VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

AMOGH KRISHNA J S (1BM23CS029)

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 **Amogh Krishna J S (1BM23CS029),** 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.

Lab faculty In charge: Mr. Basavaraj Jakalli Associate Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:

https://github.com/jsak737/OOJLAB

Program 1:

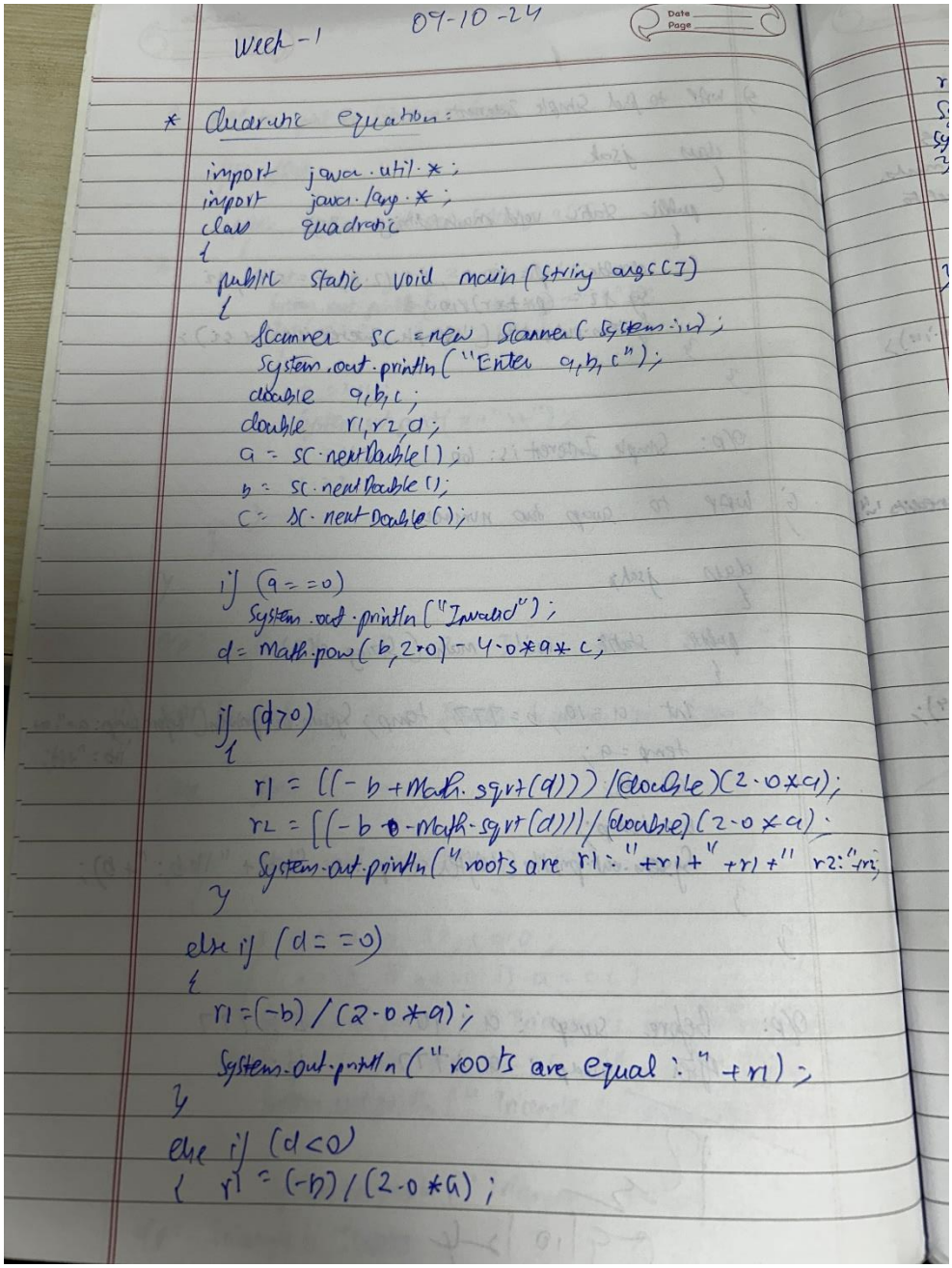
Develop a Java program that prints all real solutions to the quadratic equation ax2 +bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2 -4ac is negative, display a message stating that there are no real solutions.

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("invaild!!");
```

```
}

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("J S AMOGH KRISHNA 1BM23CS029");
}
}
```

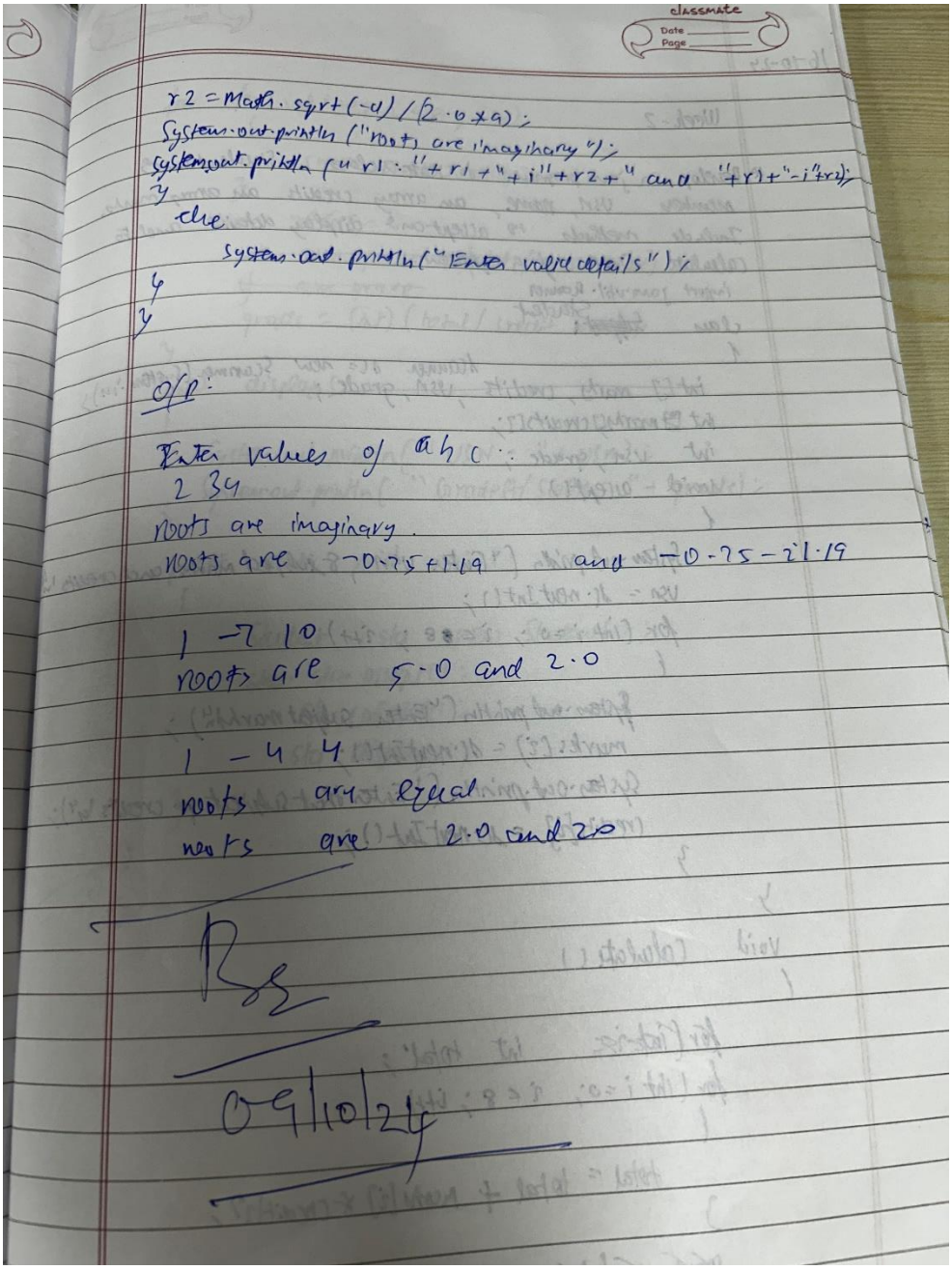
Notebook:



Week-1 09-10-24

* Quadratic equation:

```
import java.util.*;
import java.lang.*;
class Quadratic
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter a,b,c");
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+" "+r1+" r2:"+r1;
}
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 details");
}
}
```

O/P:

Enter values of a, b, c:
2 3 4
roots are imaginary.
roots are -0.75+1.19 and -0.75-i1.19

1 -7 10
roots are 5.0 and 2.0

1 -4 4
roots are Equal
roots are 2.0 and 2.0

Rs

09/10/24

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
J S AMOGH KRISHNA 1BM23CS029

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

1 -4 4
roots are equal: 2.0
J S AMOGH KRISHNA 1BM23CS029

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

6 6 6
roots are imaginary
r1:-0.5+i0.8660254037844387  and   -0.5-i0.8660254037844387
J S AMOGH KRISHNA 1BM23CS029
```

Program-2:

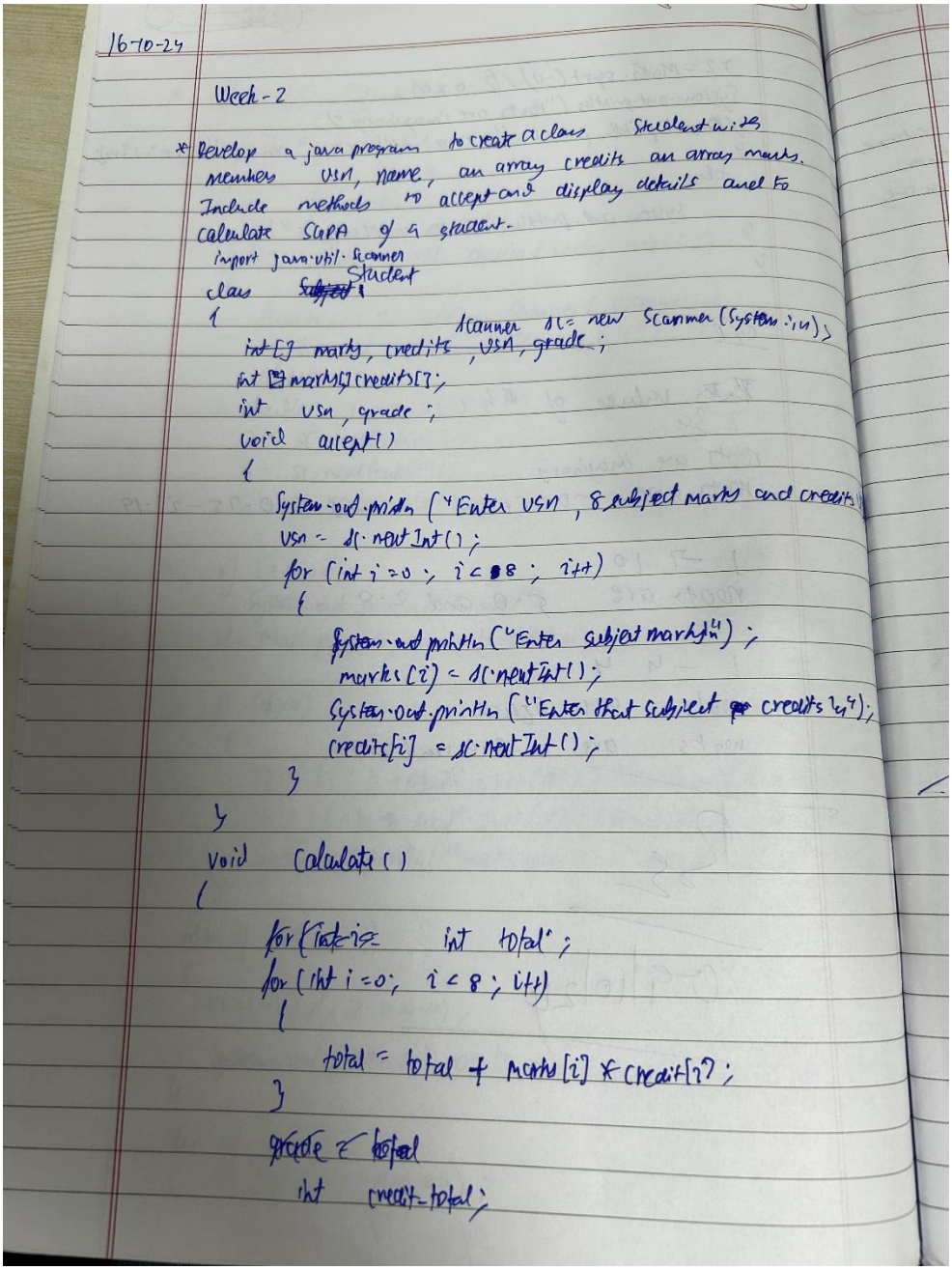
Develop a Java program to create a class Student with members usn, name, an array credits and an array marks. Include methods to accept and display details and a method to calculate SGPA of a student.

Code:

```
import java.util.Scanner;
class SGPA{
Scanner sc=new Scanner(System.in);
int marks[]=new int[8];
int credits[]=new int[8];
int USN,sum,totalcredits;
float grade;

void accept()
{
System.out.println("\nEnter Student USN: ");
USN=sc.nextInt();
for(int i=0;i<8;i++)
{
System.out.println("Enter the subject marks: ");
marks[i]=sc.nextInt();
System.out.println("Enter the respective credits: ");
credits[i]=sc.nextInt();
}
}

void calculate()
{
for(int i=0;i<8;i++)
{
sum+=marks[i]*credits[i];
}
for(int i=0;i<8;i++)
{
totalcredits+=credits[i];
}
```

```
grade=(float)(sum/(totalcredits*10));
}

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

public static void main(String args[])
{
SGPA s[]=new SGPA[3];
for(int i=0;i<3;i++)
{
s[i]=new SGPA();
}
for(int i=0;i<3;i++)
{
s[i].accept();
s[i].calculate();
s[i].display();
System.out.println("J S AMOGH KRISHNA 1BM23CS029");
}
}
}
```

Notebook:

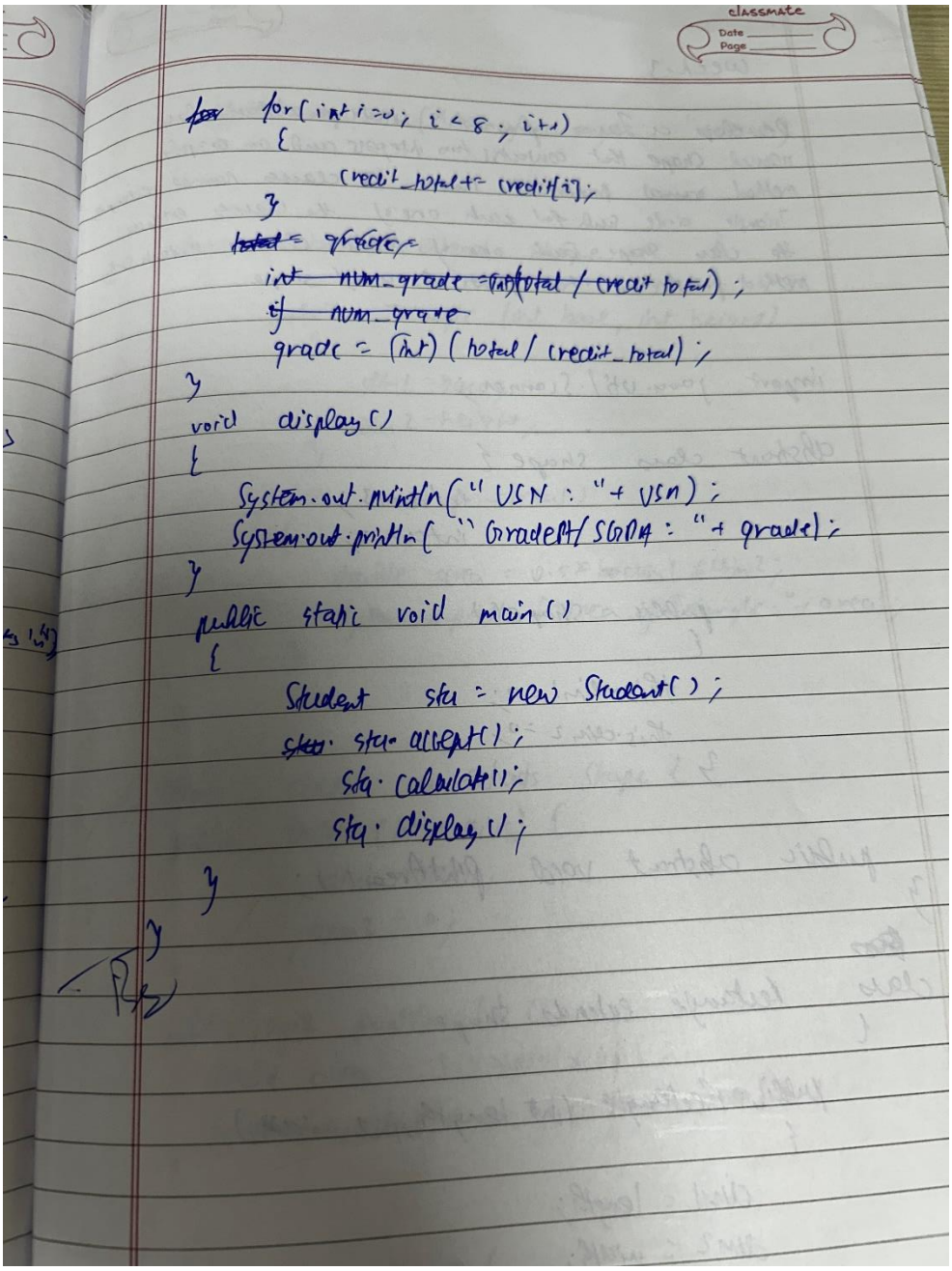


16-10-24

Week - 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 to calculate SGPA of a student.

```
import java.util.Scanner
class Student
{
    Scanner sc = new Scanner(System.in);
    int marks[], credits[], usn, grade;
    int marks[], credits[];
    int usn, grade;
    void accept()
    {
        System.out.println("Enter usn, 8 subject marks and credits");
        usn = sc.nextInt();
        for (int i=0; i<8; i++)
        {
            System.out.println("Enter subject marks");
            marks[i] = sc.nextInt();
            System.out.println("Enter that subject credits");
            credits[i] = sc.nextInt();
        }
    }
    void calculate()
    {
        for(int i=  int total;
        for (int i=0; i<8; i++)
        {
            total = total + marks[i] * credit[i];
        }
        grade = total
        int credit_total;
```



```
for(int i=0; i<8; i++)
{
    credit_total += credit[i];
}
grade = (int)(total / credit_total);
}
void display()
{
System.out.println("USN : "+usn);
System.out.println("Grade/SGPA : "+grade);
}
public static void main()
{
    Student stu = new Student();
    stu.stu_accept();
    stu.calculate();
    stu.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:

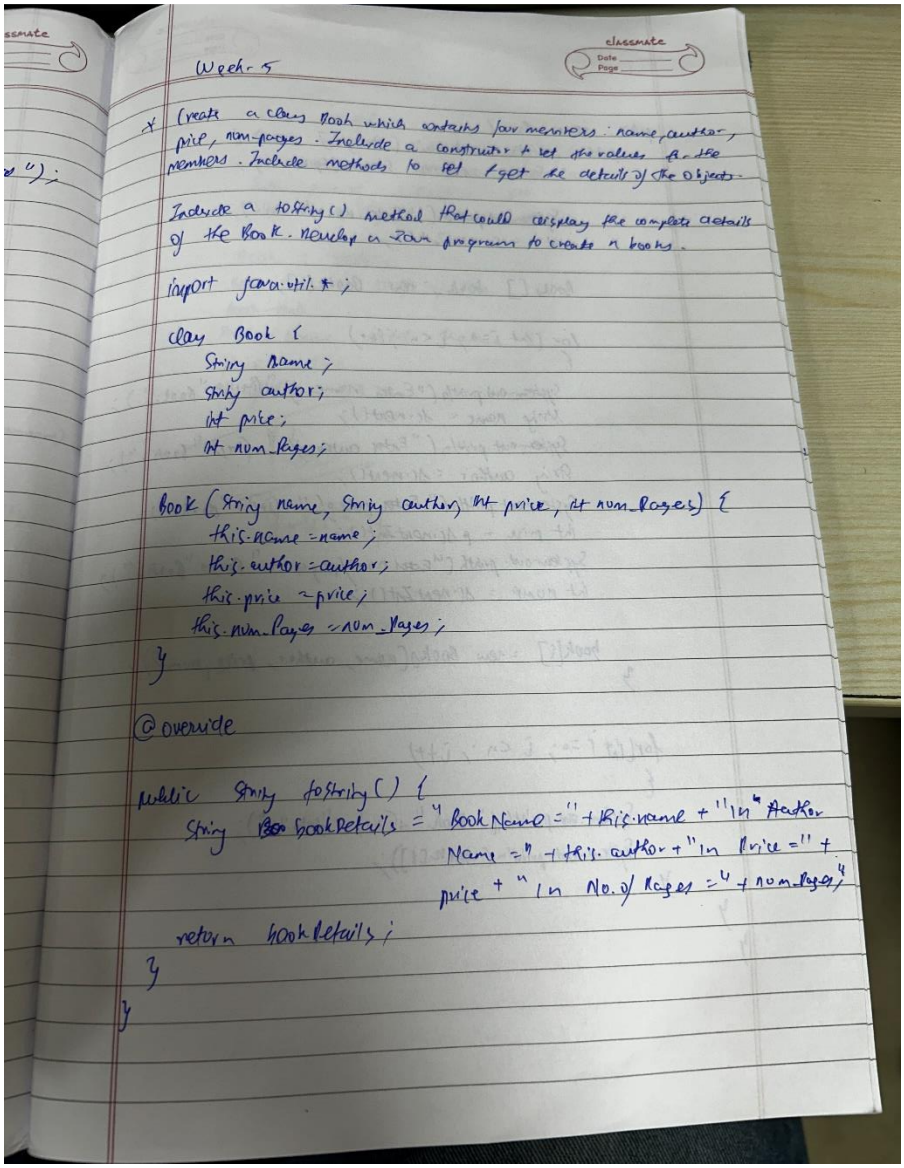
Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a toString( ) method that could display the complete details of the book. Develop a Java program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

Code:

```
import java.util.Scanner;

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

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

    @Override
    public String toString() {
        String bookDetails = "Book Name: " + this.name + "\n" +
                             "Author Name: " + this.author + "\n" +
                             "Price: " + this.price + "\n" +
                             "Number of Pages: " + this.numPages + "\n";
        return bookDetails;
    }
}

public class BooksData {
```

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

        System.out.print("Enter the Number of Books: ");
        int n = s.nextInt();

        Book[] books = new Book[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Enter name of book " + (i + 1) + ": ");
            String name = s.next();
            System.out.print("Enter author of book " + (i + 1) + ": ");
            String author = s.next();
            System.out.print("Enter price of book " + (i + 1) + ": ");
            int price = s.nextInt();
            System.out.print("Enter number of pages in book " + (i + 1) + ": ");
            int numPages = s.nextInt();

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

        System.out.println("\nBook Details:");
        for (Book book : books) {
            System.out.println(book);
        }
          System.out.println("J S AMOGH KRISHNA 1BM23CS029");


        s.close();
    }
}
```

Notebook:

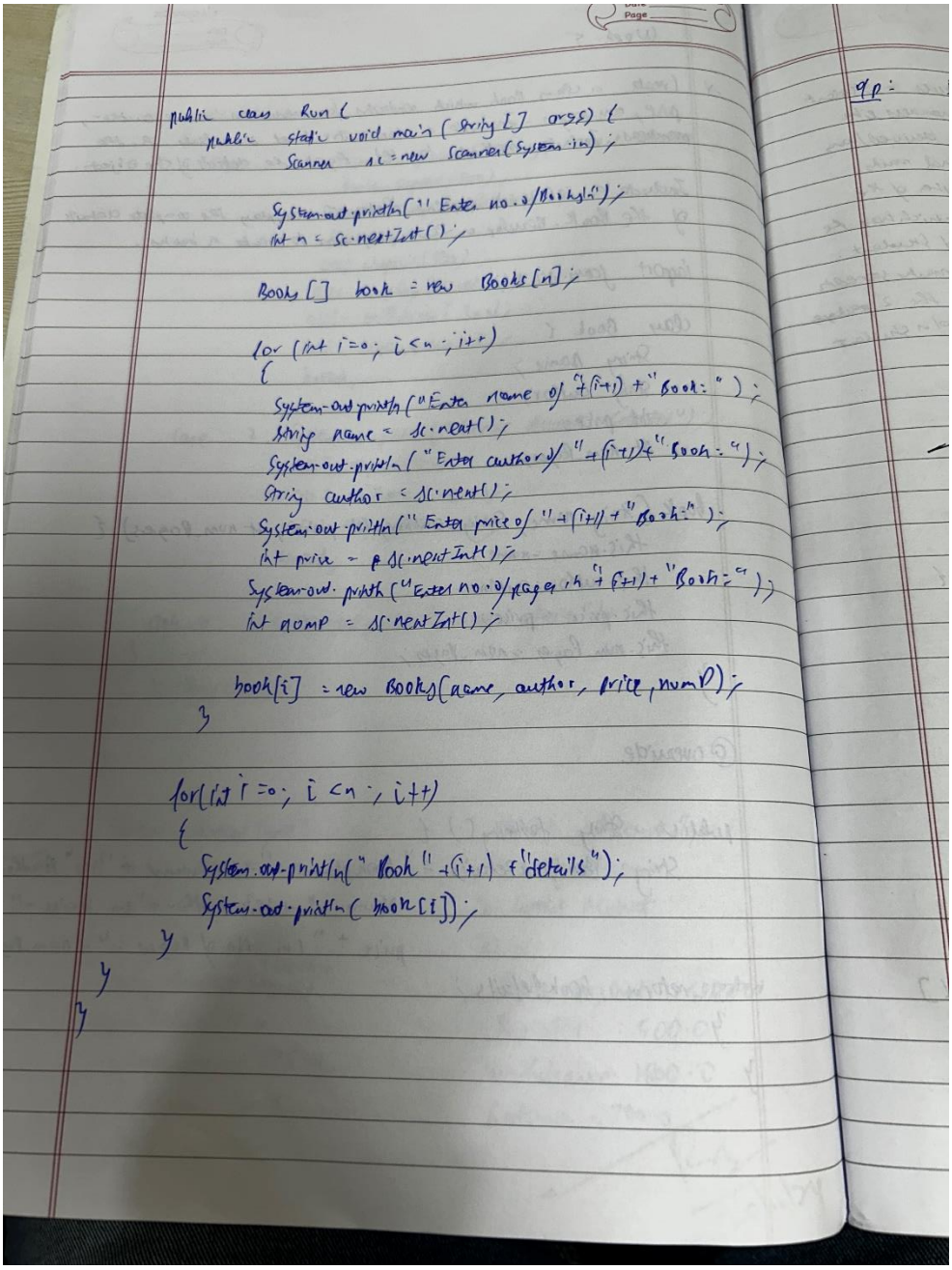


Week-5

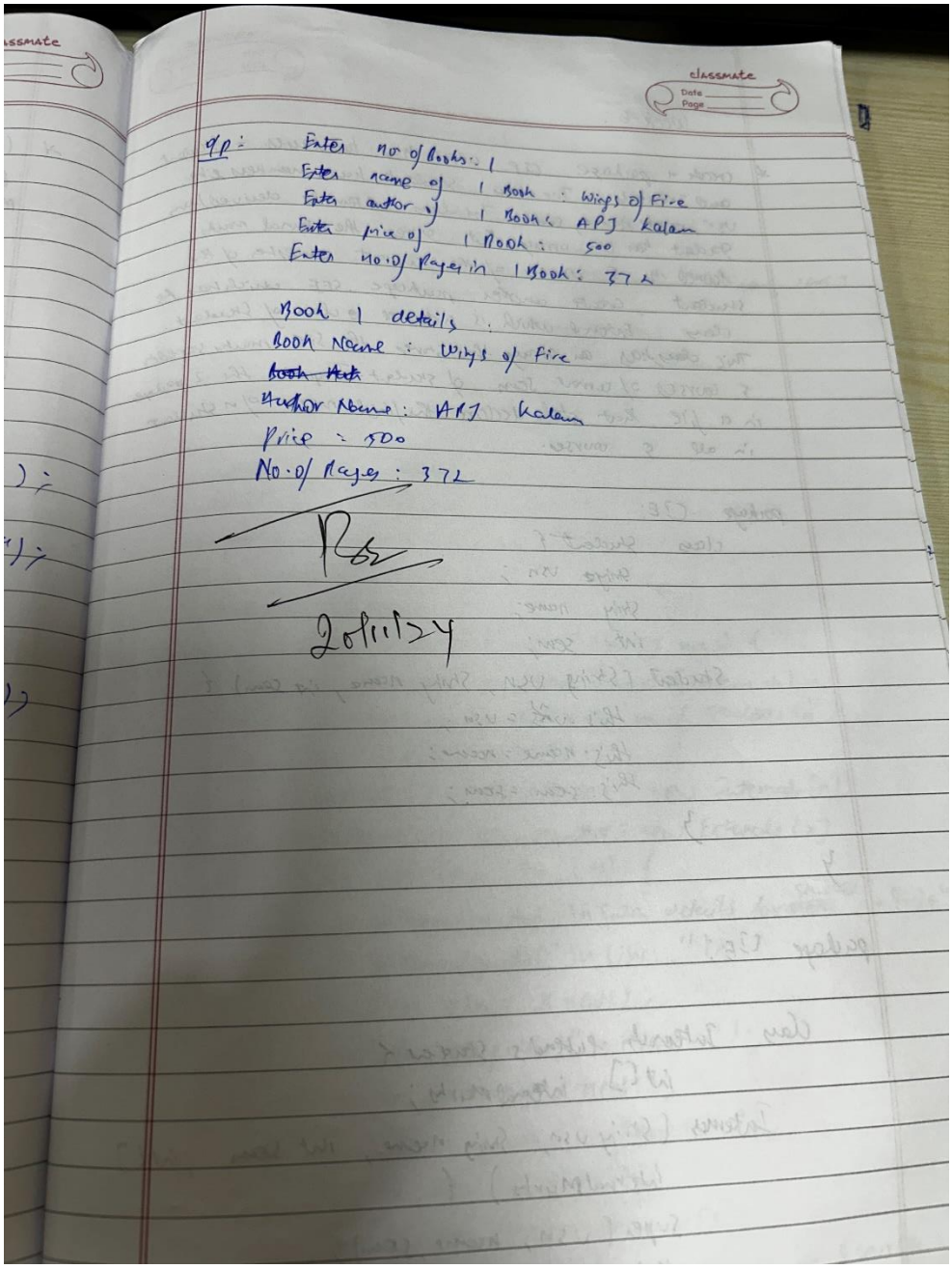
1) Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values to the members. Include methods to set / get the details of the objects.

Include a toString() method that could display the complete details of the Book. Develop a Java program to create n books.

```
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;
}

@Override

public String toString() {
    String BookDetails = "Book Name = " + this.name + "\n Author
                         Name = " + this.author + "\n Price = " +
                         price + "\n No. of Pages = " + num_Pages;

    return bookDetails;
}
}
```



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

        System.out.println("Enter no. of Books:");
        int n = sc.nextInt();

        Books[] book = new Books[n];

        for (int i=0; i<n; i++)
        {
            System.out.println("Enter name of "+(i+1)+"Book: ");
            String name = sc.next();
            System.out.println("Enter author of "+(i+1)+"Book: ");
            String author = sc.next();
            System.out.println("Enter price of "+(i+1)+"Book:");
            int price = sc.nextInt();
            System.out.println("Enter no. of pages in "+(i+1)+"Book:");
            int numP = sc.nextInt();

            book[i] = new Books(name, author, price, numP);
        }

        for(int i=0; i<n; i++)
        {
            System.out.println("Book "+(i+1)+"details");
            System.out.println(book[i]);
        }
    }
}
```



op:
Enter no of books: 1
Enter name of 1 book: Wings of Fire
Enter author of 1 book: APJ Kalam
Enter price of 1 book: 500
Enter no. of Pages in 1 book: 372

Book 1 details:
Book Name: Wings of Fire
~~Book Auth~~
Author Name: APJ Kalam
Price: 500
No. of Pages: 372

Rs

20/11/24

OUPUT:

```
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 contain only the method printArea( ) that prints the area of the given shape.

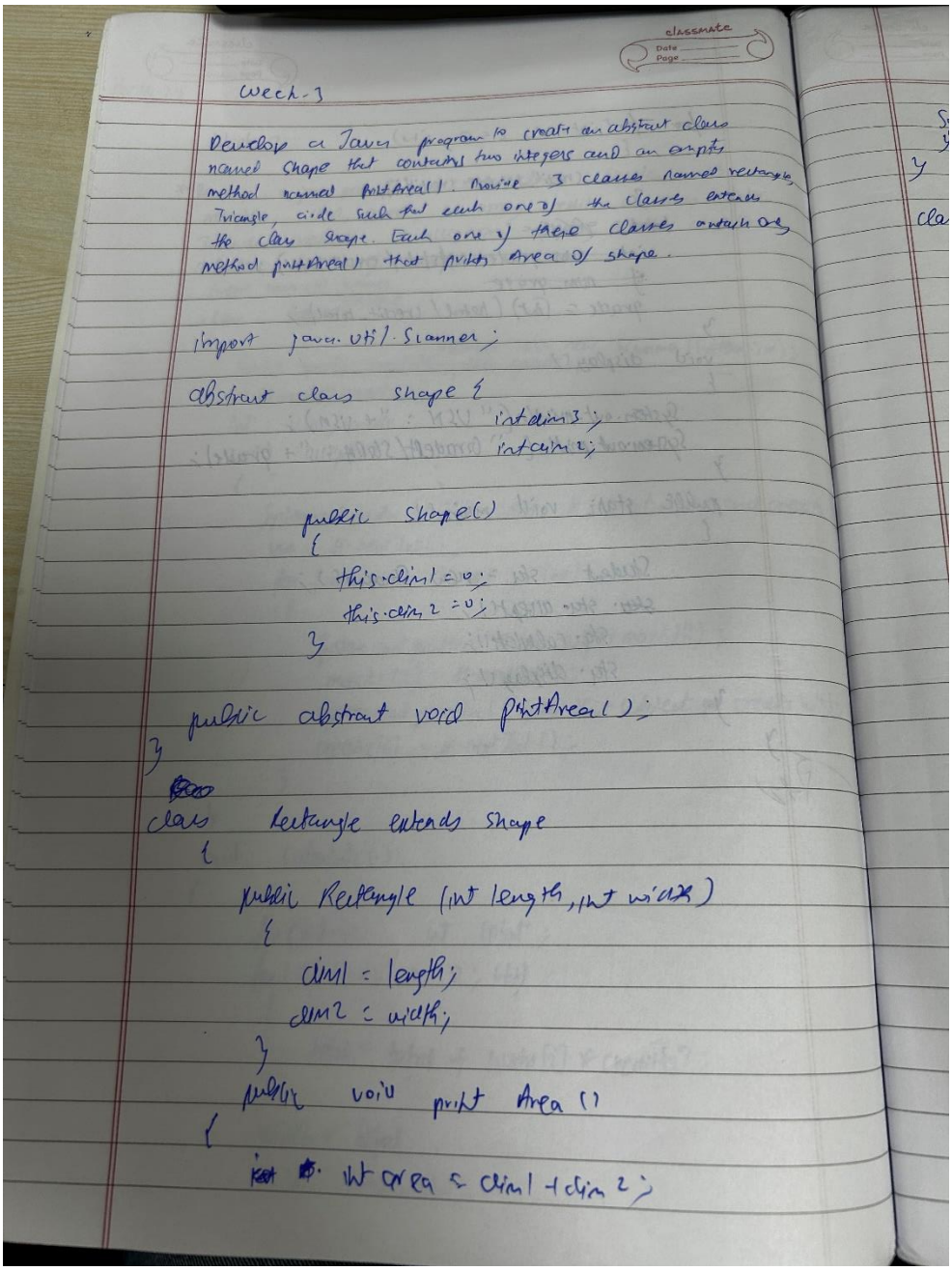
Code:

```
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("J S AMOGH KRISHNA 1BM23CS029");


        in.close();
    }
}
```

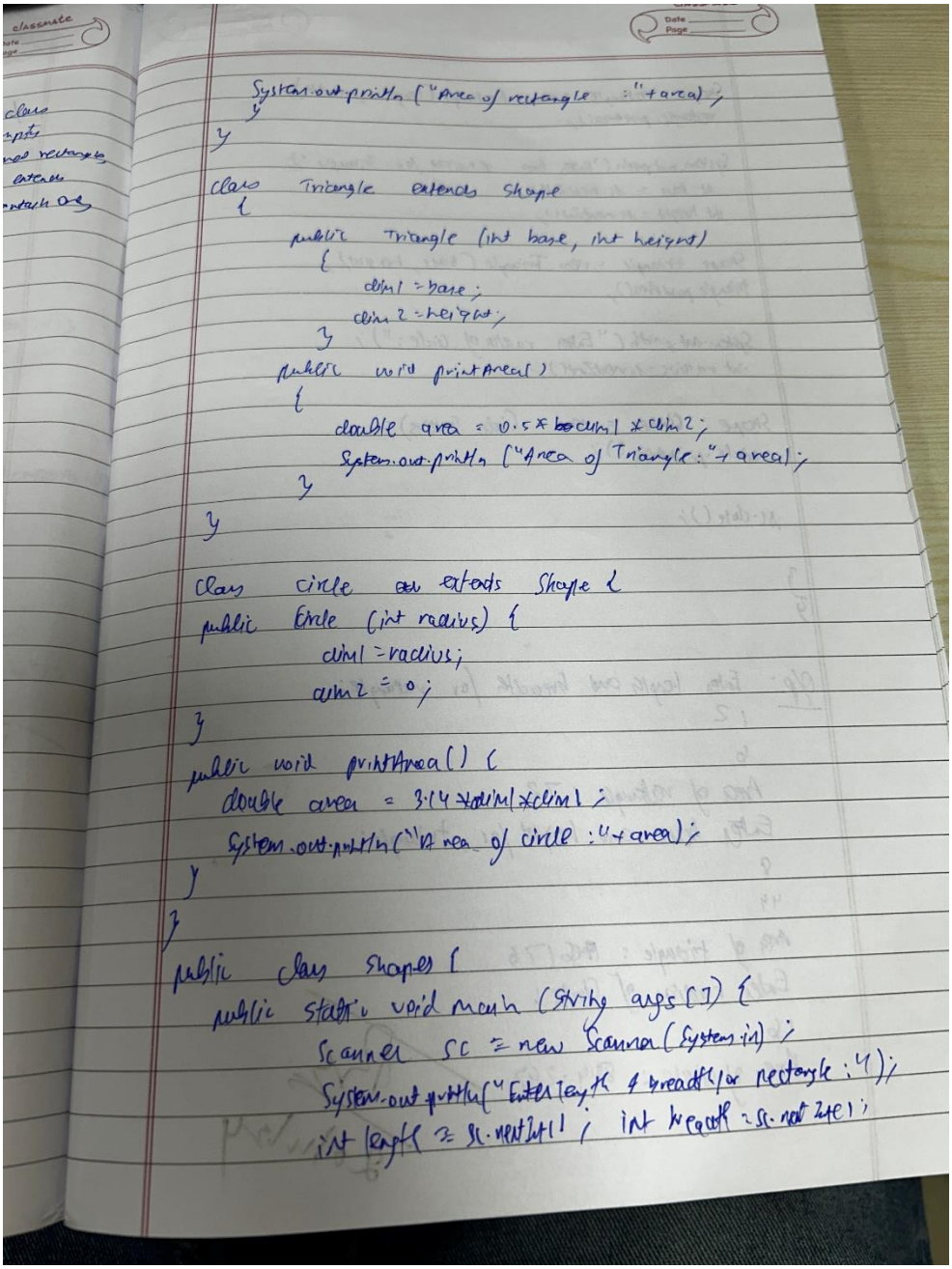
Notebook:



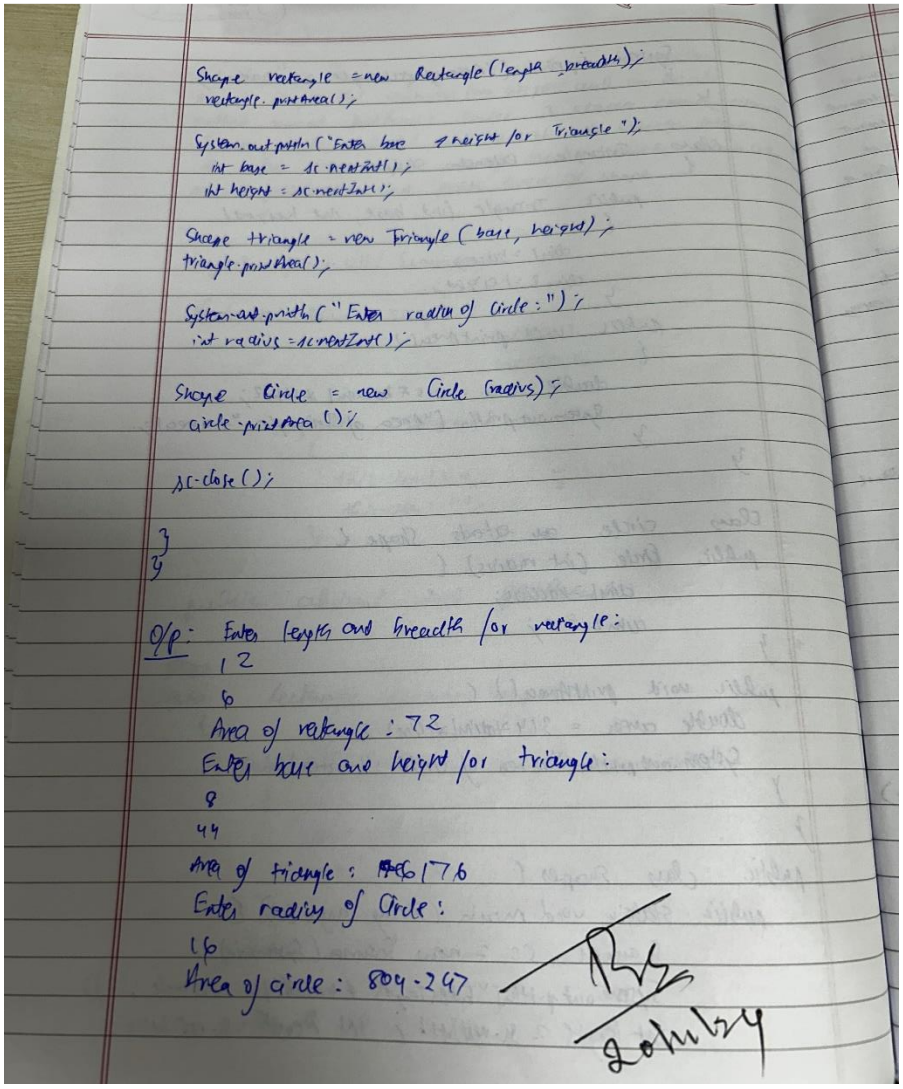
Week-3

Develop a Java program to create an abstract class named Shape that contains two integers and an empty method named printArea(). Provide 3 classes named rectangle, Triangle, circle such that each one of the classes extends the class Shape. Each one of these classes contains only method printArea() that prints Area of shape.

```
import java.util.Scanner;
abstract class Shape {
        int dim1;
        int dim2;

    public Shape()
    {
      this.dim1 = 0;
      this.dim2 = 0;
    }

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 = 3.14 * dim1 * dim1;
    System.out.println("Area of circle :"+area);
}
}
public class Shapes {
public static void main(String args[]) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter length & breadth for rectangle :");
    int length = sc.nextInt(); int breadth = sc.nextInt();
```



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:

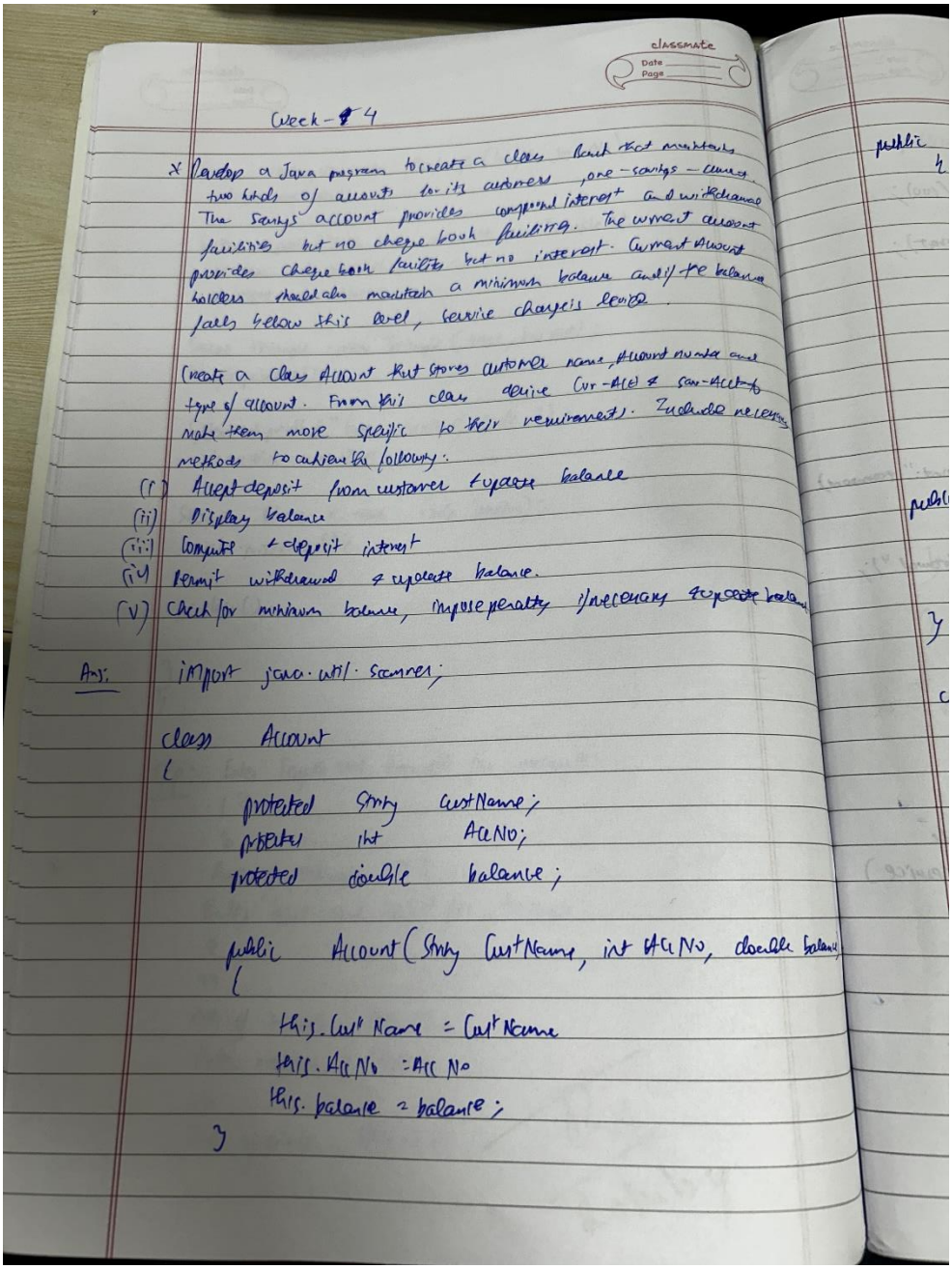
```
import java.util.Scanner;

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

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

    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() {
```

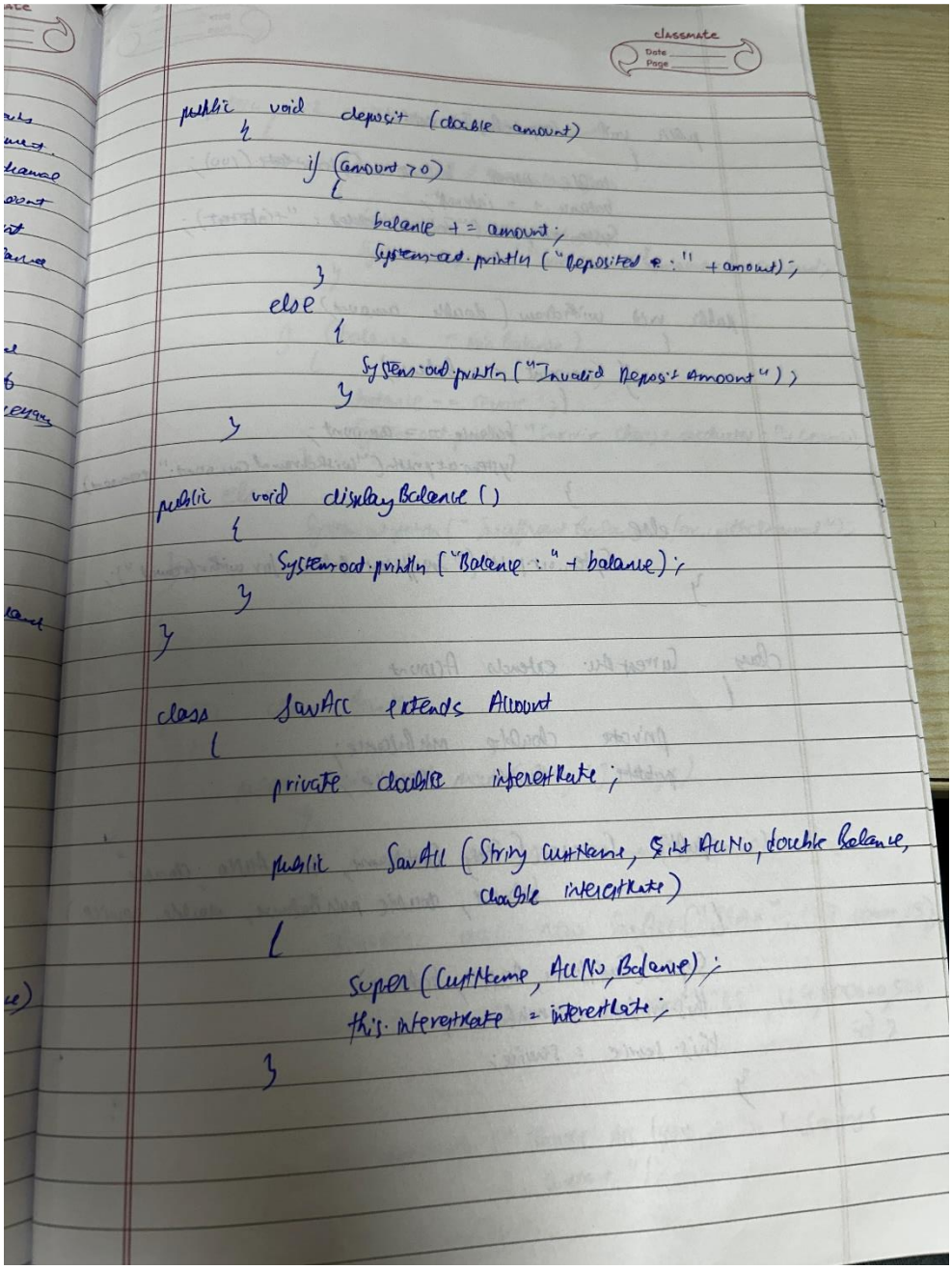
```
        System.out.println("Balance: " + balance);
    }
}

class SavAcct extends Account {
    private double interestRate;

    public SavAcct(String customerName, int accountNumber, double balance, double
interestRate) {
        super(customerName, accountNumber, balance);
        this.interestRate = interestRate;
    }

    public void computeAndDepositInterest() {
        double interest = balance * (interestRate / 100);
        balance += interest;
        System.out.println("Interest added: " + interest);
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance for withdrawal");
        }
    }
}

class CurAcct extends Account {
    private double minimumBalance;
    private double serviceCharge;

    public CurAcct(String customerName, int accountNumber, double balance, double
minimumBalance, double serviceCharge) {
        super(customerName, accountNumber, balance);
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }
```

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

            if (balance < minimumBalance) {
                balance -= serviceCharge;
                System.out.println("Service charge imposed: " + serviceCharge);
            }
        } else {
            System.out.println("Insufficient balance for withdrawal");
        }
    }
}

public class Bank {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        SavAcct savAcc = new SavAcct("Alice", 12345, 1000, 5);
        CurAcct curAcc = new CurAcct("Bob", 67890, 2000, 500, 50);

        System.out.println("Choose Account Type:\n1. Savings Account\n2. Current
Account");
        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                System.out.println("Savings Account Selected");
                savAcc.deposit(500);
                savAcc.computeAndDepositInterest();
                savAcc.withdraw(300);
                savAcc.displayBalance();
                break;

            case 2:
                System.out.println("Current Account Selected");
                curAcc.deposit(500);
                curAcc.withdraw(1800);
                curAcc.displayBalance();
                break;
```

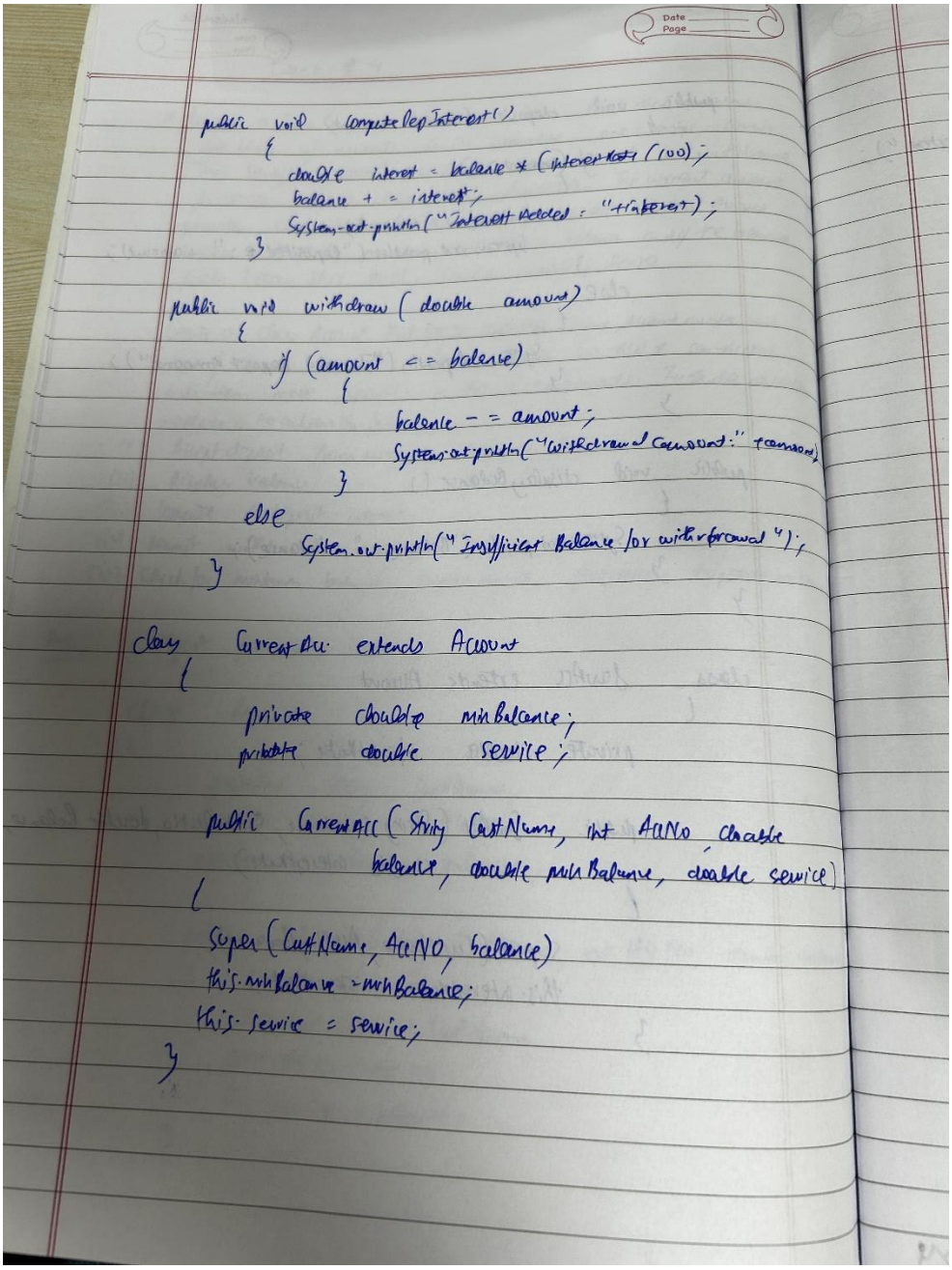
```
            default:
                System.out.println("Invalid choice");
        }
         System.out.println("J S AMOGH KRISHNA 1BM23CS029");
        sc.close();
    }
}
```

Notebook:

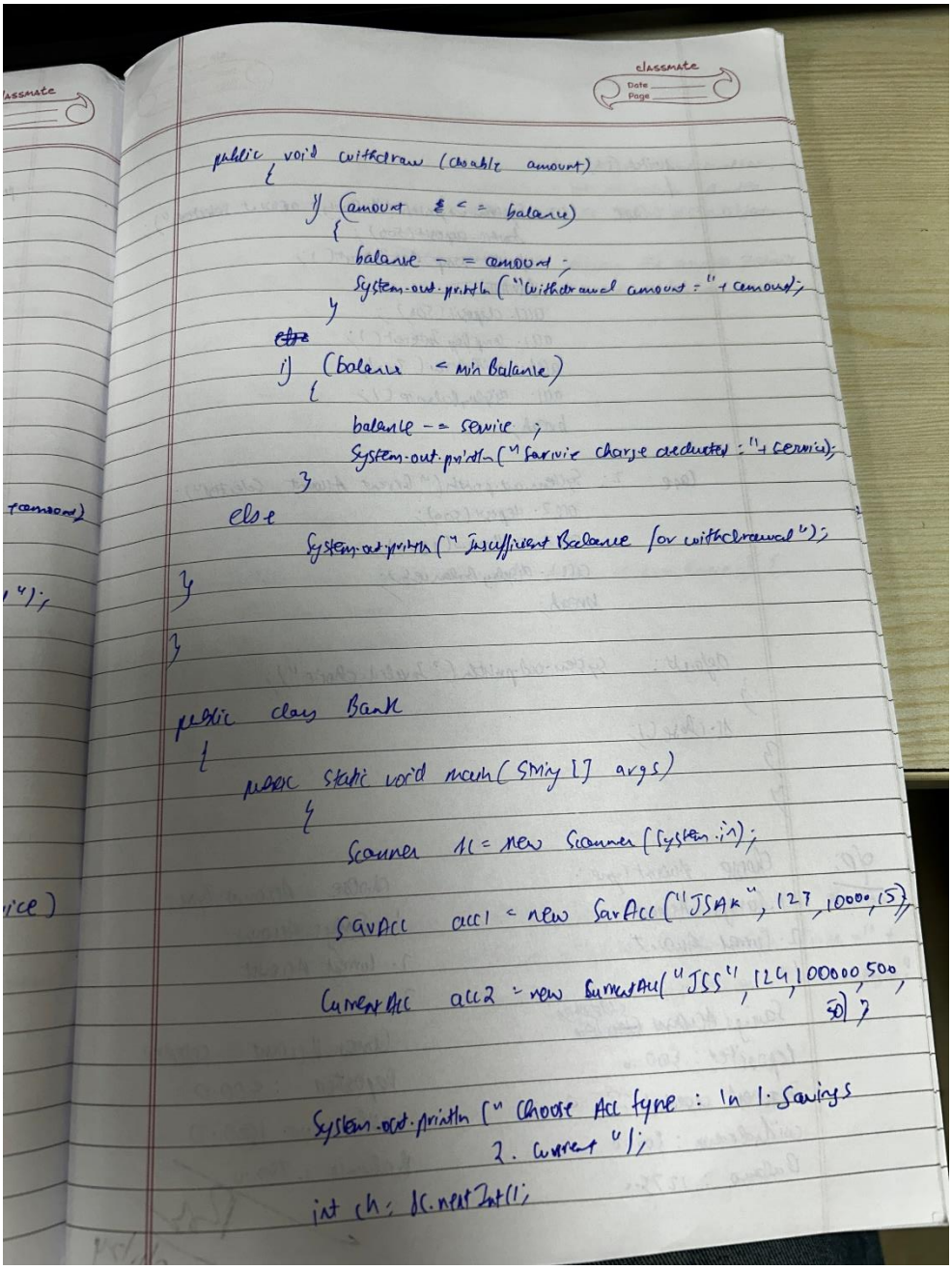




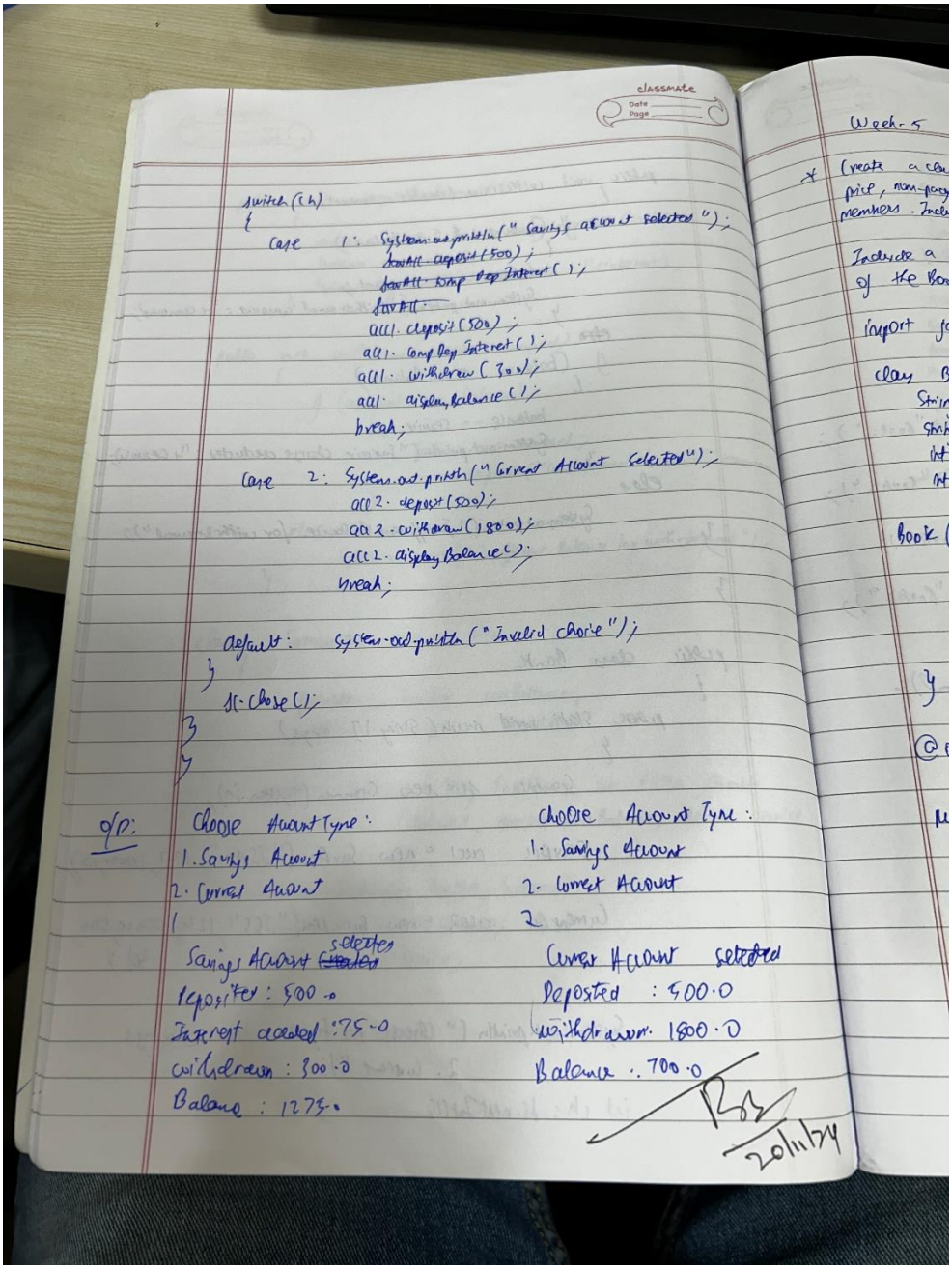
```
public void deposit (double amount)
{
    if (amount > 0)
    {
        balance += amount;
        System.out.println ("Deposited ₹ :" + amount);
    }
    else
    {
        System.out.println ("Invalid Deposit amount");
    }
}

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

class SavAcc extends Account
{
    private double interestRate;

    public SavAcc (String custName, int AccNo, double Balance,
                   double interestRate)
    {
        super (custName, AccNo, Balance);
        this.interestRate = interestRate;
    }
```



```
public void computeDepInterest()
    {
    double interest = balance * (interestRate/100);
    balance += interest;
    System.out.println("Interest added : "+interest);
    }

public void withdraw (double amount)
    {
    if (amount <= balance)
        {
            balance -= amount;
            System.out.println("withdrawal Amount:" +amount);
        }
    else
        System.out.println("Insufficient Balance for withdrawal");
    }

class CurrentAcc extends Account
{
    private double minBalance;
    private double service;

    public CurrentAcc (String CustName, int AccNo, double
            balance, double minBalance, double service)
    {
    super (CustName, AccNO, balance)
    this.minBalance = minBalance;
    this.service = service;
    }
```



```
public void withdraw (double amount)
{
    if (amount <= balance)
    {
        balance -= amount;
        System.out.println("Withdrawal amount : " + amount);
    }
    if (balance < minBalance)
    {
        balance -= servic ;
        System.out.println("Servic charge deducted : " + servic);
    }
    else
        System.out.println("Insufficient Balance for withdrawal");
}
}

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

        SavAcc acc1 = new SavAcc("JSAK", 127, 10000, 5);

        CurrentAcc acc2 = new CurrentAcc("JSS", 124, 100000, 500,
                                         50);

        System.out.println("Choose Acc type : \n 1. Savings
                            2. Current ");
        int ch = sc.nextInt();
```



```
switch(ch)
{
  case 1: System.out.println("Savings account selected");
          acc1.deposit(500);
          acc1.CompDepInterest();
          break;
          acc1.deposit(500);
          acc1.CompDepInterest();
          acc1.withdraw(300);
          acc1.displayBalance();
          break;

  case 2: System.out.println("Current Account Selected");
          acc2.deposit(500);
          acc2.withdraw(1800);
          acc2.displayBalance();
          break;

  default: System.out.println("Invalid choice");
  }
  sc.close();
}
}
```

o/p:

```
Choose Account Type:
1. Savings Account
2. Current Account
1
Savings Account selected
Deposited : 500.0
Interest created : 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
```

20/11/24

Output:

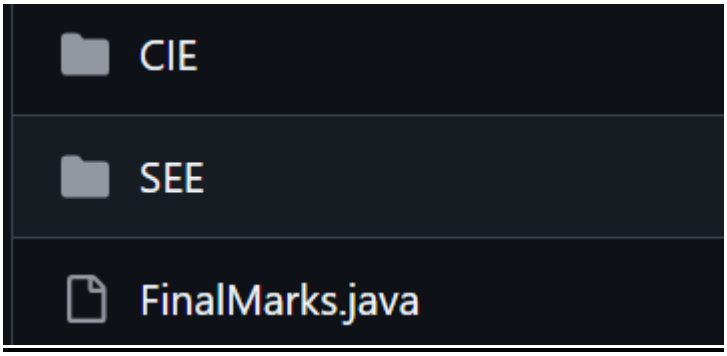
```
Choose Account Type:
1. Savings Account
2. Current Account
1
Savings Account Selected
Deposited: 500.0
Interest added: 75.0
Withdrawn: 300.0
Balance: 1275.0
```

```
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.

File Organization:



Code:

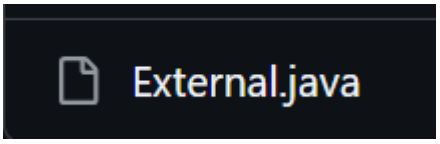
Path: Week6/CIE:


```
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:



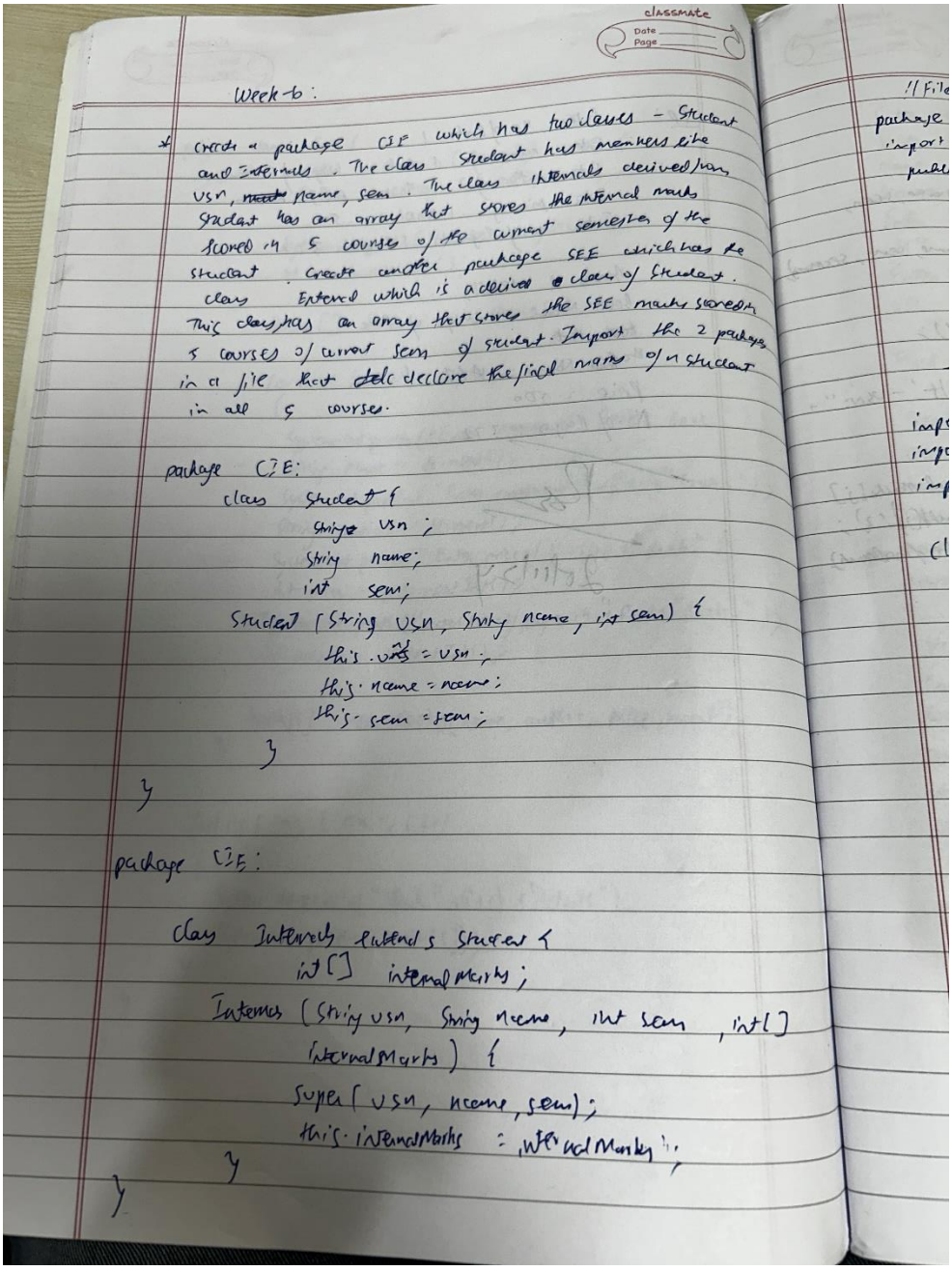

```
package SEE;
import CIE.Student;

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("J S AMOGH KRISHNA 1BM23CS029");
        scanner.close();
    }
}
```

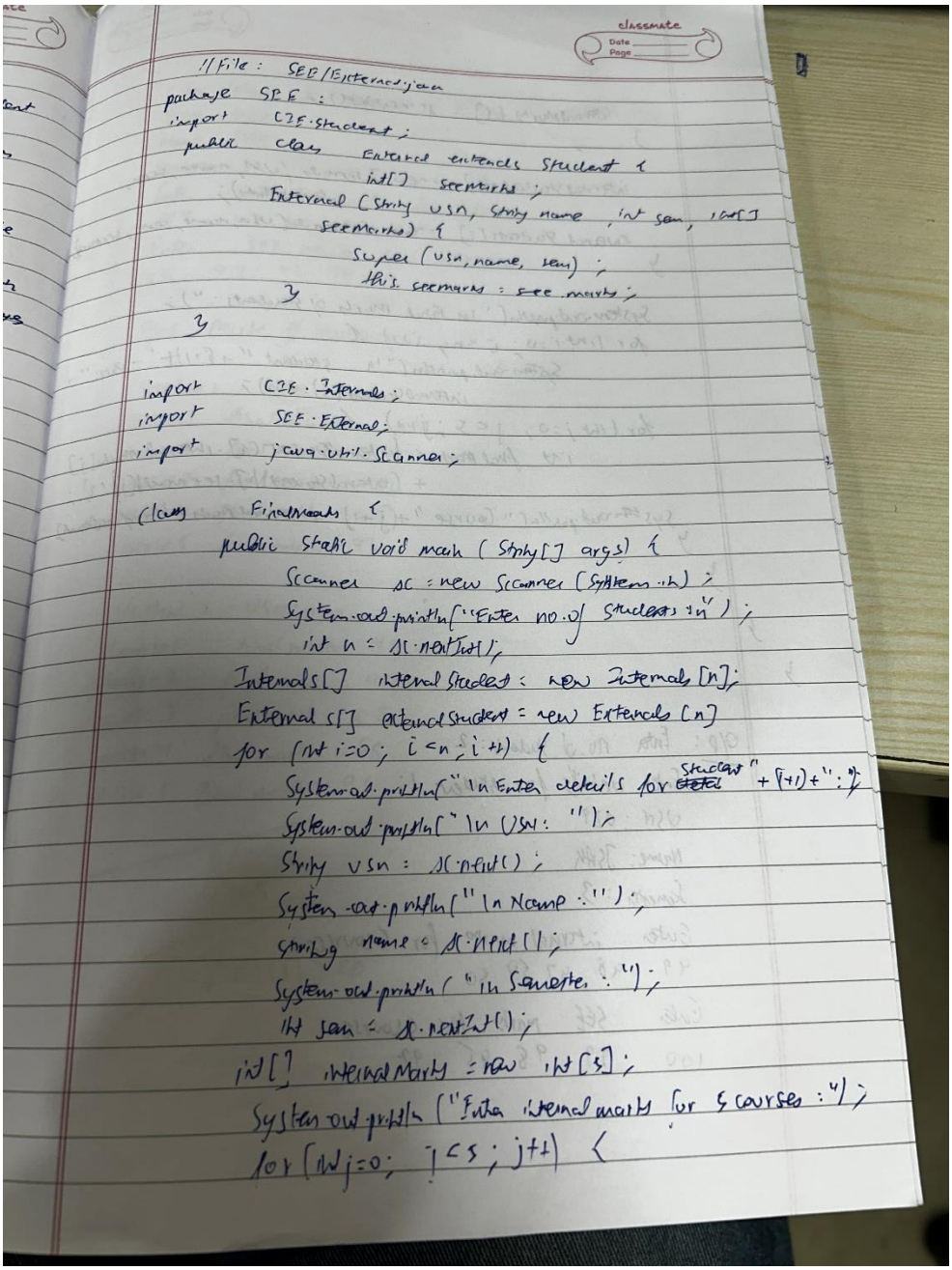
Notebook:



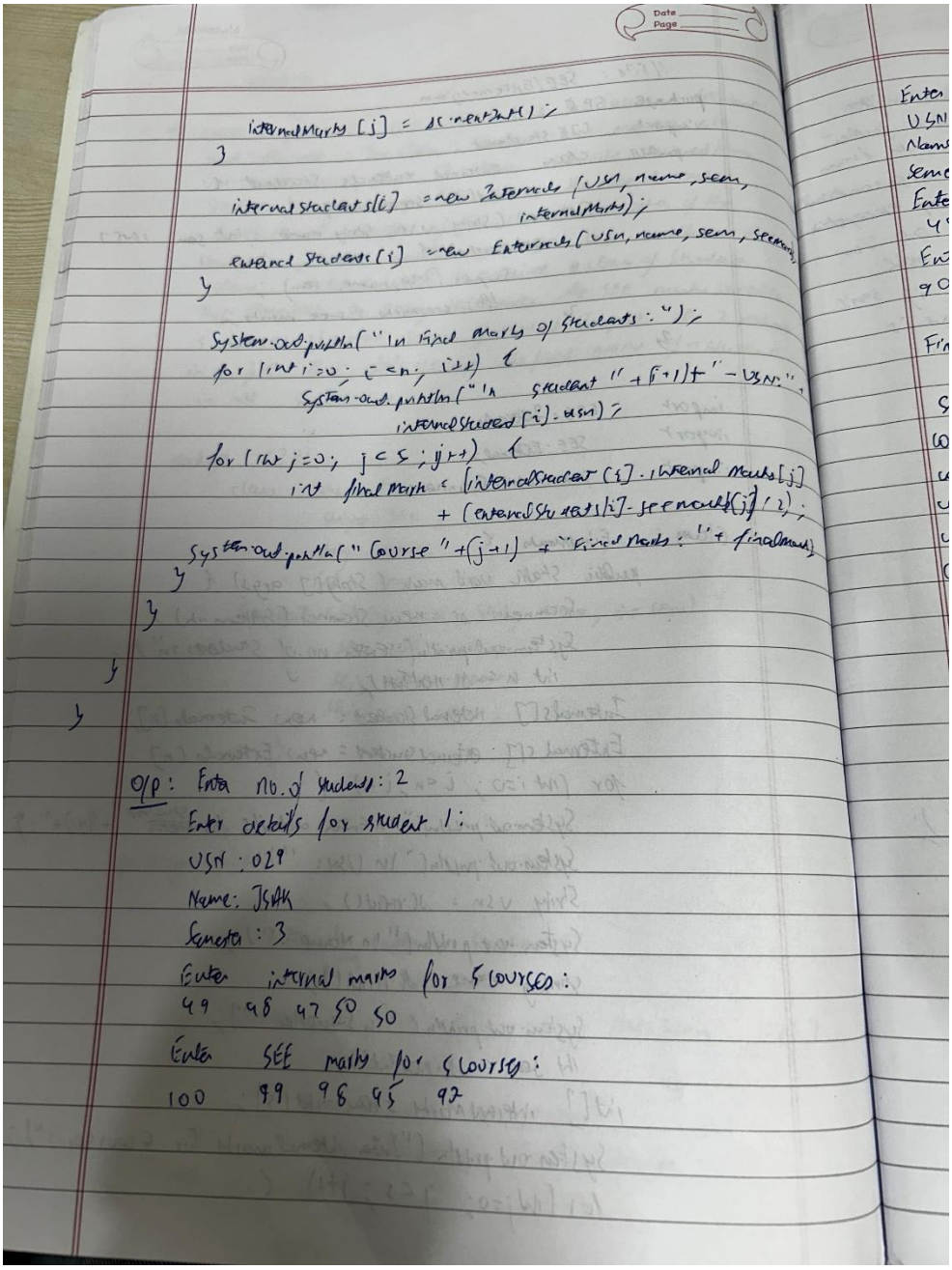
Week-6:

* Create a package CIE which has two classes - Student and Internals. The class Student has members like usn, name, sem. The class Internals derived from Student has an array that stores the internal marks scored in 5 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 5 courses of current sem of student. Import the 2 packages in a file that declare the final marks of a student in all 5 courses.

```
package CIE:
    class Student {
        String usn;
        String name;
        int sem;
        Student (String usn, String name, int sem) {
            this.usn = usn;
            this.name = name;
            this.sem = sem;
        }
    }

package CIE:

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

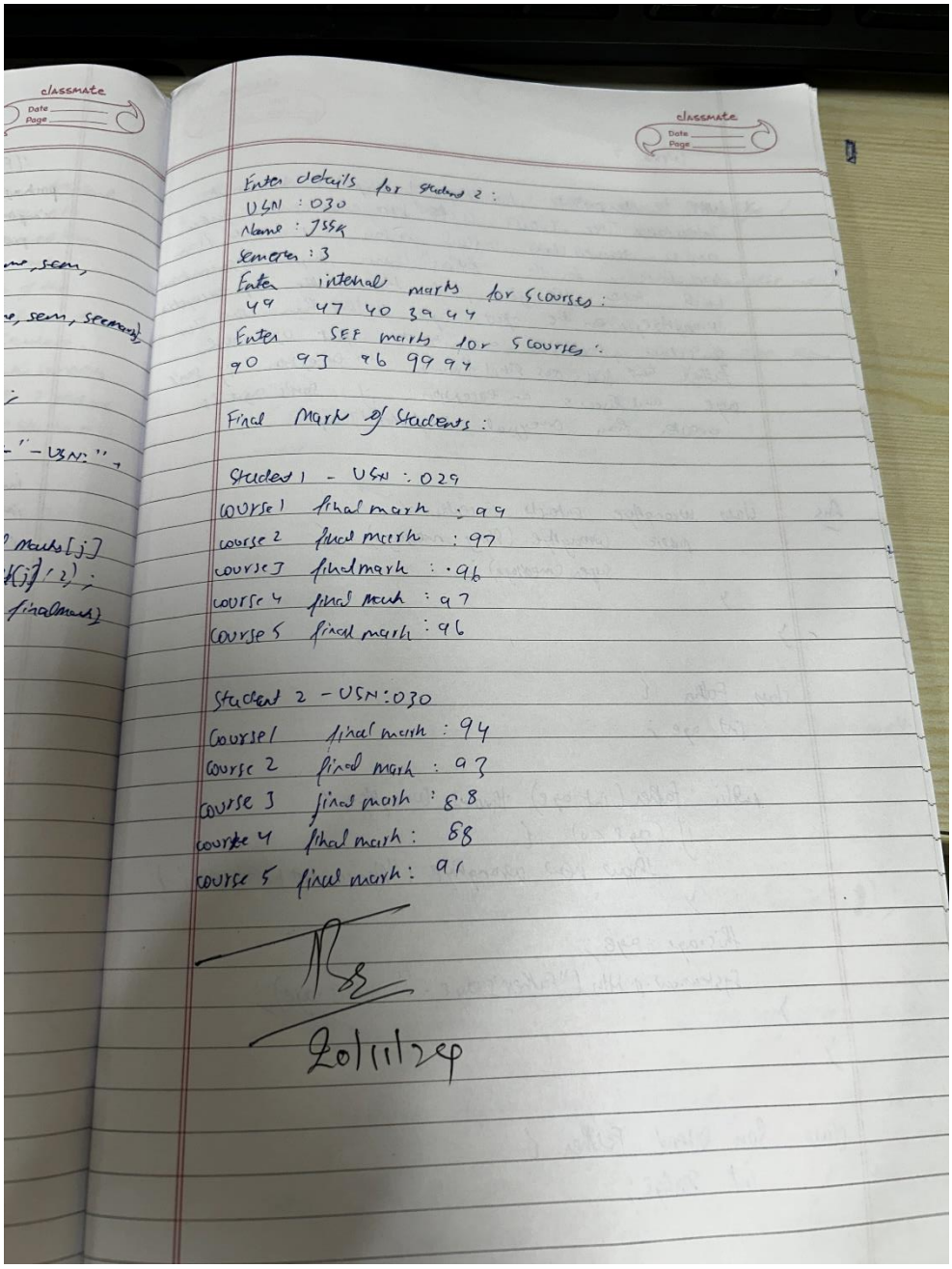


```
// File : SEE/External.java
package SEE ;
import CIE.Student ;
public class External extends Student {
        int[] seemarks ;
    External (String USN, String name, int sem, int[]
            seeMarks) {
                super (usn, name, sem) ;
                this.seemarks = see marks ;
            }
}

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

class FinalMarks {
public static void main (String[] args) {
    Scanner sc = new Scanner (System.in) ;
    System.out.println("Enter no. of students \n") ;
     int n = sc.nextInt() ;
Internals[] internalStudent = new Internals [n] ;
External[] externalStudent = new Externals [n]
for (int i=0 ; i<n ; i++) {
    System.out.println("\n Enter details for Student " + (i+1) + ": ") ;
    System.out.println("\n USN: ") ;
    String usn = sc.next() ;
    System.out.println("\n Name :") ;
    String name = sc.next() ;
    System.out.println("\n Semester :") ;
    int sem = sc.nextInt() ;
int[] internalMarks = new int[5] ;
System.out.println("Enter internal marks for 5 courses :\n") ;
for (int j=0 ; j<5 ; j++) {
```



```
internalMarks[j] = sc.nextInt();
}
internalStudents[i] = new Internals(usn, name, sem, internalMarks);
externalStudents[i] = new Externals(usn, name, sem, seemarks);
}
System.out.println("In final marks of students:");
for(int i=0; i<n; i++) {
    System.out.println("In student "+(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 Marks: "+finalMark);
    }
}
}
```

O/p: Enter no. of students: 2
Enter details for student 1:
USN: 029
Name: Jsdfk
Semester: 3
Enter internal marks for 5 courses:
49 48 47 50 50
Enter SEE marks for 5 courses:
100 99 98 95 92



Enter details for student 2:
USN : 030
Name : JSSK
Semester : 3
Enter internal marks for 5 courses:
44 47 40 39 44
Enter SEE marks for 5 courses:
90 93 96 99 94

Final Marks of Students:

Student 1 - USN : 029
Course 1 final mark : 94
course 2 final mark : 97
course 3 final mark : 96
course 4 final mark : 97
Course 5 final mark : 96

Student 2 - USN : 030
Course 1 final mark : 94
Course 2 final mark : 93
Course 3 final mark : 88
course 4 final mark : 88
course 5 final mark : 91

20/11/24

Output:

```
Enter number of students: 2

Enter details for student 1:
USN: 029
Name: JSAK
Semester: 3
Enter internal marks for 5 courses:
49 48 47 50 50
Enter SEE marks for 5 courses:
100 99 98 95 92

Enter details for student 2:
USN: 030
Name: JSSK
Semester: 3
Enter internal marks for 5 courses:
49 47 40 39 44
Enter SEE marks for 5 courses:
90 93 96 99 94

Final Marks of Students:

Student 1 - USN: 029
Course 1 Final Mark: 99
Course 2 Final Mark: 97
Course 3 Final Mark: 96
Course 4 Final Mark: 97
Course 5 Final Mark: 96

Student 2 - USN: 030
Course 1 Final Mark: 94
Course 2 Final Mark: 93
Course 3 Final Mark: 88
Course 4 Final Mark: 88
Course 5 Final Mark: 91
```

Program-7:

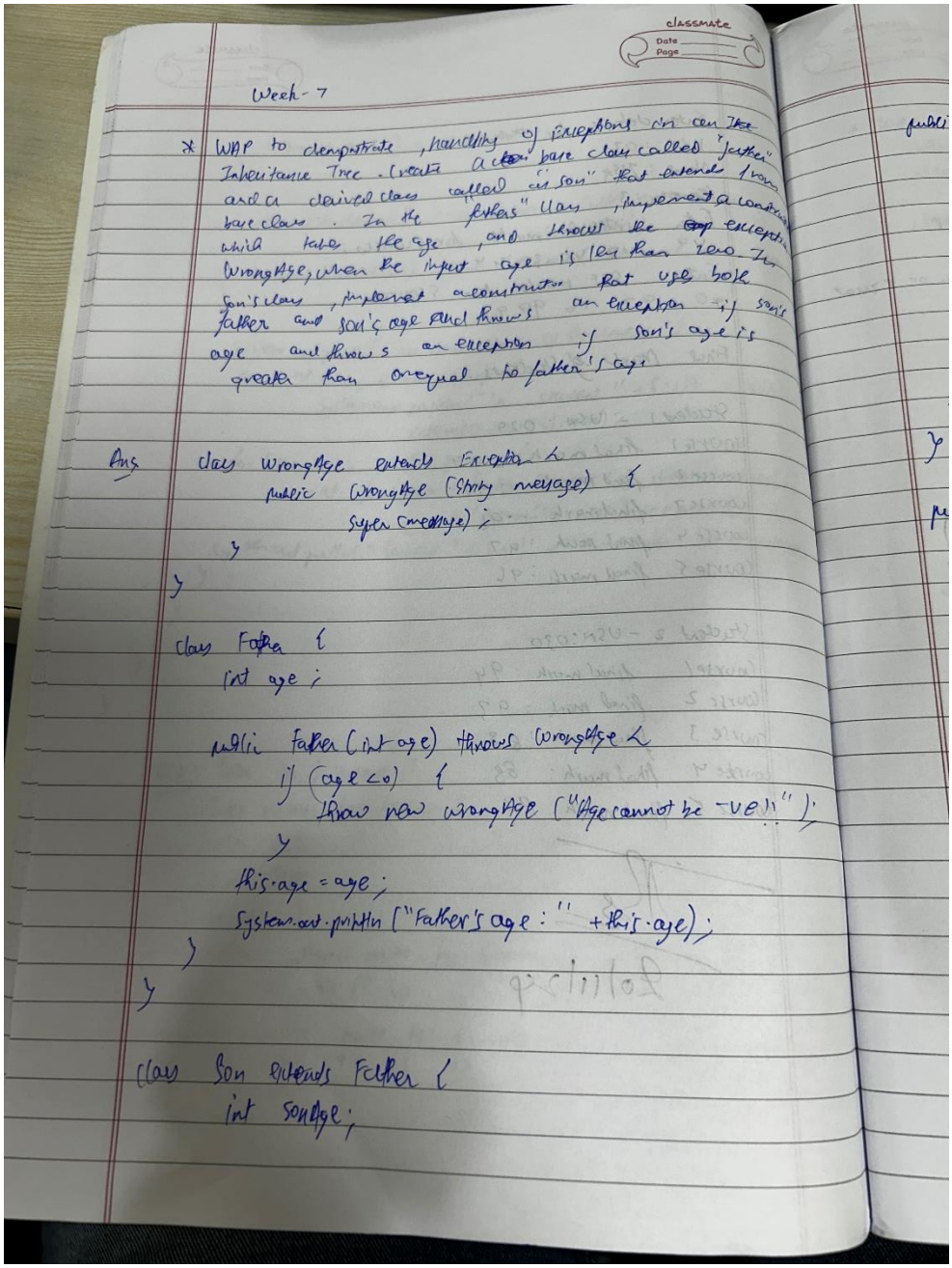
Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called “Father” and derived class called “Son” which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age=father’s 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("J S AMOGH KRISHNA 1BM23CS029");
        scanner.close();
    }
}
```

Notebook:

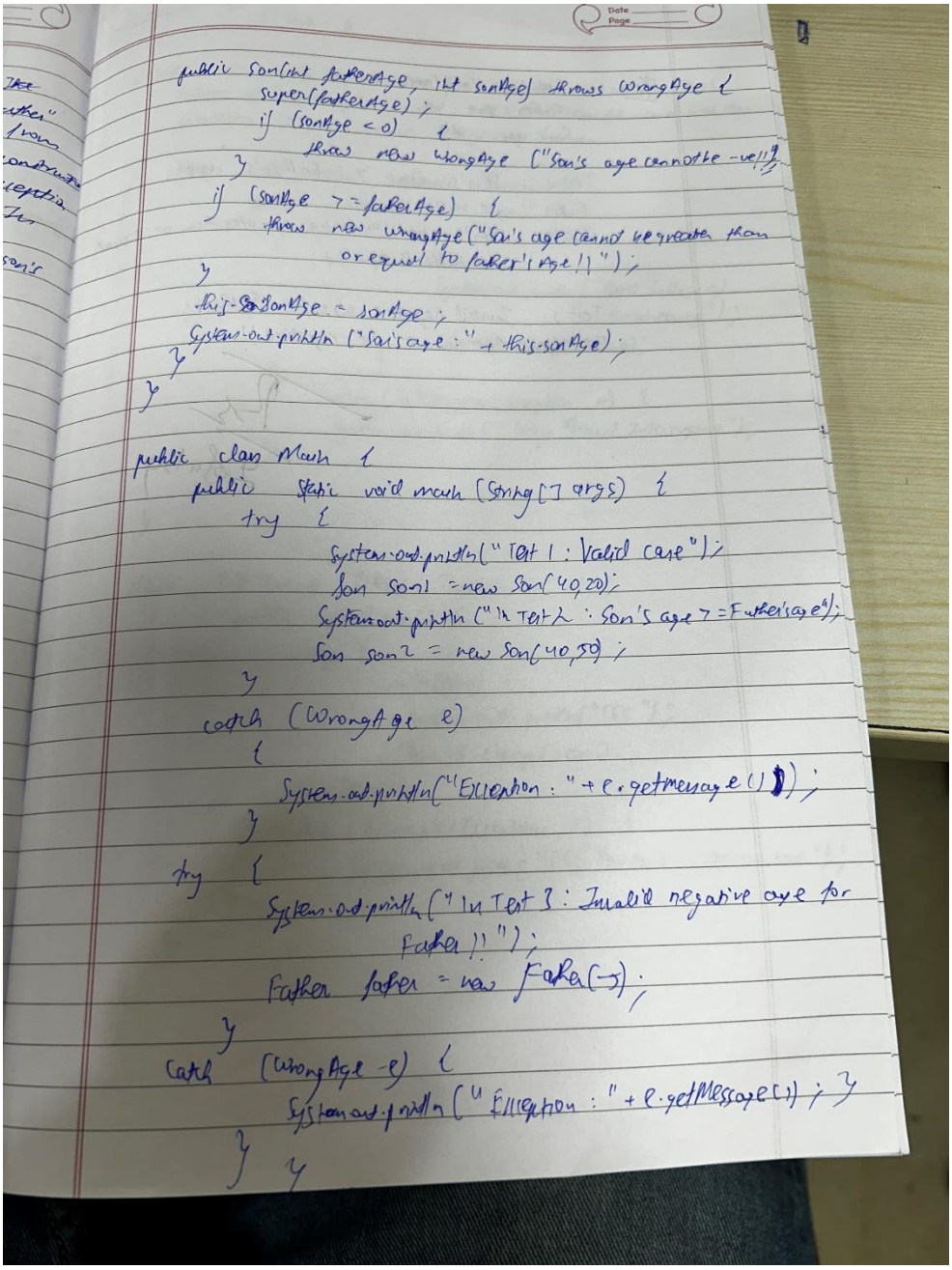


Week - 7

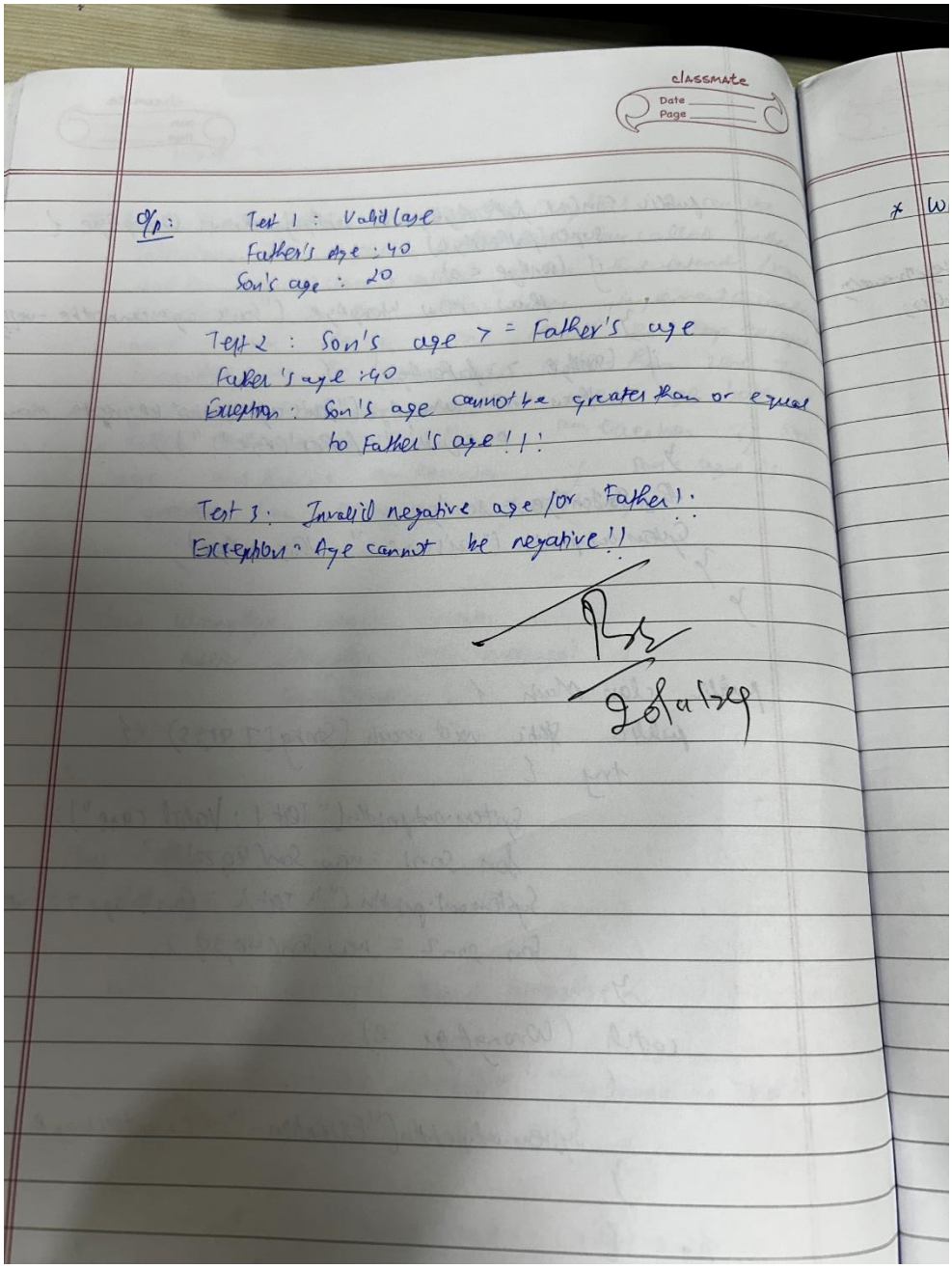
* WAP to demonstrate handling of exceptions in one Inheritance Tree. Create a base class called "Father" and a derived class called "Son" that extends the base class. In the Father's class, implement a constructor which take the age, and throws the exception WrongAge, when the input age is less than zero. In Son's class, implement a constructor that use both Father and Son's age and throws an exception if Son's age is greater than or equal to Father's age.

Ans

```
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 -ve!!");
        }
        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 -ve!!");
    }
    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 Main {
    public static void main (String[] args) {
        try {
            System.out.println("Test 1 : Valid case");
            Son son1 = new Son(40,20);
            System.out.println("\n Test 2 : Son's age >= Father's age");
            Son son2 = new Son(40,50);
        }
        catch (WrongAge e)
        {
            System.out.println("Exception : " + e.getMessage());
        }
        try {
            System.out.println("\n Test 3 : Invalid negative age for
                Father!! ");
            Father father = new Father(-5);
        }
        catch (WrongAge e) {
            System.out.println("Exception : " + e.getMessage()); }
        }
    }
```



Output:

```
Enter Father's Age: -1
Enter Son's Age: 10
Exception: Age Cannot be Negative
```

```
Enter Father's Age: 25
Enter Son's Age: -3
Father's Age: 25
Exception: Son's Age Cannot be Negative
```

```
Enter Father's Age: 25
Enter Son's Age: 30
Father's Age: 25
Exception: Son's Age Cannot be Greater than or Equal to Father's Age
```

```
Enter Father's Age: 29
Enter Son's Age: 9
Father's Age: 29
Son's Age: 9
```

Program-8:

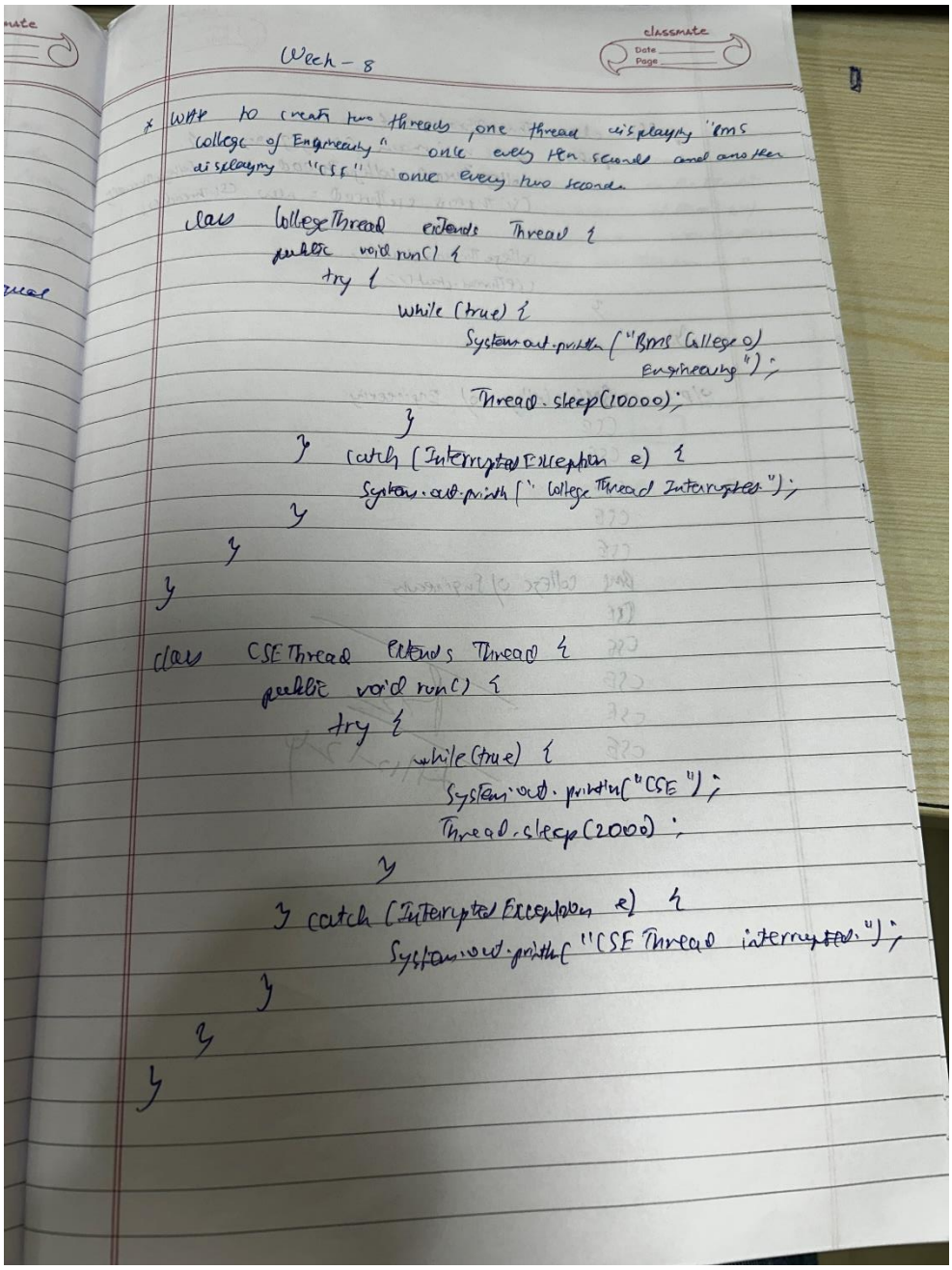
Write a program which creates two threads, one thread displaying “BMS College of Engineering” once every ten seconds and another displaying “CSE” once every two seconds.

Code:

```
class CollegeThread extends Thread {
    public void run() {
        try {
            for (int i = 0; i < 5; i++) {
                System.out.println("BMS College of Engineering");
                Thread.sleep(10000);
            }
        } catch (InterruptedException e) {
            System.out.println("CollegeThread interrupted.");
        }
    }
}

class CSEThread extends Thread {
    public void run() {
        try {
            for (int i = 0; i < 25; i++) {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            System.out.println("CSEThread interrupted.");
        }
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        CollegeThread collegeThread = new CollegeThread();
        CSEThread cseThread = new CSEThread();
        System.out.println("J S AMOGH KRISHNA 1BM23CS029");
        collegeThread.start();
```

```
        cseThread.start();
    }
}
```

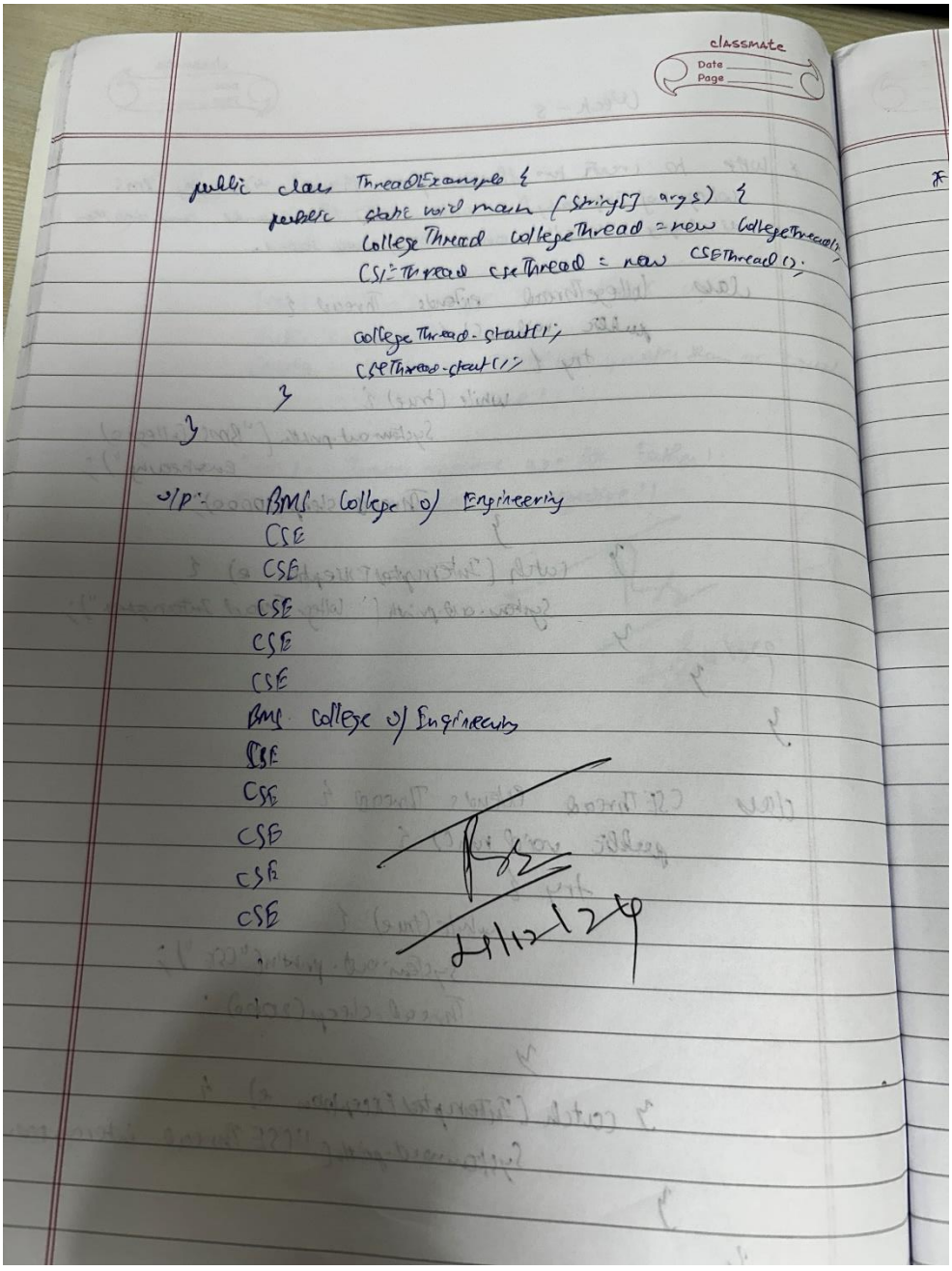
Notebook:



Week - 8

* Write to create two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

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

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



```
public class ThreadExample {
    public static void main (String[] args) {
        CollegeThread collegeThread = new CollegeThread();
        CSEThread cseThread = new CSEThread();

        collegeThread.start();
        cseThread.start();
    }
}
```

o/p:

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

21/12/24

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.

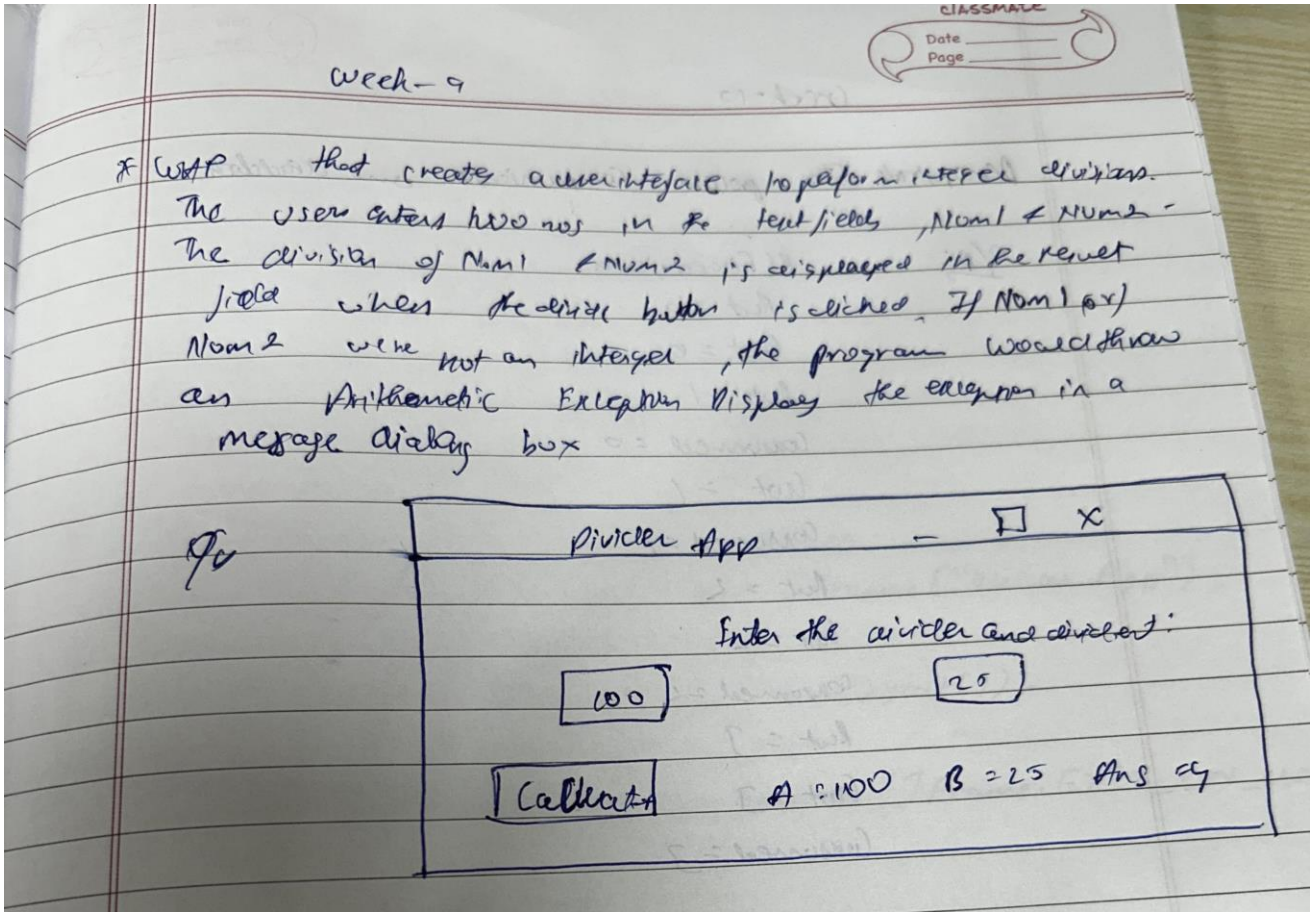
Code:

```
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!");
```

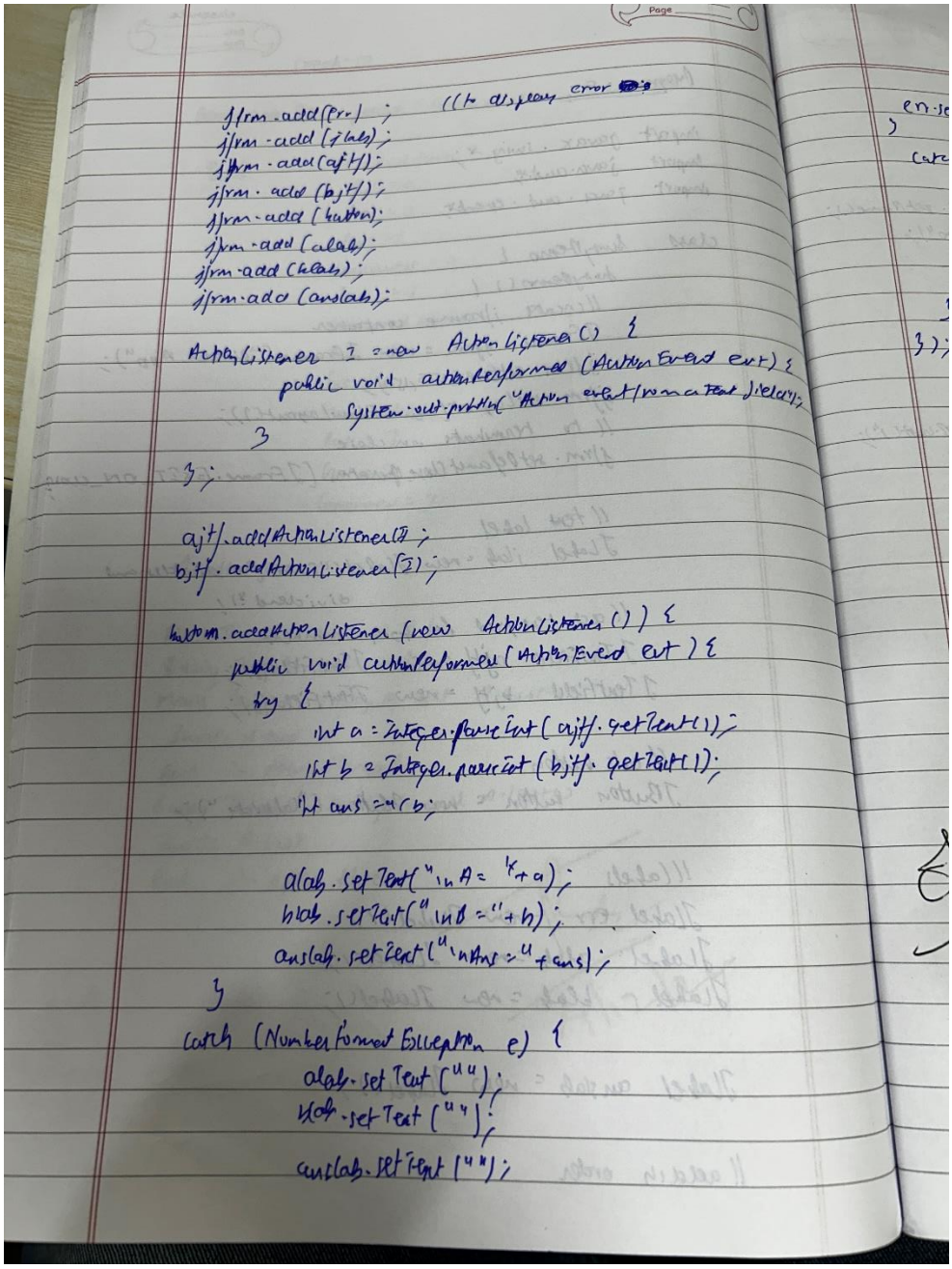
```
            }
        }
    });

    // 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("J S AMOGH KRISHNA 1BM23CS029");
  }
}
```

Notebook:

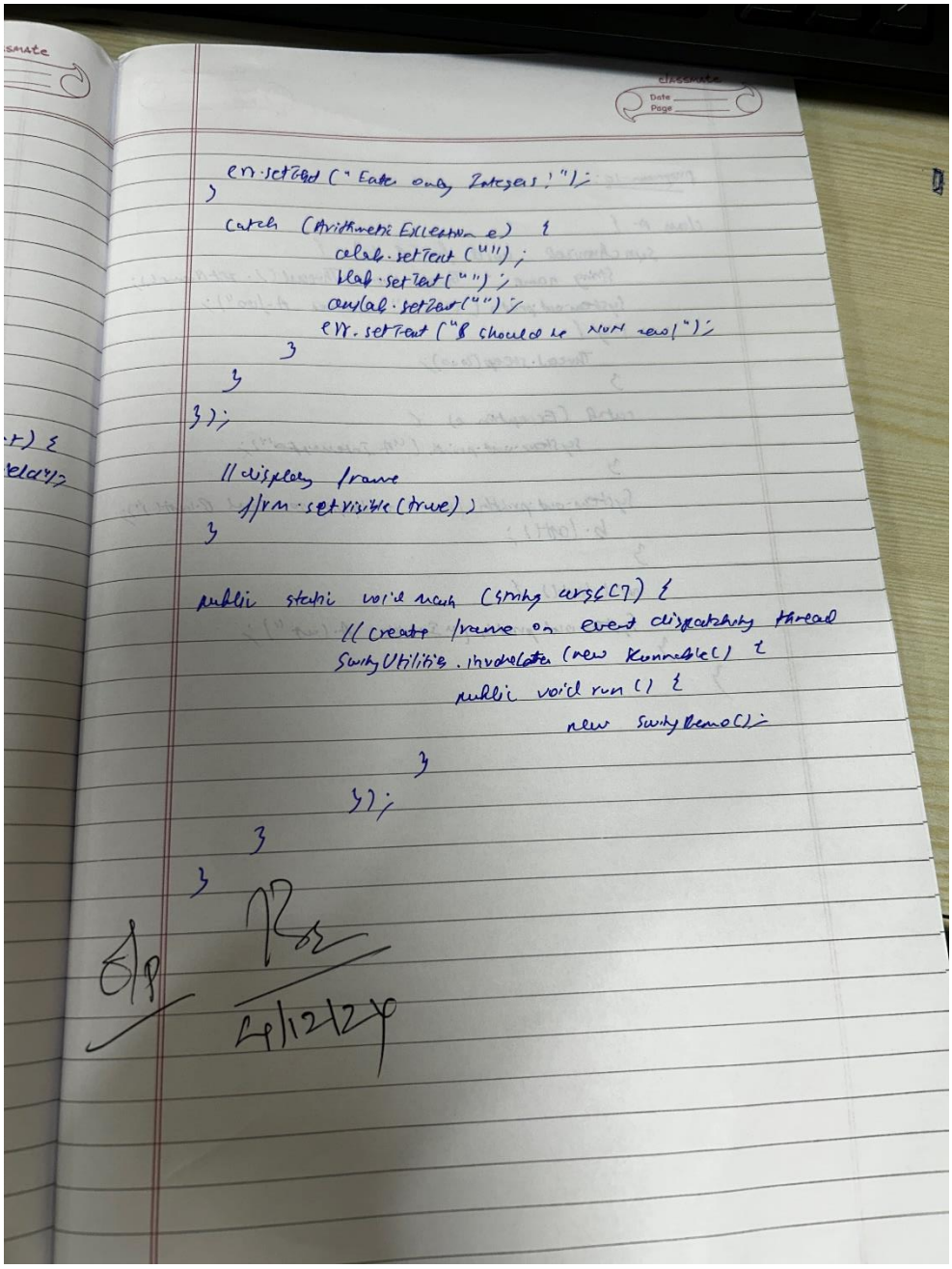

Week-9
Date
Page
Divider App
Enter the dividend and divisor:
100
25
Calculate
A = 100 B = 25 Ans =

Program - 9 :

```
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());
        // to terminate on close
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // text label
        JLabel jlab = new JLabel("Enter the divider and
                                  dividend :");
        // adding text field for both numbers
        JTextField ajtf = new JTextField(8);
        JTextField bjtf = new JTextField(8);

        // calc button
        JButton button = new JButton("Calculate");

        // labels
        JLabel err = new JLabel();
        JLabel alab = new JLabel();
        JLabel blab = new JLabel();

        JLabel anslab = new JLabel();

        // adding order
```



```
jfrm.add(fr);      //to display error
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

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("Sum= " + ans);
  }
  catch (NumberFormatException e) {
      alab.setText("");
      blab.setText("");
      anslab.setText("");
```

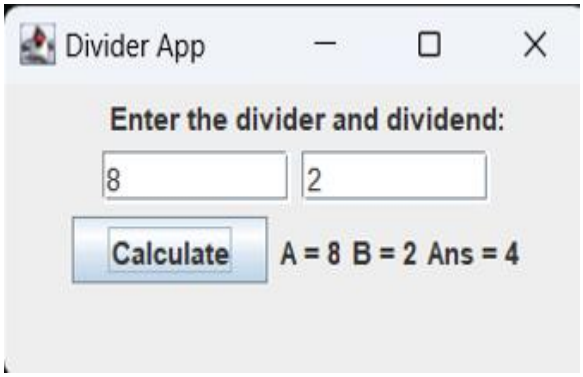


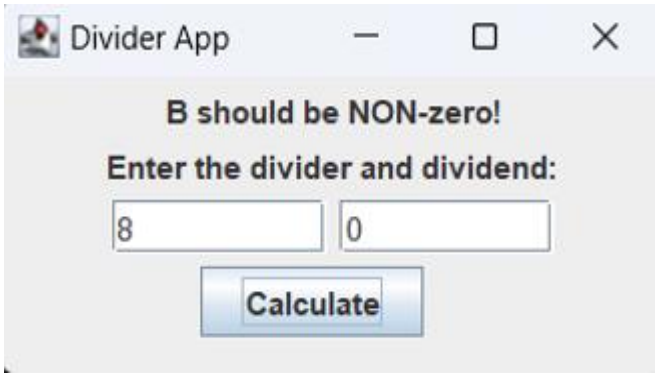
```
        err.setText("Enter only Integers!");
    }
    catch (ArithmeticException e) {
            celab.setText("");
            flab.setText("");
            cmylab.setText("");
            err.setText("B should be non zero!");
        }
    }
});

// display frame
//frm.setVisible(true);
}

public static void main(String args[]) {
        // create frame on event dispatching thread
        SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                        new SwingDemo();
                }
        });
}
}
```

Op

4/12/24

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("J S AMOGH KRISHNA 1BM23CS029");
    }
}
```

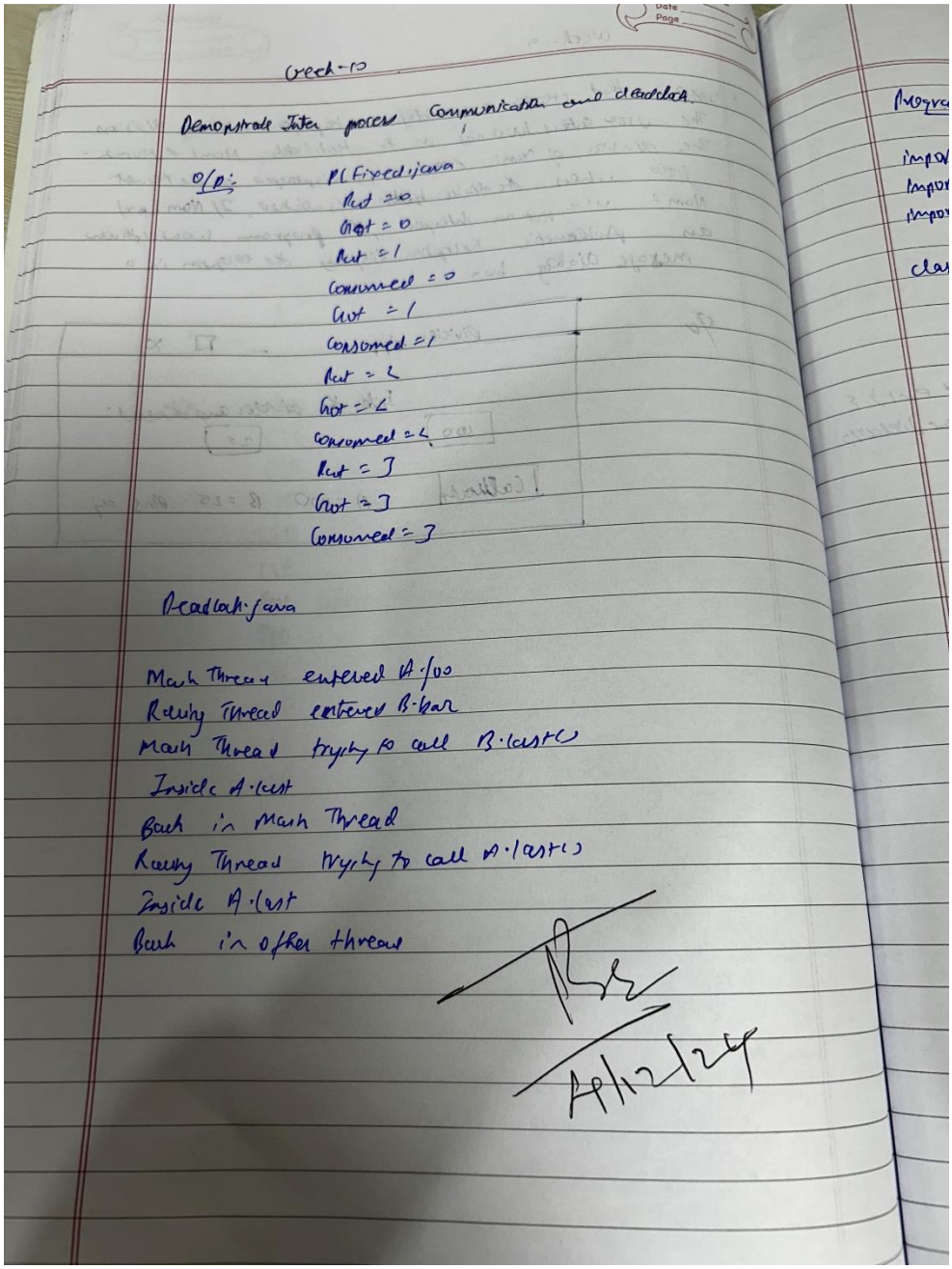
Code-2:

```
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("J S AMOGH KRISHNA 1BM23CS029");
    }
}
```

Notebook:



Prog-10

Demonstrate Inter process Communication and deadlock

O/P:

PC Fixed.java
Put = 0
Get = 0
Put = 1
Consumed = 0
Get = 1
Consumed = 1
Put = 2
Get = 2
Consumed = 2
Put = 3
Get = 3
Consumed = 3

Deadlock.java

Main Thread entered A.foo
Racing Thread entered B.bar
Main Thread trying to call B.last()
Inside A.last
Back in main Thread
Racing Thread trying to call A.last()
Inside A.last
Back in other thread

A 9/12/24

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
```