VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SIDDHARTH ARYA (1BM23CS328)

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 **SIDDHARTH ARYA (1BM23CS328),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum 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

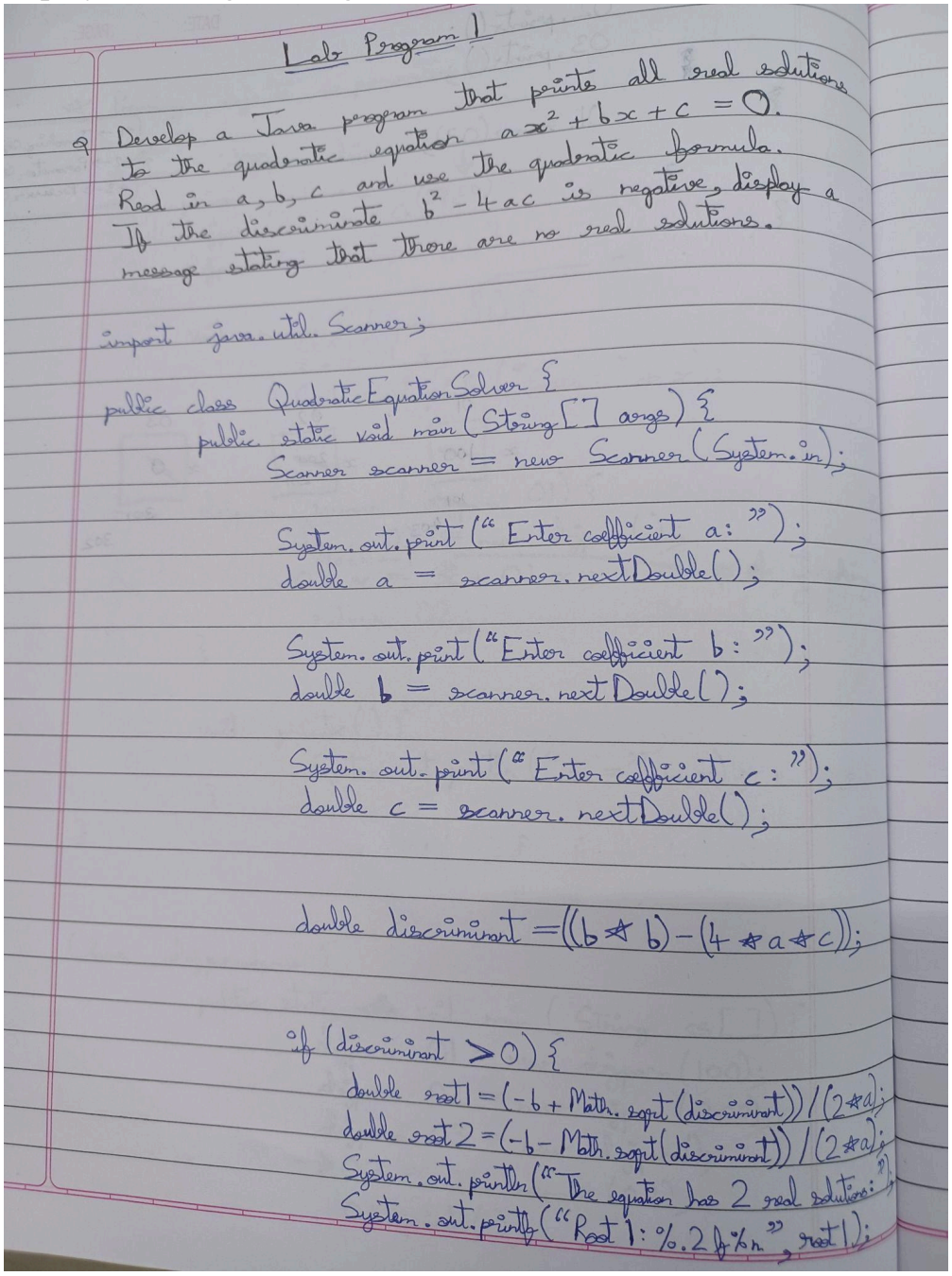
Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

LABORATORY PROGRAM - 1

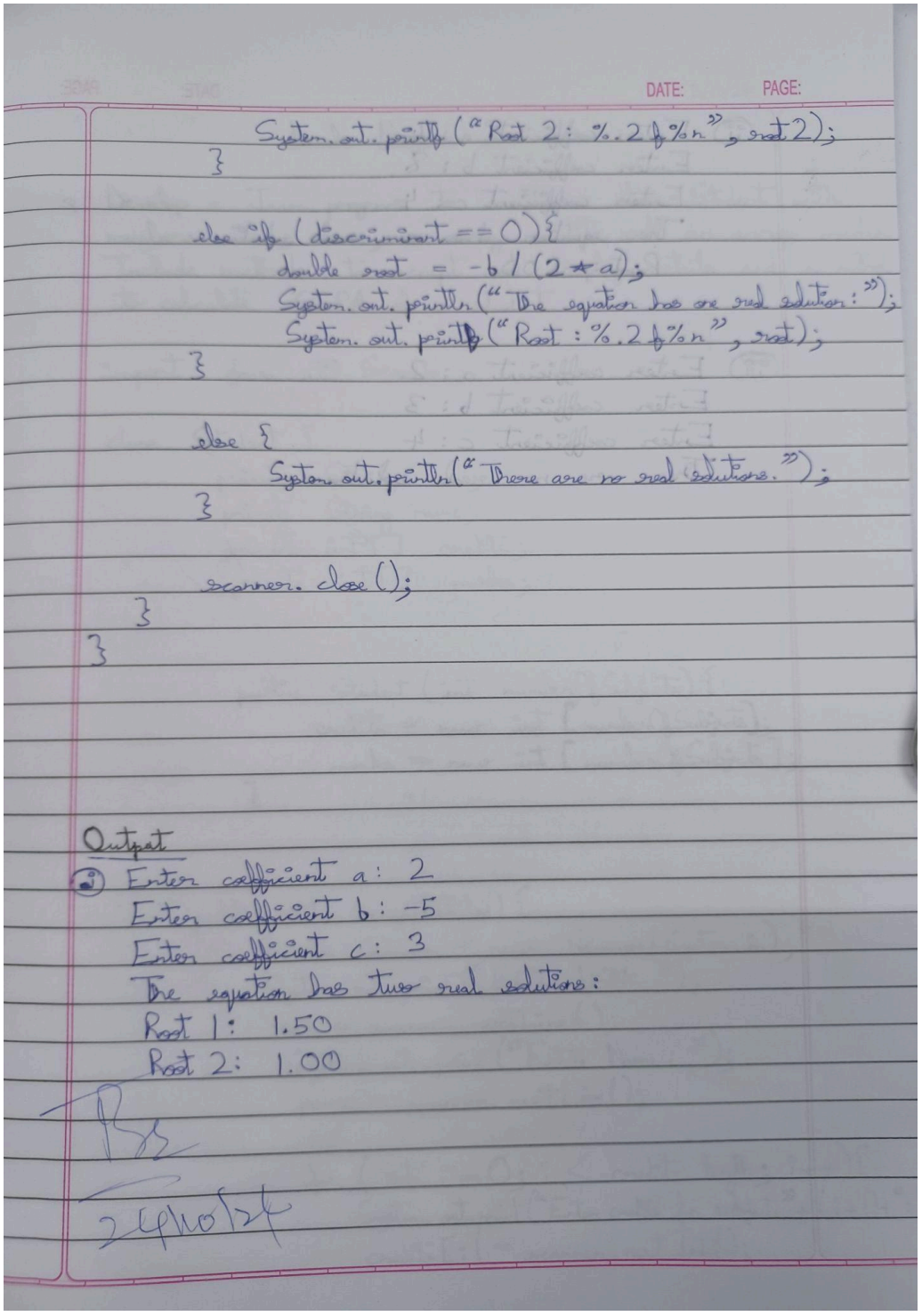
Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c = 0$. Read in a, b, c and use the quadratic formula. If the discriminant b^2-4ac is negative, display a message stating that there are no real solutions.



Lab Program 1

Q 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 discriminate b^2-4ac is negative, display a message stating that there are no real solutions.

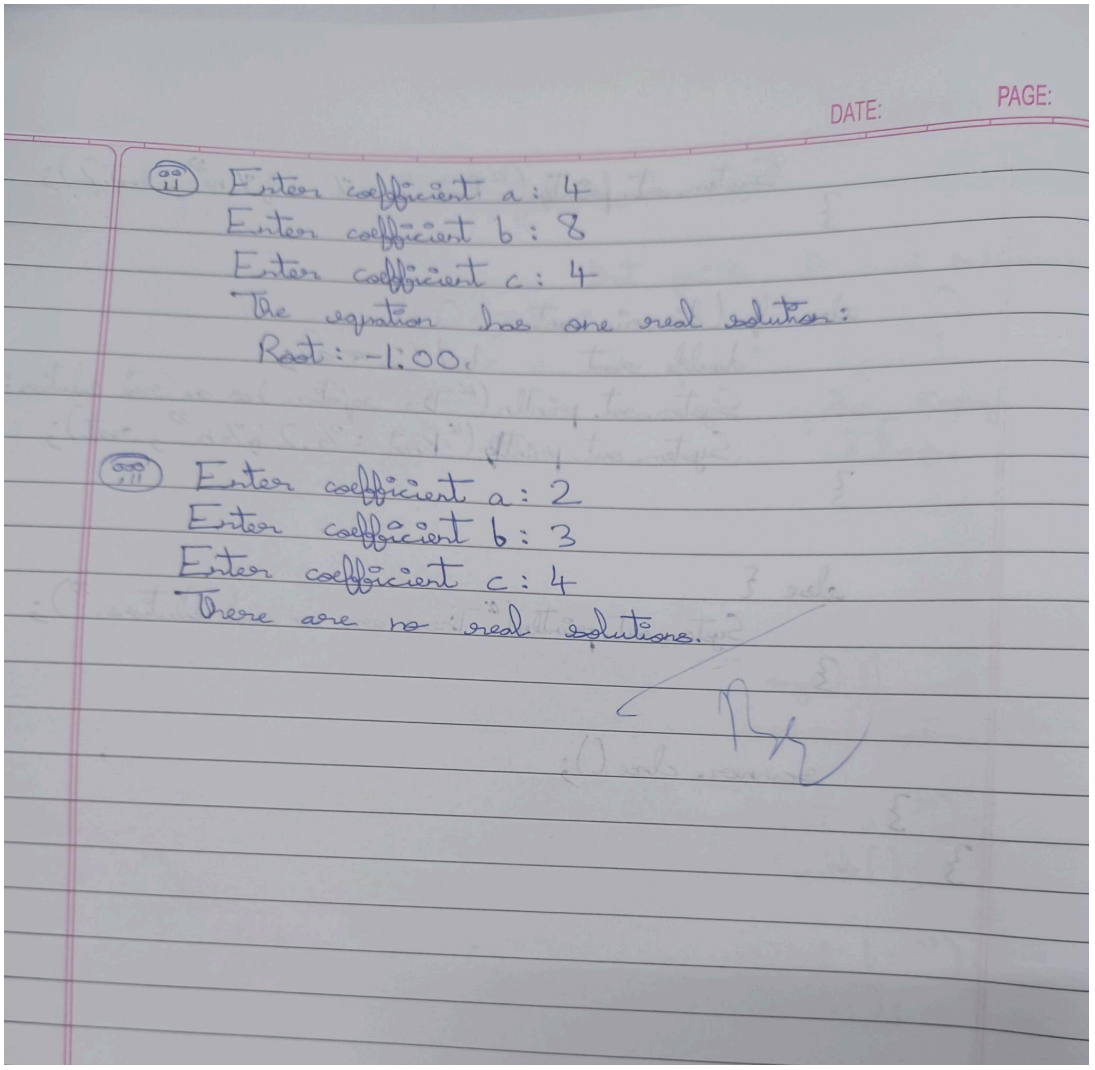
```
import java.util.Scanner;

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

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

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

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

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

        if (discriminant > 0) {
            double root1 = (-b + Math.sqrt(discriminant)) / (2*a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2*a);
            System.out.println("The equation has 2 real solutions: ");
            System.out.printf("Root 1: %.2f %n", root1);
```



```
        System.out.printf("Root 2: %.2f%n", root2);
    }

    else if (discriminant == 0) {
        double root = -b / (2 * a);
        System.out.println("The equation has one real solution:");
        System.out.printf("Root: %.2f%n", root);
    }

    else {
        System.out.println("There are no real solutions.");
    }

    scanner.close();
  }
}
```

Output

Enter coefficient a: 2
Enter coefficient b: -5
Enter coefficient c: 3
The equation has two real solutions:
Root 1: 1.50
Root 2: 1.00



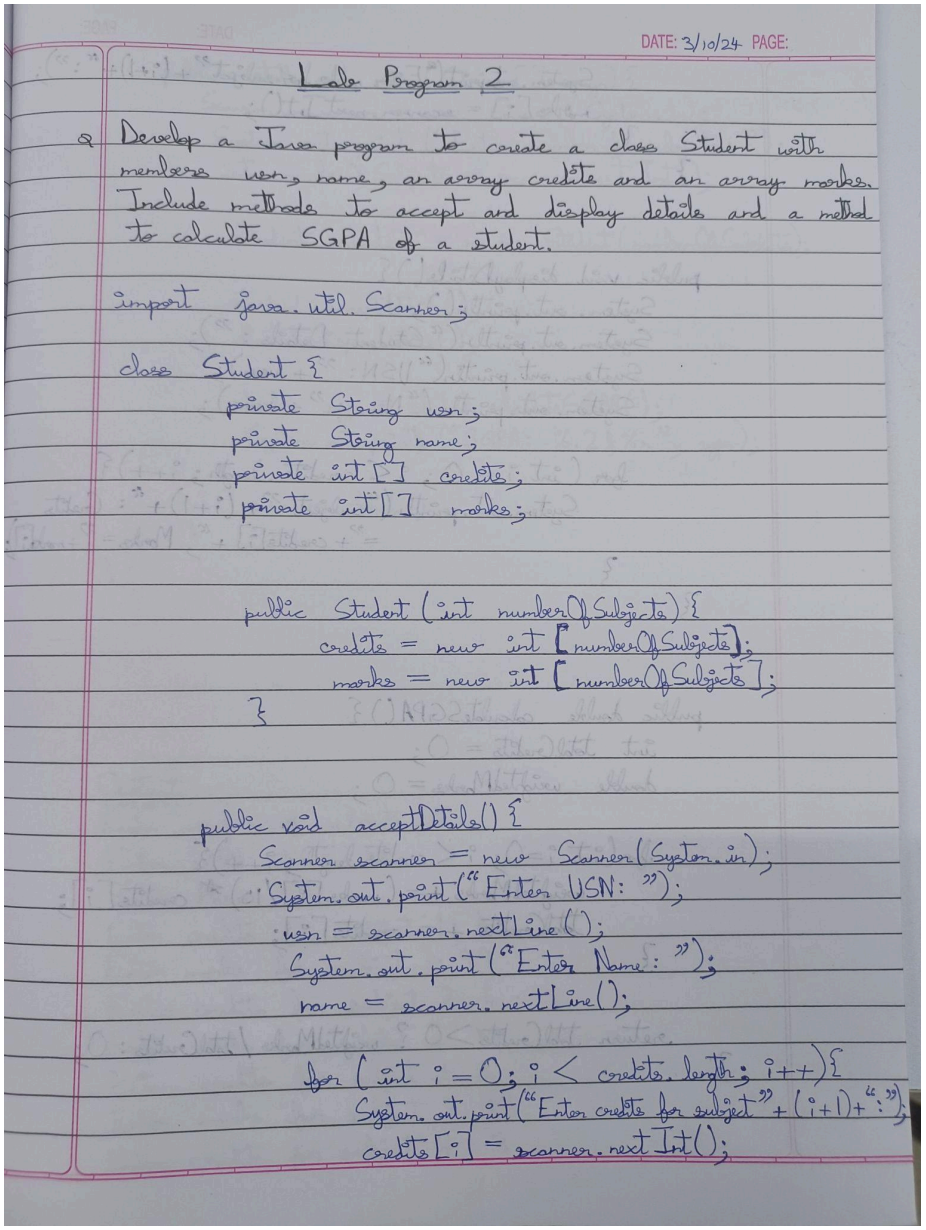

```
// Program to solve Quadratic Equation


import java.util.Scanner;


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

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

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

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


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


      if (discriminant > 0){
         double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
```

```
        double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);

        System.out.println("The equation has 2 real solutions:");
        System.out.printf("Root 1: %.2f%n", root1);
        System.out.printf("Root 2: %.2f%n", root2);
    }


    else if (discriminant == 0){
        double root = -b / (2 * a);
        System.out.println("The equation has one real solution:");
        System.out.printf("Root: %.2f%n", root);
    }

    else{
        System.out.println("There are no real solutions.");
    }

    scanner.close();
  }
}
```

```
PS D:\1BM23CS328\Week 1> javac QuadraticEquationSolver.java
PS D:\1BM23CS328\Week 1> java QuadraticEquationSolver
Enter coefficient a: 2
Enter coefficient b: -5
Enter coefficient c: 3
The equation has 2 real solutions:
Root 1: 1.50
Root 2: 1.00
PS D:\1BM23CS328\Week 1> java QuadraticEquationSolver
Enter coefficient a: 4
Enter coefficient b: 8
Enter coefficient c: 4
The equation has one real solution:
Root: -1.00
PS D:\1BM23CS328\Week 1> java QuadraticEquationSolver
Enter coefficient a:
2
Enter coefficient b: 3
Enter coefficient c: 4
There are no real solutions.
```

LABORATORY 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.

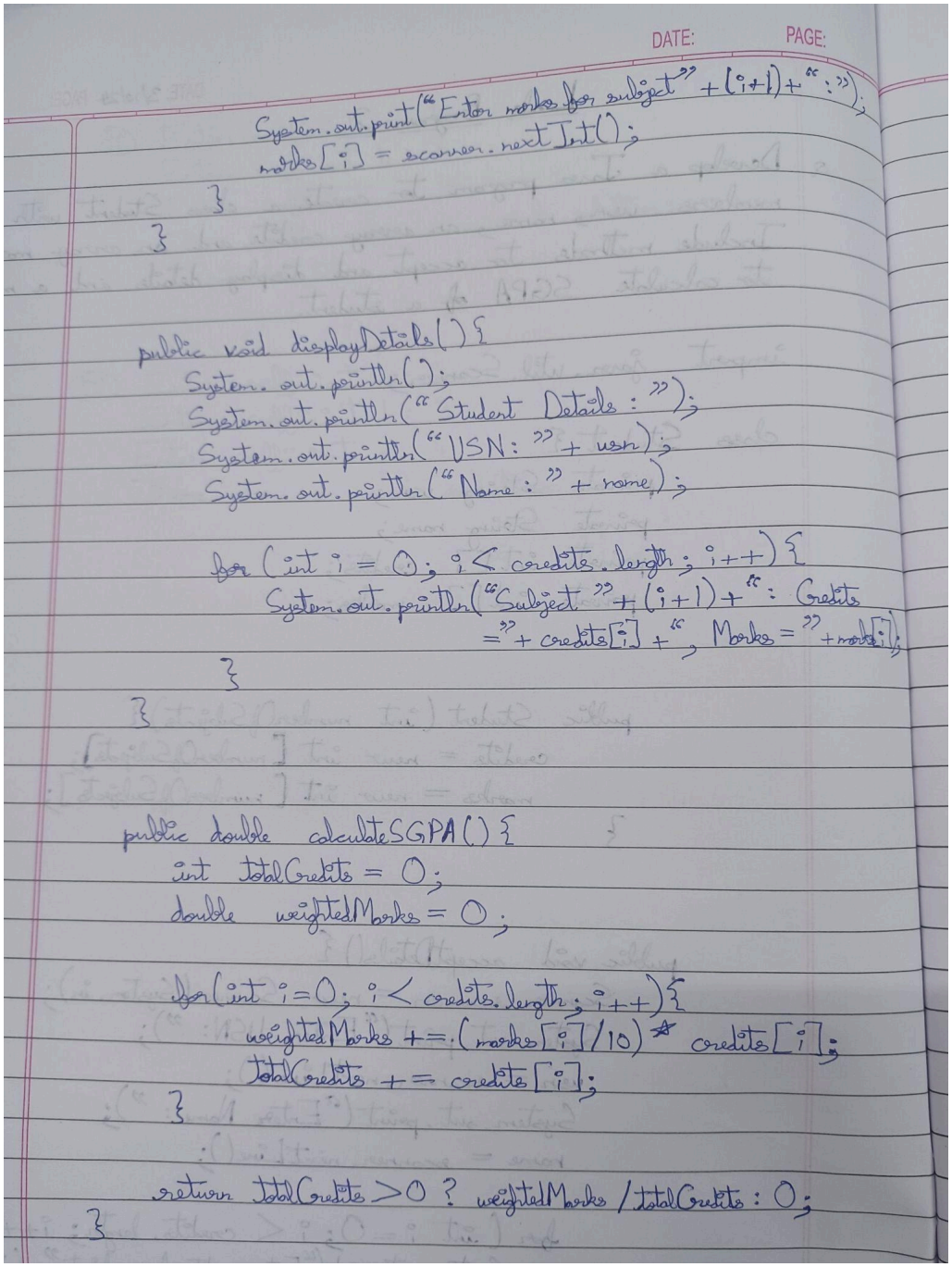


DATE: 3/10/24 PAGE:

Lab Program 2

Q 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 Student {
    private String usn;
    private String name;
    private int[] credits;
    private int[] marks;

    public Student(int numberOfSubjects) {
        credits = new int[numberOfSubjects];
        marks = new int[numberOfSubjects];
    }

public void acceptDetails() {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter USN: ");
    usn = scanner.nextLine();
    System.out.print("Enter Name: ");
    name = scanner.nextLine();

    for (int i = 0; i < credits.length; i++){
        System.out.print("Enter credits for subject" + (i+1) + ":");
        credits[i] = scanner.nextInt();
```



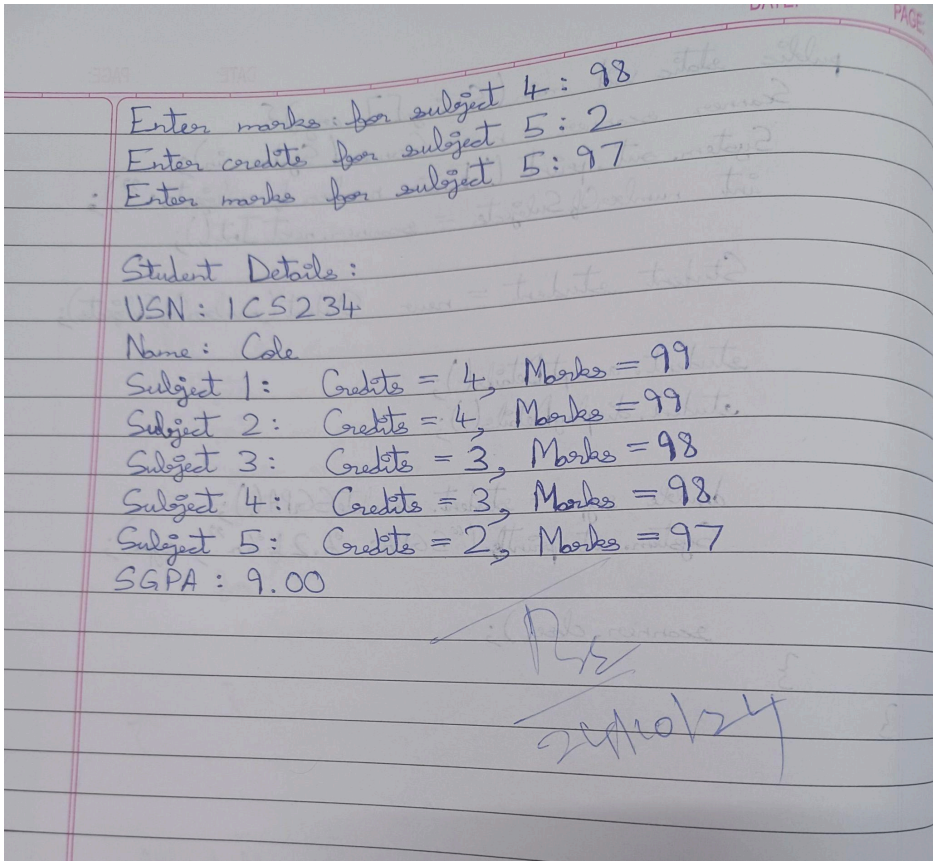
```
DATE:                PAGE:

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

public void displayDetails() {
    System.out.println();
    System.out.println("Student Details: ");
    System.out.println("USN: " + usn);
    System.out.println("Name: " + name);

    for (int i = 0; i < credits.length; i++) {
        System.out.println("Subject " + (i+1) + ": Credits
                = " + credits[i] + ", Marks = " + marks[i]);
    }
}

public double calculateSGPA() {
    int totalCredits = 0;
    double weightedMarks = 0;

    for(int i=0; i < credits.length; i++){
        weightedMarks += (marks[i]/10) * credits[i];
        totalCredits += credits[i];
    }

    return totalCredits > 0 ? weightedMarks / totalCredits : 0;
}
```

DATE: PAGE:

```
public static void main (String[] args) {
    Scanner scanner = new Scanner (System.in);
    System.out.print ("Enter number of subjects: ");
    int numberOfSubjects = scanner.nextInt();

    Student student = new Student (numberOfSubjects);

    student.acceptDetails();
    student.displayDetails();

    double sgpa = student.calculateSGPA();
    System.out.printf ("SGPA: %.2f%n", sgpa);

    scanner.close();
  }
}
```

Output:

```
Enter number of subjects: 5
Enter USN: 1CS234
Enter Name: Cole
Enter credits for subject 1: 4
Enter marks for subject 1: 99
Enter credits for subject 2: 4
Enter marks for subject 2: 99
Enter credits for subject 3: 3
Enter marks for subject 3: 98
Enter credits for subject 4: 3
```



// Program to calculate SGPA of a Student

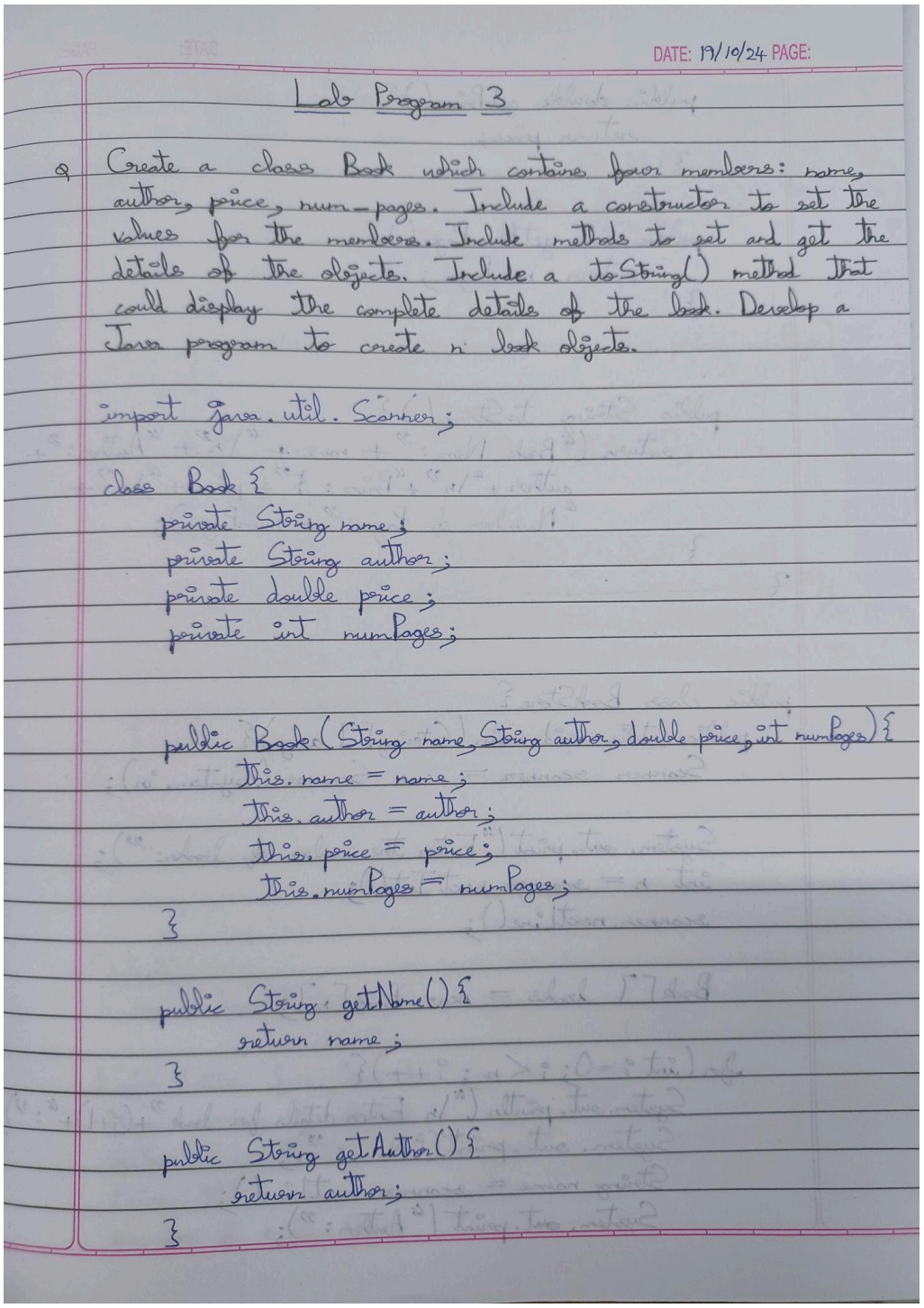
```
import java.util.Scanner;

class Student{
    private String usn;
    private String name;
    private int[] credits;
    private int[] marks;

    public Student(int numberOfSubjects){
        credits = new int[numberOfSubjects];
        marks = new int[numberOfSubjects];
    }

    public void acceptDetails(){
        Scanner scanner = new Scanner(System.in);

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

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

        for (int i = 0; i < credits.length; i++){
```

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

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


    public void displayDetails(){
        System.out.println();
        System.out.println("Student Details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);

        for(int i = 0; i < credits.length; i++){
            System.out.println("Subject " + (i + 1) + ": Credits = " + credits[i] + ", Marks = " + marks[i]);
        }
    }


    public double calculateSGPA(){
        int totalCredits = 0;
        double weightedMarks = 0;

        for (int i = 0; i < credits.length; i++){
            weightedMarks += (marks[i] / 10) * credits[i];
            totalCredits += credits[i];
        }

        return totalCredits > 0 ? weightedMarks / totalCredits : 0;
    }



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

        System.out.print("Enter number of subjects: ");
        int numberOfSubjects = scanner.nextInt();

        Student student = new Student(numberOfSubjects);

        student.acceptDetails();
        student.displayDetails();

        double sgpa = student.calculateSGPA();
        System.out.printf("SGPA: %.2f%n", sgpa);

        scanner.close();
    }
}
```

```
PS D:\1BM23CS328\Week 2> javac Student.java
PS D:\1BM23CS328\Week 2> java Student
Enter number of subjects: 5
Enter USN: 1CS234
Enter Name: Cole
Enter credits for subject 1: 4
Enter marks for subject 1: 99
Enter credits for subject 2: 4
Enter marks for subject 2: 99
Enter credits for subject 3: 3
Enter marks for subject 3: 98
Enter credits for subject 4: 3
Enter marks for subject 4: 98
Enter credits for subject 5: 2
Enter marks for subject 5: 97

Student Details:
USN: 1CS234
Name: Cole
Subject 1: Credits = 4, Marks = 99
Subject 2: Credits = 4, Marks = 99
Subject 3: Credits = 3, Marks = 98
Subject 4: Credits = 3, Marks = 98
Subject 5: Credits = 2, Marks = 97
SGPA: 9.00
```

LABORATORY PROGRAM - 3

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

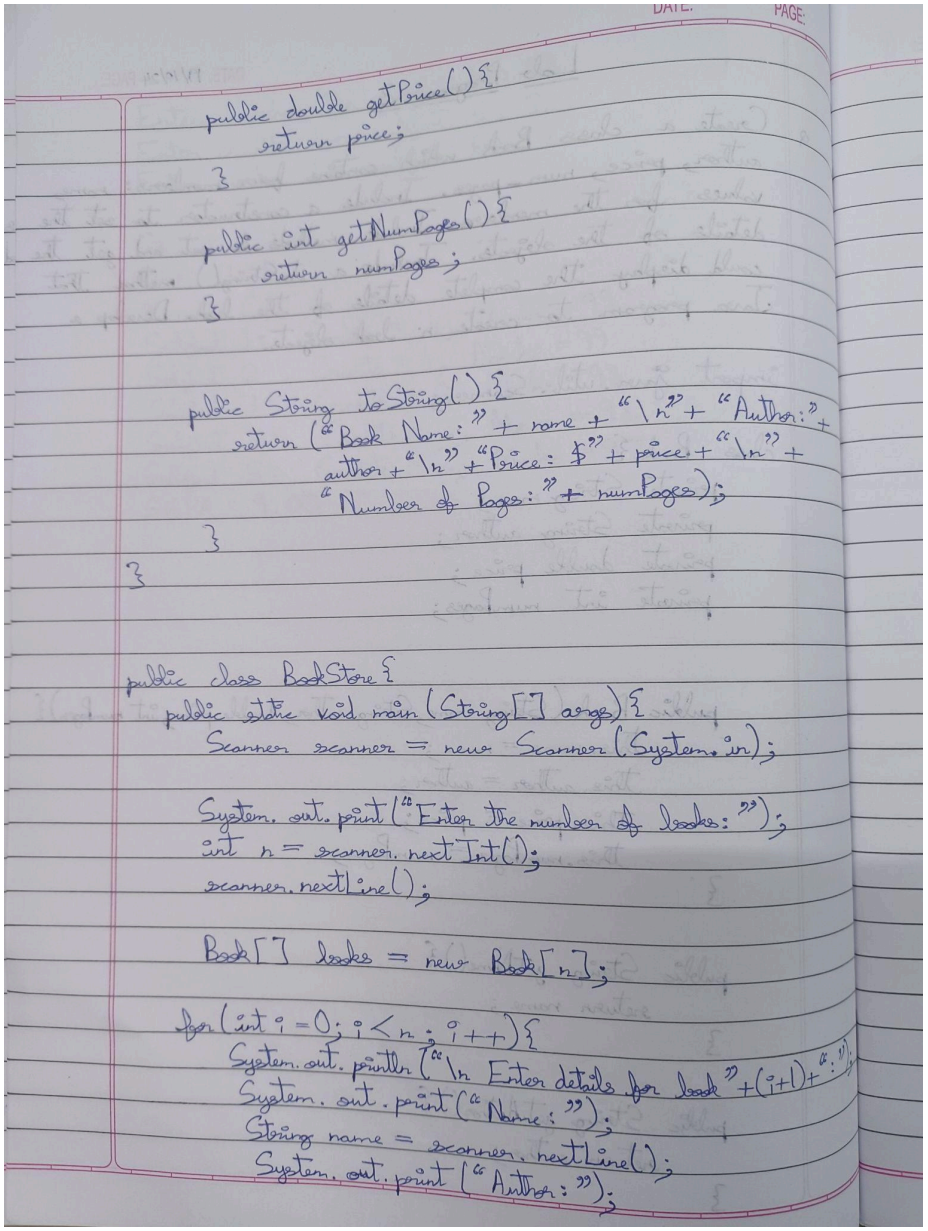


DATE: 19/10/24 PAGE:

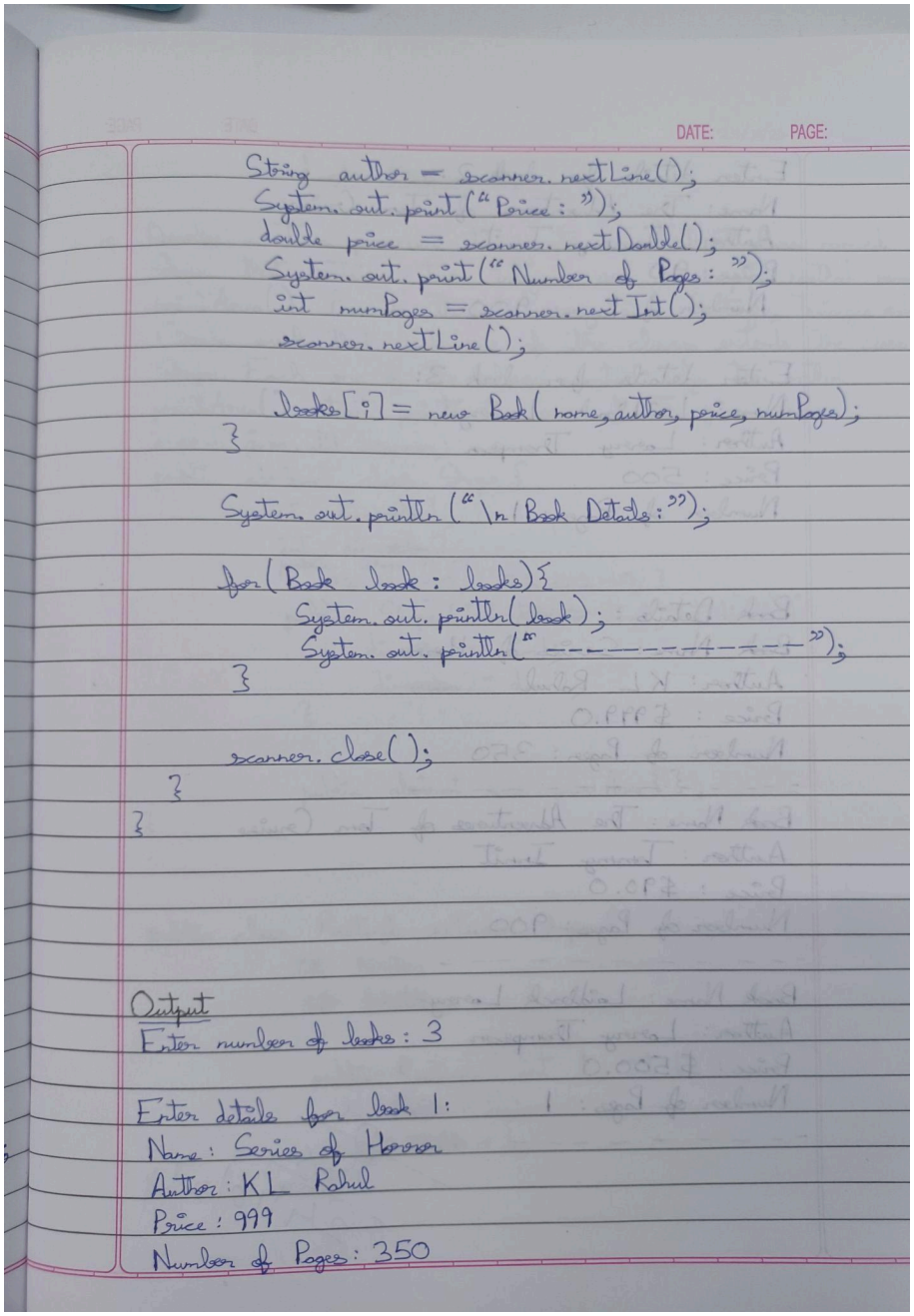
Lab Program 3

Q 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.

```
import java.util.Scanner;

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

    public Book(String name, String author, double price, int numPages){
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;
    }

    public String getName(){
        return name;
    }

    public String getAuthor(){
        return author;
    }
```



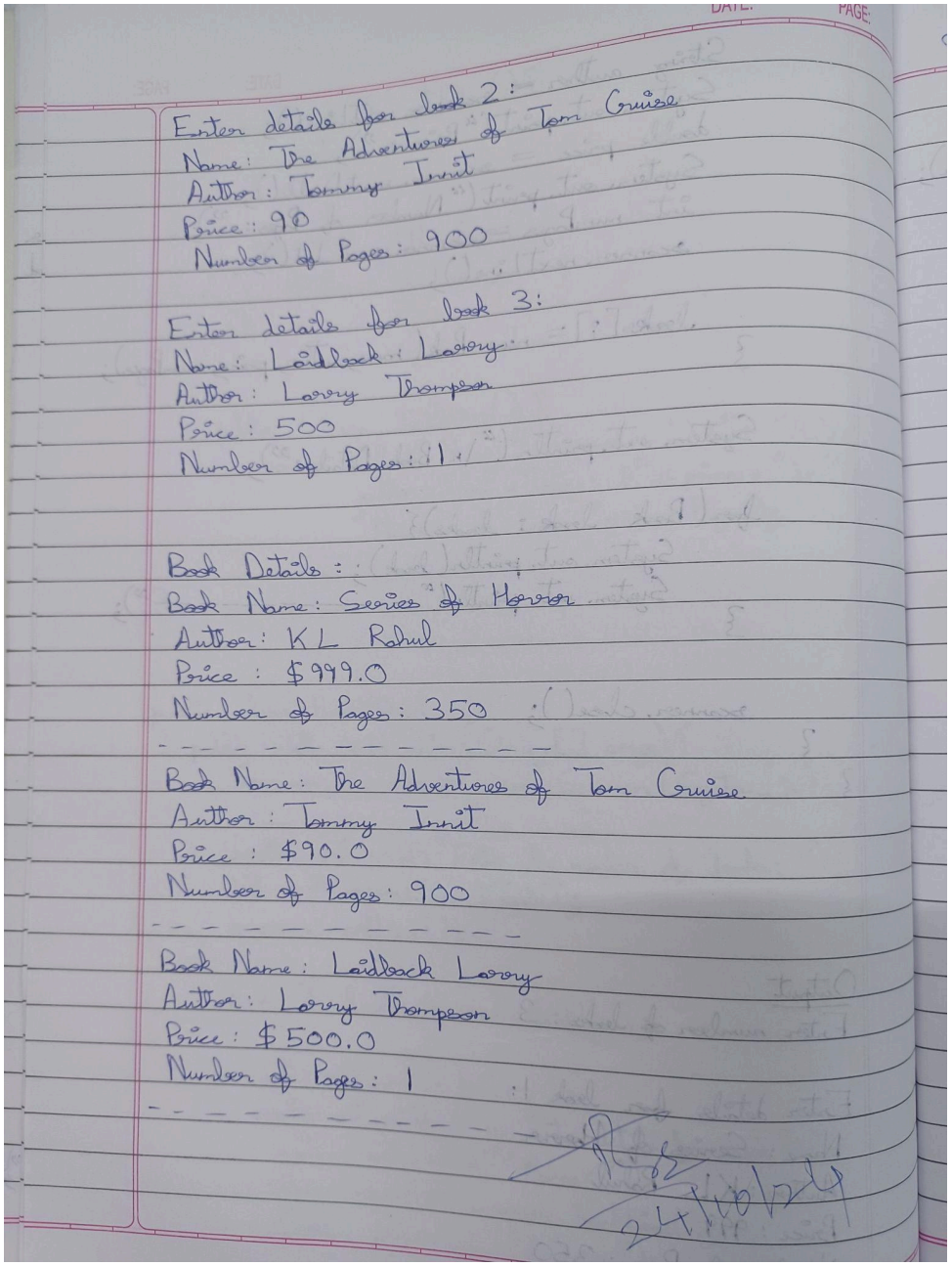
```
public double getPrice(){
    return price;
}

public int getNumPages(){
    return numPages;
}

public String toString(){
    return("Book Name: " + name + "\n" + "Author: " +
    author + "\n" + "Price: $" + price + "\n" +
    "Number of Pages: " + numPages);
}
}

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

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

Book[] books = new Book[n];

for(int i=0; i<n; i++){
    System.out.println("\n Enter details for book " +(i+1)+ ":");
    System.out.print("Name: ");
    String name = scanner.nextLine();
    System.out.print("Author: ");
```



```
        String author = scanner.nextLine();
        System.out.print("Price: ");
        double price = scanner.nextDouble();
        System.out.print("Number of Pages: ");
        int numPages = scanner.nextInt();
        scanner.nextLine();

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

    System.out.println("\n Book Details:");

    for(Book book : books){
        System.out.println(book);
        System.out.println("-----------");
    }

    scanner.close();
  }
}
```

Output

```
Enter number of books: 3

Enter details for book 1:
Name: Series of Horror
Author: KL Rahul
Price: 999
Number of Pages: 350
```



// Program to display details of various books in a BookStore

```
import java.util.Scanner;

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

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

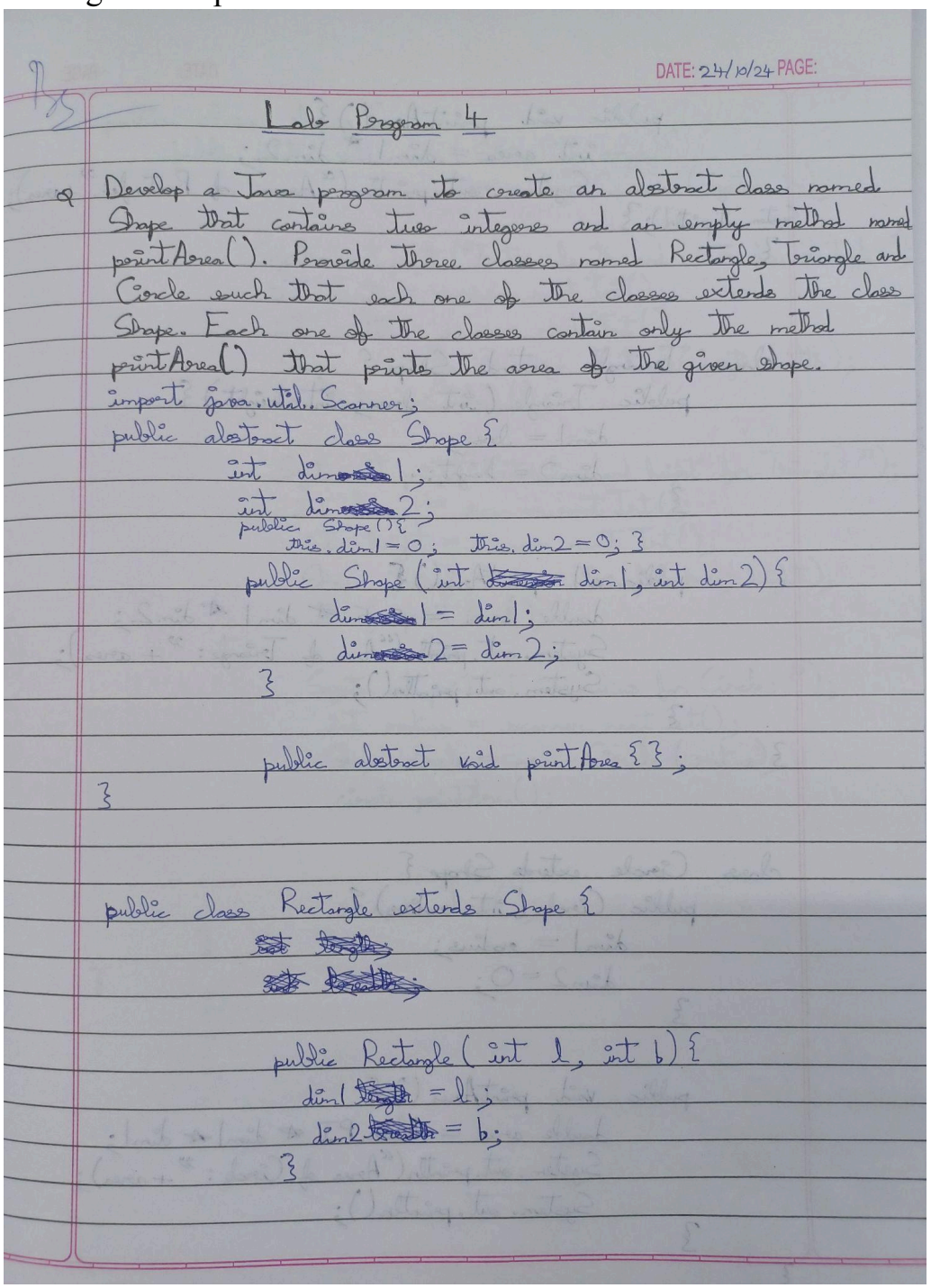
```
        this.numPages = numPages;
    }


    public String getName(){
        return name;
    }

    public String getAuthor(){
        return author;
    }

    public double getPrice(){
        return price;
    }

    public int getNumPages(){
        return numPages;
    }


    public String toString(){
        return ("Book Name: " + name + "\n" + "Author: " + author + "\n" + "Price: $" + price + "\n" + "Number of Pages: "
+ numPages);
    }
}



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

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

        Book[] books = new Book[n];

        for(int i = 0; i < n; i++){
            System.out.println("\nEnter details for book " + (i + 1) + ":");

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

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

            System.out.print("Price: ");
            double price = scanner.nextDouble();

            System.out.print("Number of Pages: ");
            int numPages = scanner.nextInt();

            scanner.nextLine();

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


        System.out.println("\n Book Details: ");
```

```
        for (Book book : books){
            System.out.println(book);
            System.out.println("------------------------------");
        }


        scanner.close();
    }

}
```

```
Enter the number of books: 2

Enter details for book 1:
Name: The Adventures of Tom Cruise
Author: Tommy Innit
Price: 90
Number of Pages: 900

Enter details for book 2:
Name: Laidback Larry
Author: Larry Thompson
Price: 5001
Number of Pages:
1

 Book Details:
Book Name: The Adventures of Tom Cruise
Author: Tommy Innit
Price: $90.0
Number of Pages: 900
------------------------------
Book Name: Laidback Larry
Author: Larry Thompson
Price: $500.0
Number of Pages: 1
------------------------------
```

LABORATORY 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 contains only the method printArea( ) that prints the area of the given shape.

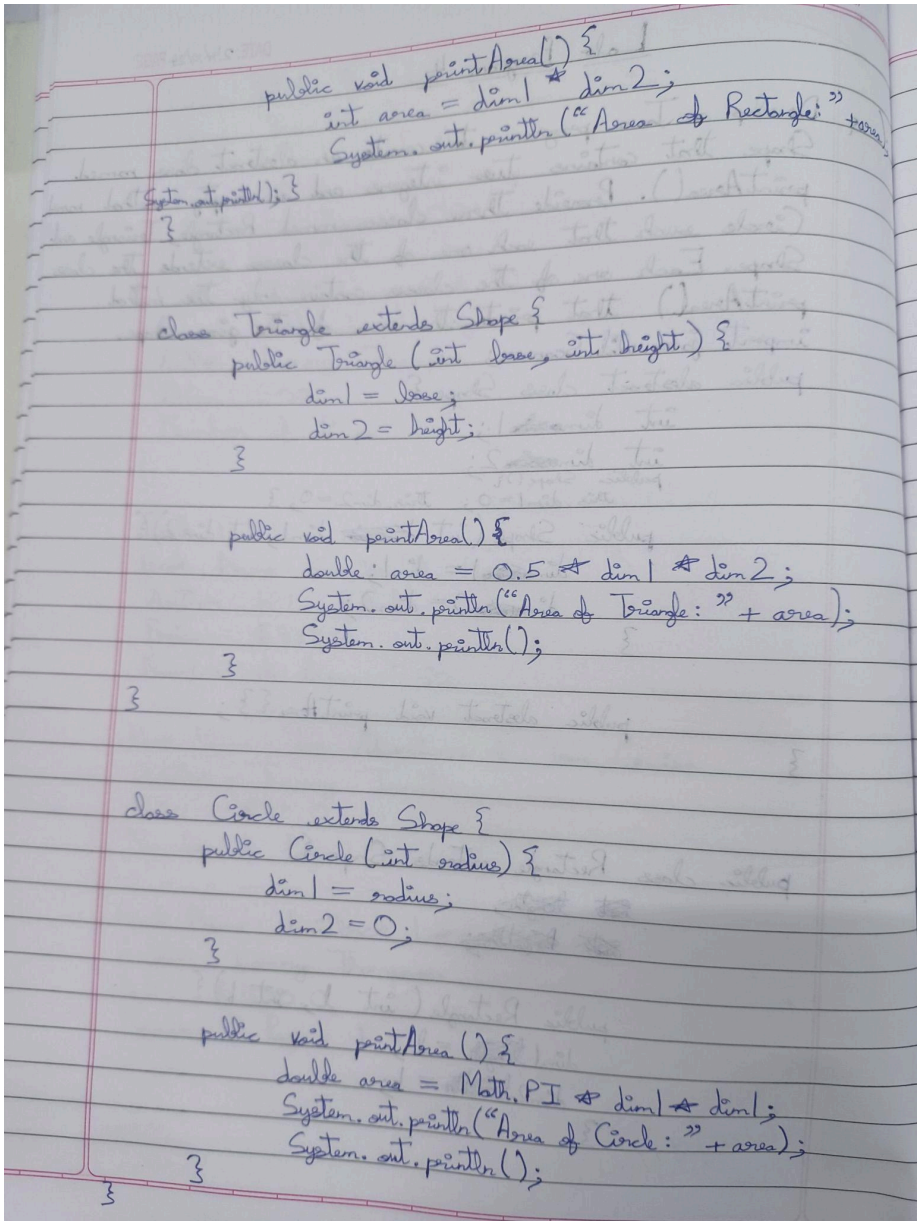


DATE: 24/10/24 PAGE:

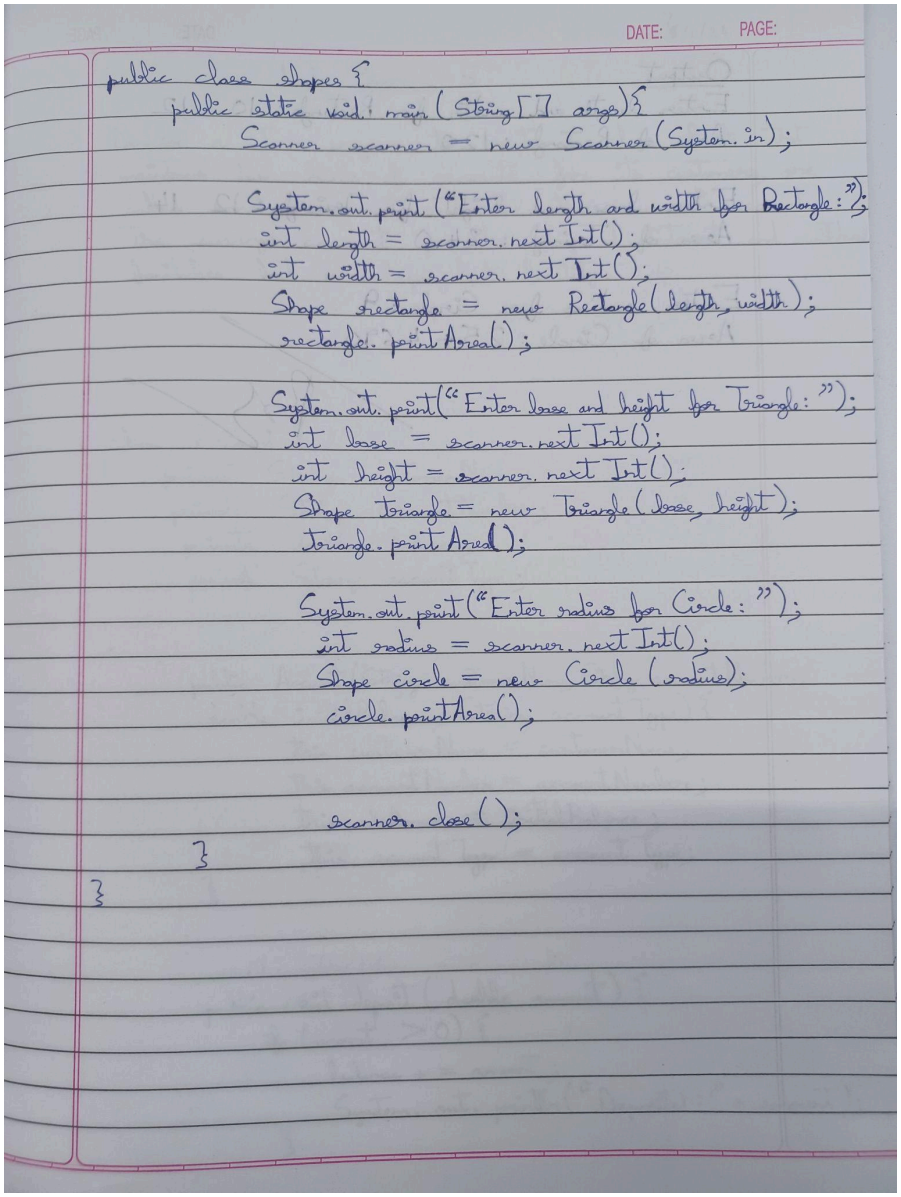
Lab Program 4

Q 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.

```
import java.util.Scanner;
public abstract class Shape {
    int dim1;
    int dim2;
    public Shape(){
        this.dim1=0; this.dim2=0; }
    public Shape (int dim1, int dim2) {
        dim1 = dim1;
        dim2 = dim2;
    }

    public abstract void printArea{};
}

public class Rectangle extends Shape {

    public Rectangle ( int l, int b) {
        dim1 = l;
        dim2 = b;
    }
```

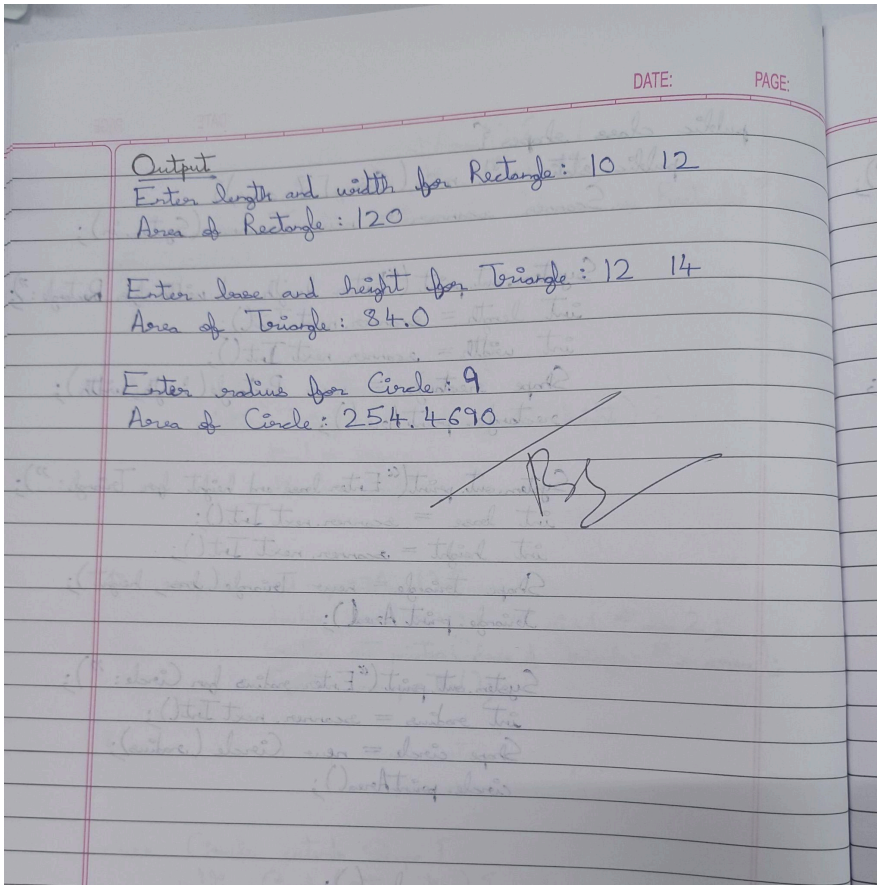


```
    public void printArea() {
        int area = dim1 * dim2;
        System.out.println("Area of Rectangle: " + area);
    System.out.println(); }
    }

class Triangle extends Shape {
    public Triangle (int base, int height) {
        dim1 = base;
        dim2 = height;
    }

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

class Circle extends Shape {
    public Circle (int radius) {
        dim1 = radius;
        dim2 = 0;
    }

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



DATE: PAGE:

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

        System.out.print("Enter length and width for Rectangle: ");
        int length = scanner.nextInt();
        int width = scanner.nextInt();
        Shape rectangle = new Rectangle(length, width);
        rectangle.printAreal();

        System.out.print("Enter base and height for Triangle: ");
        int base = scanner.nextInt();
        int height = scanner.nextInt();
        Shape triangle = new Triangle(base, height);
        triangle.printAreal();

        System.out.print("Enter radius for Circle: ");
        int radius = scanner.nextInt();
        Shape circle = new Circle(radius);
        circle.printAreal();


        scanner.close();
    }
}
```



// Program to Print Area of different Shapes

```
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) {
        dim1 = length;
        dim2 = 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) {
        dim1 = base;
        dim2 = 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) {
        dim1 = radius;
        dim2 = 0;
    }


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


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


        System.out.println("Enter length and width for Rectangle:");
        int length = scanner.nextInt();
        int width = scanner.nextInt();
        Shape rectangle = new Rectangle(length, width);
        rectangle.printArea();
```

```
        System.out.println("Enter base and height for Triangle:");
        int base = scanner.nextInt();
        int height = scanner.nextInt();
        Shape triangle = new Triangle(base, height);
        triangle.printArea();


        System.out.println("Enter radius for Circle:");
        int radius = scanner.nextInt();
        Shape circle = new Circle(radius);
        circle.printArea();



        scanner.close();
    }
}
```

```
Enter length and width for Rectangle:
10 12
Area of Rectangle: 120
Enter base and height for Triangle:
12 14
Area of Triangle: 84.0
Enter radius for Circle:
9
Area of Circle: 254.46900494077323
```

LABORATORY PROGRAM - 5

Develop a Java program to create a class Bank that maintains two kinds of accounts 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 deposits from customers and update the balance.
b) Display the balance.
c) Compute and deposit interest
d) Permit withdrawal and update the balance

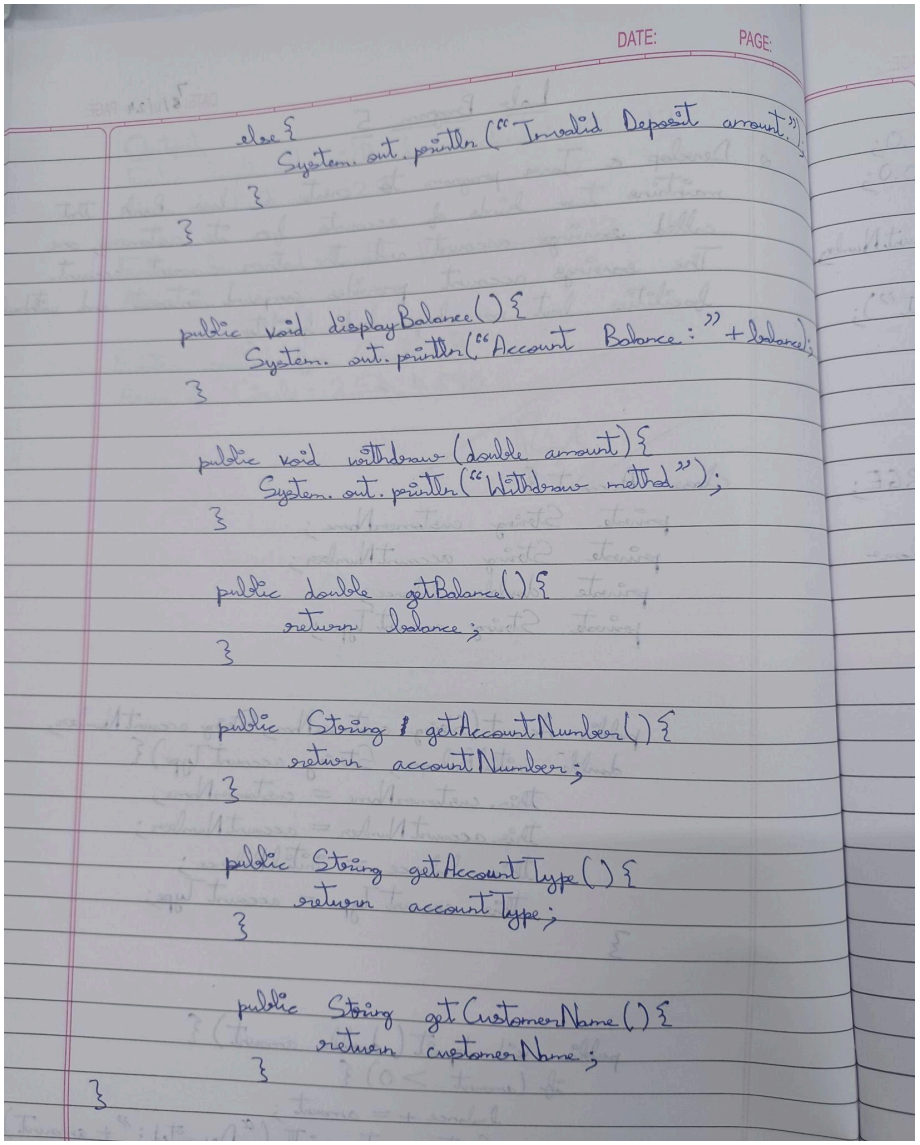
Check for the minimum balance, impose a penalty if necessary and update the balance.

DATE: 7/11/24 PAGE:

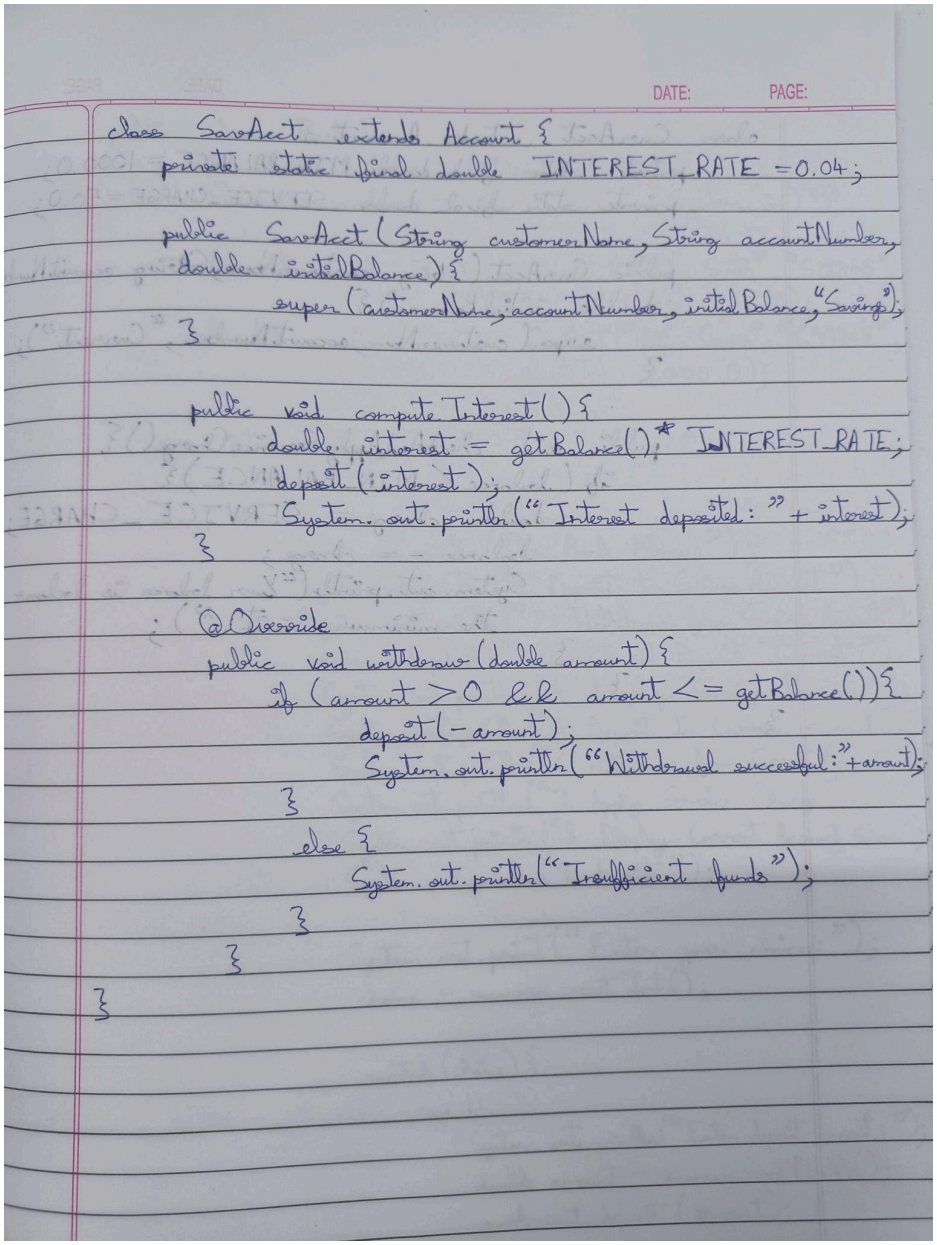
Lab Program 5

Q Develop a Java program to create a class Bank that maintains two kinds of accounts 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.

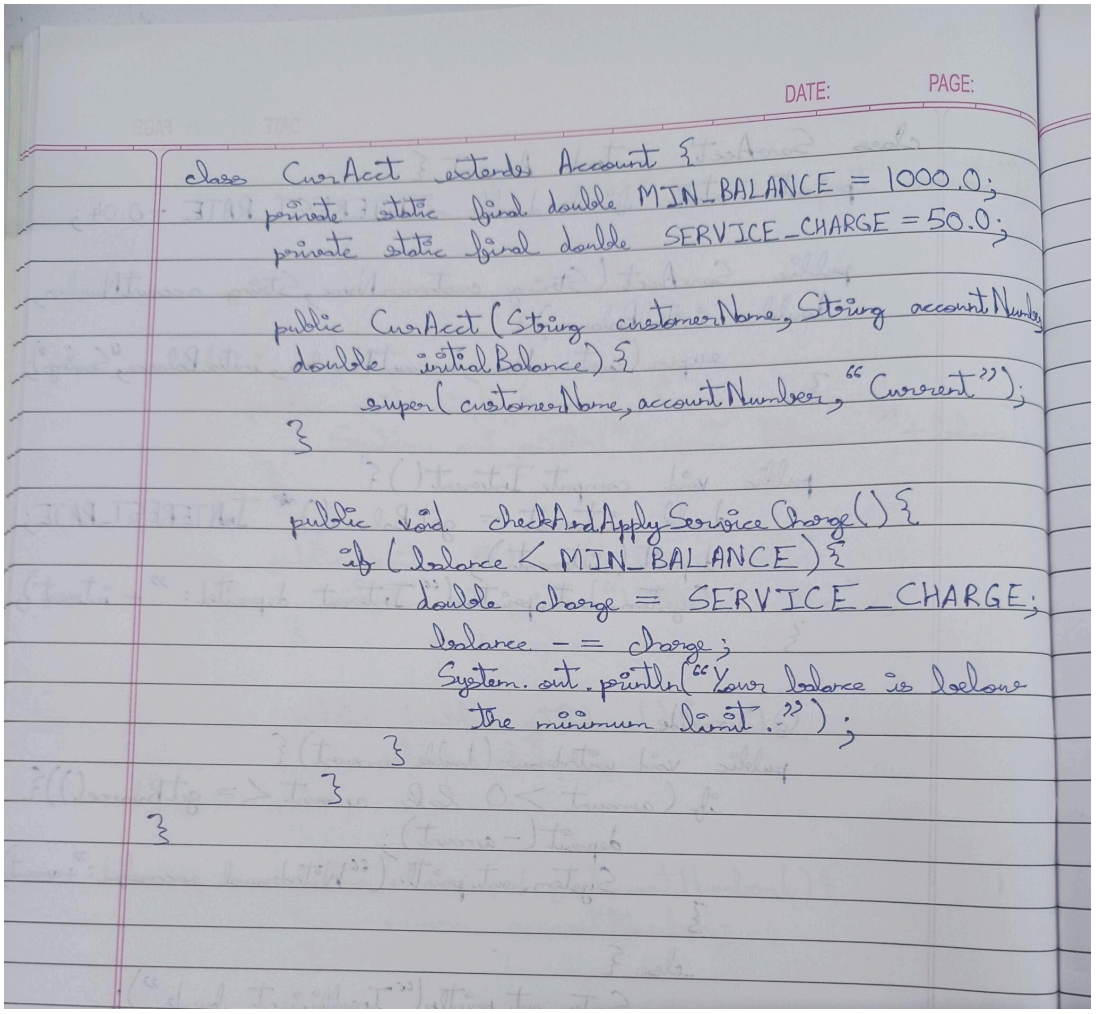
```
class Account {
    private String customerName;
    private String accountNumber;
    private double balance;
    private String accountType;

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

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
    }
```



```
        else {
            System.out.println("Invalid Deposit amount.");
        }
    }

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

    public void withdraw(double amount) {
        System.out.println("Withdraw method");
    }

    public double getBalance() {
        return balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public String getAccountType() {
        return accountType;
    }

    public String getCustomerName() {
        return customerName;
    }
}
```

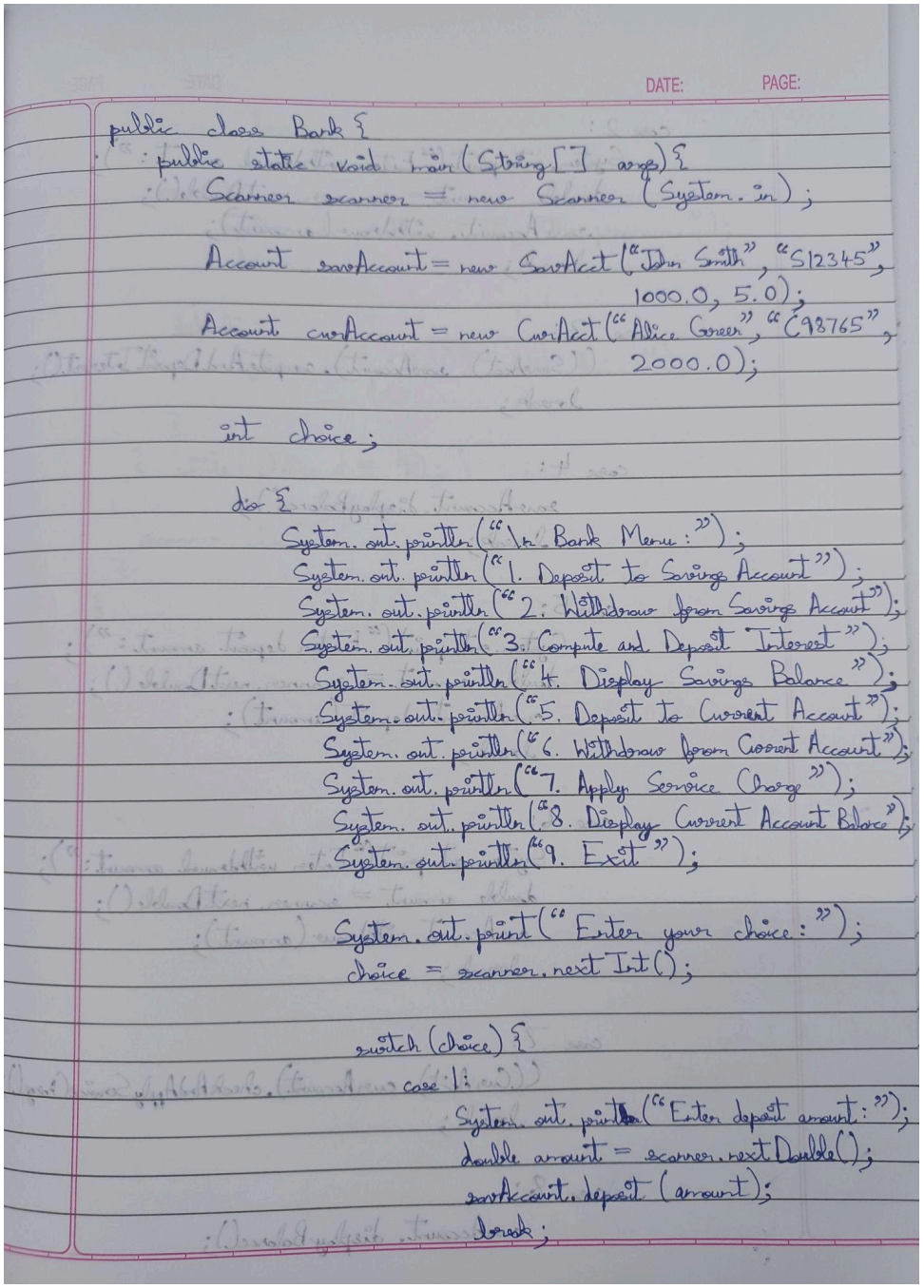


```
class SavAcct extends Account {
    private static final double INTEREST_RATE = 0.04;

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

    public void computeInterest() {
        double interest = getBalance() * INTEREST_RATE;
        deposit (interest);
        System.out.println("Interest deposited: " + interest);
    }

    @Override
    public void withdraw (double amount) {
        if (amount > 0 && amount <= getBalance()) {
            deposit (-amount);
            System.out.println("Withdrawal successful: "+amount);
        }
        else {
            System.out.println("Insufficient funds");
        }
    }
}
```



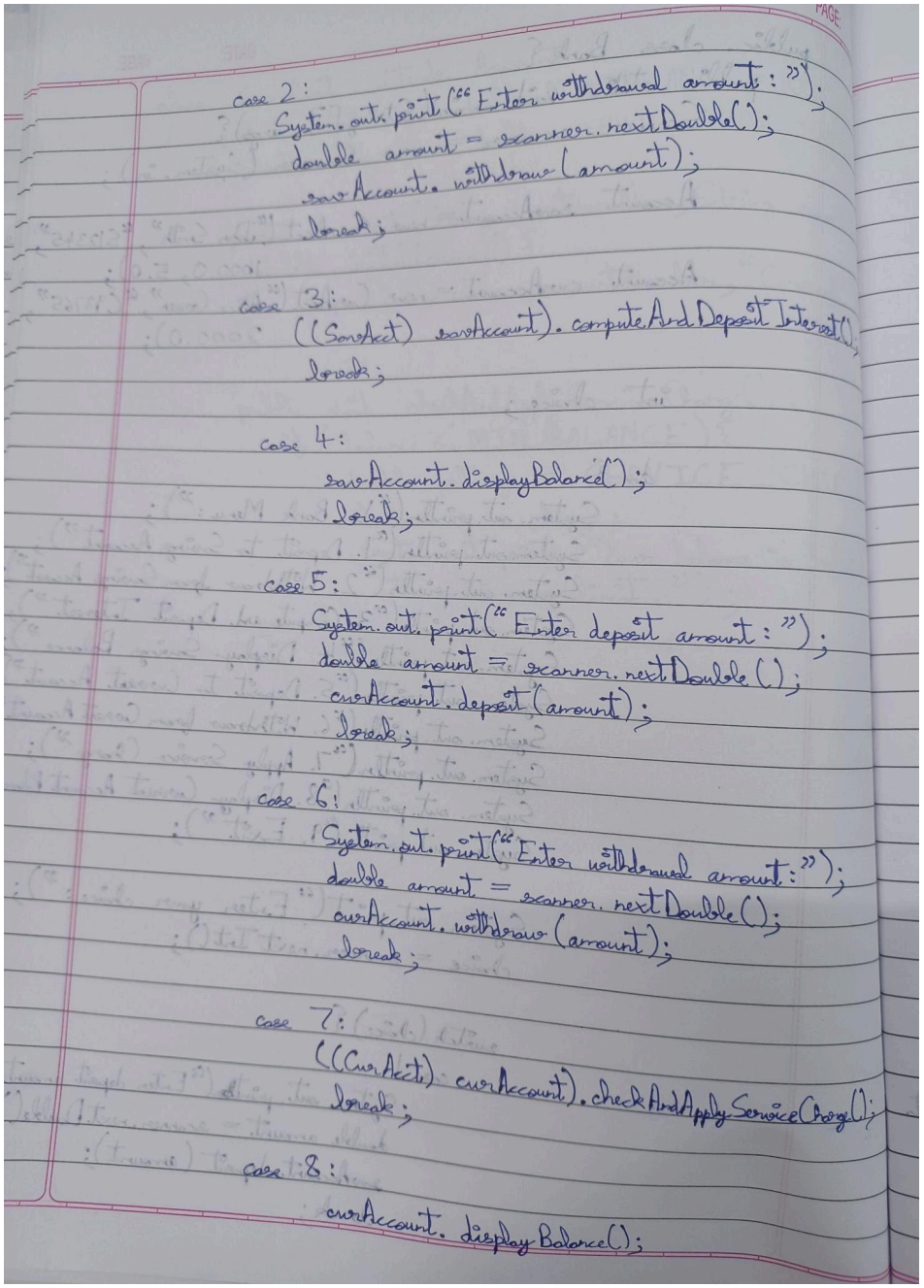
DATE: PAGE:

```
class CurAcct extends Account {
private static final double MIN_BALANCE = 1000.0;
private static final double SERVICE_CHARGE = 50.0;

public CurAcct (String customerName, String accountNumber,
double initialBalance) {
    super(customerName, accountNumber, "Current");
}

public void checkAndApplyServiceCharge() {
    if (balance < MIN_BALANCE) {
        double charge = SERVICE_CHARGE;
        balance -= charge;
        System.out.println("Your balance is below
        the minimum limit.");
    }
}
}
```

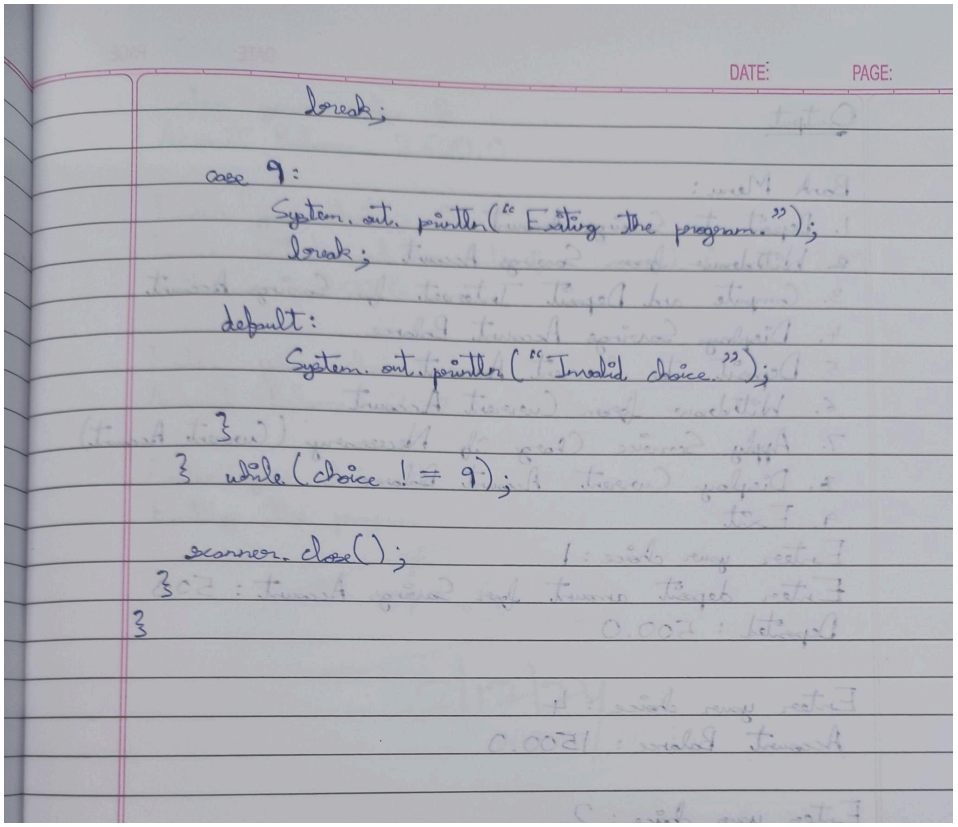


DATE: PAGE:

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

        Account savAccount = new SavAcct("John Smith", "S12345",
                                         1000.0, 5.0);
        Account curAccount = new CurAcct("Alice Green", "C98765",
                                         2000.0);

        int choice;

        do {
            System.out.println("\n Bank Menu:");
            System.out.println("1. Deposit to Savings Account");
            System.out.println("2. Withdraw from Savings Account");
            System.out.println("3. Compute and Deposit Interest");
            System.out.println("4. Display Savings Balance");
            System.out.println("5. Deposit to Current Account");
            System.out.println("6. Withdraw from Current Account");
            System.out.println("7. Apply Service Charge");
            System.out.println("8. Display Current Account Balance");
            System.out.println("9. Exit");

            System.out.print("Enter your choice:");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    System.out.print("Enter deposit amount:");
                    double amount = scanner.nextDouble();
                    savAccount.deposit(amount);
                    break;
```



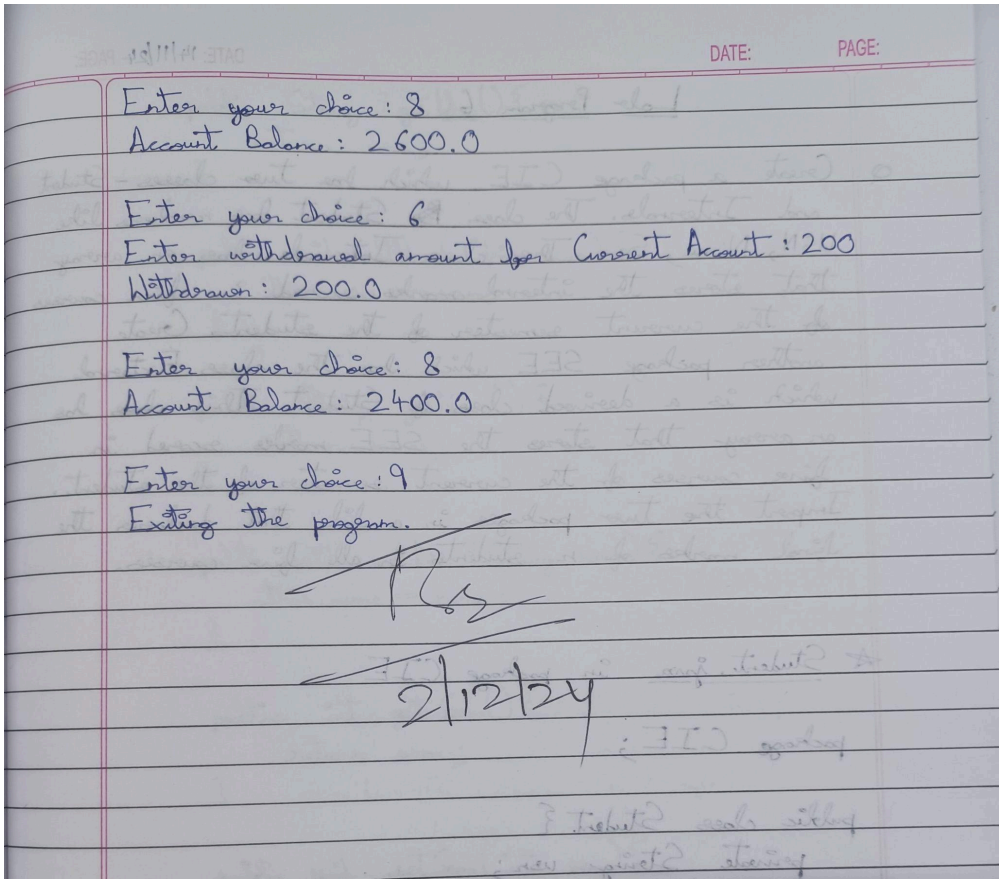
```
case 2:
    System.out.print("Enter withdrawal amount: ");
    double amount = scanner.nextDouble();
    savAccount.withdraw(amount);
    break;

case 3:
    ((SavAcct) savAccount).computeAndDepositInterest();
    break;

case 4:
    savAccount.displayBalance();
    break;

case 5:
    System.out.print("Enter deposit amount: ");
    double amount = scanner.nextDouble();
    curAccount.deposit(amount);
    break;

case 6:
    System.out.print("Enter withdrawal amount: ");
    double amount = scanner.nextDouble();
    curAccount.withdraw(amount);
    break;

case 7:
    ((CurAcct) curAccount).checkAndApplyServiceCharge();
    break;

case 8:
    curAccount.displayBalance();
```



```
                break;

            case 9:
                System.out.println("Exiting the program.");
                break;

            default:
                System.out.println("Invalid choice.");

        }
    } while (choice != 9);

    scanner.close();
}
}
```

Output

```
Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 1
Enter deposit amount for Savings Account: 500
Deposited: 500.0

Enter your choice: 4
Account Balance: 1500.0

Enter your choice: 2
Enter withdrawal amount for Savings Account: 250
Withdrawn: 250.0

Enter your choice: 4
Account Balance: 1250.0

Enter your choice: 3
Interest of 62.5 has been credited to your savings account

Enter your choice: 5
Enter deposit amount for Current Account: 600
Deposited: 600.0
```



// Program to create a Bank Class

```
import java.util.Scanner;

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

    // Constructor to initialize the account details
    public Account(String customerName, String accountNumber, double initialBalance, String accountType) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
        this.accountType = accountType;
    }

    // Method to accept deposit
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: " + amount);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }
```

```
    // Method to display the balance
    public void displayBalance() {
        System.out.println("Account Balance: " + balance);
    }

    // Method to withdraw money
    public void withdraw(double amount) {
        if (amount > 0 && balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance or invalid withdrawal amount.");
        }
    }
}

// Savings Account Class (Subclass of Account)
class SavAcct extends Account {
    private double interestRate;

    // Constructor to initialize a savings account
    public SavAcct(String customerName, String accountNumber, double initialBalance, double interestRate) {
        super(customerName, accountNumber, initialBalance, "Savings");
        this.interestRate = interestRate;
    }

    // Method to compute and deposit interest
    public void computeAndDepositInterest() {
        double interest = balance * interestRate / 100;
        balance += interest;
        System.out.println("Interest of " + interest + " has been credited to your savings account.");
    }
}

// Current Account Class (Subclass of Account)
class CurAcct extends Account {
    private static final double MIN_BALANCE = 500.0;
    private static final double SERVICE_CHARGE = 50.0;

    // Constructor to initialize a current account
    public CurAcct(String customerName, String accountNumber, double initialBalance) {
        super(customerName, accountNumber, initialBalance, "Current");
    }

    // Method to apply service charge if the balance is below the minimum
    public void checkAndApplyServiceCharge() {
        if (balance < MIN_BALANCE) {
            double charge = SERVICE_CHARGE;
            balance -= charge;
            System.out.println("Your balance is below the minimum limit. A service charge of " + charge + " has been
applied.");
        }
    }
}

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

        // Creating instances for savings and current accounts
        Account savAccount = new SavAcct("John Doe", "S12345", 1000.0, 5.0);
```

```
Account curAccount = new CurAcct("Alice Smith", "C98765", 2000.0);

// Menu-driven interface
int choice;
do {
   System.out.println("\nBank Menu:");
   System.out.println("1. Deposit to Savings Account");
   System.out.println("2. Withdraw from Savings Account");
   System.out.println("3. Compute and Deposit Interest for Savings Account");
   System.out.println("4. Display Savings Account Balance");
   System.out.println("5. Deposit to Current Account");
   System.out.println("6. Withdraw from Current Account");
   System.out.println("7. Apply Service Charge if Necessary (Current Account)");
   System.out.println("8. Display Current Account Balance");
   System.out.println("9. Exit");

   System.out.print("Enter your choice: ");
   choice = scanner.nextInt();

   switch (choice) {
      case 1: {
         System.out.print("Enter deposit amount for Savings Account: ");
         double amount = scanner.nextDouble();
         savAccount.deposit(amount);
         break;
      }
      case 2: {
         System.out.print("Enter withdrawal amount for Savings Account: ");
         double amount = scanner.nextDouble();
         savAccount.withdraw(amount);
         break;
      }
      case 3: {
         ((SavAcct) savAccount).computeAndDepositInterest();
         break;
      }
      case 4: {
         savAccount.displayBalance();
         break;
      }
      case 5: {
         System.out.print("Enter deposit amount for Current Account: ");
         double amount = scanner.nextDouble();
         curAccount.deposit(amount);
         break;
      }
      case 6: {
         System.out.print("Enter withdrawal amount for Current Account: ");
         double amount = scanner.nextDouble();
         curAccount.withdraw(amount);
         break;
      }
      case 7: {
         ((CurAcct) curAccount).checkAndApplyServiceCharge();
         break;
      }
      case 8: {
         curAccount.displayBalance();
         break;
      }
      case 9: {
         System.out.println("Exiting the program.");
```

```
                break;
            }
            default: {
                System.out.println("Invalid choice. Please try again.");
            }
        }
    } while (choice != 9);

    scanner.close();
  }
}
```

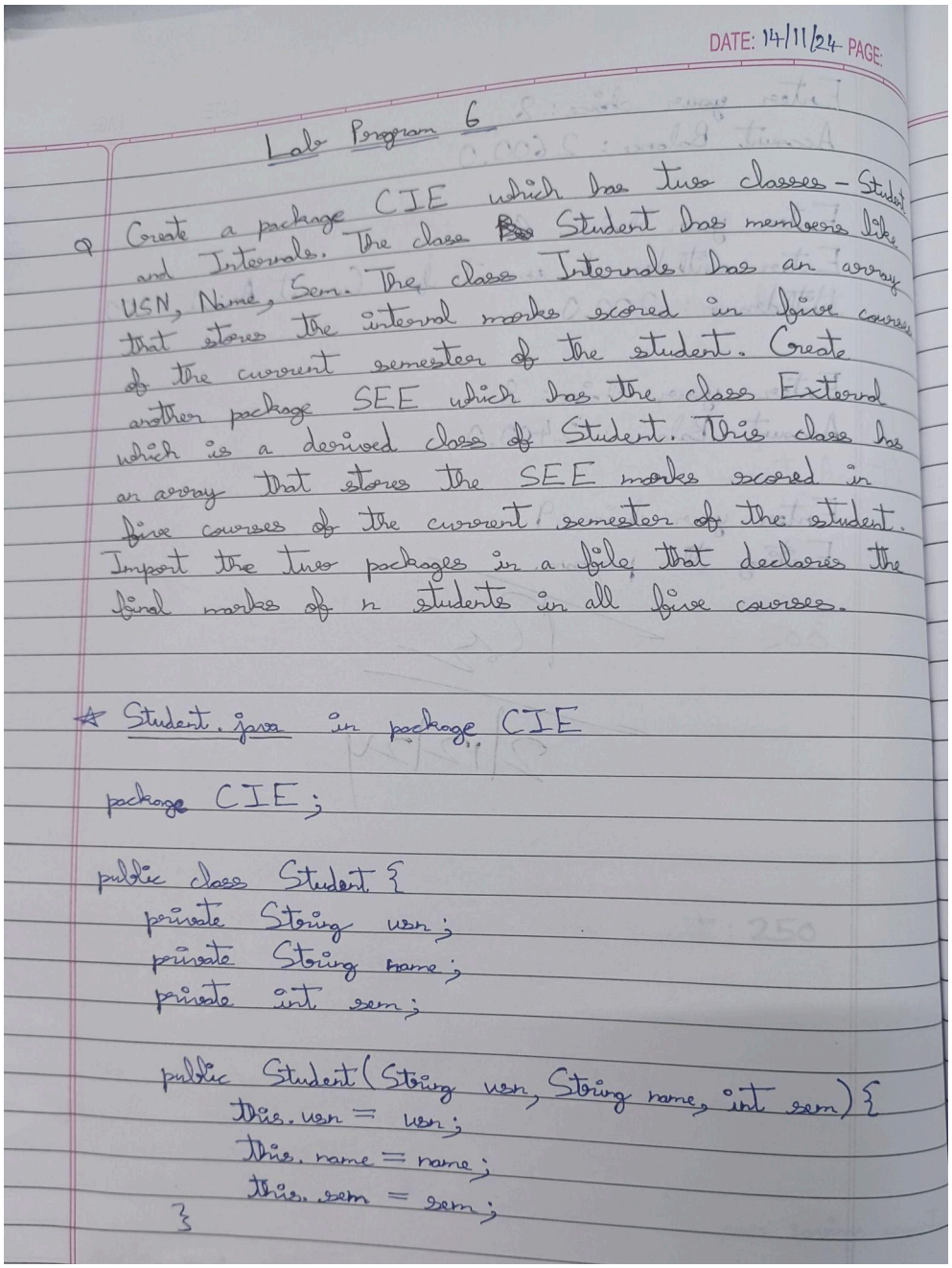
```
Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 1
Enter deposit amount for Savings Account: 500
Deposited: 500.0

Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 4
Account Balance: 1500.0
```

```
Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 2
Enter withdrawal amount for Savings Account: 250
Withdrawn: 250.0

Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 3
Interest of 62.5 has been credited to your savings account.
```

```
Enter your choice: 6
Enter withdrawal amount for Current Account: 200
Withdrawn: 200.0

Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 8
Account Balance: 1800.0

Bank Menu:
1. Deposit to Savings Account
2. Withdraw from Savings Account
3. Compute and Deposit Interest for Savings Account
4. Display Savings Account Balance
5. Deposit to Current Account
6. Withdraw from Current Account
7. Apply Service Charge if Necessary (Current Account)
8. Display Current Account Balance
9. Exit
Enter your choice: 9
Exiting the program.
```

LABORATORY PROGRAM - 6

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



DATE: 14/11/24 PAGE:

Lab Program 6

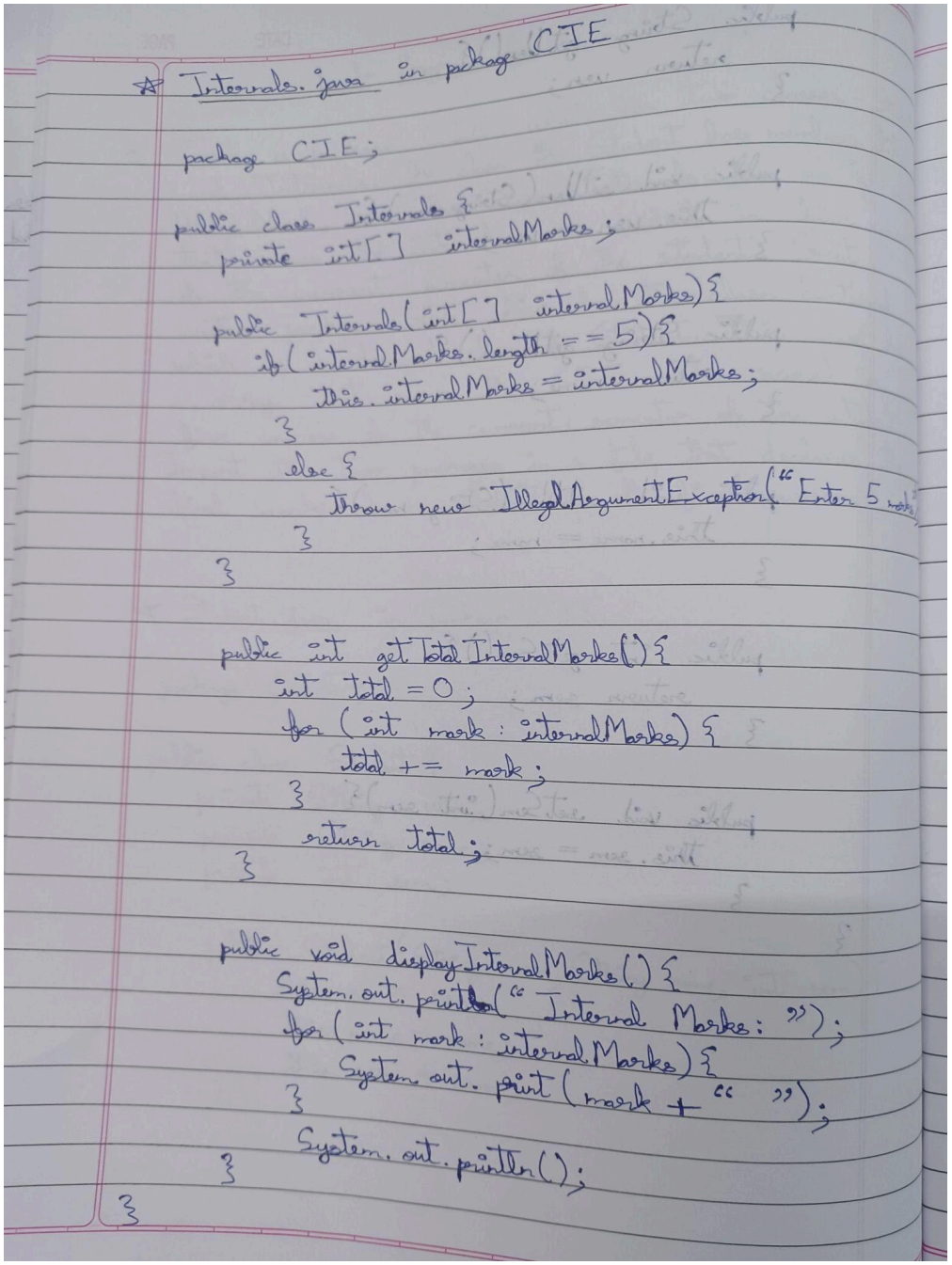
Q Create a package CIE which has two classes - Student and Internals. The class Student has members like USN, Name, Sem. The class Internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External 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.

☆ Student.java in package CIE

```
package CIE;

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

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

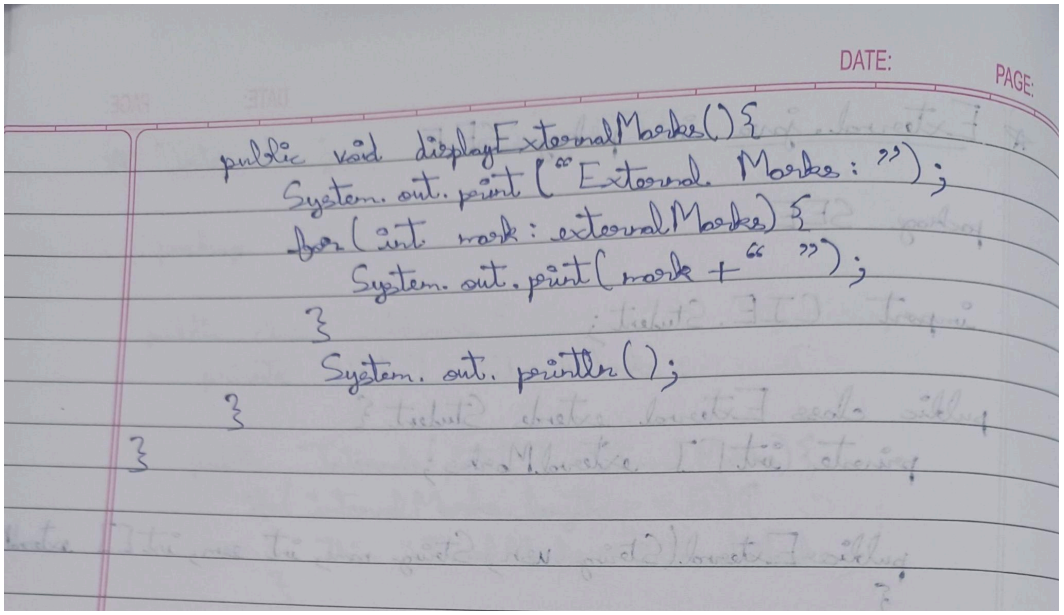
DATE: PAGE:

```
public String getUsn(){
    return usn;
}

public void setUsn(String usn){
    this.usn = usn;
}

public String getName(){
    return name;
}

public void setName(String name){
    this.name = name;
}

public int getSem(){
    return sem;
}

public void setSem(int sem){
    this.sem = sem;
}
}
```



* Internals.java in package CIE

```
package CIE;

public class Internals {
    private int[] internalMarks;

    public Internals(int[] internalMarks){
        if(internalMarks.length == 5){
            this.internalMarks = internalMarks;
        }
        else {
            throw new IllegalArgumentException("Enter 5 marks");
        }
    }

    public int getTotalInternalMarks(){
        int total = 0;
        for (int mark : internalMarks) {
            total += mark;
        }
        return total;
    }

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



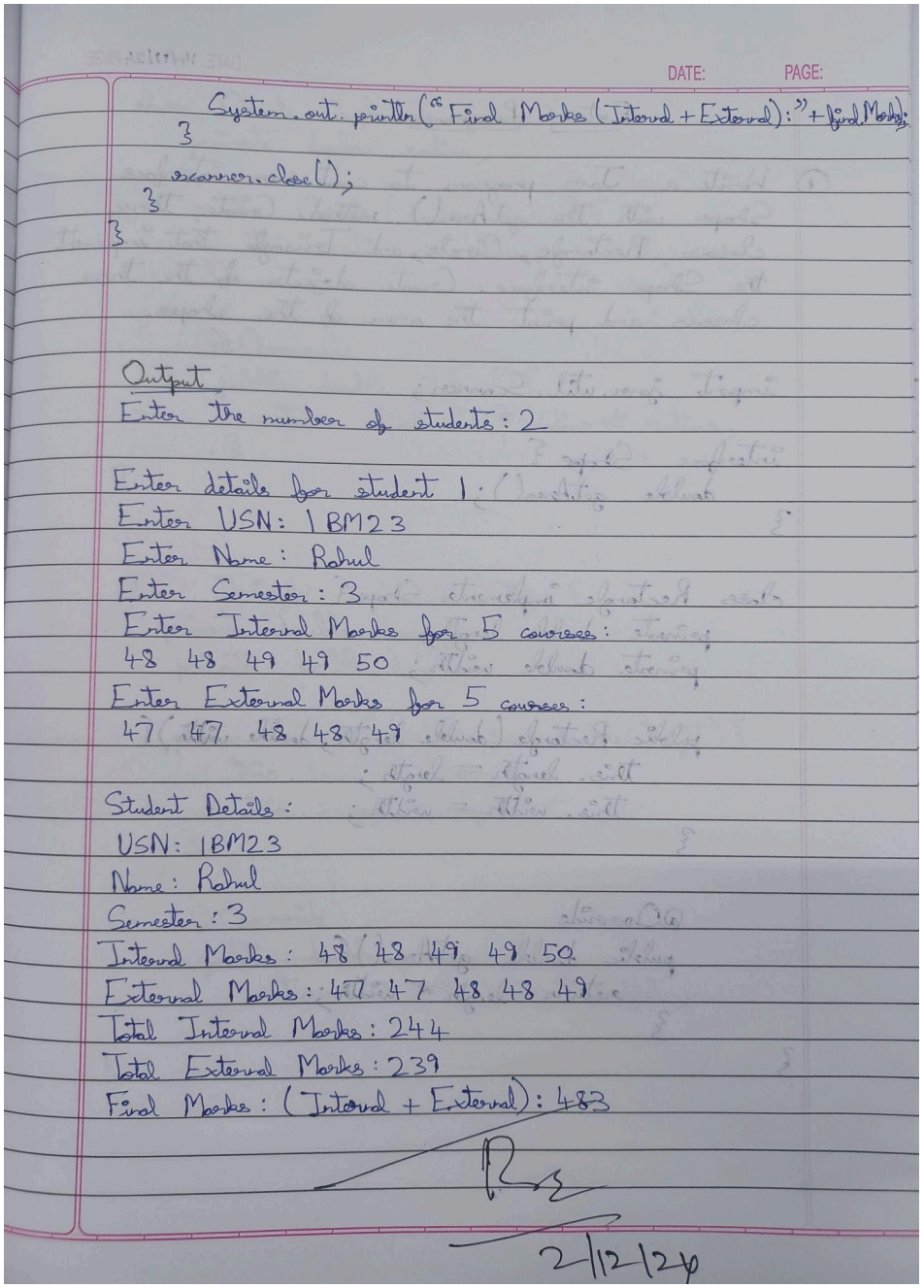
```
public void displayExternalMarks(){
    System.out.print("External Marks: ");
    for(int mark: externalMarks){
        System.out.print(mark + " ");
    }
    System.out.println();
}
}
```

DATE: PAGE:

A Main.java in package SEE

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

import java.util.Scanner;

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

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

        External[] students = new External[n];

        for(int i=0; i<n; i++) {
            System.out.println("\n Enter details for student "+(i+1));

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

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

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

            int[] internalMarks = new int[5];
```

```
System.out.println("Enter Internal Marks");
for(int j=0; j<5; j++){
    internalMarks[j]=scanner.nextInt();
}

Internals internals = new Internals(internalMarks);

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

students[i] = new External(usn, name, sem, externalMarks);

System.out.println("\n Student Details:");
System.out.println("USN: "+students[i].getUsn());
System.out.println("Name: "+students[i].getName());
System.out.println("Semester: "+students[i].getSem());

internals.displayInternalMarks();

students[i].displayExternalMarks();

int totalInternalMarks = internals.getTotalInternalMarks();
int totalExternalMarks = students[i].getTotalExternalMarks();
int finalMarks = totalInternalMarks + totalExternalMarks;

System.out.println("Total Internal Marks: "+
                   totalInternalMarks);
System.out.println("Total External Marks: "+
                   totalExternalMarks);
```



```
System.out.println("Final Marks (Internal + External):" + finalMarks);
        }
        scanner.close();
    }
}
```

Output

```
Enter the number of students: 2

Enter details for student 1:
Enter USN: 1BM23
Enter Name: Rahul
Enter Semester: 3
Enter Internal Marks for 5 courses:
48 48 49 49 50
Enter External Marks for 5 courses:
47 47 48 48 49

Student Details:
USN: 1BM23
Name: Rahul
Semester: 3
Internal Marks: 48 48 49 49 50
External Marks: 47 47 48 48 49
Total Internal Marks: 244
Total External Marks: 239
Final Marks: (Internal + External): 483
```

2/12/24

// Student.java in package CIE

```
package CIE;

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


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

```
    public String getUsn() {
        return usn;
    }

    public void setUsn(String usn) {
        this.usn = usn;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getSem() {
        return sem;
    }

    public void setSem(int sem) {
        this.sem = sem;
    }
}
```

```
// Internals.java in CIE

package CIE;

public class Internals {
    private int[] internalMarks;


    public Internals(int[] internalMarks) {
        if (internalMarks.length == 5) {
            this.internalMarks = internalMarks;
        } else {
            throw new IllegalArgumentException("Internal marks must be an array of size 5.");
        }
    }


    public int getTotalInternalMarks() {
        int total = 0;
        for (int mark : internalMarks) {
            total += mark;
        }
        return total;
    }


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

```
        System.out.println();
    }
}
```

```
// External.java in SEE

package SEE;

import CIE.Student;

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


    public External(String usn, String name, int sem, int[] externalMarks) {
        super(usn, name, sem);
        if (externalMarks.length == 5) {
            this.externalMarks = externalMarks;
        } else {
            throw new IllegalArgumentException("External marks must be an array of size 5.");
        }
    }


    public int getTotalExternalMarks() {
        int total = 0;
        for (int mark : externalMarks) {
            total += mark;
        }
        return total;
    }


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

```
// Main.java in SEE

import CIE.Internals;
import SEE.External;

import java.util.Scanner;

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


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

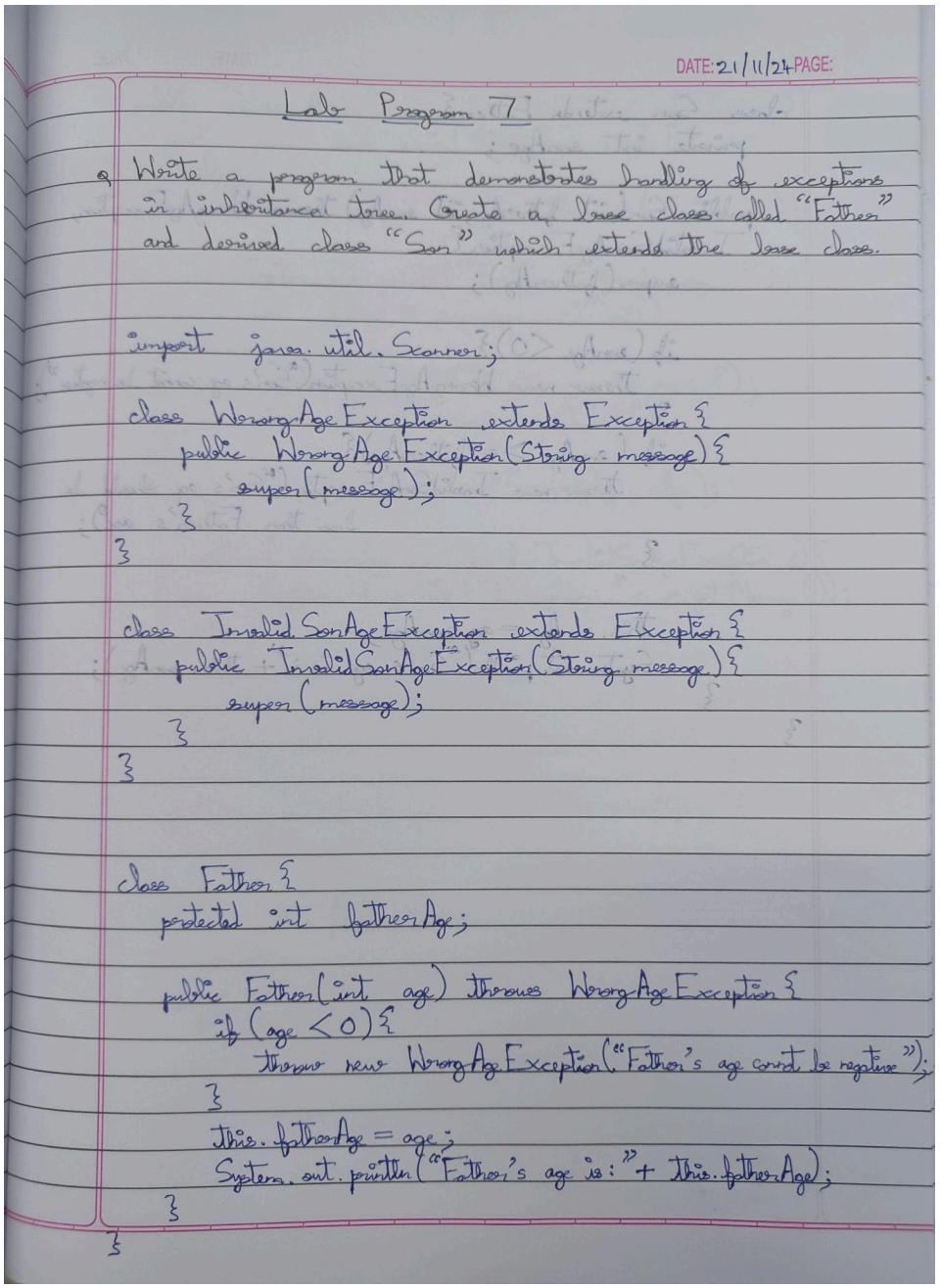
```
        scanner.nextLine();

        External[] students = new External[n];

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

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

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

            Internals internals = new Internals(internalMarks);

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

            students[i] = new External(usn, name, sem, externalMarks);

            System.out.println("\nStudent Details: ");
            System.out.println("USN: " + students[i].getUsn());
            System.out.println("Name: " + students[i].getName());
            System.out.println("Semester: " + students[i].getSem());

            internals.displayInternalMarks();

            students[i].displayExternalMarks();

            int totalInternalMarks = internals.getTotalInternalMarks();
            int totalExternalMarks = students[i].getTotalExternalMarks();
            int finalMarks = totalInternalMarks + totalExternalMarks;
            System.out.println("Total Internal Marks: " + totalInternalMarks);
            System.out.println("Total External Marks: " + totalExternalMarks);
            System.out.println("Final Marks (Internal + External): " + finalMarks);
        }

        scanner.close();
    }
}
```

```
Enter the number of students: 2

Enter details for student 1
Enter USN: 1RV20CS001
Enter Name: Alice
Enter Semester: 5
Enter Internal Marks for 5 courses:
70 80 90 75 85
Enter External Marks for 5 courses:
60 70 80 65 90

Student Details:
USN: 1RV20CS001
Name: Alice
Semester: 5
Internal Marks: 70 80 90 75 85
External Marks: 60 70 80 65 90
Total Internal Marks: 400
Total External Marks: 365
Final Marks (Internal + External): 765
```

```
Enter details for student 2
Enter USN: 1RV20CS002
Enter Name: Bob
Enter Semester: 5
Enter Internal Marks for 5 courses:
65 75 80 70 90
Enter External Marks for 5 courses:
55 85 90 80 70

Student Details:
USN: 1RV20CS002
Name: Bob
Semester: 5
Internal Marks: 65 75 80 70 90
External Marks: 55 85 90 80 70
Total Internal Marks: 380
Total External Marks: 380
Final Marks (Internal + External): 760
```

LABORATORY PROGRAM - 7

Write a program that demonstrates handling of exceptions in inheritance trees. Create a base class called “Father” and a 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.

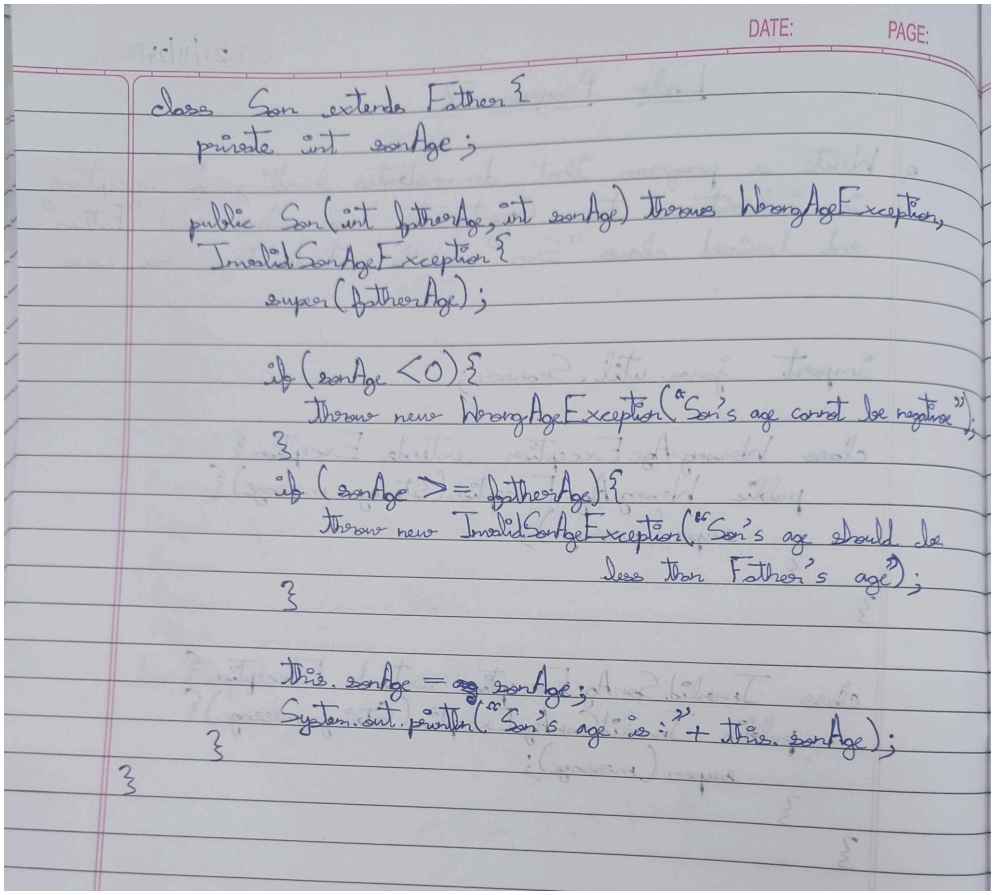


DATE: 21/11/24 PAGE:

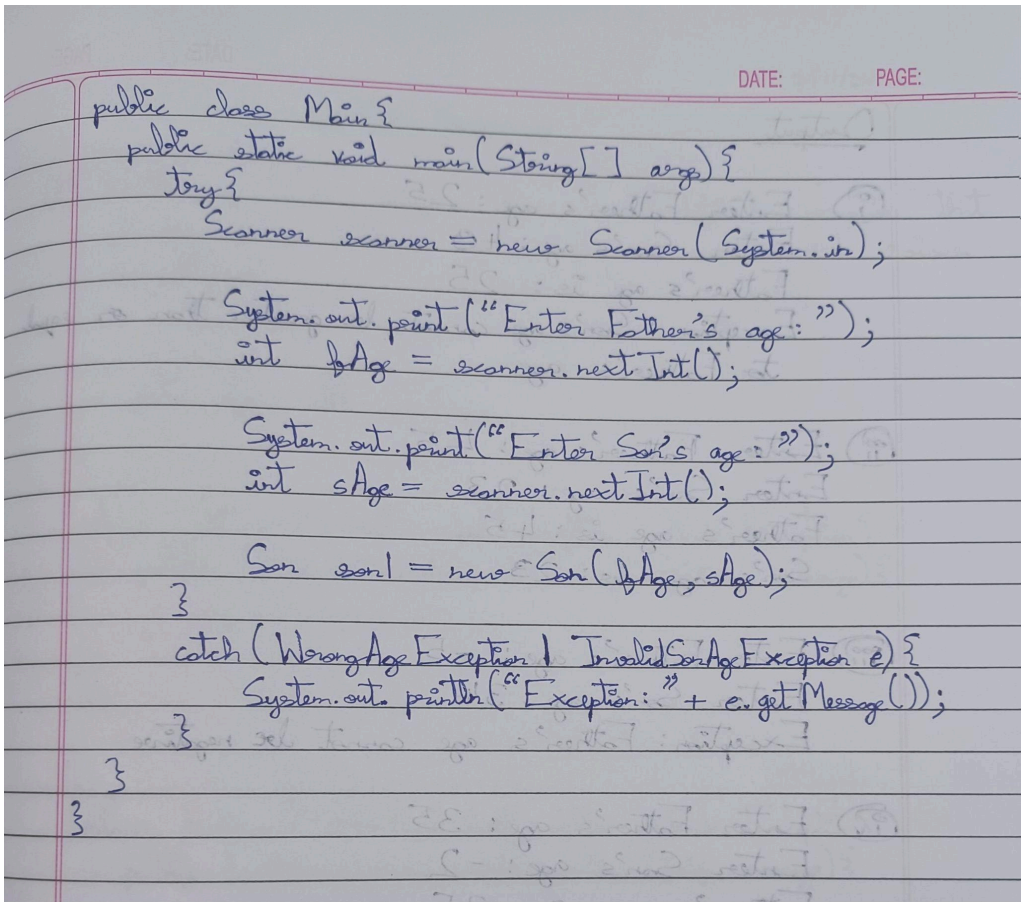
Lab Program 7

Q. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class "Son" which extends the base class.

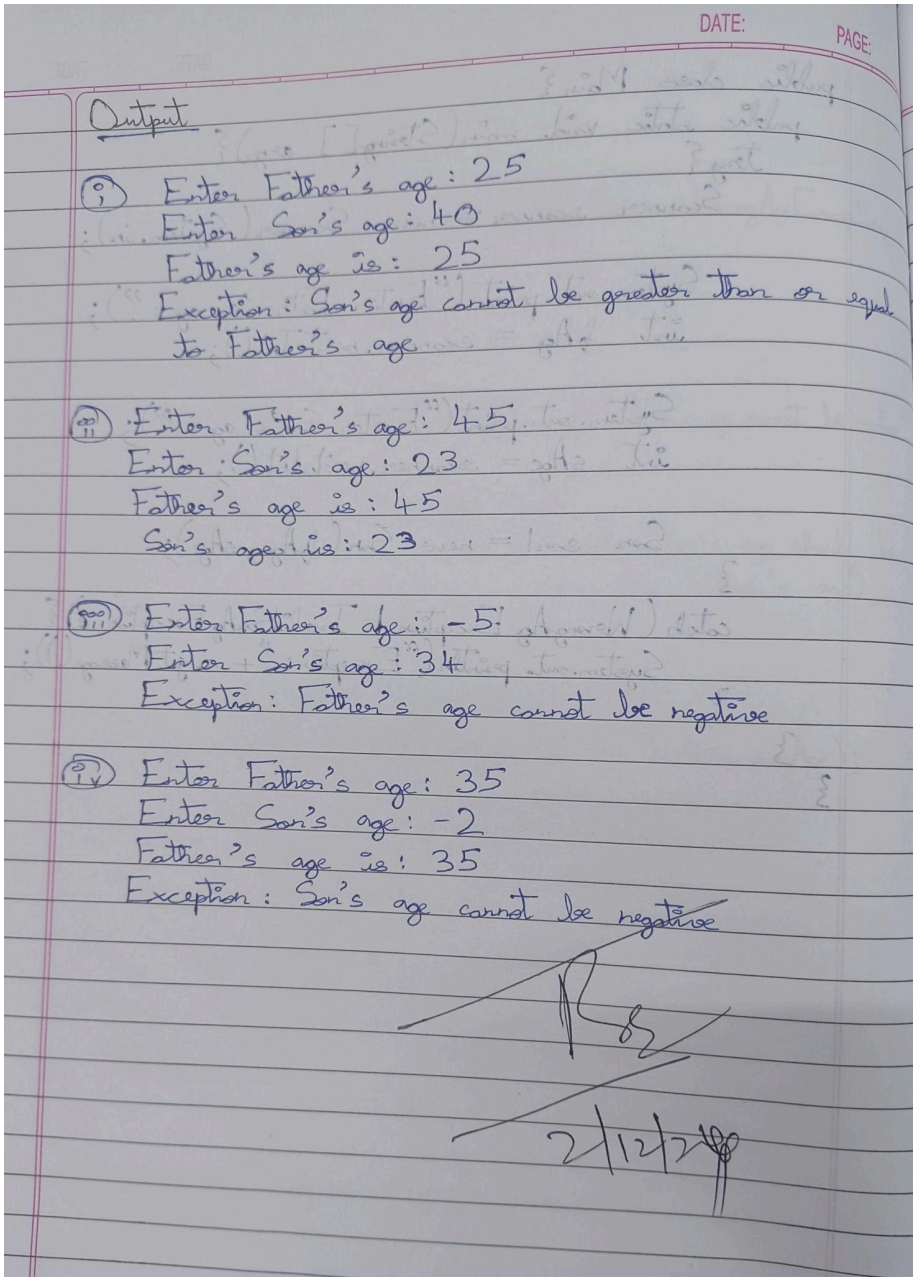
```
import java.util.Scanner;

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

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

class Father {
    protected int fatherAge;

    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Father's age cannot be negative");
        }
        this.fatherAge = age;
        System.out.println("Father's age is: " + this.fatherAge);
    }
}
```



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

    public Son(int fatherAge, int sonAge) throws WrongAgeException,
    InvalidSonAgeException {
        super(fatherAge);

        if (sonAge < 0) {
            throw new WrongAgeException("Son's age cannot be negative");
        }
        if (sonAge >= fatherAge) {
            throw new InvalidSonAgeException("Son's age should be
                                    less than Father's age");
        }

        this.sonAge = sonAge;
        System.out.println("Son's age is: " + this.sonAge);
    }
}
```



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

            System.out.print("Enter Father's age: ");
            int fAge = scanner.nextInt();

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

            Son son1 = new Son(fAge, sAge);
        }
        catch (WrongAgeException | InvalidSonAgeException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```




```
import java.util.Scanner;


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


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

```
class Father {
    protected int fatherAge;


    public Father(int age) throws WrongAgeException {
        if (age < 0) {
            throw new WrongAgeException("Father's age cannot be negative: " + age);
        }
        this.fatherAge = age;
        System.out.println("Father's age is: " + this.fatherAge);
    }
}


class Son extends Father {
    private int sonAge;


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

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


            System.out.print("Enter Father's age: ");
                int fAge = scanner.nextInt();

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


            Son son1 = new Son(fAge, sAge);

        } catch (WrongAgeException | InvalidSonAgeException e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

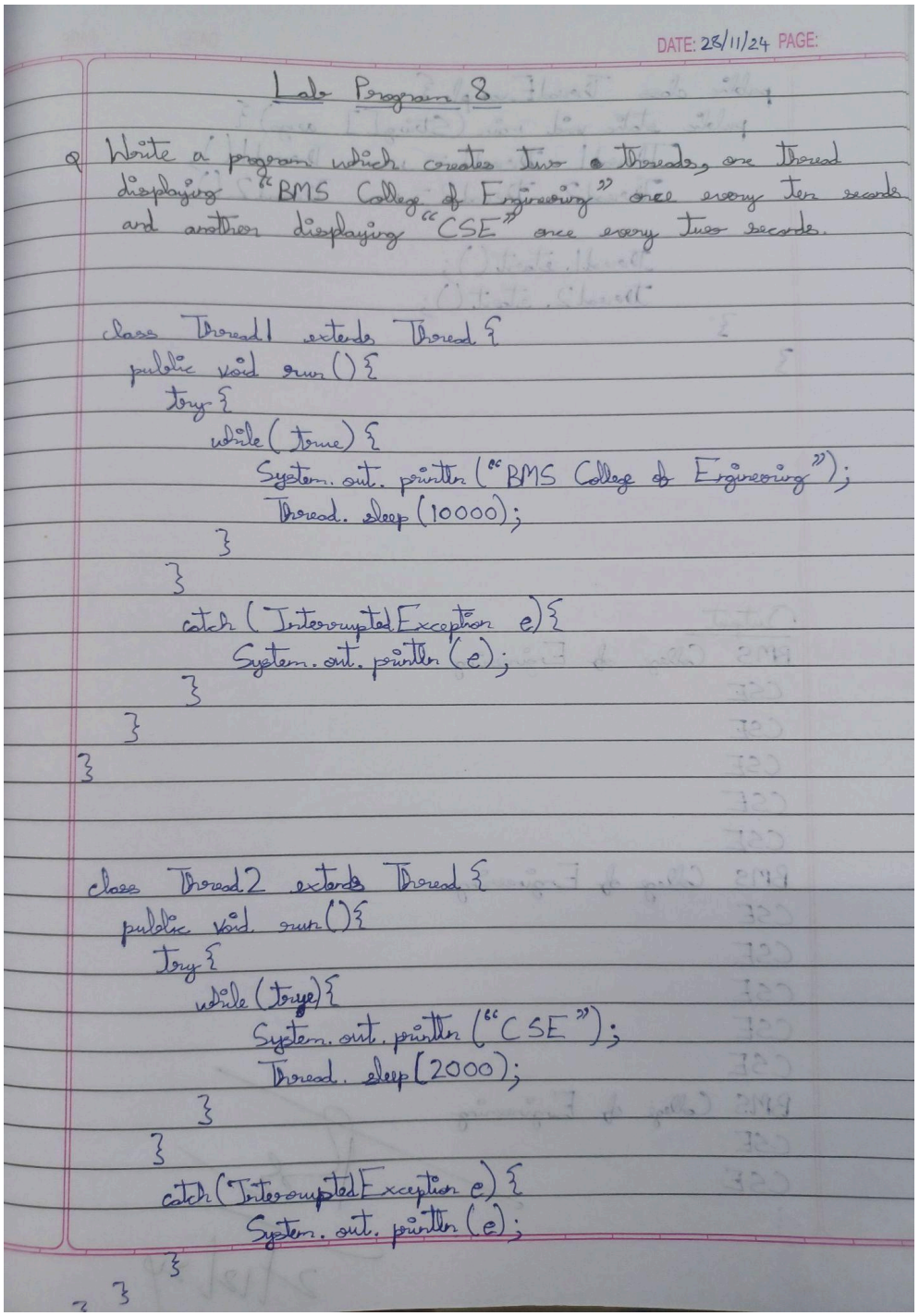
```
Enter Father's age: 25
Enter Son's age: 40
Father's age is: 25
Exception: Son's age cannot be greater than or equal to Father's age.
```

```
Enter Father's age: 45
Enter Son's age: 23
Father's age is: 45
Son's age is: 23
```

```
Enter Father's age: -5
Enter Son's age: 34
Exception: Father's age cannot be negative: -5
```

LABORATORY 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.

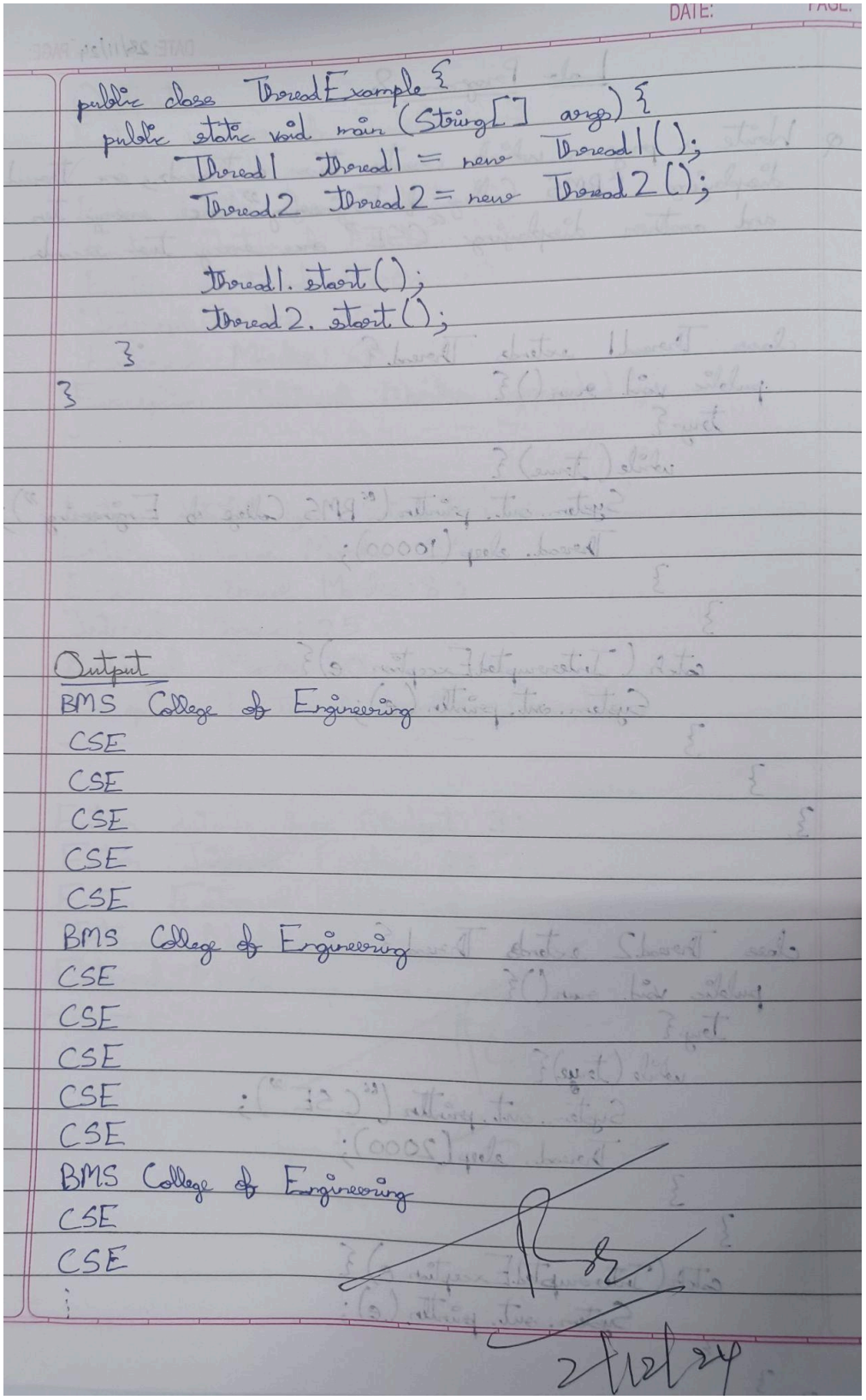


DATE: 28/11/24 PAGE:

Lab Program 8

Q 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 Thread1 extends Thread {
  public void run() {
    try {
      while (true) {
        System.out.println("BMS College of Engineering");
        Thread.sleep(10000);
      }
    }
    catch (InterruptedException e) {
      System.out.println(e);
    }
  }
}

class Thread2 extends Thread {
  public void run() {
    try {
      while (true) {
        System.out.println("CSE");
        Thread.sleep(2000);
      }
    }
    catch (InterruptedException e) {
      System.out.println(e);
    }
  }
}
```



```
public class ThreadExample {
    public static void main(String[] args) {
        Thread1 thread1 = new Thread1();
        Thread2 thread2 = new Thread2();

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

Output

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

2/12/24

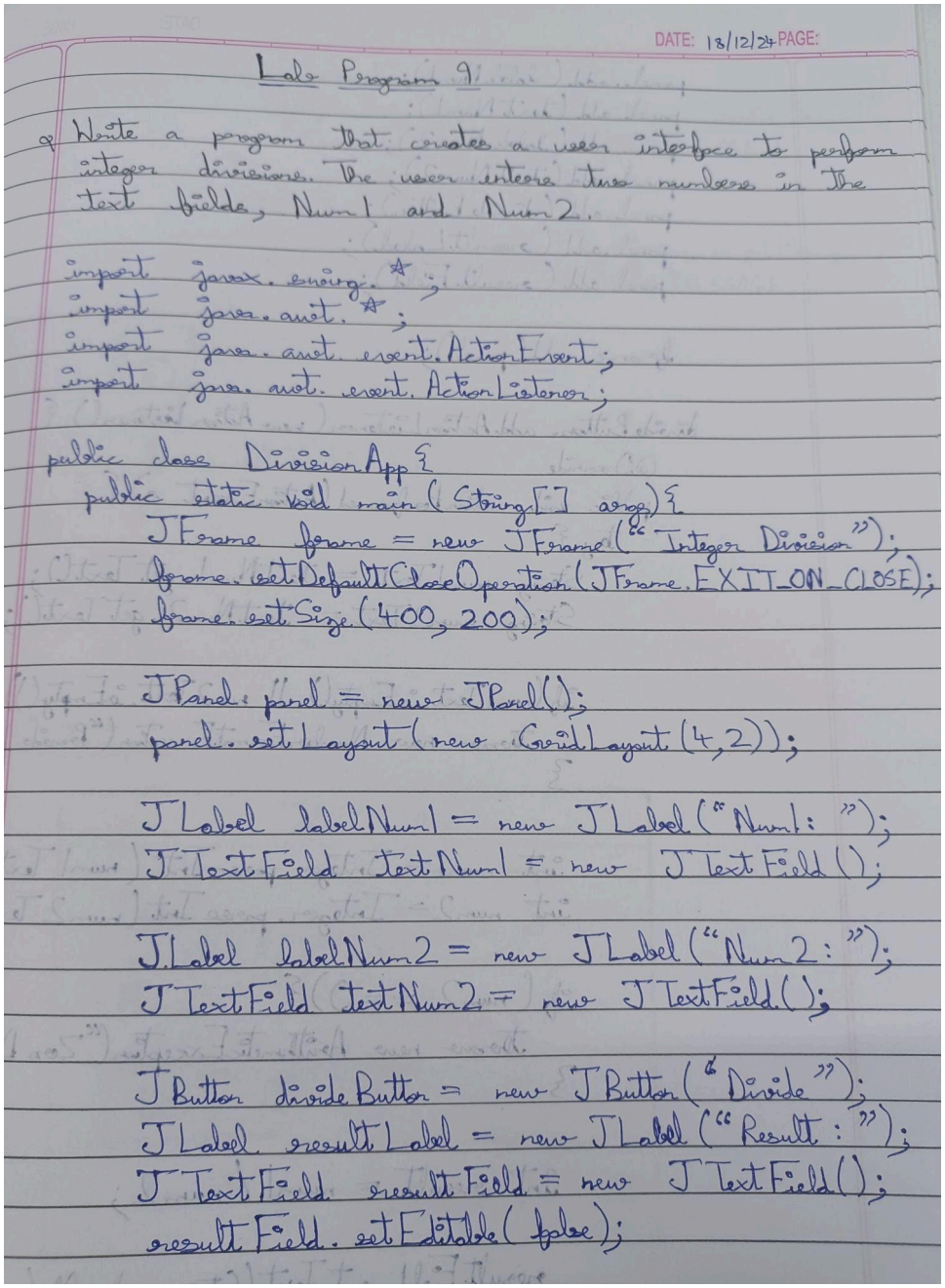
```
// Class for the first thread that prints "BMS College of Engineering" every 10 seconds
class Thread1 extends Thread {
    String a;
    int timer;
```

```
    int i = 0;
    int numTimesEx;

    Thread1(String inp, int time, int numTimes){
            this.a = inp;
            this.timer = time;
            this.numTimesEx = numTimes;
    }

    public void run() {
       try {
          while (true) {
             System.out.println(this.a);
             Thread.sleep(this.timer);
                        i++;

                        if(i >= numTimesEx){
                                this.interrupt();
                        }
          }
       } catch (InterruptedException e) {
          System.out.println(e);
       }
    }
}



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

        Thread1 thread1 = new Thread1("BMS College of Engineering", 10000, 5);
        Thread1 thread2 = new Thread1("CSE", 2000, 25);


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

```
C:\Users\Language lab 28\Desktop\CS328_Java\Week 8>javac ThreadExample.java

C:\Users\Language lab 28\Desktop\CS328_Java\Week 8>java ThreadExample
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
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
java.lang.InterruptedException: sleep interrupted
CSE
java.lang.InterruptedException: sleep interrupted
```

LABORATORY 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.



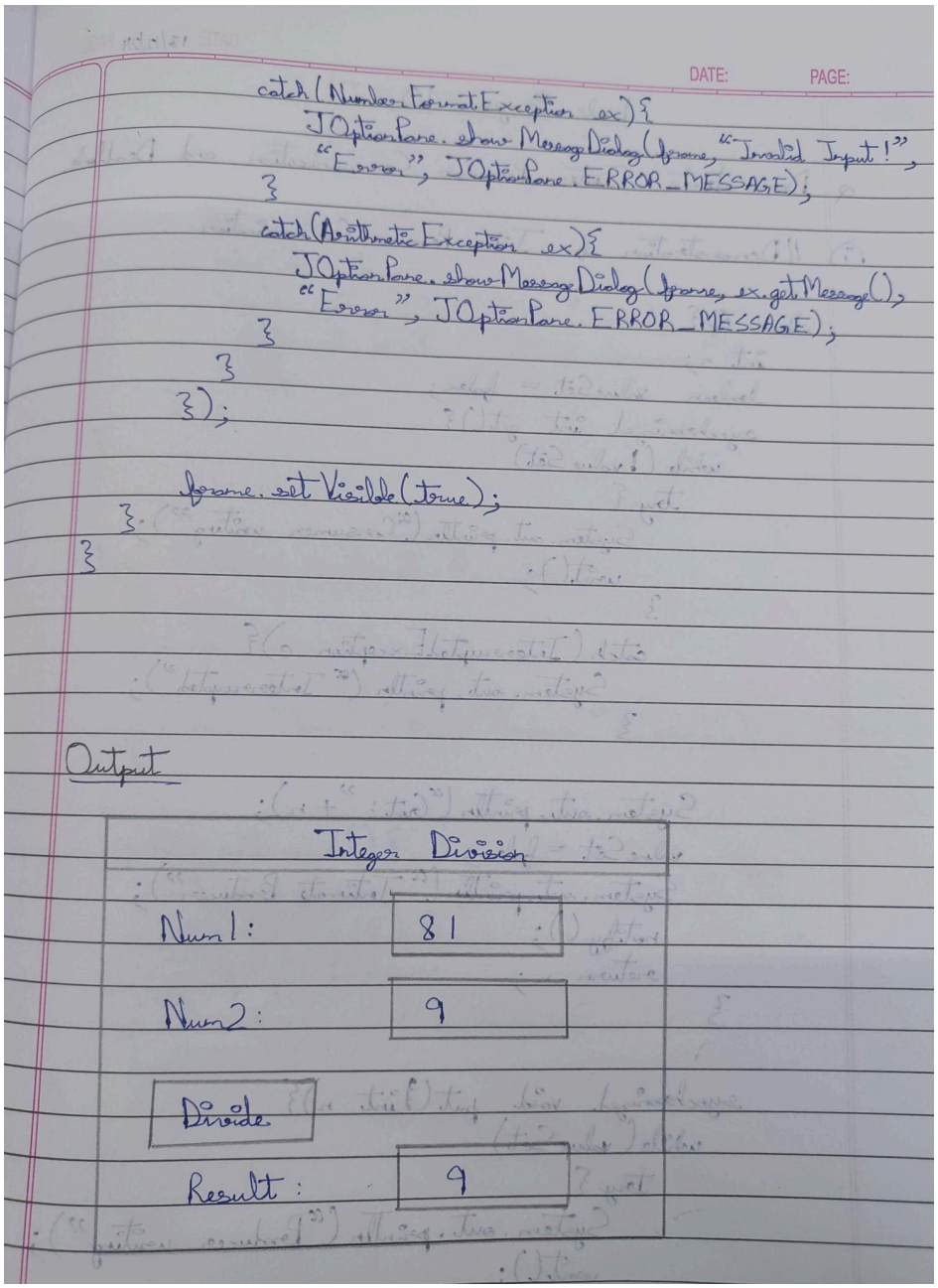
DATE: 13/12/24 PAGE:

Lab Program 9

Q Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2.

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

public class DivisionApp {
    public static void main (String[] args) {
        JFrame frame = new JFrame("Integer Division");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 200);

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(4,2));

        JLabel labelNum1 = new JLabel("Num1:");
        JTextField textNum1 = new JTextField();

        JLabel labelNum2 = new JLabel("Num2:");
        JTextField textNum2 = new JTextField();

        JButton divideButton = new JButton("Divide");
        JLabel resultLabel = new JLabel("Result:");
        JTextField resultField = new JTextField();
        resultField.setEditable(false);
```

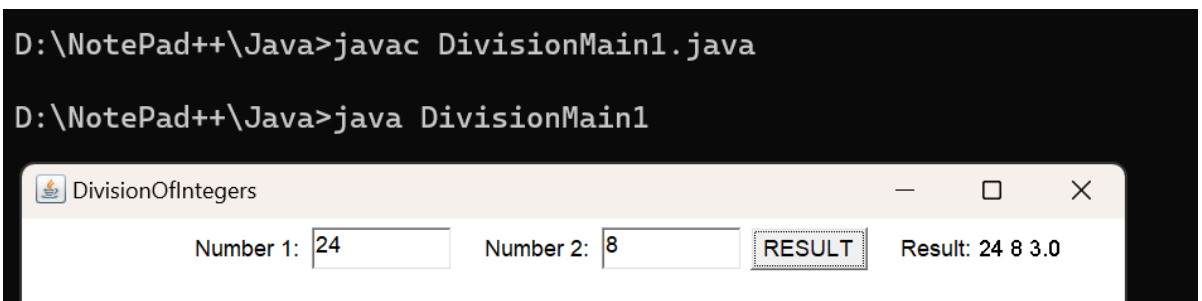
```
panel.add(labelNum1);
panel.add(textNum1);
panel.add(labelNum2);
panel.add(textNum2);
panel.add(divideButton);
panel.add(resultLabel);
panel.add(resultField);

frame.add(panel);

divideButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String num1Text = textNum1.getText();
            String num2Text = textNum2.getText();

            if(num1Text.isEmpty() || num2Text.isEmpty()){
                throw new NumberFormatException("Provide both");
            }

            int num1 = Integer.parseInt(num1Text);
            int num2 = Integer.parseInt(num2Text);

            if(num2 == 0){
                throw new ArithmeticException("Zero Division");
            }

            int result = num1 / num2;

            resultField.setText(String.valueOf(result));
```




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

public class DivisionMain1 extends Frame implements ActionListener
{
TextField num1,num2; Button dResult;
Label outResult; String out=""; double resultNum; int flag=0;

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

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

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

num1.addActionListener(this); num2.addActionListener(this); dResult.addActionListener(this); addWindowListener(new
WindowAdapter()
{
public void windowClosing(WindowEvent we)
{
System.exit(0);
}
});
}
public void actionPerformed(ActionEvent ae)
{
int n1,n2; try
{
if (ae.getSource() == dResult)
{
n1=Integer.parseInt(num1.getText()); n2=Integer.parseInt(num2.getText());

/*if(n2==0)
throw new ArithmeticException();*/ out=n1+" "+n2+" ";
resultNum=n1/n2; out+=String.valueOf(resultNum); repaint();

}
}
catch(NumberFormatException e1)
{
flag=1;
out="Number Format Exception! "+e1; repaint();
}
catch(ArithmeticException e2)
{
flag=1;
out="Divide by 0 Exception! "+e2; repaint();
}

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

public static void main(String[] args)
{
DivisionMain1 dm=new DivisionMain1(); dm.setSize(new Dimension(800,400)); dm.setTitle("DivisionOfIntegers");
dm.setVisible(true);
}
}
```

OUTPUT



LABORATORY PROGRAM - 10

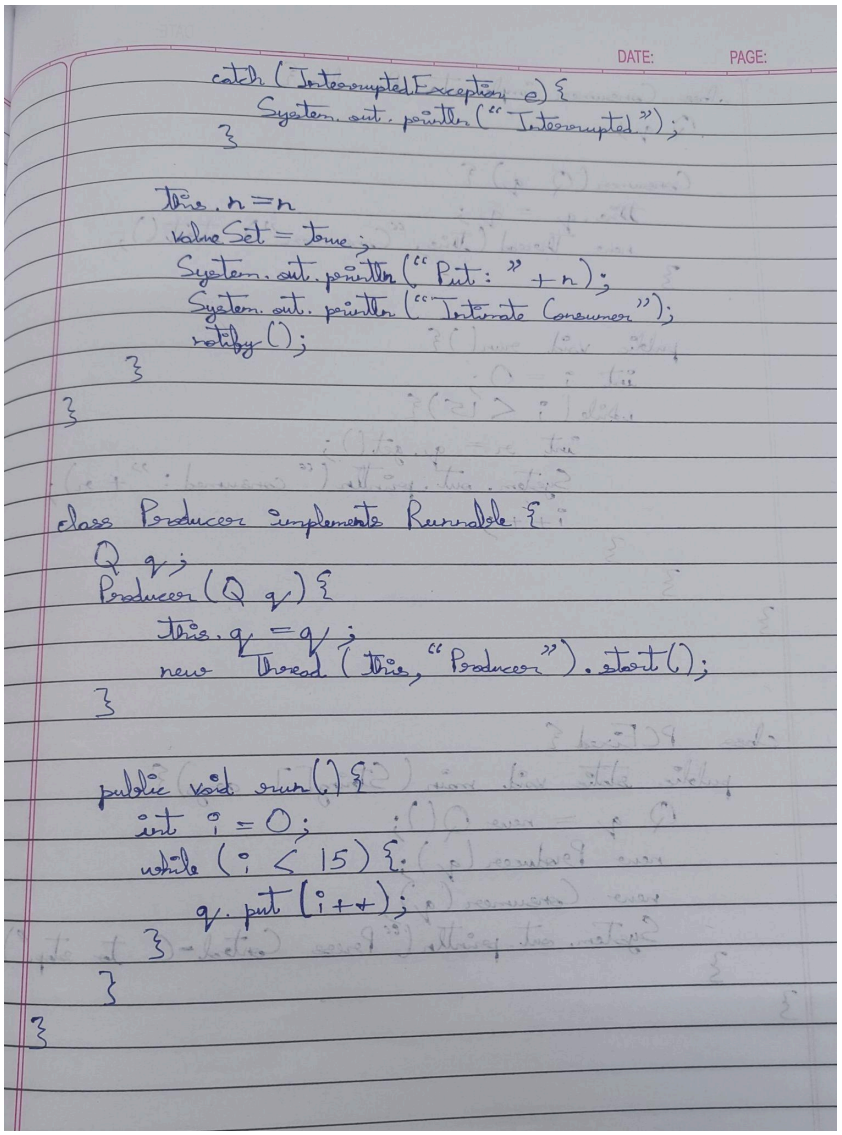
Demonstrate Interprocess communication and deadlock

DATE: 18/12/24 PAGE:

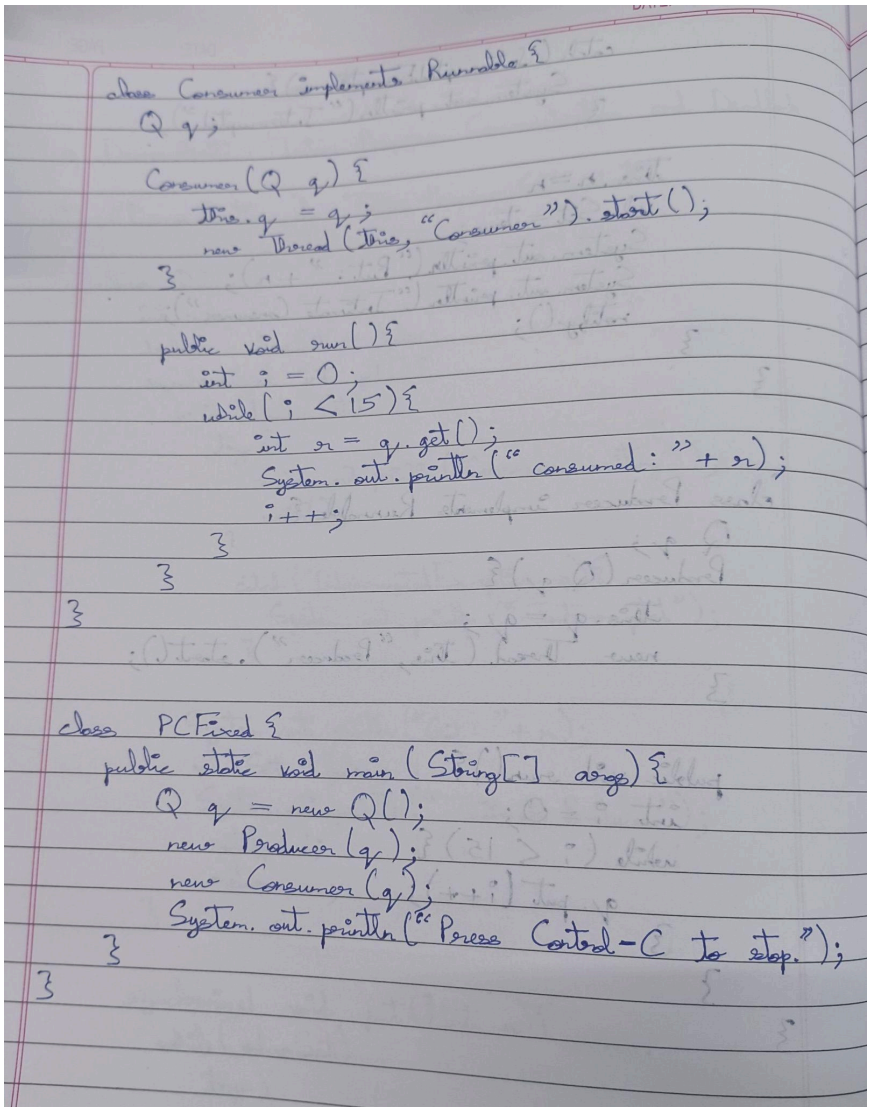
Lab Program 10

Q Demonstrate Inter Process Communication and Deadlock.

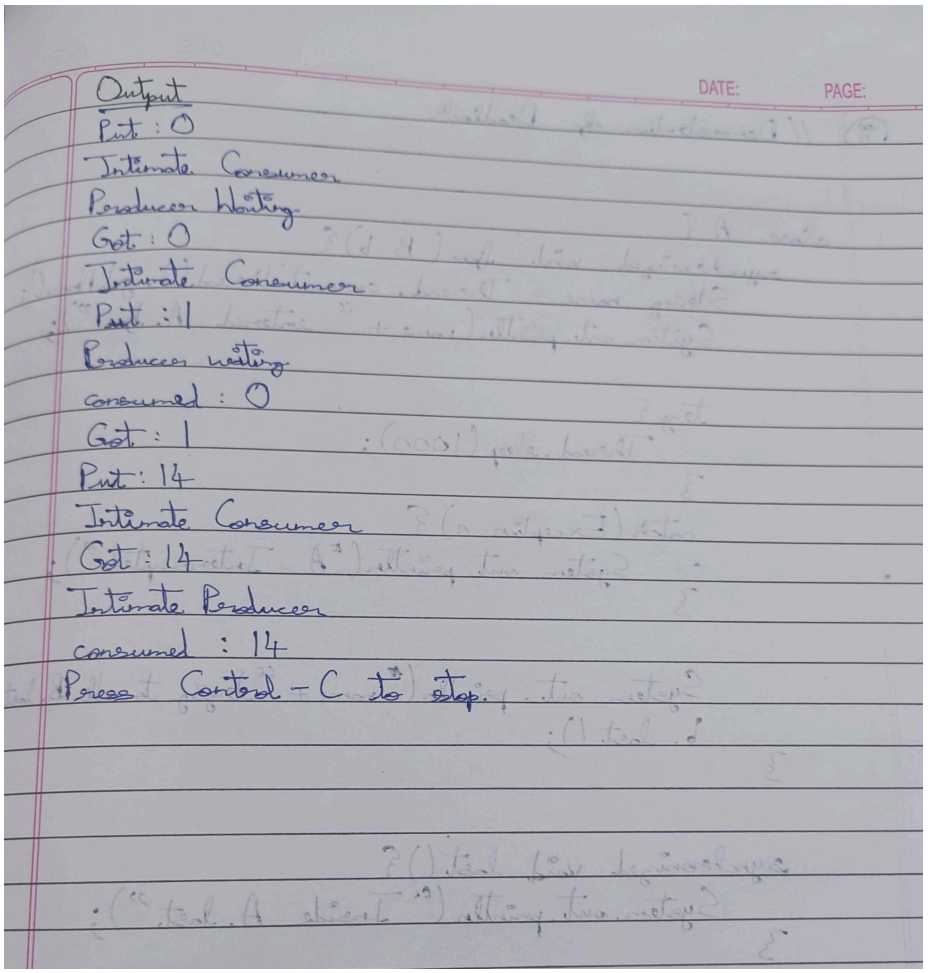
```
// Demonstration of Inter Process Communication
class Q{
  int n;
  boolean valueSet = false;
  synchronized int get(){
    while(!valueSet)
      try{
        System.out.println("Consumer waiting");
        wait();
      }
      catch(InterruptedException e){
        System.out.println("Interrupted");
      }

    System.out.println("Got: " + n);
    valueSet = false;
    System.out.println("Intimate Producer");
    notify();
    return n;
  }

  synchronized void put(int n){
    while(valueSet)
      try{
        System.out.println("Producer waiting");
        wait();
      }
```



```
DATE:  PAGE:
        catch (InterruptedException e){
            System.out.println("Interrupted");
        }

        this.n=n
        valueSet = true;
        System.out.println("Put: "+n);
        System.out.println("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 < 15){
            q.put(i++);
        }
    }
}
```

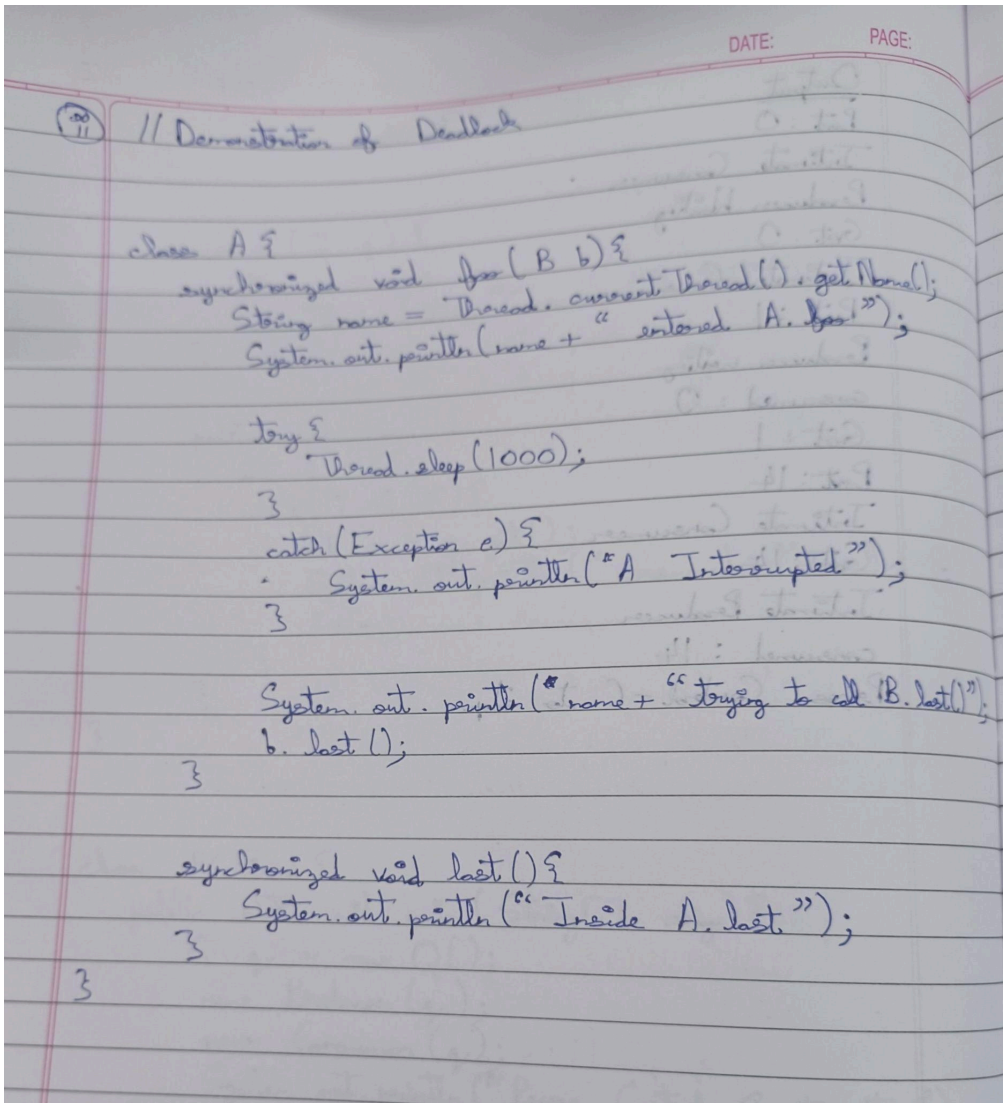


```
class Consumer implements Runnable {
    Q q;

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

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

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



Output

Put : 0
Intimate Consumer
Producer Waiting
Got : 0
Intimate Consumer
Put : 1
Producer waiting
consumed : 0
Got : 1
Put : 14
Intimate Consumer
Got : 14
Intimate Producer
consumed : 14
Press Control - C to stop.

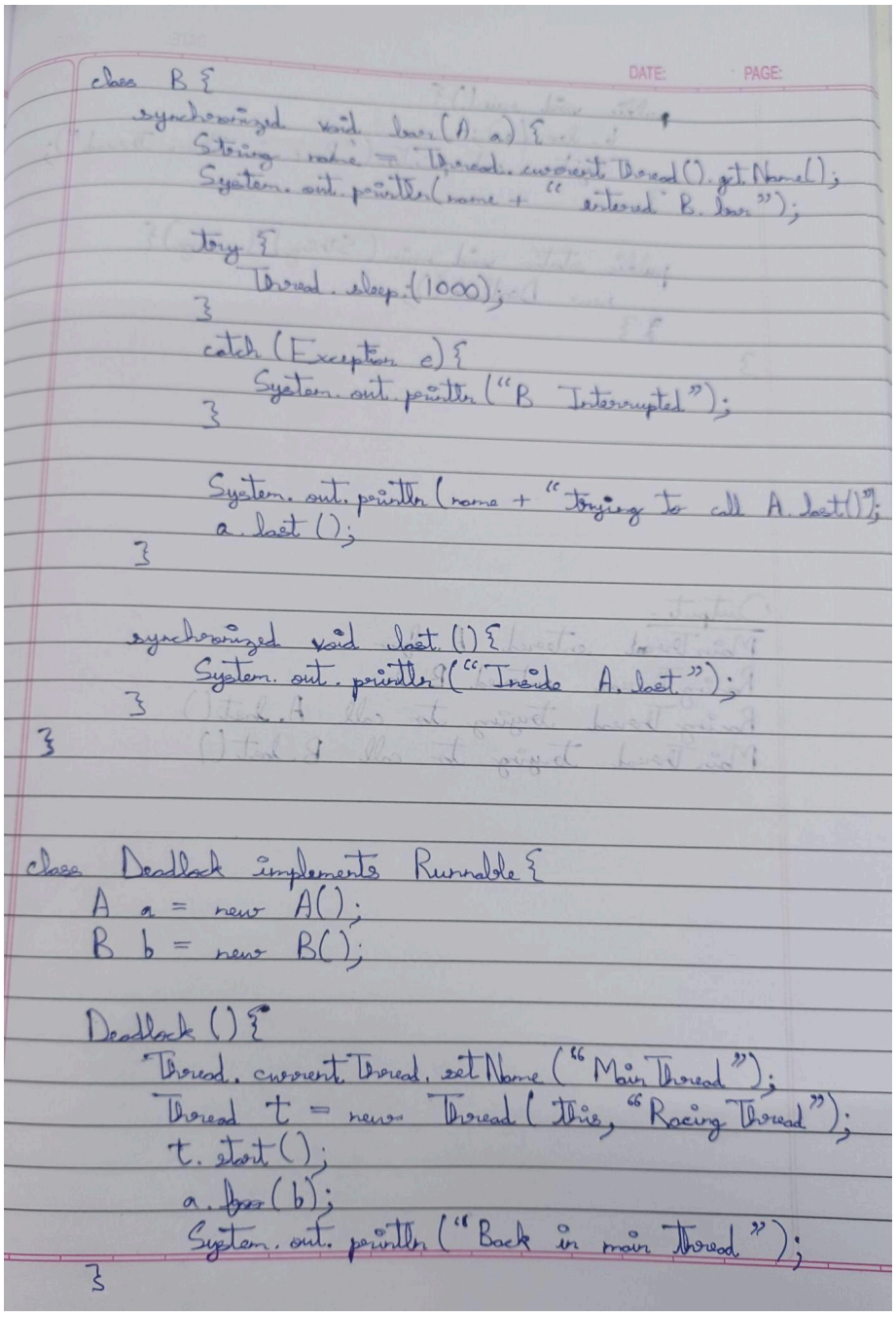


DATE: PAGE:

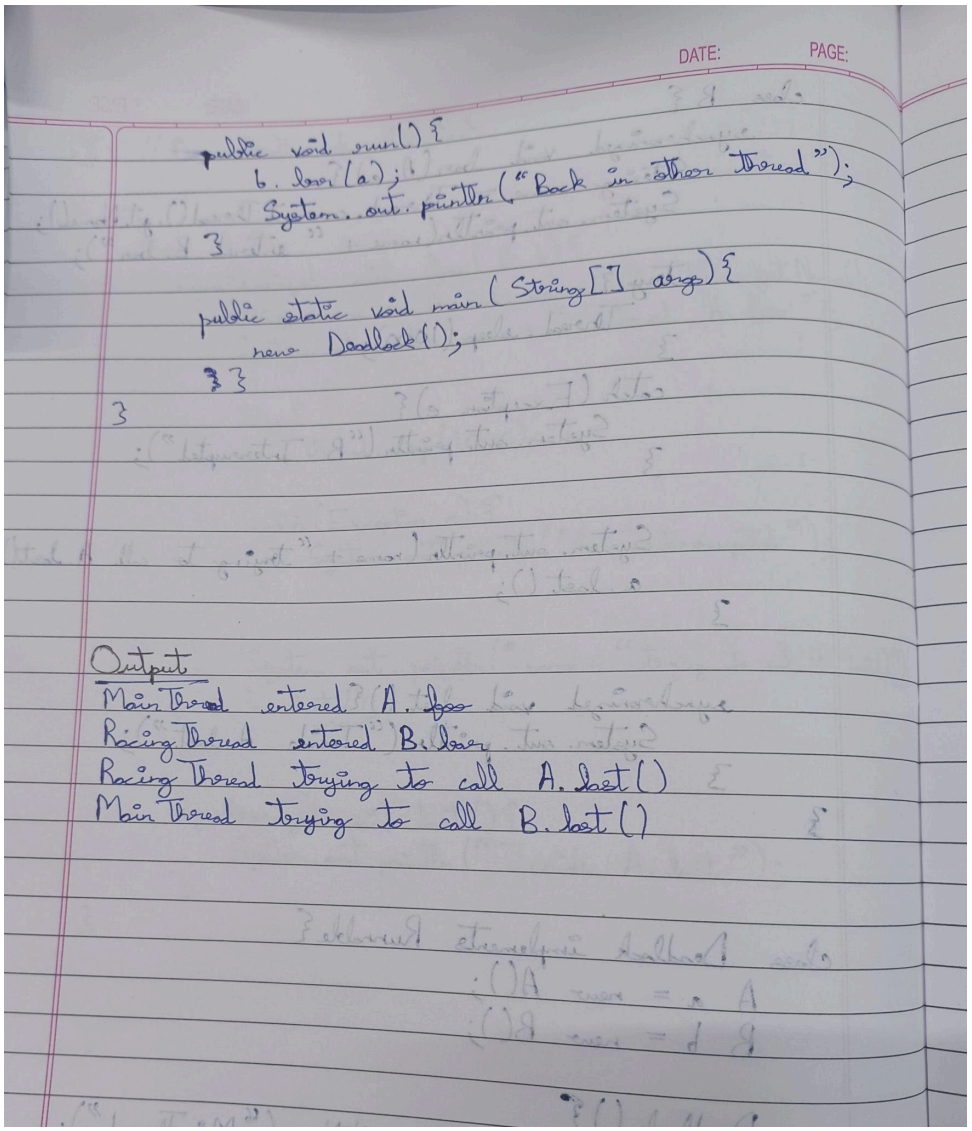
30
11

// Demonstration of Deadlock

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

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

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

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



```
class B {
    synchronized void bar(A a) {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");
        try {
            Thread.sleep(1000);
        }
        catch (Exception e) {
            System.out.println("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("Main Thread");
        Thread t = new Thread(this, "Racing Thread");
        t.start();
        a.bar(b);
        System.out.println("Back in main thread");
    }
```




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

ii. Demonstration of deadlock

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

```
      catch(Exception e) { System.out.println("A Interrupted"); }
      System.out.println(name + " trying to call B.last()"); b.last(); }
      synchronized void last() { System.out.println("Inside A.last"); }
}

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

```
Put: 0
Intimate Consumer
Producer waiting
Press Control-C to stop.
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
Got: 3
Intimate Producer
consumed:3
Consumer waiting
Put: 4
Intimate Consumer
Got: 4
Intimate Producer
consumed:4
Consumer waiting
```

```
Intimate Producer
consumed:9
Put: 10
Intimate Consumer
Got: 10
Intimate Producer
consumed:10
Consumer waiting
Put: 11
Intimate Consumer
Got: 11
Intimate Producer
consumed:11
Put: 12
Intimate Consumer
Producer waiting
Got: 12
Intimate Producer
consumed:12
Put: 13
Intimate Consumer
Got: 13
Intimate Producer
consumed:13
Put: 14
Intimate Consumer
Got: 14
Intimate Producer
consumed:14

Process finished with exit code 0
```

```
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last()
MainThread trying to call B.last()
```