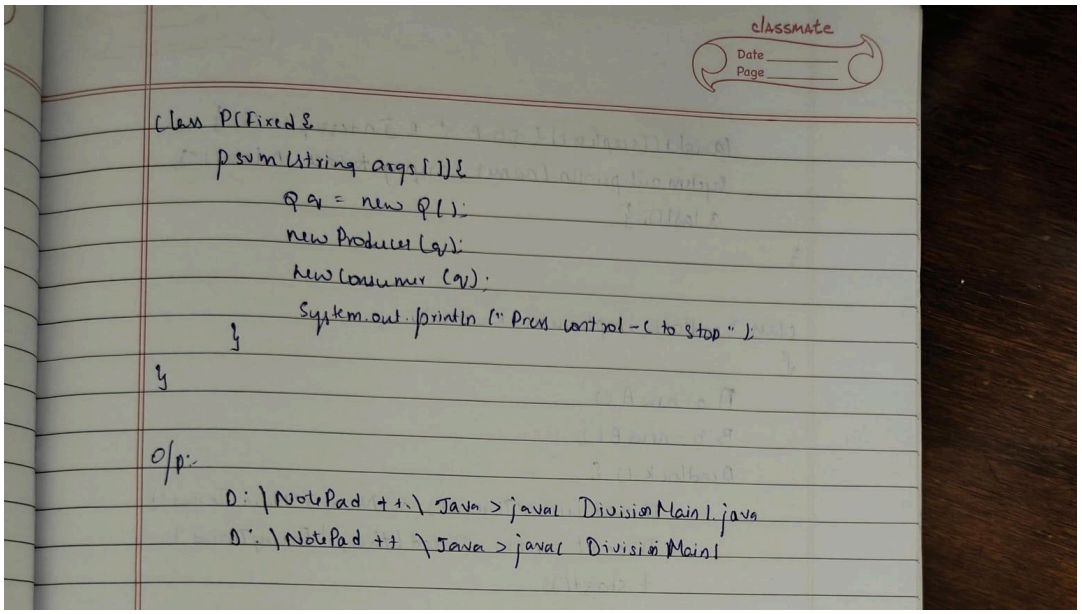
10. Demonstrate Interprocess communication and deadlock.



```
Lab Program 10:-

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();
  }
  catch (InterruptedException e) {
      System.out.println("InterruptedException caught");
  }
  this.n = n;
  valueSet = true;
```



```
        System.out.println("Put: "+n);
        System.out.println("\n Intimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0;
        while(i<15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i=0
        while(i<15) {
        int r = q.get();
        System.out.println("Consumed: "+r);
        i++;
        }
    }
}
```



```
class PCFixed {
    psvm(String args[]) {
        Q q = new Q();
        new Producer(q);
        New Consumer(q);
        System.out.println("Press control-C to stop");
    }
}
```

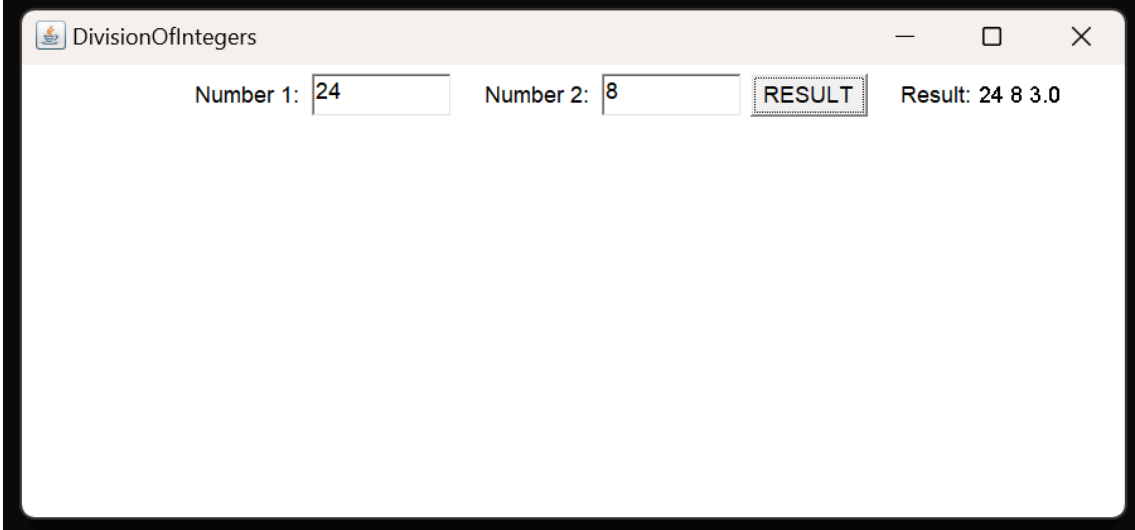
O/p:

```
D:\NotePad ++\Java> javac DivisionMain1.java
D:\NotePad ++\Java> java DivisionMain1
```

```
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.");
}
}
```

```
D:\NotePad++\Java>javac DivisionMain1.java

D:\NotePad++\Java>java DivisionMain1
```



```
D:\1BM23CS330>java PCFixed
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer


Producer waiting

consumed:0
Got: 1

Intimate Producer

consumed:1
Put: 2

Intimate Consumer


Producer waiting
```

```
Got: 2

Intimate Producer

consumed:2
Put: 3

Intimate Consumer


Producer waiting

Got: 3

Intimate Producer

consumed:3
Put: 4

Intimate Consumer


Producer waiting

Got: 4

Intimate Producer

consumed:4
Put: 5

Intimate Consumer
```

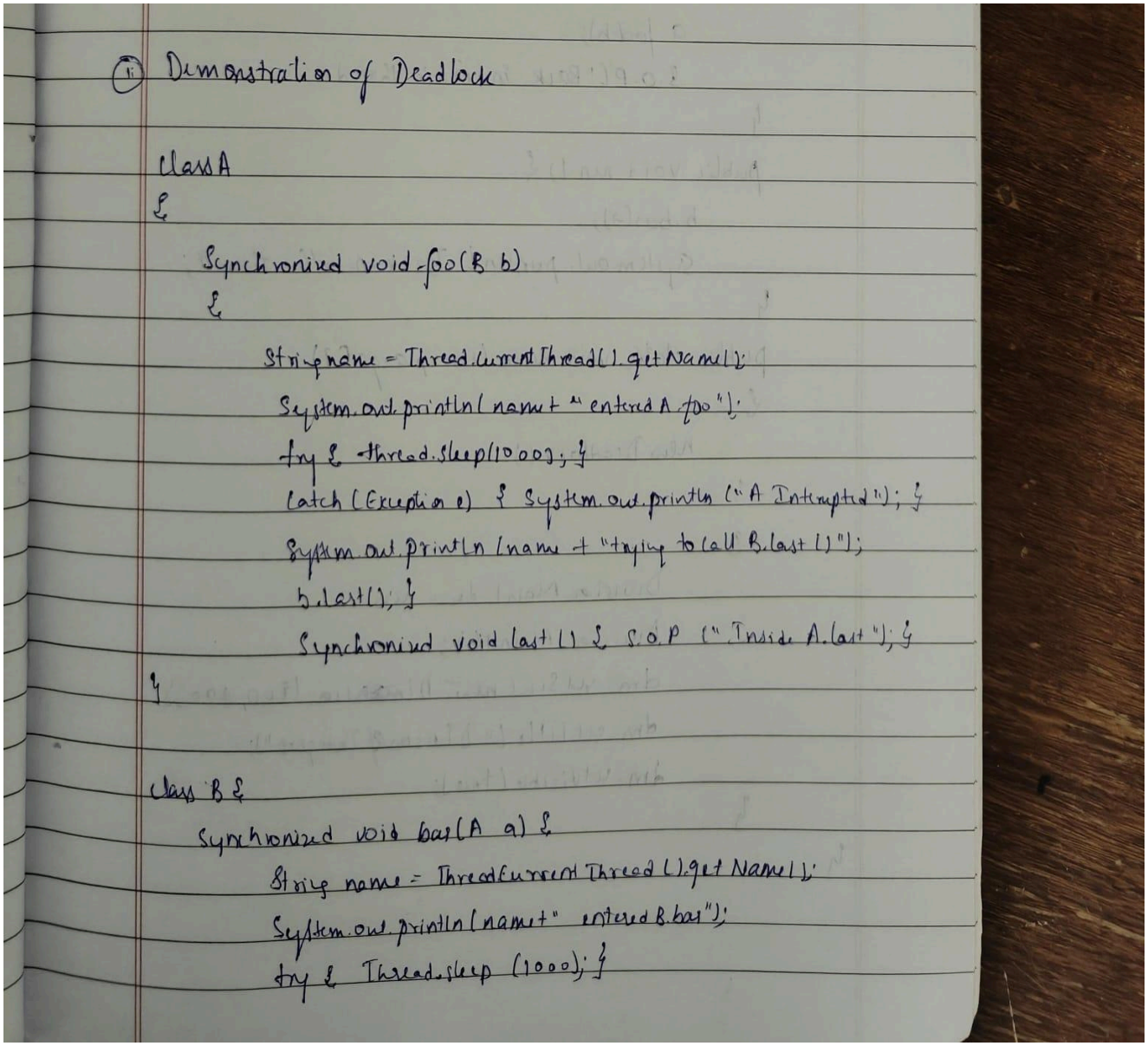
```
Producer waiting

Got: 5

Intimate Producer

consumed:5
Put: 6

Intimate Consumer


Producer waiting

Got: 6

Intimate Producer

consumed:6
Put: 7

Intimate Consumer


Producer waiting

Got: 7

Intimate Producer

consumed:7
Put: 8
```

```
Intimate Consumer


Producer waiting

Got: 8

Intimate Producer

consumed:8
Put: 9

Intimate Consumer


Producer waiting

Got: 9

Intimate Producer

consumed:9
Put: 10

Intimate Consumer


Producer waiting

Got: 10

Intimate Producer
```

```
consumed:10
Put: 11

Intimate Consumer


Producer waiting

Got: 11

Intimate Producer

consumed:11
Put: 12

Intimate Consumer


Producer waiting

Got: 12

Intimate Producer

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:\1BM23CS330>
```

10.b Demonstration of deadlock



① 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() { S.O.P ("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); }
```

```
        Caught(Exception e){ S.O.P ("B Interrupted "); }
        System.out.println(name+ "trying to call A.last()");
        a.last(); }
}

class Deadlock implements Runnable
{
    A a=new A();
    B b=new B();
    Deadlock() {
        Thread.currentThread().setName("Main Thread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        S.O.P("Back in main thread");
    }
    public void run() {
        b.bar(a);
        System.out.println("Back in other thread");
    }
    public static void main(String args[])
    {
        new Deadlock();
    }
    {
        DivisionMain1 dm = new
        DivisionMain1();
        dm.setSize(new Dimension(800, 400));
        dm.setTitle("Divison Of Integers");
        dm.setVisible(true);
    }
}
```

```
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 A.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(); }
        {
                DivisionMain1 dm=new
                DivisionMain1(); dm.setSize(new
                Dimension(800,400));
                dm.setTitle("DivisionOfIntegers");
                dm.setVisible(true);
        }
}
```

```
D:\1BM23CS330>javac Deadlock.java

D:\1BM23CS330>java Deadlock
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last()
MainThread trying to call B.last()
```