VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Aryan Kolur Gowda (1BM23CS054)

in partial fulfilment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by **Aryan Kolur Gowda(1BM23CS054),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object-Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Dr Prasad G R Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:

https://github.com/AryanG-netizen/Java-Lab

Program 1:

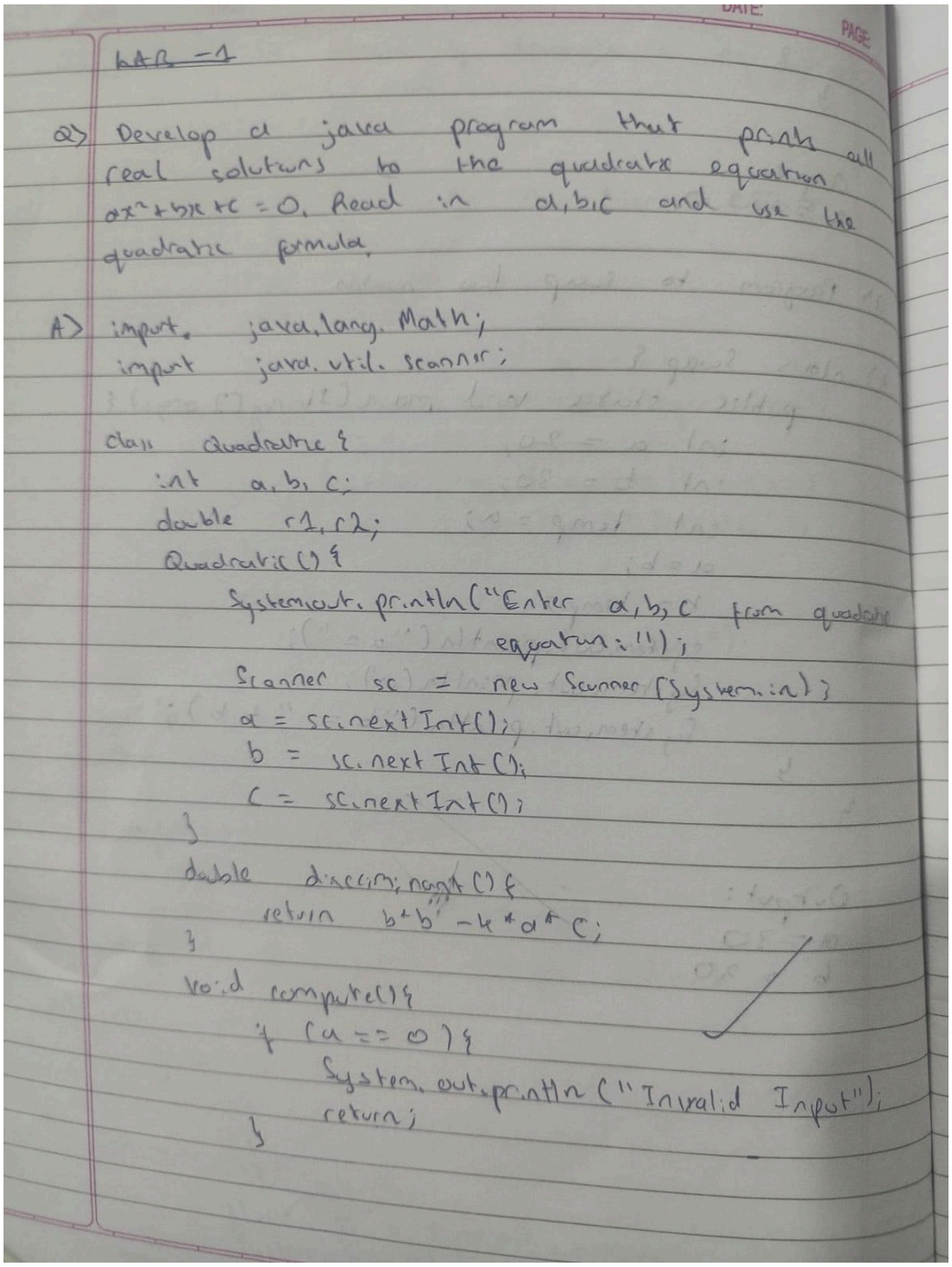
Develop a Java program that prints all real solutions to the quadratic equation ax2 +bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminant b2 -4ac is negative, display a message stating that there are no real solutions.

Code:

```
import java.util.*;
import java.lang.*;
class quadratic
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("enter values of a,b,c: \n");
double a,b,c;
double r1,r2,d;
a=sc.nextDouble();
b=sc.nextDouble();
c=sc.nextDouble();

if (a==0)
{
System.out.println("invalid!!");

}

d=Math.pow(b,2.0)-4.0*a*c;
```

```
if (d>0)
 {
r1=((-b+Math.sqrt(d)))/(double)(2.0*a);
r2=((-b-Math.sqrt(d)))/(double)(2.0*a);
System.out.println("roots are r1:"+r1+" r2:"+r2);
 }
else if (d==0)
 {
r1=(-b)/(2.0*a);
System.out.println("roots are equal: "+r1);
 }
else if (d<0)
 {
r1=(-b)/(2.0*a);
r2=Math.sqrt(-d)/(2.0*a);
System.out.println("roots are imaginary");
System.out.println("r1:"+r1+"+i"+r2+" and "+r1+"-i"+r2);
 }
else
System.out.println("enter valid items");
System.out.println("Aryan Gowda 1BM23CS054"); }
 }
```

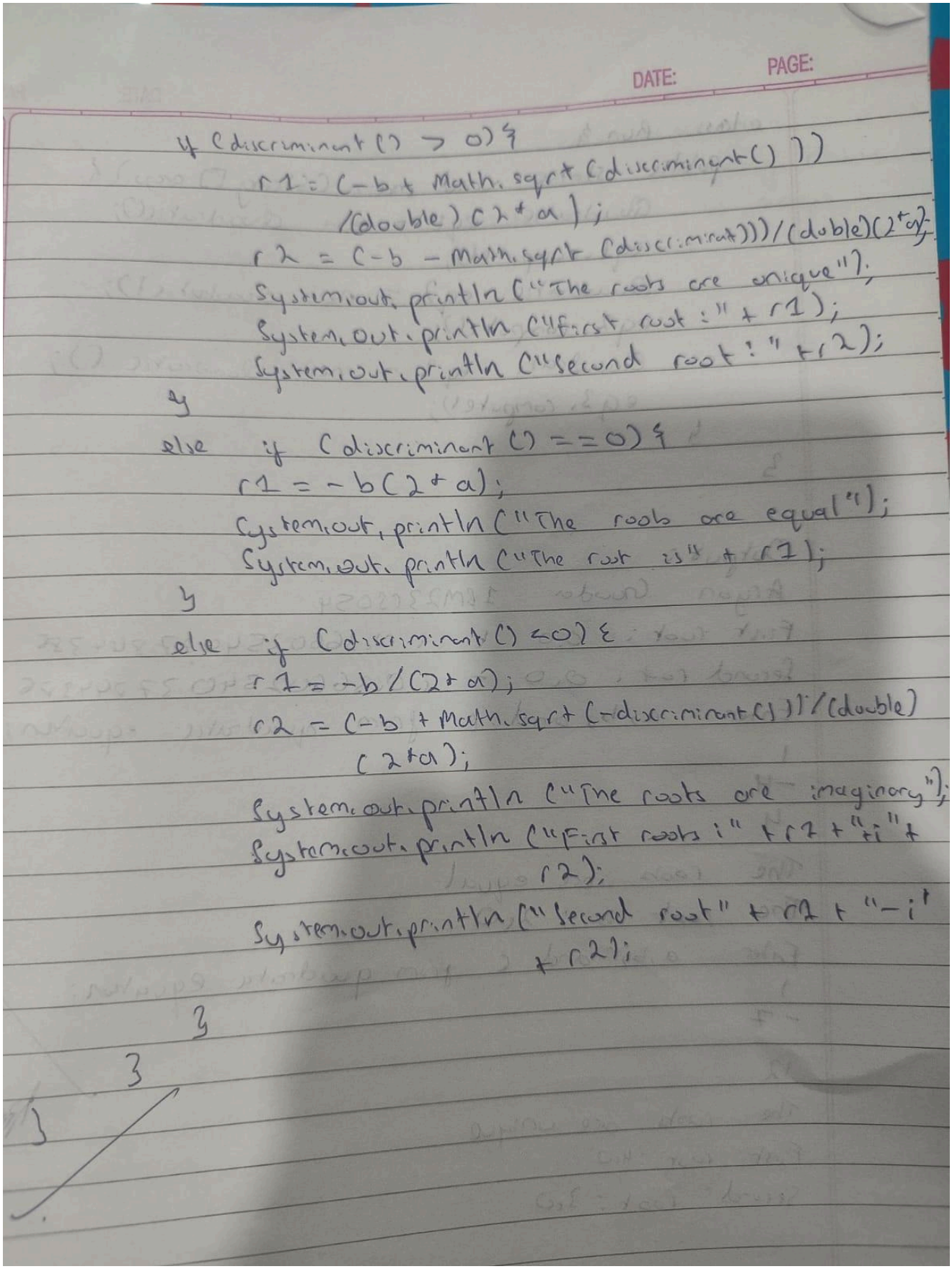
Notebook:



LAB - 1

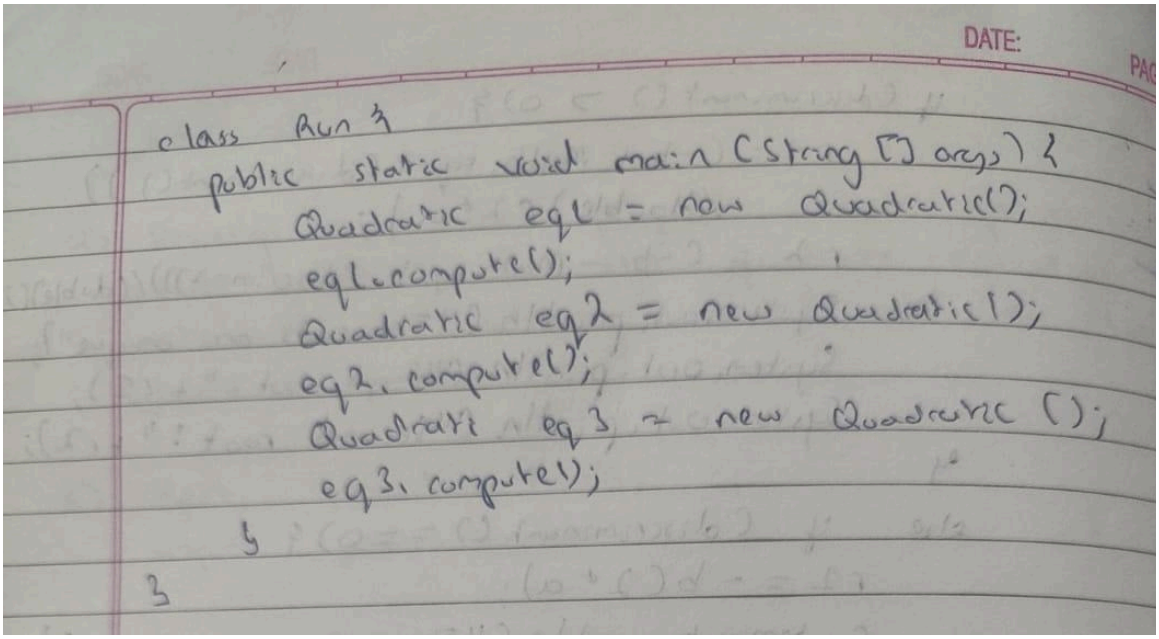
Q) Develop a java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a, b, c and use the quadratic formula.

A)
```
import java.lang.Math;
import java.util.Scanner;

class Quadratic {
    int a, b, c;
    double r1, r2;
    Quadratic() {
        System.out.println("Enter a, b, c from quadratic
                            equation:");
        Scanner sc = new Scanner(System.in);
        a = sc.nextInt();
        b = sc.nextInt();
        c = sc.nextInt();
    }
    double discriminant() {
        return b*b - 4*a*c;
    }
    void compute() {
        if (a == 0) {
            System.out.println("Invalid Input");
            return;
        }
```



DATE: PAGE:

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



```
class Run {
    public static void main (String [] args) {
        Quadratic eq1 = new Quadratic();
        eq1.compute();
        Quadratic eq2 = new Quadratic();
        eq2.compute();
        Quadratic eq3 = new Quadratic ();
        eq3.compute();
    }
}
```

Output:

```
C:\Users\Admin\Desktop>javac quadratic.java

C:\Users\Admin\Desktop>java quadratic
enter values of a,b,c:

1 -7 10
roots are r1:5.0  r2:2.0
```

```
Enter a,b,c:
4
-4
1
Equal roots.r1=r2=0.5
```

```
Enter a,b,c:
-1
-10
-30
Roots are imaginary
```

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.

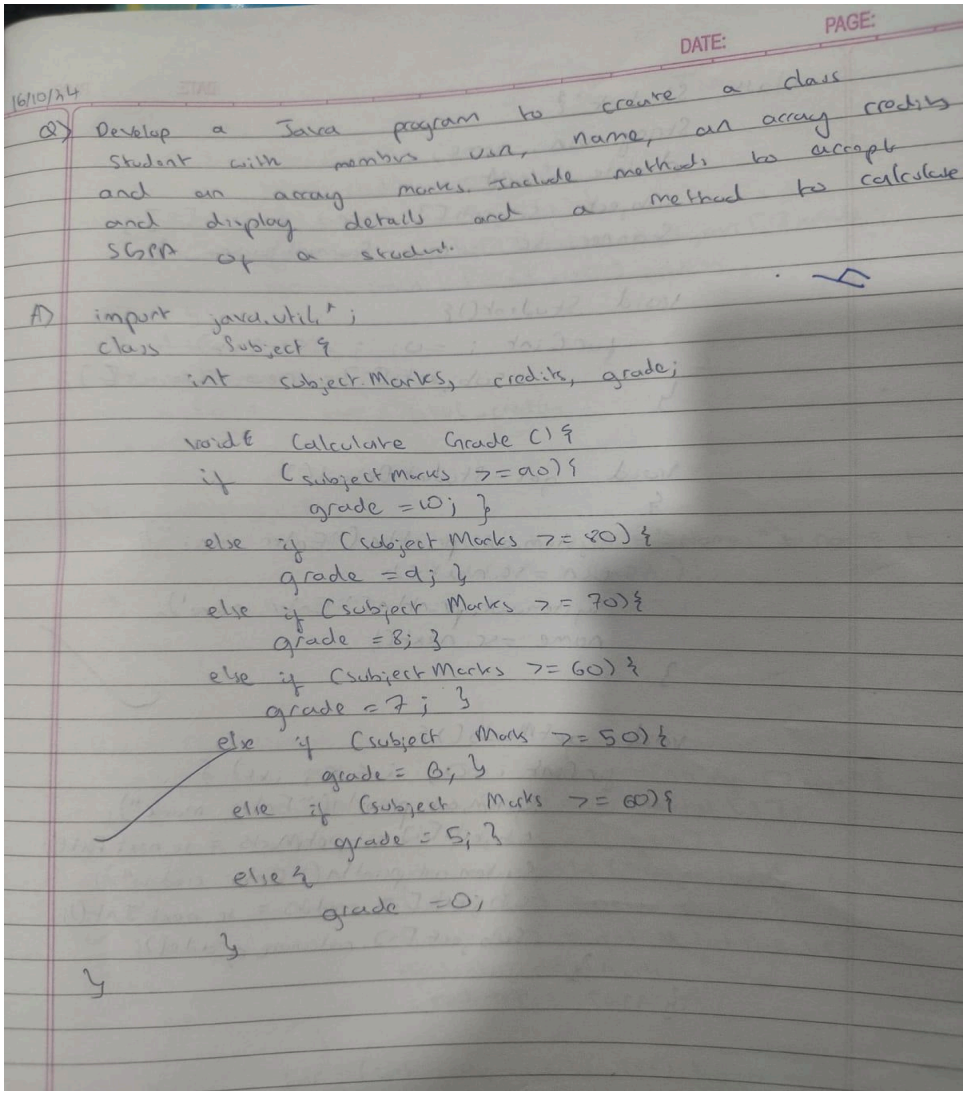
Code:

```
import java.util.*;

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

    // Method to get student details
    void getdetails() {
        System.out.println("Enter the usn and name");
        usn = sc.next();
        name = sc.next();

        System.out.println("Enter the marks");
        for (int i = 0; i < 8; i++) {
            marks[i] = sc.nextInt();
        }

        System.out.println("Enter the credits");
        for (int i = 0; i < 8; i++) {
            credit[i] = sc.nextInt();
        }
    }

    // Method to display student details
    void display() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);

        for (int i = 0; i < 8; i++) {
            System.out.println("Marks of subject " + (i + 1) + ": " + marks[i]);
```

```
        }

        System.out.println("SGPA: " + calculateSGPA());
    }

    // Method to calculate grade points based on marks
    double getgradepoint(int mark) {
        if (mark >= 90) return 10.0;
        else if (mark >= 80) return 9.0;
        else if (mark >= 70) return 8.0;
        else if (mark >= 60) return 7.0;
        else if (mark >= 50) return 6.0;
        else if (mark >= 40) return 5.0;
        else return 0.0;
    }

    // Method to calculate SGPA
    double calculateSGPA() {
        int totalCredits = 0;
        double gradePoints = 0;

        // Calculate total credits
        for (int i = 0; i < 8; i++) {
            totalCredits += credit[i];
        }

        // Calculate grade points
        for (int i = 0; i < 8; i++) {
            gradePoints += getgradepoint(marks[i]) * credit[i];
        }

        // Calculate SGPA
        return (gradePoints / totalCredits);
    }
}

class Student {
    public static void main(String args[]) {
        // Create an array of 3 Stud_details objects
        Stud_details s1[] = new Stud_details[3];
```

```
        // Initialize the array of student objects
        for (int j = 0; j < 3; j++) {
            s1[j] = new Stud_details();
        }

        // Input details for each student
        for (int j = 0; j < 3; j++) {
            System.out.println("Enter details of student " + (j + 1));
            s1[j].getdetails();
        }

        // Display details of each student
        for (int j = 0; j < 3; j++) {
            s1[j].display();
        }

        // Print a final line with a name and USN
        System.out.println("\nAryan Kolur Gowda 1BM23CS054");
    }
}
```

Notebook:

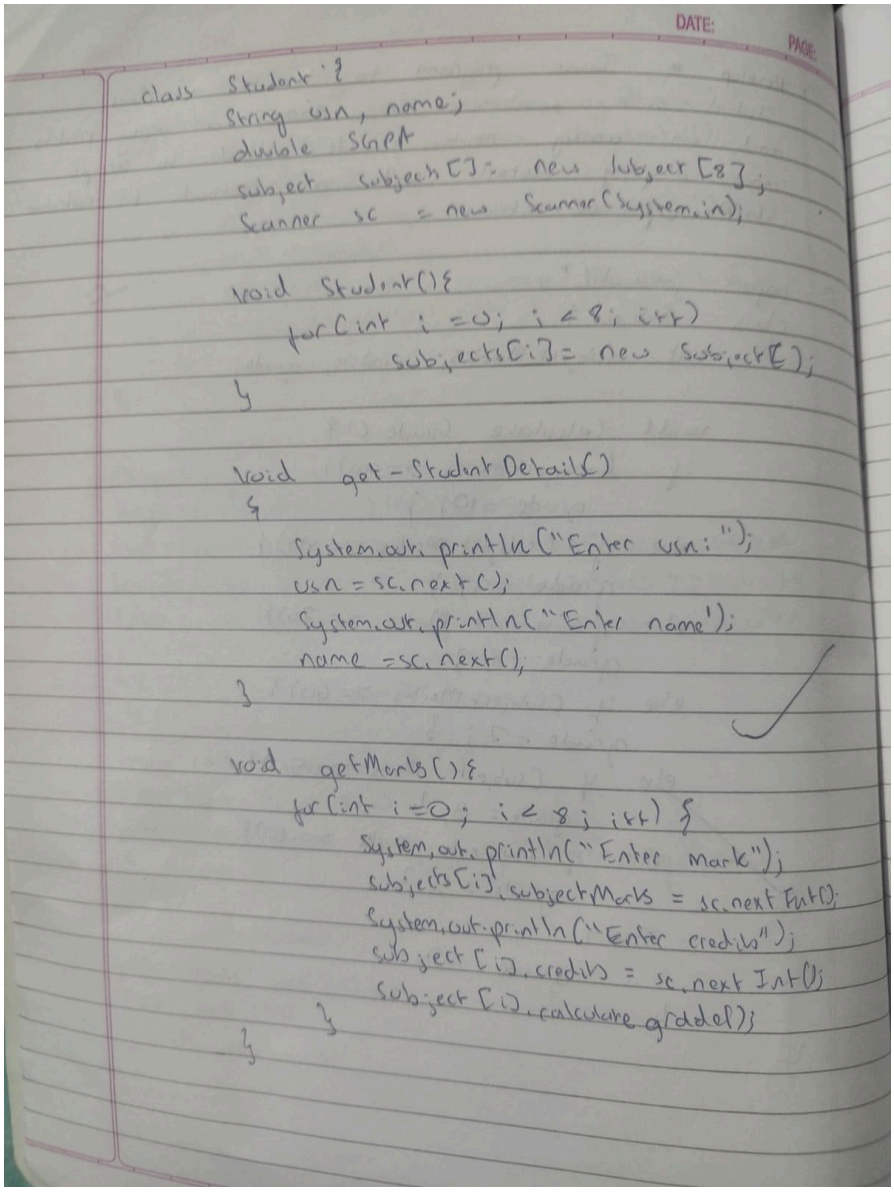


16/10/24

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.

A)

```
import java.util.*;
class Subject {
    int subjectMarks, credits, grade;

    void Calculate Grade () {
    if (subjectMarks >= 90) {
        grade = 10; }
    else if (subjectMarks >= 80) {
        grade = 9; }
    else if (subjectMarks >= 70) {
        grade = 8; }
    else if (subjectMarks >= 60) {
        grade = 7; }
    else if (subject Marks >= 50) {
        grade = 6; }
    else if (subject Marks >= 40) {
        grade = 5; }
    else {
        grade = 0;
    }
}
```



```
class Student {
    String usn, name;
    double sgpa
    subject subjects[] = new subject[8];
    Scanner sc = new Scanner(System.in);

    void Student(){
        for(int i =0; i < 8; i++)
            subjects[i] = new subject();
    }

    void getStudentDetail()
    {
        System.out.println("Enter usn:");
        usn = sc.next();
        System.out.println("Enter name");
        name = sc.next();
    }

    void getMarks(){
        for(int i=0; i < 8; i++) {
            System.out.println("Enter Mark");
            subjects[i].subjectMarks = sc.nextInt();
            System.out.println("Enter credits");
            subject[i].credits = sc.nextInt();
            subject[i].calculateGrade();
        }
    }
```

```
void SGPA(){
    duble score =0;
    int total_credits =0;
    for(int i=0; i<8; i++){
        score += subjects[i].credits * subjects[i].grade;
        total-credits += subject[i].credits;
    }
void SGPA(){
    double sre
    SGPA = score / total_credits;
}

void display(){
    System.out.println("USN" + USN + "Name:" + name +
                       "SGPA" + SGPA);
}

}

class StudDetails {
    public static void main(String[] args){
        Student students[] = new Student[3];
        for(int j=0; j<3; j++){
            System.out.println("Enter student Details");
            students[j] = new student();
            students[j] = getStudentDetails();
            student[j].getmarks();
            students[j].SGPA();
        }
        for(int i=0; i<3; i++){
            students[j].display();
        }
```

OUTPUT:

```
Enter Student USN:
45
Enter the subject marks:
42
Enter the respective credits:
1
Enter the subject marks:
24
Enter the respective credits:
4
Enter the subject marks:
7
Enter the respective credits:
5
Enter the subject marks:
7
Enter the respective credits:
67
Enter the subject marks:
8
Enter the respective credits:
1
Enter the subject marks:
99
Enter the respective credits:
4
Enter the subject marks:
99
Enter the respective credits:
4
Enter the subject marks:
99
Enter the respective credits:
4
USN: 45
SGPA=2.0
```

```
Enter Student USN:
1
Enter the subject marks:
90
Enter the respective credits:
2
Enter the subject marks:
99
Enter the respective credits:
4
Enter the subject marks:
100
Enter the respective credits:
4
Enter the subject marks:
92
Enter the respective credits:
1
Enter the subject marks:
98
Enter the respective credits:
4
Enter the subject marks:
89
Enter the respective credits:
2
Enter the subject marks:
97
Enter the respective credits:
4
Enter the subject marks:
99
Enter the respective credits:
3
USN: 1
SGPA=9.0
```

```
Enter Student USN:
2
Enter the subject marks:
34
Enter the respective credits:
2
Enter the subject marks:
67
Enter the respective credits:
2
Enter the subject marks:
46
Enter the respective credits:
4
Enter the subject marks:
22
Enter the respective credits:
9
Enter the subject marks:
34
Enter the respective credits:
7
Enter the subject marks:
34
Enter the respective credits:
6
Enter the subject marks:
45
Enter the respective credits:
1
Enter the subject marks:
34
Enter the respective credits:
1
USN: 2
SGPA=3.0
```

PROGRAM-3:

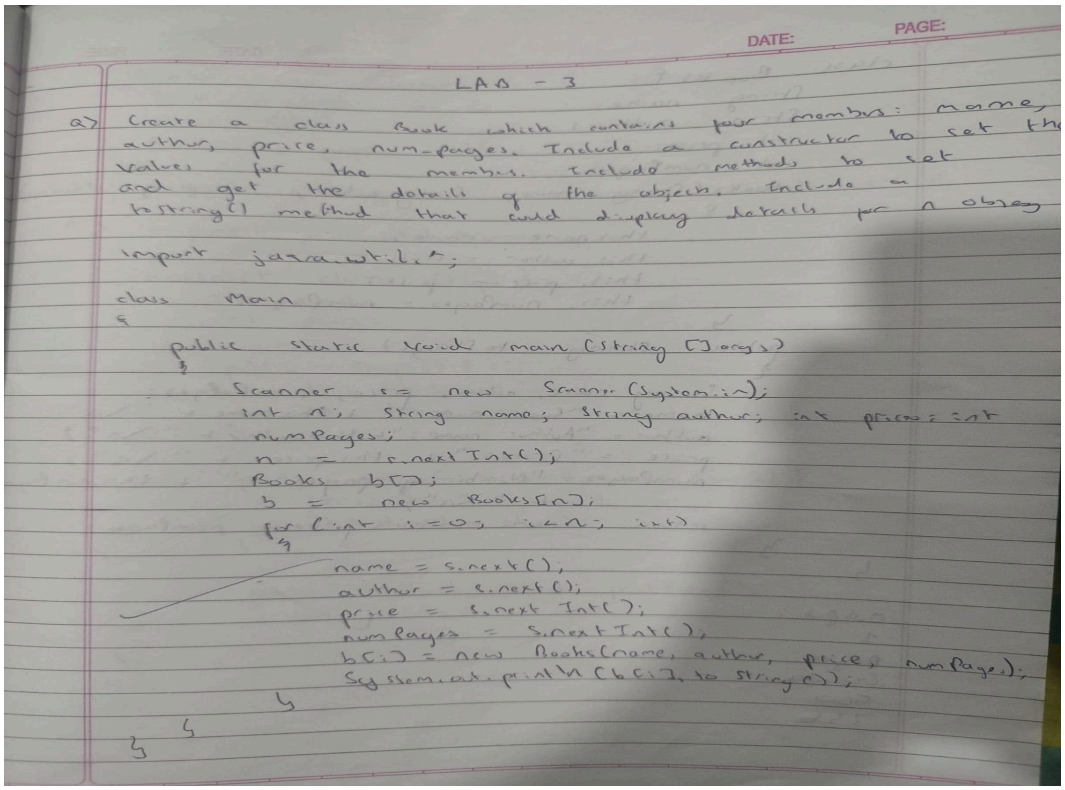
Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program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

Code:

```
import java.util.*;
class Book{
String name;
String author;
int price;
int num_pages;
Book(String name,String author,int price,int num_pages){
this.name=name;
this.author=author;
this.price=price;
this.num_pages=num_pages;
}
public String toString(){
return "name:"+name+",author:"+author+",price:"+price+",no of pages:"+num_pages;
}
}
class BookDetails{
public static void main(String[] args){
13
Scanner sc=new Scanner(System.in);
System.out.println("Enter no of books:");
int n=sc.nextInt();
Book[] book=new Book[n];
System.out.println("Enter name,author,price and no of pages for each book
respectively:");
for(int i=0;i<n;i++){
System.out.println("Enter details of book "+(i+1)+":");
```

```
String name=sc.next();
String author=sc.next();
int price=sc.nextInt();
int num_pages=sc.nextInt();
book[i]=new Book(name,author,price,num_pages);
}
for(int i=0;i<n;i++){
System.out.println("Book "+(i+1)+":");
System.out.println(book[i].toString());
}
}
}
```

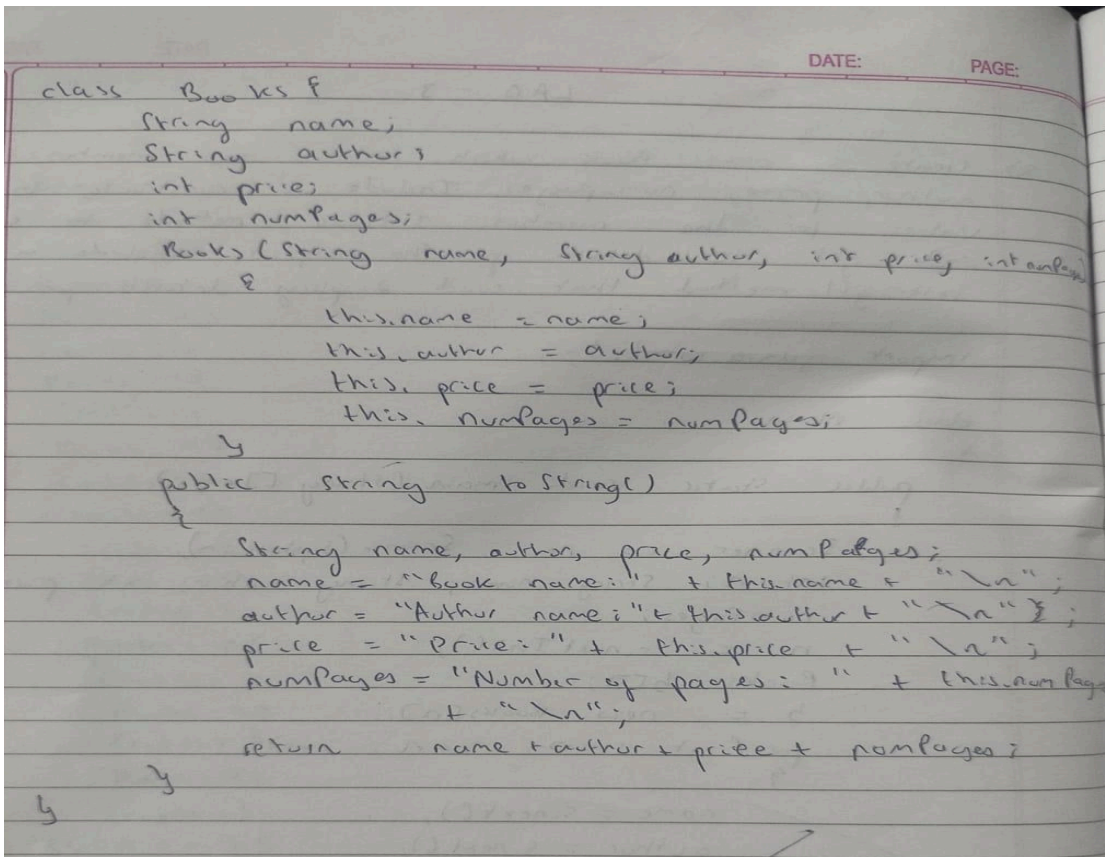
Notebook:



LAB - 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 details for an object

```
import java.util.*;

class Main
{
    public static void main (String [] args)
    {
        Scanner s = new Scanner (System.in);
        int n; String name; String author; int price; int
        numPages;
        n = s.nextInt();
        Books b[];
        b = new Books[n];
        for (int i=0; i<n; i++)
        {
            name = s.next();
            author = s.next();
            price = s.nextInt();
            numPages = s.nextInt();
            b[i] = new Books(name, author, price, numPages);
            System.out.println(b[i].toString());
        }
    }
}
```



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

OUTPUT:

```
Enter the Number of Books: 3
Enter name of book 1: Divergent
Enter author of book 1: R.Veronica
Enter price of book 1: 500
Enter number of pages in book 1: 390
Enter name of book 2: Emma
Enter author of book 2: J.Austen
Enter price of book 2: 399
Enter number of pages in book 2: 300
Enter name of book 3: Rebecca
Enter author of book 3: D.Maurier
Enter price of book 3: 699
Enter number of pages in book 3: 469

Book Details:
Book Name: Divergent
Author Name: R.Veronica
Price: 500
Number of Pages: 390

Book Name: Emma
Author Name: J.Austen
Price: 399
Number of Pages: 300

Book Name: Rebecca
Author Name: D.Maurier
Price: 699
Number of Pages: 469
```

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.

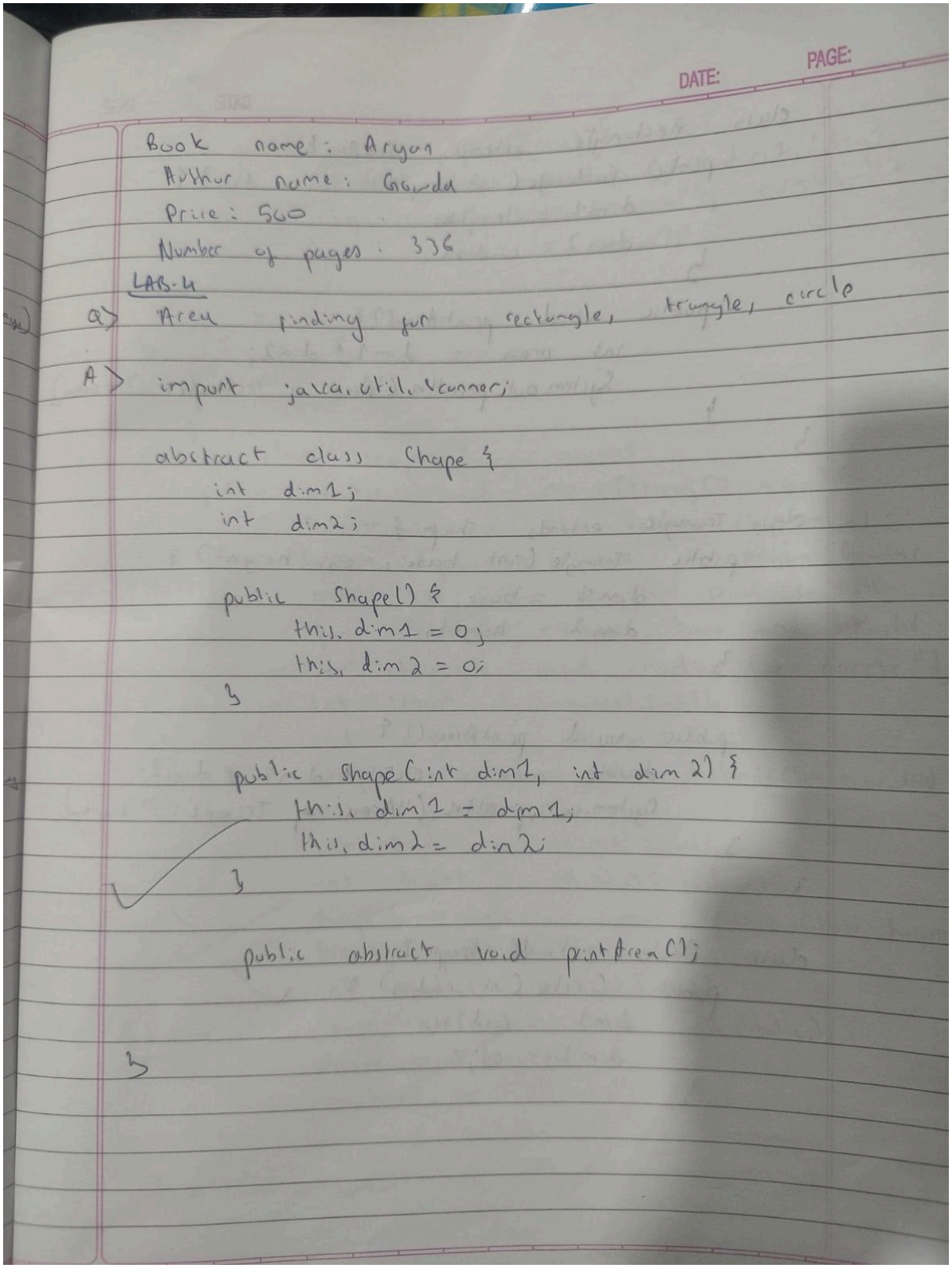
Code:

```
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 in = new Scanner(System.in);
        System.out.println("Enter length and width for Rectangle:");
 int length = in.nextInt();
 int width = in.nextInt();
```

```
 Shape rectangle = new Rectangle(length, width);
rectangle.printArea();

 System.out.println("Enter base and height for Triangle:");

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

 System.out.println("Enter radius for Circle:");

 int radius = in.nextInt();
 Shape circle = new Circle(radius);
 circle.printArea();
 System.out.println("Aryan Gowda 1BM23CS054");


 in.close();
 }
}
```

Notebook:



DATE: PAGE:

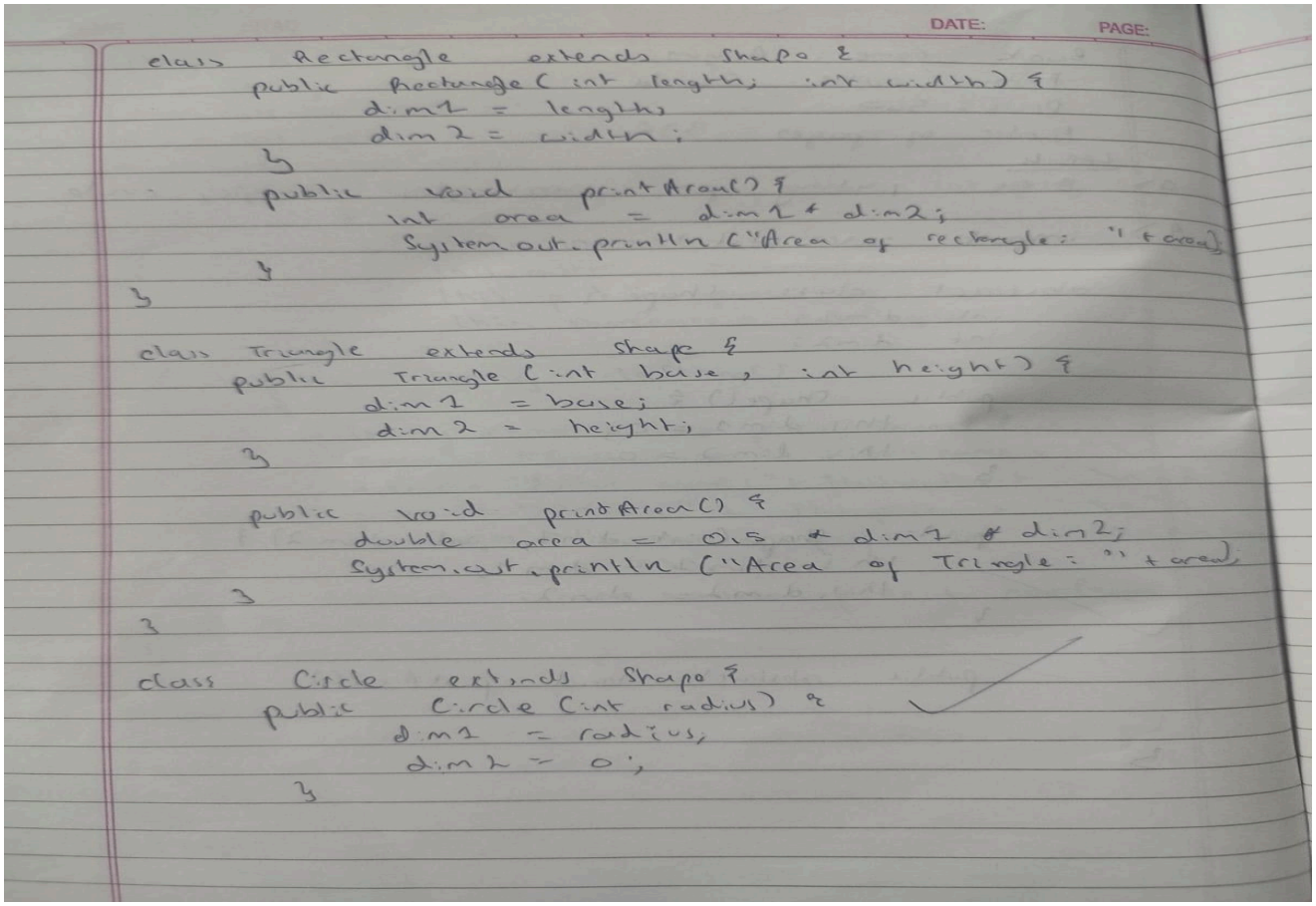
Book name : Aryan
Author name : Gowda
Price : 560
Number of pages : 336

LAB-4

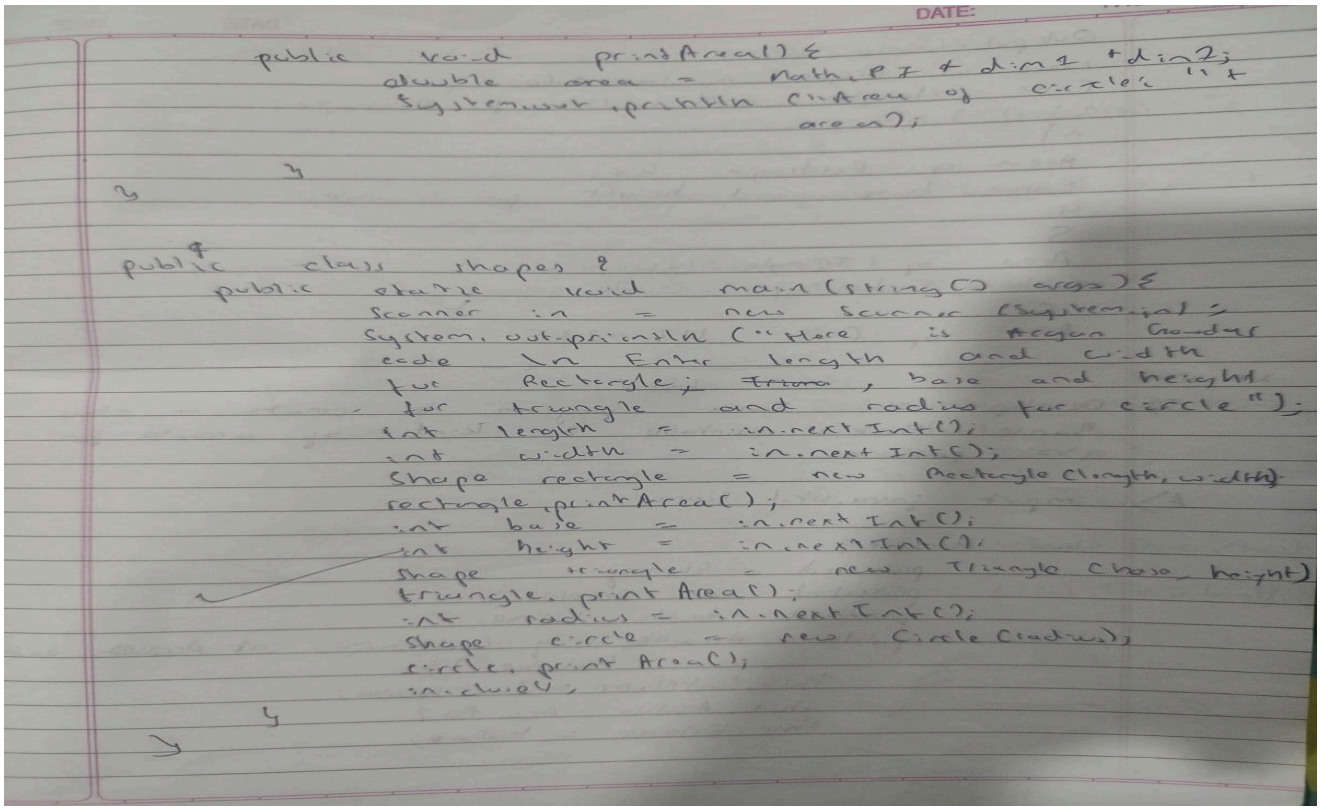
Q) Area finding for rectangle, triangle, circle

A)

```
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();

}
```

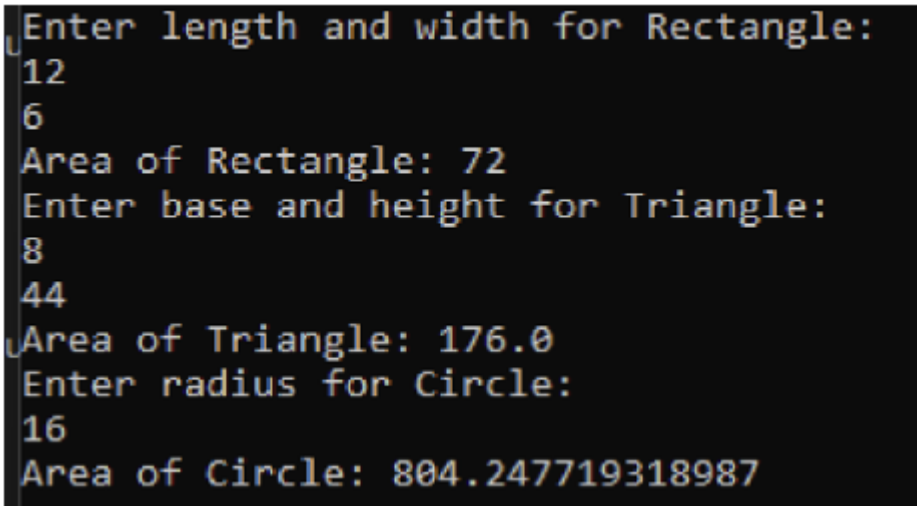


```
class Rectangle extends Shape {
    public Rectangle(int length, int width) {
        dim1 = length;
        dim2 = width;
    }
    public void printArea() {
        int area = dim1 * dim2;
        System.out.println("Area of rectangle: " + area);
    }
}

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 * 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("Here is Area Calculator
        code \n Enter length and width
        for Rectangle, base and height
        for triangle and radius for circle");
        int length = in.nextInt();
        int width = in.nextInt();
        Shape rectangle = new Rectangle(length, width);
        rectangle.printArea();
        int base = in.nextInt();
        int height = in.nextInt();
        Shape triangle = new Triangle(base, height);
        triangle.printArea();
        int radius = in.nextInt();
        Shape circle = new Circle(radius);
        circle.printArea();
        in.close();
    }
}
```

Output:


Enter length and width for Rectangle:
12
6
Area of Rectangle: 72
Enter base and height for Triangle:
8
44
Area of Triangle: 176.0
Enter radius for Circle:
16
Area of Circle: 804.247719318987

Program-5:

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks: a) Accept deposit from customer and update the balance. b) Display the balance. c) Compute and deposit interest d) Permit withdrawal and update the balance Check for the minimum balance, impose penalty if necessary and update the balance.

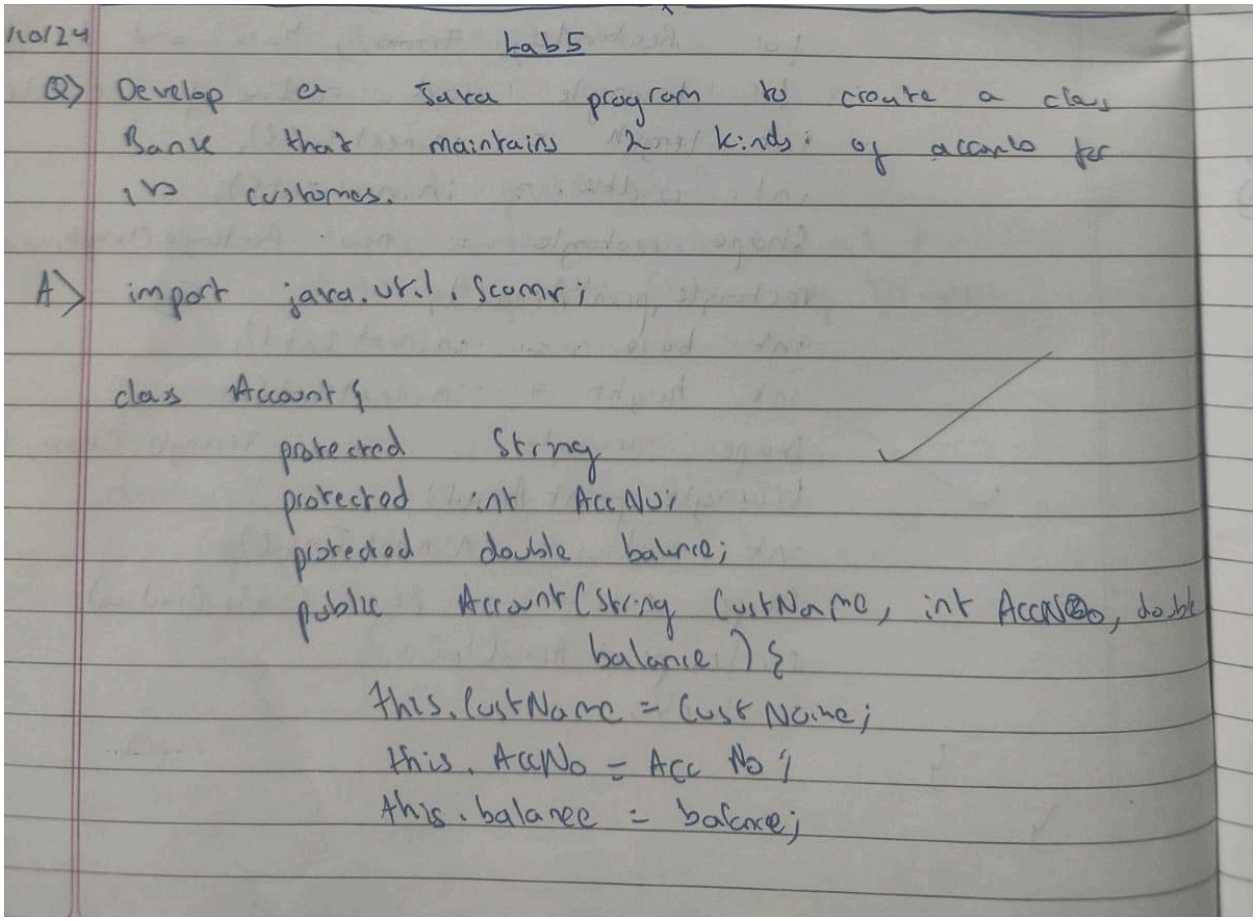
Code:

```
class Account {
    String cname;
    int accno;
    String acctype;
    double balance;

    // Constructor
    Account(String cName, int accNo, String accType, double initialBalance) {
        cname = cName;
        accno = accNo;
        acctype = accType;
        balance = initialBalance;
    }

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

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

class SavAcct extends Account {
    final double ir = 0.05; // Interest rate

    // Constructor for Savings Account
    SavAcct(String cName, int accNo, double initialBalance) {
        super(cName, accNo, "Savings", initialBalance);
    }

    // Method to add interest
    void getInterest() {
        double interest = balance * ir;
        balance += interest;
        System.out.println("Interest added: " + interest);
    }

    // Method to withdraw amount
    void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Not enough balance");
        } else {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        }
    }
}

class CurAcct extends Account {
    final double minBalance = 500.0; // Minimum balance for current account
    final double serviceCharge = 50.0; // Service charge for low balance

    // Constructor for Current Account
    CurAcct(String cName, int accNo, double initialBalance) {
        super(cName, accNo, "Current", initialBalance);
```

```
    }

    // Method to check minimum balance and apply service charge
    void checkMinimumBalance() {
        if (balance < minBalance) {
            balance -= serviceCharge;
            System.out.println("Service charge imposed: " + serviceCharge);
        }
    }

    // Method to withdraw amount
    void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient balance");
        } else {
            balance -= amount;
            checkMinimumBalance();
            System.out.println("Amount withdrawn: " + amount);
        }
    }
}

public class Bank {
    public static void main(String args[]) {
        // Creating instances of Savings and Current Accounts
        SavAcct sa = new SavAcct("John", 101, 2000);
        CurAcct ca = new CurAcct("Sejil", 102, 1000);

        System.out.println("Savings account:");
        sa.deposit(500);
        sa.getInterest();
        sa.withdraw(200);
        sa.displayBalance();

        System.out.println("\nCurrent account:");
        ca.deposit(200);
        ca.withdraw(800);
        ca.displayBalance();
    }
}
```

Notebook:



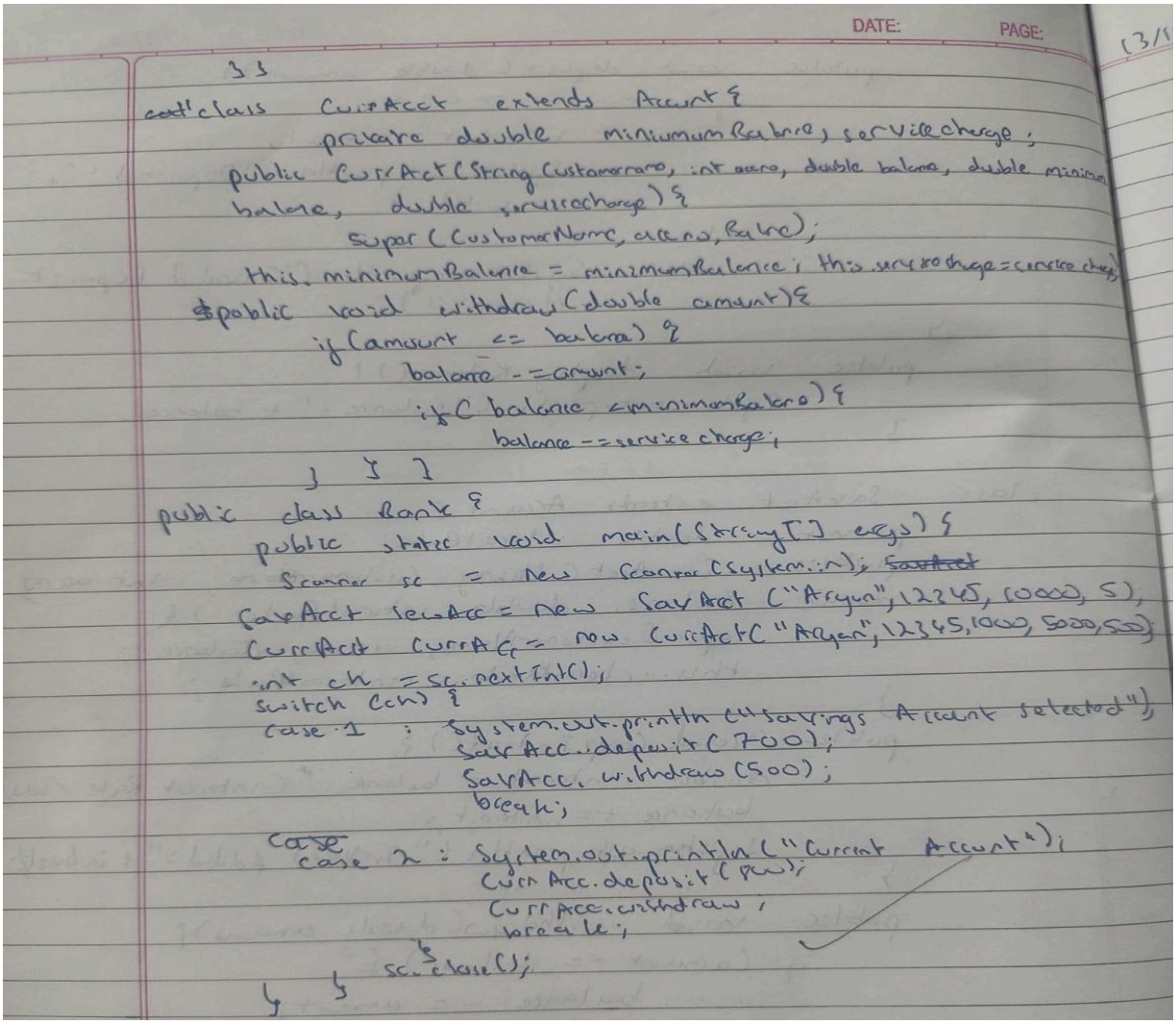
11/12/24

Lab 5

Q) Develop a Java program to create a class Bank that maintains 2 kinds of accounts for 'n' customers.

A)

```
import java.util.Scanner;

class Account {
    protected String
    protected int AccNo;
    protected double balance;
    public Account(String CustName, int AccNo, double
                   balance ) {
        this.CustName = CustName;
        this.AccNo = AccNo;
        this.balance = balance;
```



```
}}
cont'class CurrAcct extends Account{
        private double minimumBalance, serviceCharge;
    public CurrAct(String CustomerName, int accno, double balance, double minimum
    balance, double serviceCharge){
            super(CustomerName, accno, Balance);
        this.minimumBalance = minimumBalance; this.serviceCharge = serviceCharge;
    public void withdraw(double amount){
            if(amount <= balance){
                balance -= amount;
                if(balance < minimumBalance){
                    balance -= serviceCharge;
        } } }
public class Bank{
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in); SavAcct
    SavAcct savAcc = new SavAct("Aryan", 12345, 10000, 5);
    CurrAct currAc = new CurrAct("Aryan", 12345, 1000, 5000, 500);
    int ch = sc.nextInt();
    switch(ch){
    case 1 : System.out.println("Savings Account selected");
             savAcc.deposit(700);
             savAcc.withdraw(500);
             break;

    case
    case 2 : System.out.println("Current Account");
             currAcc.deposit(pw);
             currAcc.withdraw;
             break;
        sc.close();
}}
```

Output:

```
Choose Account Type:
1. Savings Account
2. Current Account
1
Savings Account Selected
Deposited: 500.0
Interest added: 75.0
Withdrawn: 300.0
Balance: 1275.0
```

```
Choose Account Type:
1. Savings Account
2. Current Account
2
Current Account Selected
Deposited: 500.0
Withdrawn: 1800.0
Balance: 700.0
```

Program-6:

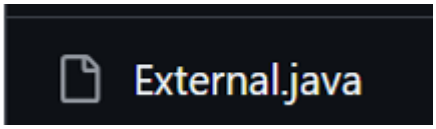
Create a package CIE which has two classes- Student and Internals. The class Personal has members like usn, name, sem. The class internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External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

CODE

```
package CIE;

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

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

```
package CIE;

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

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

Path: Week6/SEE:



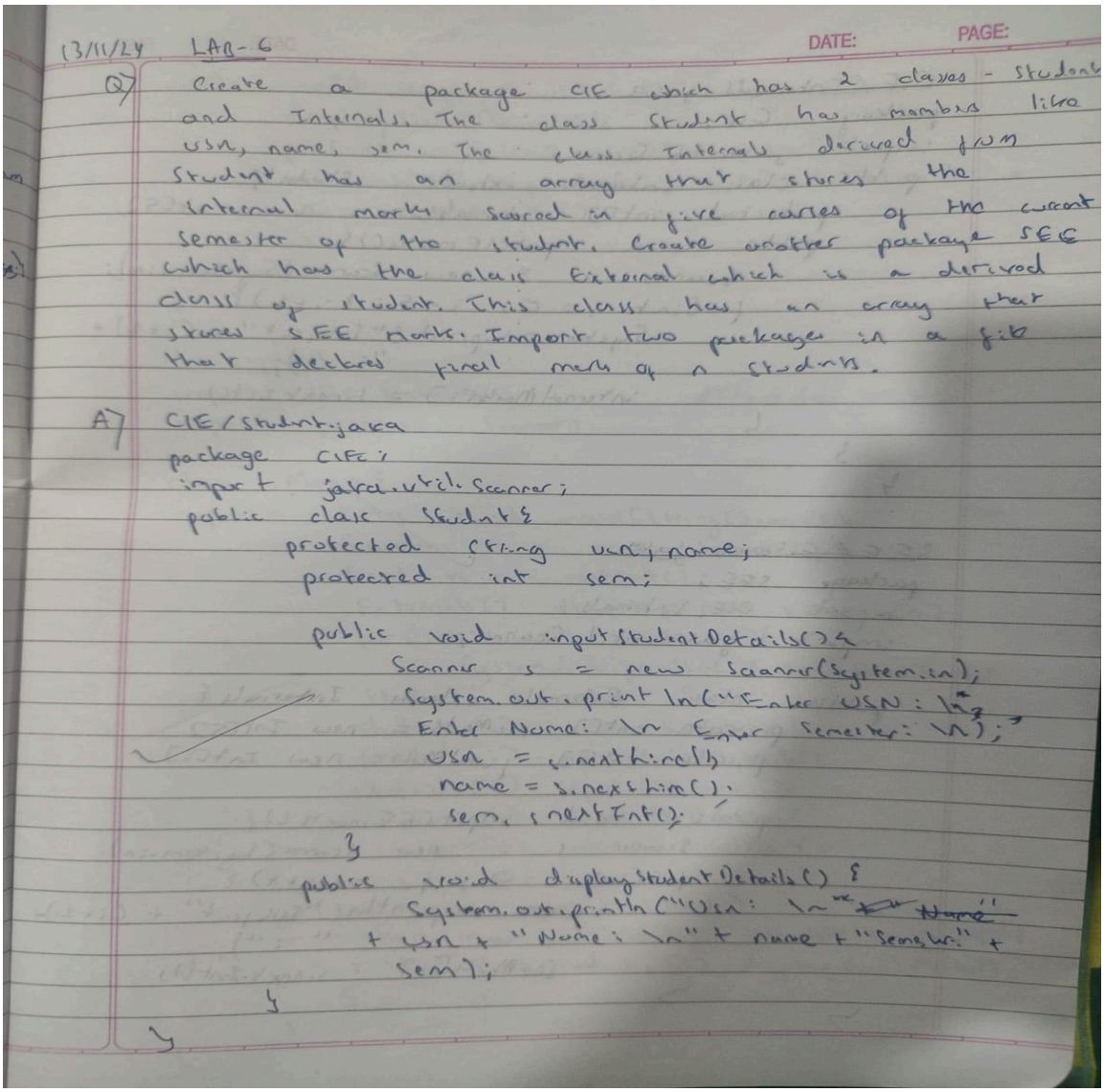

```
package SEE;
import CIE.Student;

public class External extends Student {
 public int[] seeMarks;

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

Path: Week6:

```
import CIE.Internals;
import SEE.External;
import java.util.Scanner;

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

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

 Internals[] internalStudents = new Internals[n];
 External[] externalStudents = new External[n];

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

 System.out.print("USN: ");
 String usn = scanner.next();
```

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

 System.out.print("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();
 }

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



 // Creating objects for Internals and External
 internalStudents[i] = new Internals(usn, name, sem, internalMarks);
externalStudents[i] = new External(usn, name, sem, seeMarks);  }

 // Display final marks
 System.out.println("\nFinal Marks of Students:");
 for (int i = 0; i < n; i++) {
 System.out.println("\nStudent " + (i + 1) + " - USN: " +
internalStudents[i].usn);
 for (int j = 0; j < 5; j++) {
 int finalMark = (internalStudents[i].internalMarks[j] +
(externalStudents[i].seeMarks[j]) / 2);
 System.out.println("Course " + (j + 1) + " Final Mark: " + finalMark);  }
 }
 System.out.println("Aryan Kolur Gowda 1BM23CS054");
scanner.close();
 }
 }
```

Notebook:



13/11/24 LAB-6 DATE: PAGE:

Q) Create a package CIE which has 2 classes - Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of student. This class has an array that stores SEE marks. Import two packages in a file that declares final marks of a student.

A)

```
CIE/Student.java
package CIE;
import java.util.Scanner;
public class Student {
    protected String usn, name;
    protected int sem;

    public void inputStudentDetails() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter USN: \n
        Enter Name: \n Enter Semester: \n");
        usn = s.nextLine();
        name = s.nextLine();
        sem = s.nextInt();
    }
    public void displayStudentDetails() {
        System.out.println("USN: \n" + usn + "Name: \n" + name + "Semester:" +
        sem);
    }
}
```

CIE / Internals.java

```
package CIE;
import java.util.Scanner;
public class Internals extends Student{
    protected int[] internalMarks = new int[5];
    public void inputCIEmarks(){
        Scanner s = new Scanner(System.in);
        for(int i = 0; i < 5; i++){
            System.out.print("Subject" + (i+1) +
                ": ");
            internalMarks[i] = s.nextInt();
        }
    }
}
```

SEE / Externals.java

```
package SEE;
import CIE.Internals;
import java.util.Scanner;

public class Externals extends Internals {
    private int[] seeMarks = new int[5];
    private int[] finalMarks = new int[5];

    public void inputSEEmarks(){
        Scanner s = new Scanner(System.in);
        for(int i=0; i<5; i++){
            System.out.println("Subject" + (i+1)
                + ":");
            seeMarks[i] = s.nextInt();
        }
    }
```

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

Main.java

```
import SEE.Externals;          //Import SEE Externals
import java.util.Scanner;

class Main{
    public static void main(String[] args){
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        Externals[] students = new Externals[n];
        for(int i=0; i<n; i++){
            System.out.println("Enter Details for
            Student" + (i+1) + " :");
            students[i] = new Externals();
            students[i].inputStudentDetails();
            students[i].inputCIEmarks();
            students[i].inputSEEmarks();
            students[i].calculateFinalMarks();
        }
        for(int i=0; i<n; i++){
            System.out.println("\n Student:" + (i+1) +
            " :" + students[i].displayFinalMarks();
        }
    }
}
```

Output:

```
Enter number of students: 2

Enter details for student 1:
USN: 1
Name: A
Semester: 3
Enter internal marks for 5 courses:
45 49 48 47 46
Enter SEE marks for 5 courses:
90 98 97 94 93

Enter details for student 2:
USN: 2
Name: B
Semester: 2
Enter internal marks for 5 courses:
34 35 2 23 49
Enter SEE marks for 5 courses:
90 98 45 3 46

Final Marks of Students:

Student 1 - USN: 1
Course 1 Final Mark: 90
Course 2 Final Mark: 98
Course 3 Final Mark: 96
Course 4 Final Mark: 94
Course 5 Final Mark: 92

Student 2 - USN: 2
Course 1 Final Mark: 79
Course 2 Final Mark: 84
Course 3 Final Mark: 24
Course 4 Final Mark: 24
Course 5 Final Mark: 72
```

Program-7:

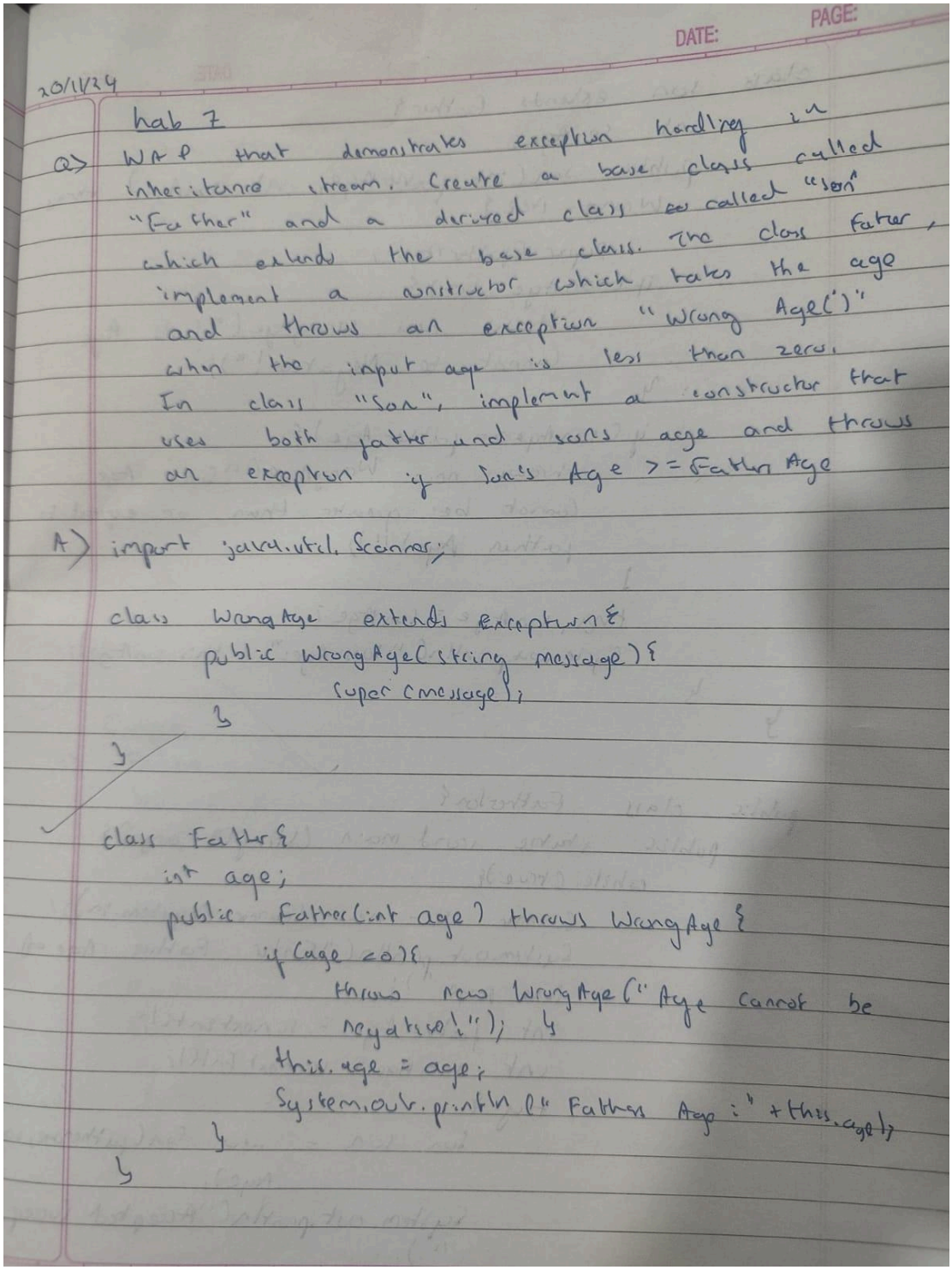
Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age=father’s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

Code:

```
import java.util.Scanner;

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

class Father {
 int age;

 public Father(int age) throws WrongAge {
 if (age < 0) {
 throw new WrongAge("Age Cannot be Negative");
 }
 this.age = age;
 System.out.println("Father's Age: " + this.age);
 }
}

class Son extends Father {
 int sonAge;

 public Son(int fatherAge, int sonAge) throws WrongAge {
super(fatherAge);
 if (sonAge < 0) {
 throw new WrongAge("Son's Age Cannot be Negative");  }
 if (sonAge >= fatherAge) {
```

```
 throw new WrongAge("Son's Age Cannot be Greater than or Equal to

Father's Age");
 }
 this.sonAge = sonAge;
 System.out.println("Son's Age: " + this.sonAge);  }
}

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

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

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

 try {
 Son son = new Son(fatherAge, sonAge);   }
catch (WrongAge e){
 System.out.println("Exception: " + e.getMessage());  }
 System.out.println("Aryan Gowda 1BM23CS054");
scanner.close();
 }
}
```

Notebook:



20/11/24

DATE: PAGE:

lab 7

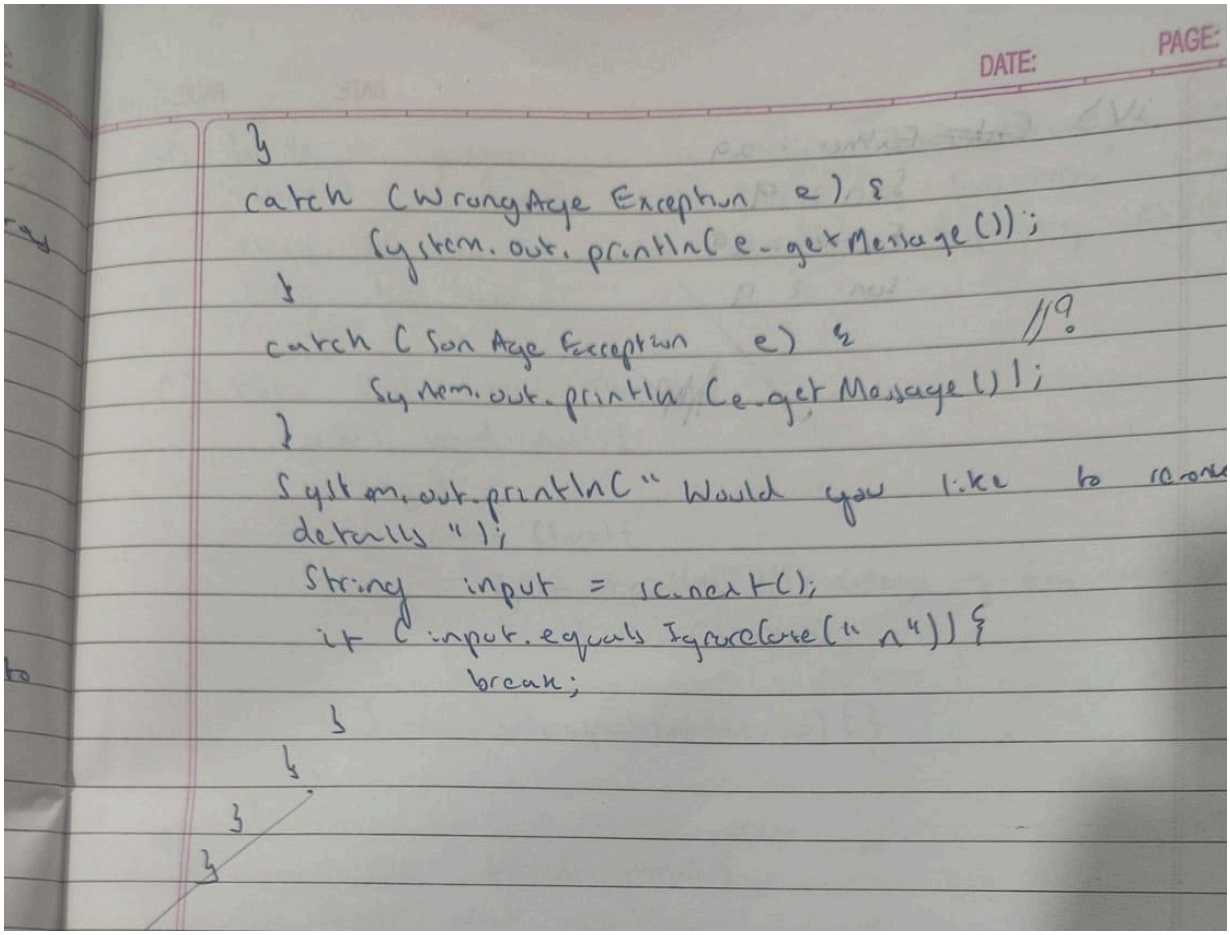
Q> WAP that demonstrates exception handling in inheritance stream. Create a base class called "Father" and a derived class so called "son" which extends the base class. The class Father, implement a constructor which takes the age and throws an exception "Wrong Age()" when the input age is less than zero. In class "Son", implement a constructor that uses both father and sons age and throws an exception if Son's Age >= Father Age

A>

```
import java.util.Scanner;

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

class Father{
    int age;
    public Father(int age) throws WrongAge{
        if(age<0){
            throw new WrongAge("Age cannot be negative!");}
        this.age = age;
        System.out.println("Fathers Age:" + this.age);
    }
}
```

DATE: PAGE:

```
class son extends father {
    int sonAge;
    public son(int fatherAge, int sonAge) throws
      WrongAge {
        super(fatherAge);
        if(sonAge < 0) {
            throw new WrongAge("Son's Age
            cannot be Negative!");
        }
        if(sonAge >= fatherAge) {
            throws new WrongAge("Son's Age
            cannot be greater than or equal to
            father Age!");
        }
        this.sonAge = sonAge;
        System.out.println("Sons Age:" + this.sonAge);
    }
}

public class FatherSon {
    public static void main(String[] args) {
        while(true) {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter Father Age at
            Sons Age");
            int fatherAge = sc.nextInt();
            int sonAge = sc.nextInt();
            try {
                Son son = new Son(fatherage, son
                Age);
                System.out.println("Accepted Succesfull
                ");
```



```
}
catch (WrongAge Exception e) {
        System.out.println(e.getMessage());
}
catch (Son Age Exception e) {
        System.out.println(e.getMessage());
}
System.out.println("Would you like to re
details");
String input = sc.next();
if (input.equalsIgnoreCase("n")) {
        break;
}
}
}
}
```

Output:

```
Enter Father's Age:
34
Enter Son's Age: 4
Accepted
Enter details again:(Y/n)
y
Enter Father's Age:
54
Enter Son's Age: 21
Accepted
Enter details again:(Y/n)
y
Enter Father's Age:
45
Enter Son's Age: 46
Son's age should be less than father's age
Enter details again:(Y/n)
n
```

Program-8:

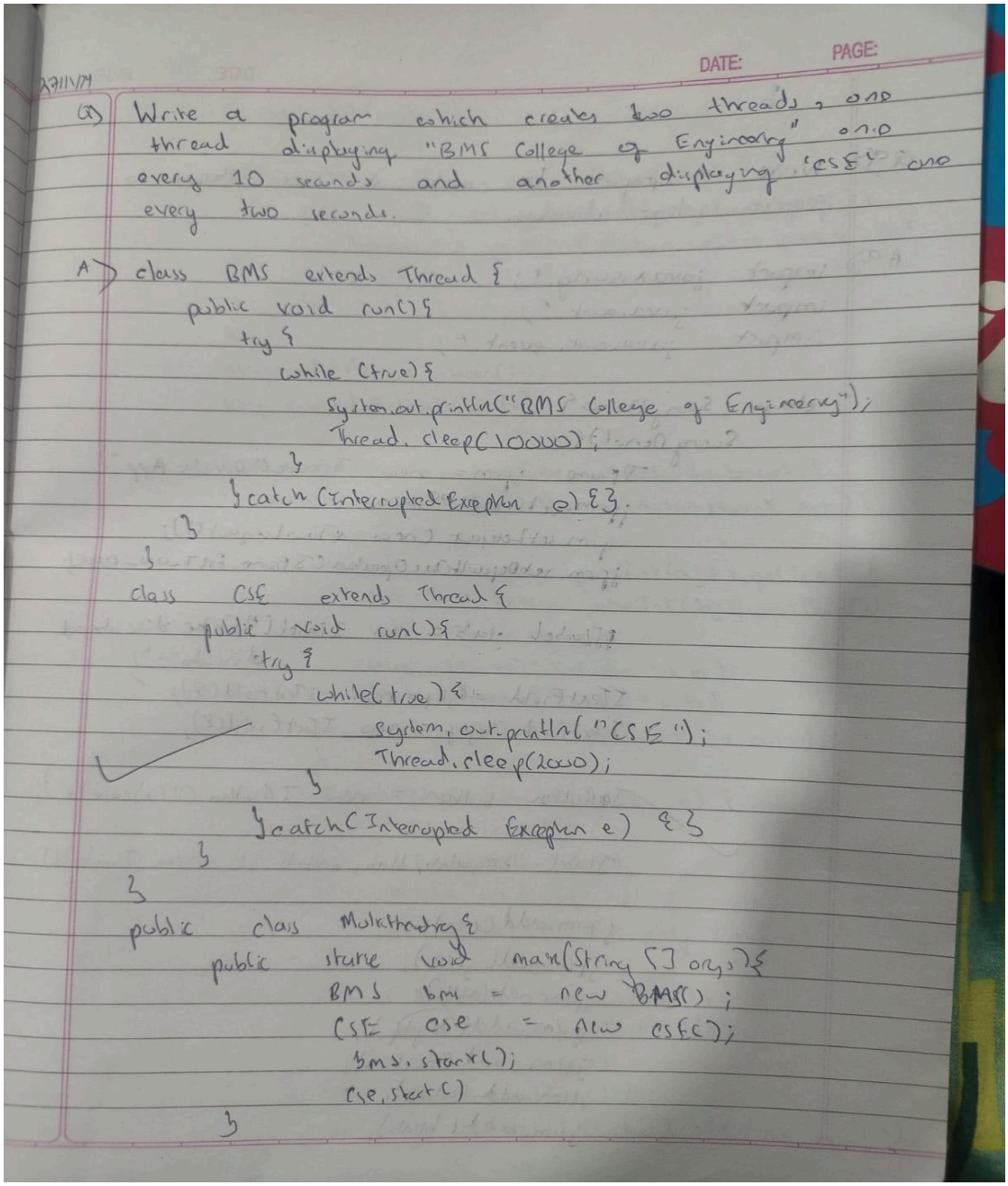
Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

Code:

```
class BMS extends Thread {

    public void run() {

        try {

            while (true) {

                System.out.println("BMS College of Engineering");

                Thread.sleep(10 * 1000);  // Sleep for 10 seconds

            }

        } catch (InterruptedException e) {}

    }

}


class CSE extends Thread {

    public void run() {

        try {

            while (true) {

                System.out.println("CSE");
```

```
                Thread.sleep(2000);  // Sleep for 2 seconds
            }
        } catch (InterruptedException e) {}
    }
}

public class Multithread {
    public static void main(String[] args) {
        BMS bms = new BMS();  // Create an instance of the BMS thread
        CSE cse = new CSE();  // Create an instance of the CSE thread
        bms.start();  // Start the BMS thread
        cse.start();  // Start the CSE thread
    }
}
```

Notebook:



Q) Write a program which creates two threads, one thread displaying "BMS College of Engineering" one every 10 seconds and another displaying "CSE" one every two seconds.

A)
```
class BMS extends Thread {
    public void run(){
        try {
            while (true){
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000);
            }
        }catch (InterruptedException e){}.
    }
}
class CSE extends Thread{
    public void run(){
        try {
            while(true){
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        }catch(InterruptedException e) {}
    }
}
public class Multithreading{
    public static void main(String[] args){
        BMS bms = new BMS();
        CSE cse = new CSE();
        bms.start();
        cse.start()
    }
```

Output:

```
CSE
BMS College of Engineering
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
```

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.

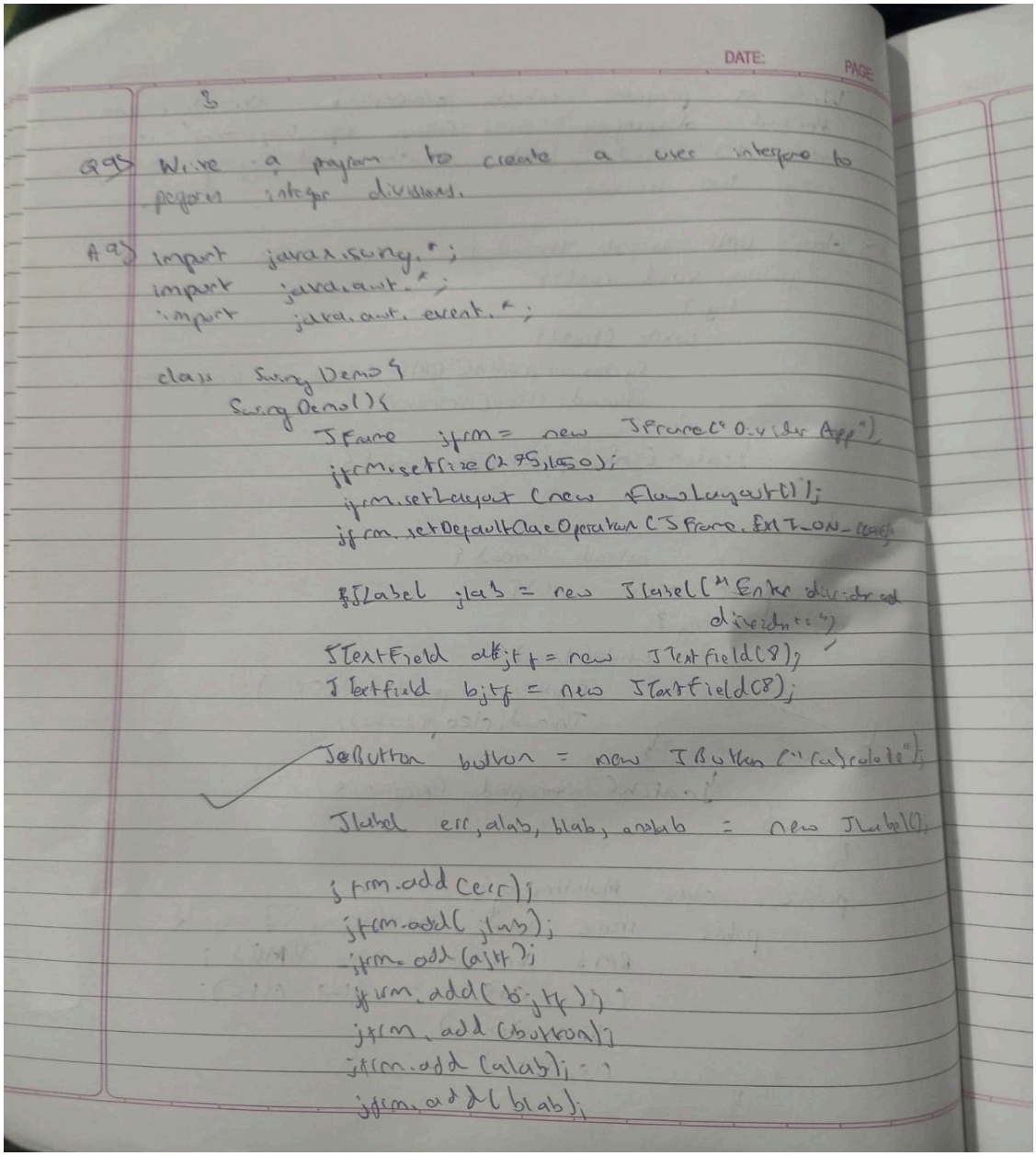
Code:

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

class SwingDemo {
 SwingDemo() {
 // Create JFrame container
 JFrame jfrm = new JFrame("Divider App");
 jfrm.setSize(275, 150);
 jfrm.setLayout(new FlowLayout());
 // Terminate on close
 jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 // Create components
 JLabel jlab = new JLabel("Enter the divider and dividend:");
JTextField ajtf = new JTextField(8);
 JTextField bjtf = new JTextField(8);
 JButton button = new JButton("Calculate");
 JLabel err = new JLabel();
 JLabel alab = new JLabel();
 JLabel blab = new JLabel();
 JLabel anslab = new JLabel();


 // Add components in order
 jfrm.add(err); // To display errors
 jfrm.add(jlab);
```

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

 // Add ActionListeners
 ActionListener l = new ActionListener() {
 public void actionPerformed(ActionEvent evt) {
System.out.println("Action event from a text field");  }
 };
 ajtf.addActionListener(l);
 bjtf.addActionListener(l);

 button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {  try {
  int  a  =  Integer.parseInt(ajtf.getText());    int  b  =
Integer.parseInt(bjtf.getText());  int ans = a / b;

 alab.setText("A = " + a);
 blab.setText("B = " + b);
 anslab.setText("Ans = " + ans);
 err.setText(""); // Clear error message  } catch
(NumberFormatException e) {  alab.setText("");
 blab.setText("");
 anslab.setText("");
 err.setText("Enter Only Integers!");  } catch
(ArithmeticException e) {
 alab.setText("");
 blab.setText("");
 anslab.setText("");
 err.setText("B should be NON-zero!");


 }
 }
 });

 // Display the frame
```

```
 jfrm.setVisible(true);
 }

 public static void main(String args[]) {
 // Create frame on Event Dispatching Thread
SwingUtilities.invokeLater(new Runnable() {
public void run() {
 new SwingDemo();
 }
 });
 System.out.println("Aryan Gowda 1BM23CS054");  }
}
```

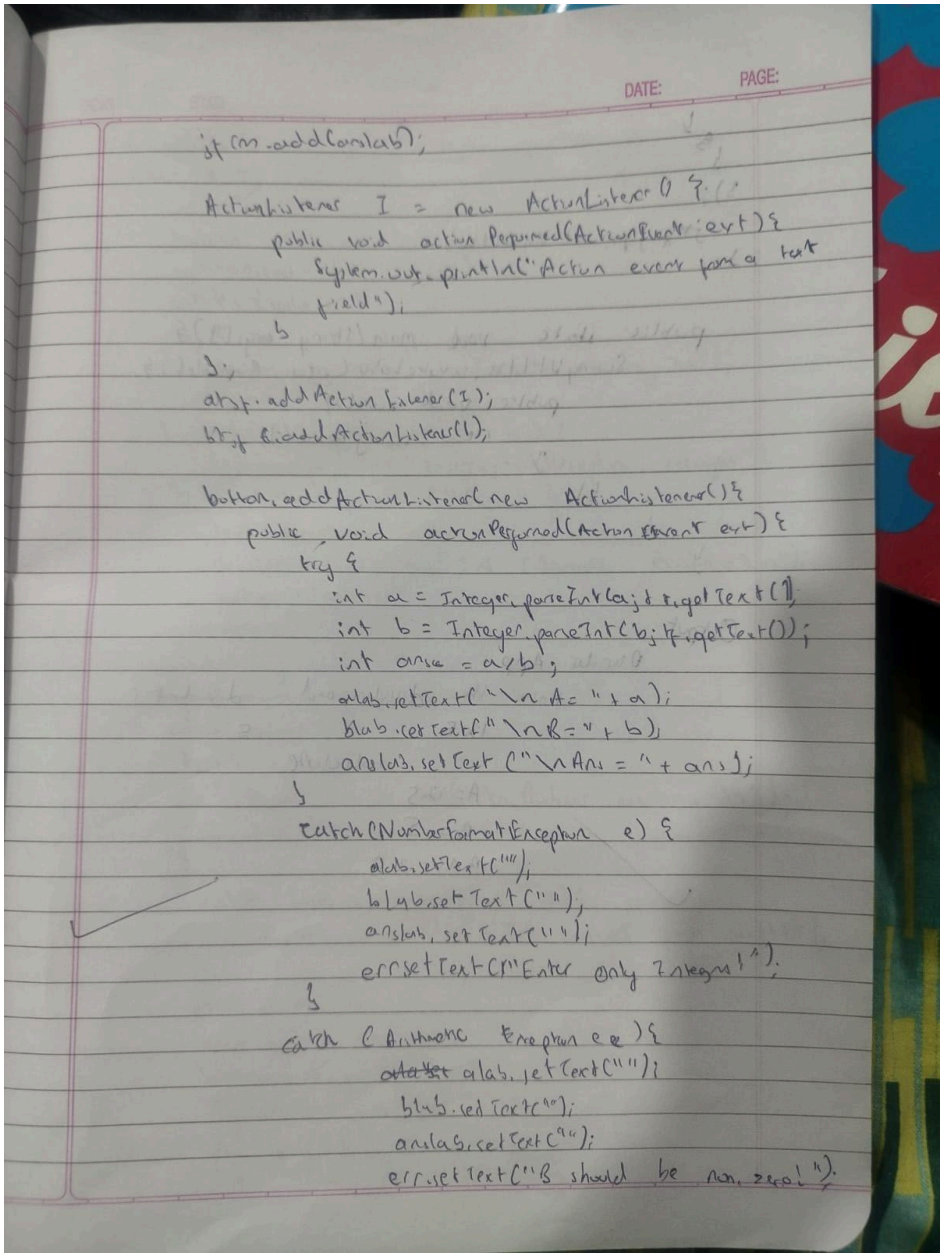
Notebook:



Q9) Write a program to create a user interface to perform integer division.

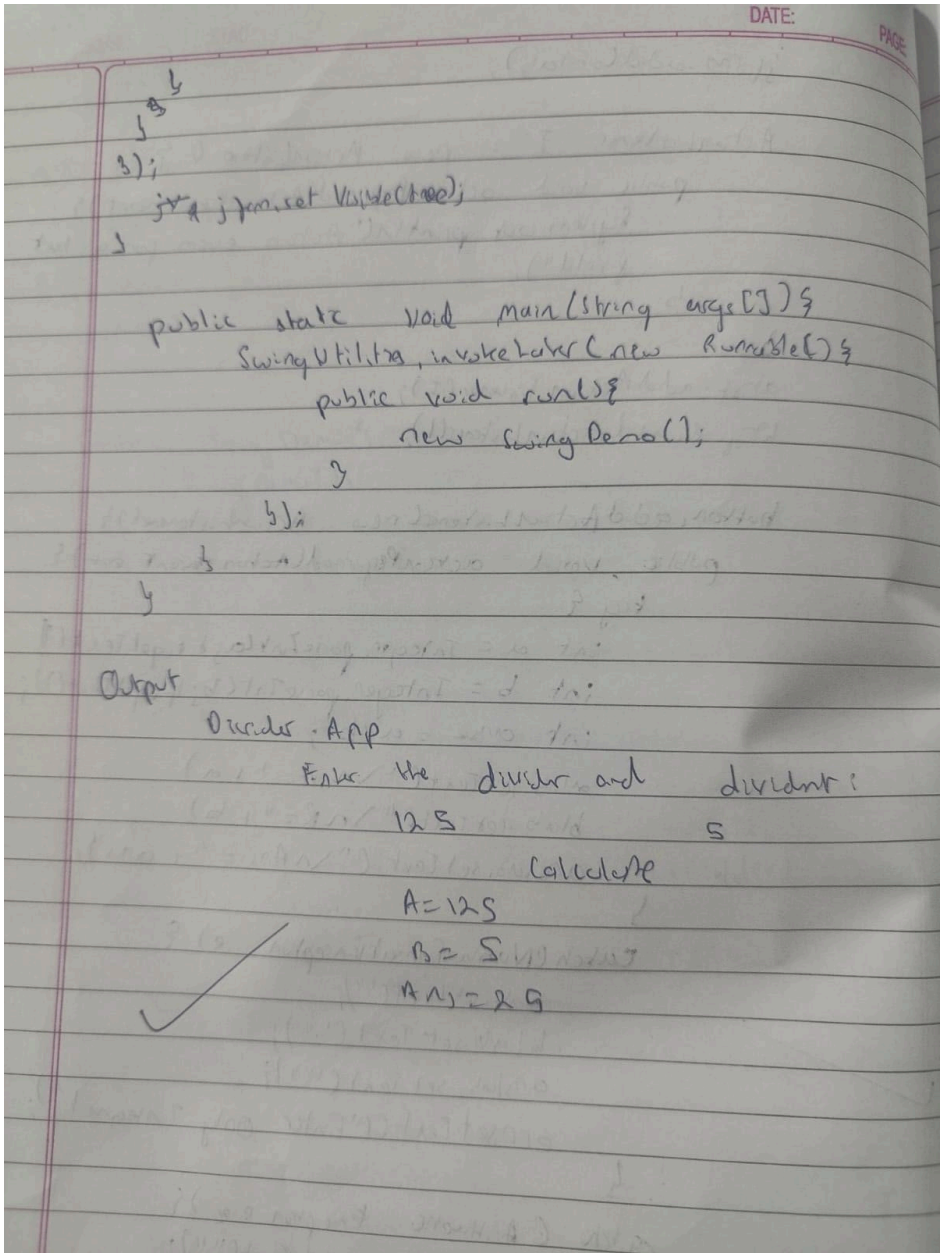
Ans)

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

class SwingDemo{
    SwingDemo(){
        JFrame jfrm = new JFrame("Divider App");
        jfrm.setSize(275,150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel jlab = new JLabel("Enter dividend and divisor:");
        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(8);

        JButton button = new JButton("Calculate");

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

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



DATE: PAGE:

```
jf(rn.add(ansLab);

ActionListener I = new ActionListener() {
    public void actionPerformed(ActionEvent evt){
        System.out.println("Action event from a text
        field");
    }
};
atxt.addActionListener(I);
btxt.addActionListener(I);

button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent evt){
        try{
            int a = Integer.parseInt(atxt.getText());
            int b = Integer.parseInt(btxt.getText());
            int ans = a/b;
            alab.setText("\n A= "+a);
            blab.setText("\n B= "+b);
            ansLab.setText("\n Ans = "+ans);
        }
        catch(NumberFormatException e){
            alab.setText("");
            blab.setText("");
            ansLab.setText("");
            errsetText("Enter only Integers!");
        }
        catch(ArithmeticException ee){
            alab.setText("");
            blab.setText("");
            ansLab.setText("");
            errsetText("B should be non zero!");
```



```
    }
});
    jf.setVisible(true);
}

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

Output

Divider.App

Enter the divisor and dividend:

12 5

5

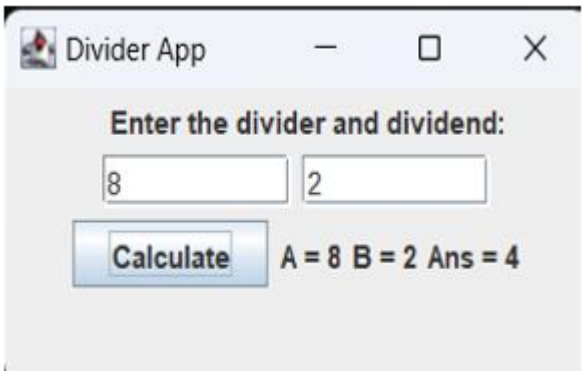
Calculate

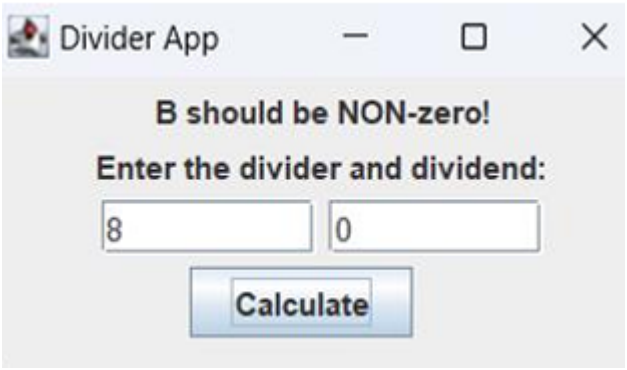
A=125

B=5

A/B=25

Output:


Divider App
Enter the divider and dividend:
8
2
Calculate
A = 8 B = 2 Ans = 4


Divider App
B should be NON-zero!
Enter the divider and dividend:
8
0
Calculate


Divider App
Enter Only Integers!
Enter the divider and dividend:
8.2
1.3
Calculate

Program-10:

Demonstrate Inter process Communication and deadlock

Code:

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

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

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

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

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

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

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

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

```
 }
}

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

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

 a.foo(b);
 System.out.println("Back in main thread");
 }

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

 public static void main(String args[]) {
 new Deadlock();
 System.out.println("1BM23CS054 Aryan Gowda");  }
}
```

Code-2:

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

 synchronized int get() {
 while (!valueSet) {
 try {
 System.out.println("\nConsumer waiting\n");
```

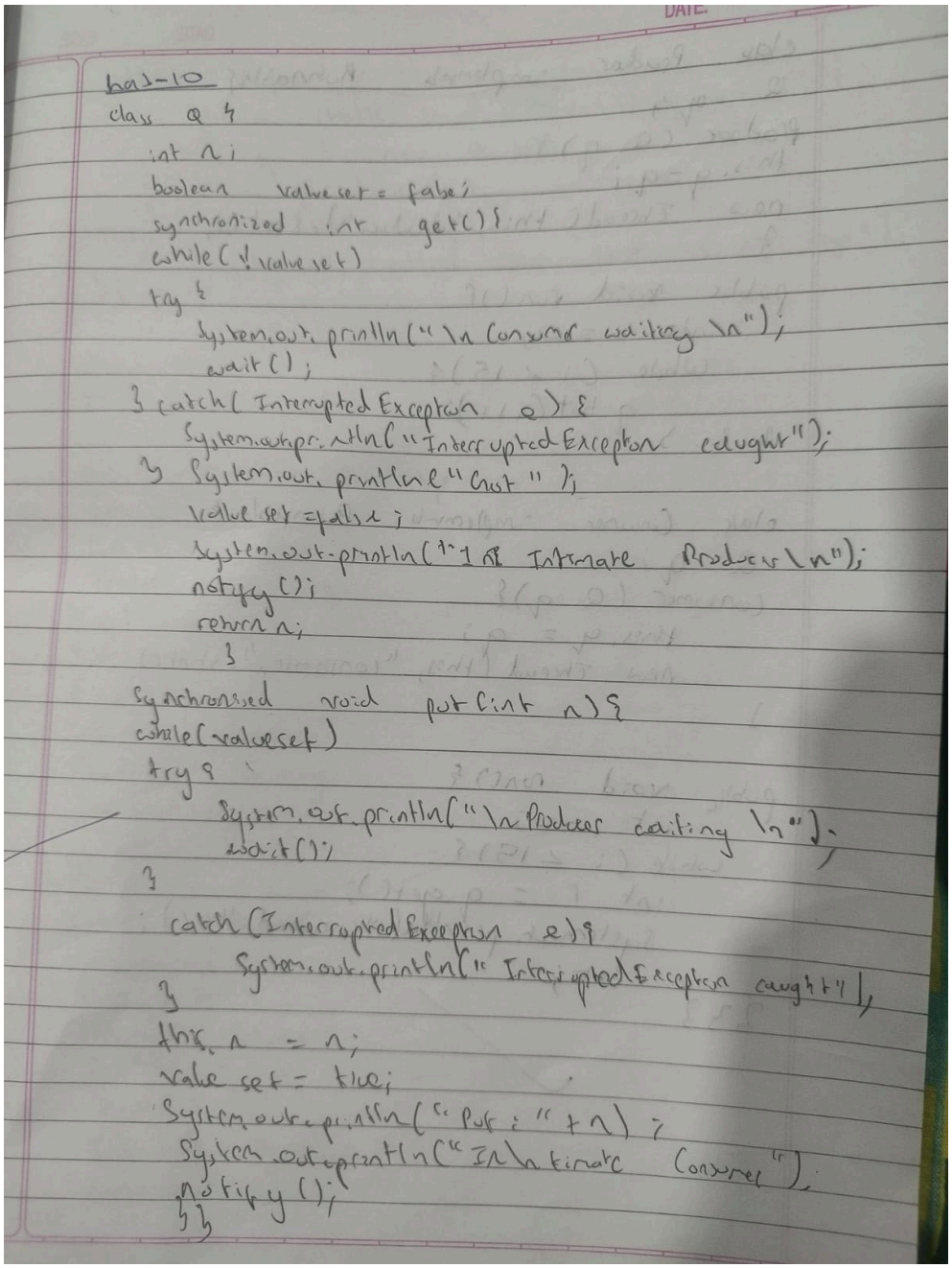


```
 wait();
 } catch (InterruptedException e) {
 System.out.println("InterruptedException caught");  }
 }
 System.out.println("Got: " + n);
```

```
 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++;
 }
 }
}

public 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.");
System.out.println("Aryan Kolur Gowda 1BM23CS054");  }
 }
```

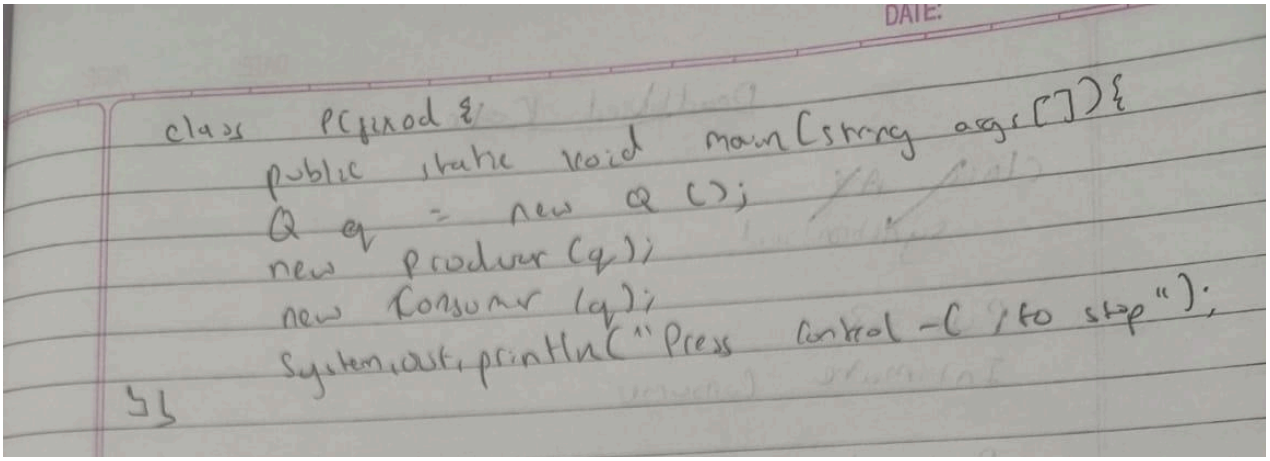
Notebook:



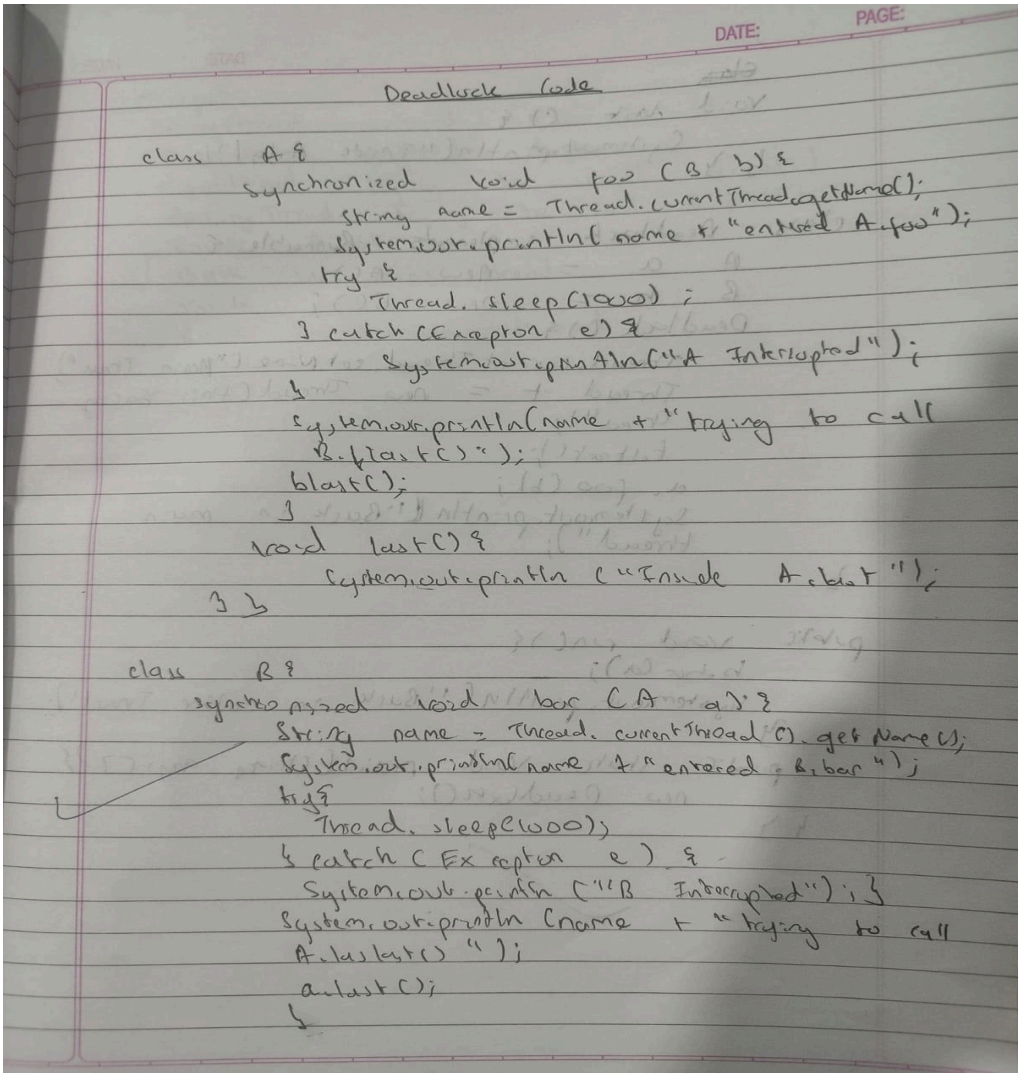
ba1-10

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



Deadlock Code

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

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

```
class
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.foo(b);
        System.out.println("Back in main
        thread");
    }

public void run(){
    b.bar(a);
    System.out.println("Back in other Thread");
}
public static void main(String args[]){
    new Deadlock();
} }
```

Output:

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

```
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

Put: 5

Intimate Consumer


Producer waiting

Consumed: 4
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
```

```
Consumed: 8
Got: 9

Intimate Producer

Put: 10

Intimate Consumer


Producer waiting

Consumed: 9
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
```