VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Aishwarya R (1BM23CS018)

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

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by **Aishwarya R (1BM23CS018),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Lab faculty Incharge Name Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Github Link:

https://github.com/Aishnir/Javalab

Program 1
Implement Quadratic Equation

Algorithm:

WEEK 1

Date 9/10/24
Page

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

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

```
r2 = (-b) - (Math.sqrt(d)) / (double) (2*a);
System.out.println("Roots are " + r1 + "and" + r2);
}
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("Roots are" + r1 + "i" + r2 +
"and" + r1 + "-i" + r2);
}

else
{
System.out.println("Roots are Equal");
r1 = (-b) / (2.0*a);
System.out.println("Roots are" + r1 + "and"
" + r2);
}
}
else }
```

O/p

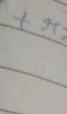
Enter the values for a b and c :
2 3 4
Roots are imaginary.
Roots are -0.75 + 1.19 and -0.75 - i 1.19

1 -7 10
Roots are 5.0 and 2.0

1 -4 4
roots are equal

Date / /
Page

roots are 2.0 and 2.0



0 c 1 1 0 1 2 4

Code:

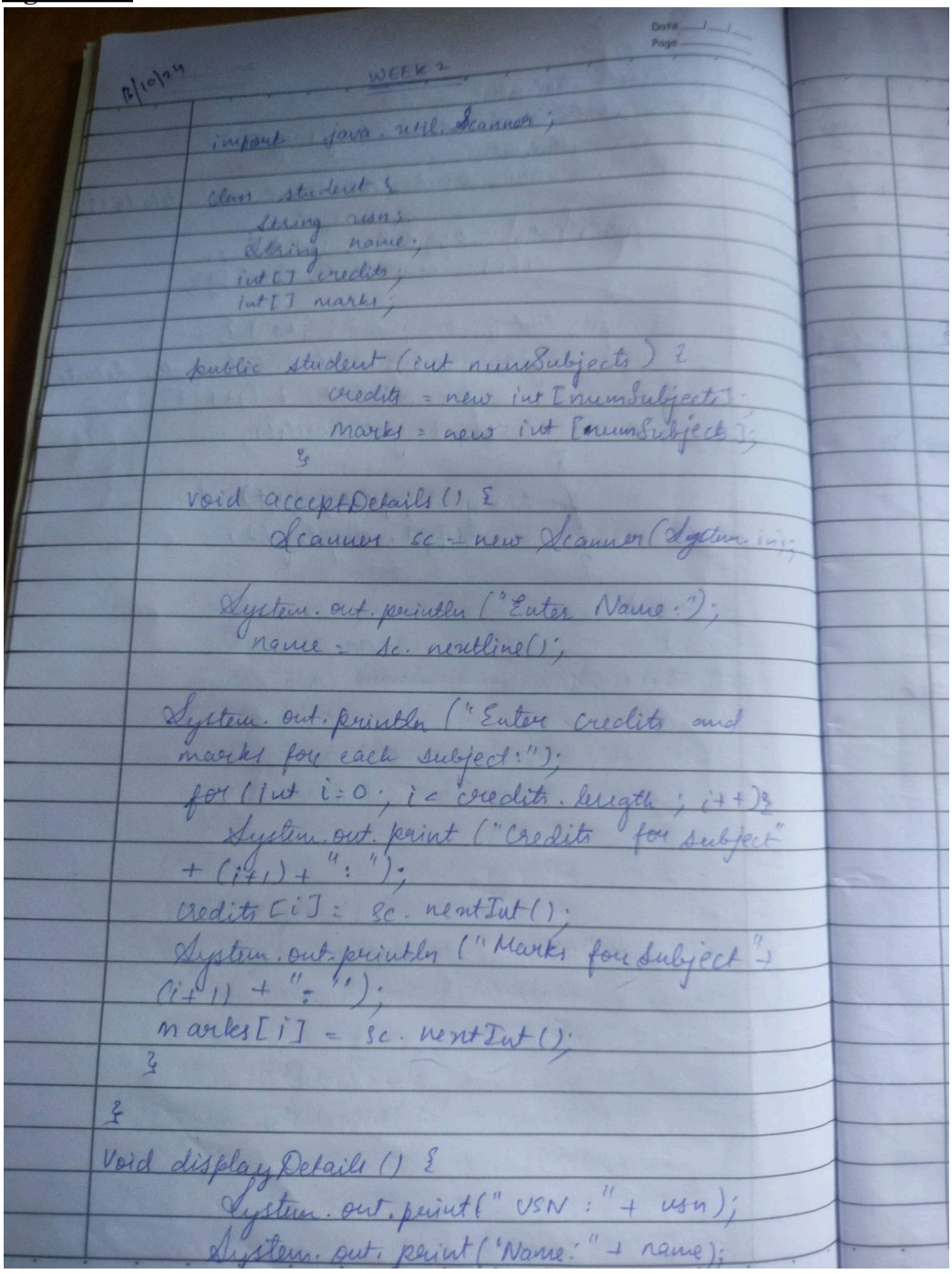
```
import java.util.scanner;
import java.lang.*;
class quad
{
public static void main(Strin args[])
{ double a,ab,c;
double r1,r2,d;
Scanner sc =new Scanner(System.in);
System.out.println("Enter the values for a , b and c:\n);
a=sc.nextDouble();
b=sc.nextDouble();
c=sc.nextDouble();

if (a==0)
{ System.out.println("invalid");
}
d=Math.pow(b,2.0)-4.0*a*c;

if(d>0)
{ r1=((-b) + (Math.sprt(d))/(double)(2.0*a);
System.out.println("roots are"+r1+"and"+r2);
}
else if (d<0)
{r1=(-b)/(2.0*a);
r2=Math.sprt(-d)/(2.0*a);
System.out.println("roots are imaginary");
System.out.println("roots are"+r1+"i"+r2+"and"+r1+"-i"+r2);
}
else
{ System.out.println("roots are equal");
r1=(-b)/(2.0*a);
System.out.println("roots are"+r1+"and"+r2);}
}
}
```

output:

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\1BM23CS018 JAVA> javac QuadraticEquations.java
PS D:\1BM23CS018 JAVA> java QuadraticEquations
Enter a: 2
Enter b: 3
Enter c: 4
aishwarya r
Roots are complex
PS D:\1BM23CS018 JAVA> java QuadraticEquations
Enter a: 1
Enter b: -7
Enter c: 10
aishwarya r
Real Roots
Root 1: 5.0
Root 2: 2.0
PS D:\1BM23CS018 JAVA> java QuadraticEquations
Enter a: 1
Enter b: -4
Enter c: 4
aishwarya r
Roots are real and equal
Root: 2.0
PS D:\1BM23CS018 JAVA>
```

Program 2
Calculate SGPA

algorithm:



```
8/10/24                    WEEK 2

import java.util.Scanner;

class student {
    String usn;
    String name;
    int[] credits;
    int[] marks;

public student(int numSubjects) {
        credits = new int[numSubjects];
        marks = new int[numSubjects];
    }

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

    System.out.println("Enter Name:");
    name = sc.nextLine();

System.out.println("Enter credits and
marks for each subject:");
for(int i=0; i< credits.length; i++){
    System.out.print("Credits for subject
+ (i+1) + ": ");
credits[i] = sc.nextInt();
System.out.println("Marks for subject "+
(i+1) + ": ");
marks[i] = sc.nextInt();
}

}

Void displayDetails() {
    System.out.print(" USN : " + usn);
    System.out.print("Name:" + name);
```

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

System.out.printf("SGPA : %.2f %n",
        calculateSGPA());
}

double calculateSGPA() {
        double totalCredits = 0;
        double totalpoints = 0;

for (int i = 0; i < credits.length ; i++) {
    double gradepoint = calculateGradePoint
    (marks[i]);
        totalpoints += gradepoint * credits[i];
        totalCredits += credits[i];
}

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

double calculateGradePoint (int mark) {
    if (mark >= 90) return 10;
else if (mark >= 80) return 9;
else if (mark >= 70) return 8;
else if (mark >= 60) return 7;
else if (mark >= 50) return 6;
else if (mark >= 40) return 5;
else return 0;
}
}
```

```
public class StudentSGPACalculator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of subjects");
        int numSubjects = sc.nextInt();
        sc.nextLine();

        Student student = new Student(numSubjects);
        student.acceptDetails();
        student.displayDetails();
        System.out.println("AishwaryaR" + "Name");
        sc.close();
        System.out.println("Name - Aishwarya\n" +
                "USN - 1BM23CS018");
```

O/P

```
Enter the number of subjects : 3
Enter the number of USN : 1BM23CS018
Enter Name : Aishwarya
Enter credits and marks for each subject :
credits for subject 1 : 4
Marks for subject 1 : 92
credits for subject 2 : 4
Marks for subject 2 : 95
credits for subject 3 : 3
Marks for subject 3 : 86
USN : 1BM23CS018
Name : Aishwarya
Subject1 - credits : 4, Marks : 92
Subject2 - credits : 4, Marks : 95
Subject3 - credits : 3, Marks : 86
```

SGPA : 9.727

Name- Aishwarya

USN - 1bm23cs018

16/10/24

Algorithm:

```
import java.util.Scanner;

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

    // Constructor to initialize the arrays based on the number of subjects
    public Student(int numSubjects) {
        credits = new int[numSubjects];
        marks = new int[numSubjects];
    }

    // Method to accept student details
    void acceptDetails() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter USN: ");
        usn = sc.nextLine();  // Using nextLine to capture full input
        System.out.print("Enter Name: ");
        name = sc.nextLine(); // Using nextLine to capture full input

        System.out.println("Enter credits and marks for each subject:");
        for (int i = 0; i < credits.length; i++) {
            System.out.print("Credits for subject " + (i + 1) + ": ");
            credits[i] = sc.nextInt();
            System.out.print("Marks for subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
        }
    }

    // Method to display student details
    void displayDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        for (int i = 0; i < credits.length; i++) {
            System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i] + ", Marks: " + marks[i]);
        }
        System.out.printf("SGPA: %.2f%n", calculateSGPA());
    }

    // Method to calculate SGPA
```

```
    double calculateSGPA() {
        double totalCredits = 0;
        double totalPoints = 0;

        for (int i = 0; i < credits.length; i++) {
            double gradePoint = calculateGradePoint(marks[i]);
            totalPoints += gradePoint * credits[i];
            totalCredits += credits[i];
        }

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

    // Method to convert marks to grade points (Assuming a simple scale)
    double calculateGradePoint(int mark) {
        if (mark >= 90) return 10;
        else if (mark >= 80) return 9;
        else if (mark >= 70) return 8;
        else if (mark >= 60) return 7;
        else if (mark >= 50) return 6;
        else if (mark >= 40) return 5;
        else return 0; // Fail
    }
}

public class StudentSGPACalculator {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of subjects: ");
        int numSubjects = sc.nextInt();
        sc.nextLine();  // Consume the newline character after the integer input

        Student student = new Student(numSubjects);
        student.acceptDetails();
        student.displayDetails();

        sc.close();
    }
}
```

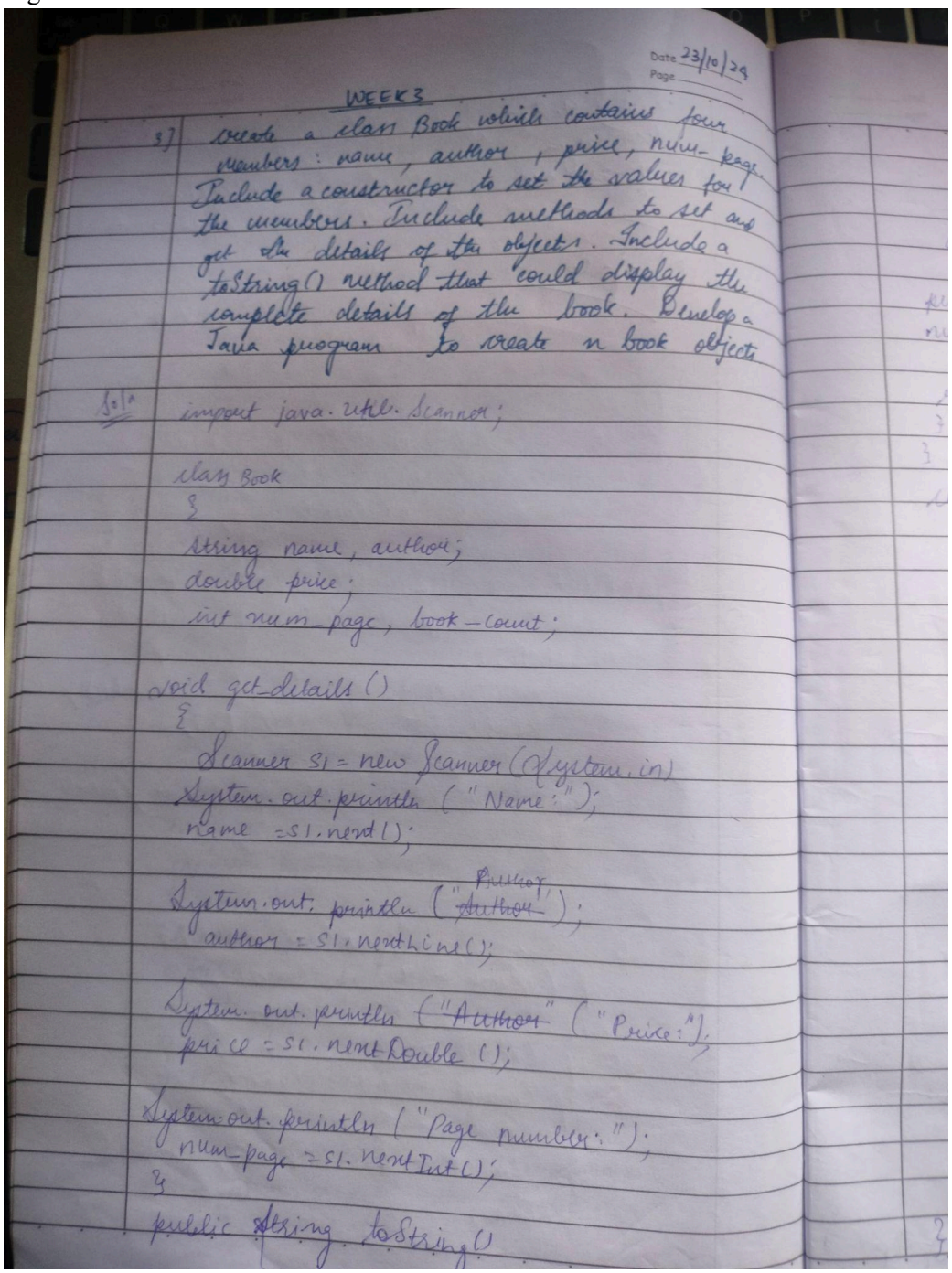
output:

```
D:\java>javac StudentSGPACalculator.java

D:\java>java StudentSGPACalculator
Enter the number of subjects: 3
Enter USN: 1BM23CS018
Enter Name: Aishwarya
Enter credits and marks for each subject:
Credits for subject 1: 4
Marks for subject 1: 92
Credits for subject 2: 4
Marks for subject 2: 95
Credits for subject 3: 3
Marks for subject 3: 86
USN: 1BM23CS018
Name: Aishwarya
Subject 1 - Credits: 4, Marks: 92
Subject 2 - Credits: 4, Marks: 95
Subject 3 - Credits: 3, Marks: 86
SGPA: 9.73
Aishwarya
D:\java>
```

Program 3

N book objects

Algorithm:



WEEK 3

3] create a class Book which contains four members: name, author, price, num_page. 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

Sol^n

```
import java.util.Scanner;

class Book
{
String name, author;
double price;
int num_page, book_count;

void get_details()
{
Scanner S1 = new Scanner(System.in)
System.out.println("Name:");
name = S1.next();

System.out.println("Author");
author = S1.nextLine();

System.out.println("Author" ("Price:");
price = S1.nextDouble();

System.out.println("Page number:");
num_page = S1.nextInt();
}
public String toString()
```

```
    {
        String name, author, price, num_page;
        name = "Name: " + this.name + "\n";
        author = "Author: " + this.author + "\n";
        price = "Price: " + this.price + "\n";
        num_page = "Number of pages: " + this.num_page + "\n";
        return name + author + price + num_page;
    }
}
class demo
{
    public static void main (String[] args)
    {
        Scanner s2 = new Scanner (System.in);
        int n;
        System.out.println ("Number of books: ");
        n = s2.nextInt();
        Book[] b = new Book[n];
        for (int i=0; i<n; i++)
        {
            b[i] = new Book();
            b[i].get_details();
        }
        for (int i=0; i<n; i++)
        {
            System.out.println (b[i].toString());
        }
    }
}
```

Name: Aishwarya.K
USN: 1BM23CS018

Number of books
2

Name:
The Subtle Art of Not giving a fuck
Author:
Mark
Price:
350
Page Number:
520

Name:
Deep work
Author:
Cal
Price:
650
Page Number:
420

Name:
The Subtle Art of Not giving a fuck
Author:
Mark
Price:
350
Page Number:
520

Name:
Deep work
Author:
Cal
Price:
650
Page Number:
420

code:

```
import java.util.Scanner;

class Book
{
String name,author;
double price;
int num_page,book_count;


void get_details()
{
Scanner s1=new Scanner(System.in);
System.out.println("Name: ");
name=s1.nextLine();

System.out.println("Author: ");
author=s1.nextLine();

System.out.println("Price: ");
price=s1.nextDouble();

System.out.println("Page number: ");
num_page=s1.nextInt();

}
public String toString()
{
String name,author,price,num_page;
name="Name: "+this.name+"\n";
author = "Author: " + this.author + "\n";

price = "Price: " + this.price + "\n";

num_page = "Number of pages: " + this.num_page + "\n";

return name + author + price + num_page;
}
}

class dontrun
{
public static void main(String[] args)
{
System.out.println("Name:
Aisharya r");
System.out.println("USN: 1BM23CS018");

Scanner s2=new Scanner(System.in);
int n;
System.out.println("Number of books: ");
```

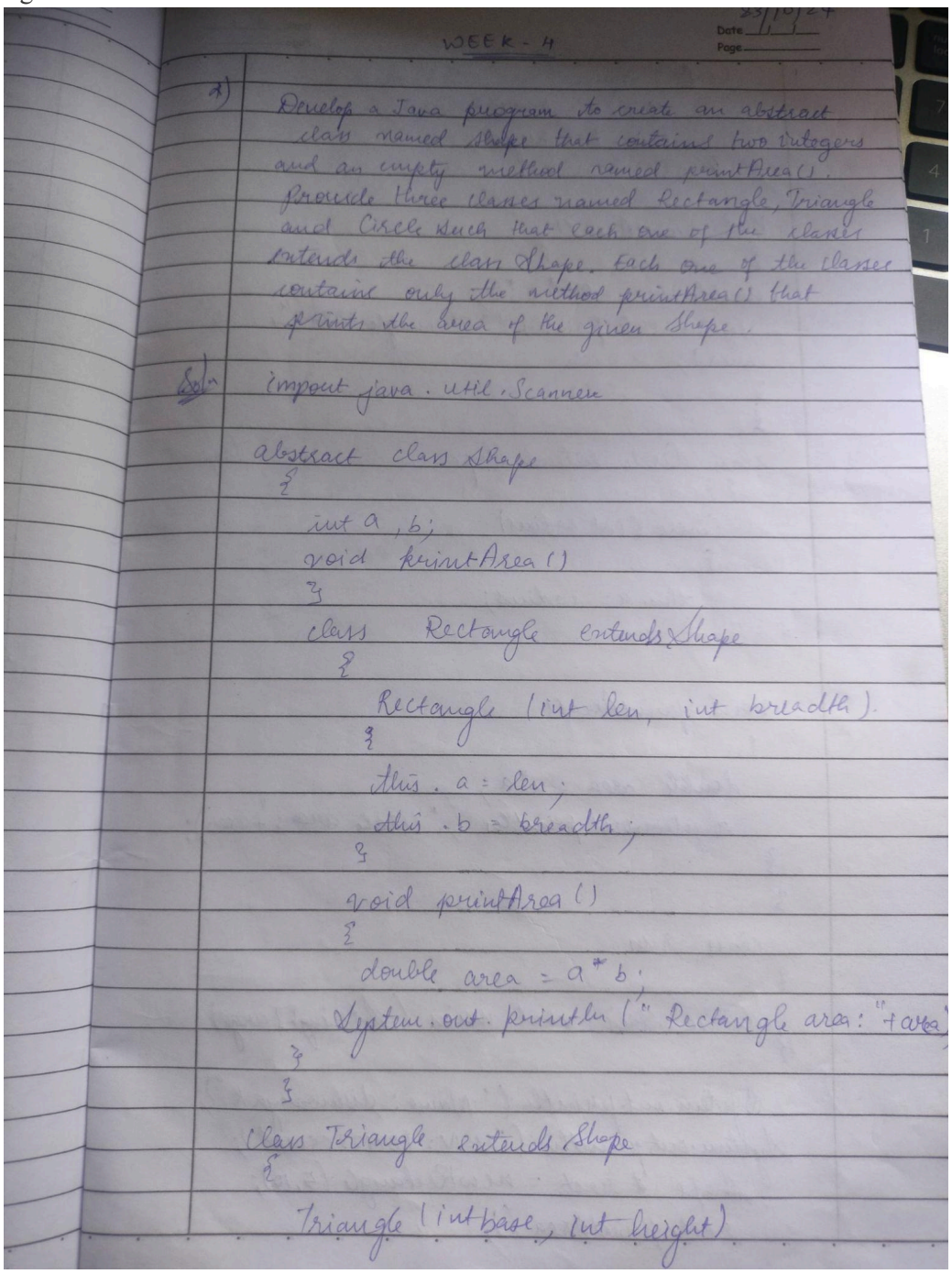
```
n=s2.nextInt();
Book[] b=new Book[n];
for(int i=0;i<n;i++)
{
b[i]=new Book();
b[i].get_details();


}
for(int i=0;i<n;i++)

{
System.out.println(b[i].toString());
}
}
}
```

output:

```
Name: Aisharya r
USN: 1BM23CS018
Number of books:
2
Name:
the subtle art of not giving a fuck
Author:
mark
Price:
350
Page number:
520
Name:
deep work
Author:
lee
Price:
650
Page number:
420
Name: the subtle art of not giving a fuck
Author: mark
Price: 350.0
Number of pages: 520

Name: deep work
Author: lee
Price: 650.0
Number of pages: 420
```

program 4

algorithm:



WEEK - 4

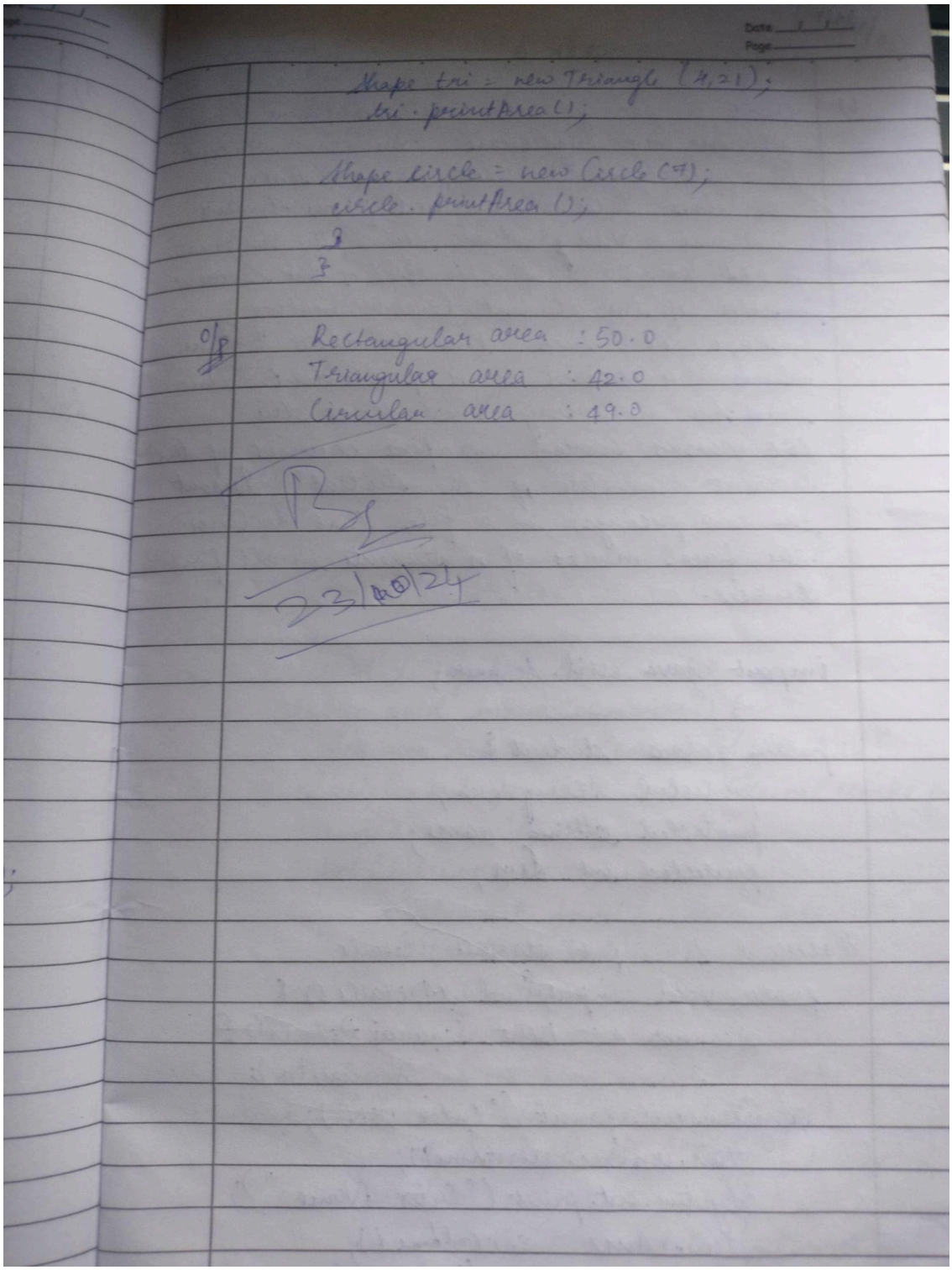
Date 23/10/24
Page

1) 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.

Sol:

```
import java.util.Scanner

abstract class Shape
{
    int a, b;
    void printArea()
    {
    }
    class Rectangle extends Shape
    {
        Rectangle(int len, int breadth)
        {
        this.a = len;
        this.b = breadth;
        }
        void printArea()
        {
        double area = a*b;
        System.out.println("Rectangle area: "+area)
        }
    }
    class Triangle extends Shape
    {
        Triangle(int base, int height)
```

```
    {
        this.a = base;
        this.b = height;
    }

void printArea()
{
double area = 0.5*a*b;
System.out.println("Triangle area:" +
area);
}
}
class Circle extends Shape
  {
    Circle(int radius)
    {
      this.a = radius;
    }

void printArea()
 {
   double area = a*a;
   System.out.println("Circle area:" +area);
}
}
  class Run
{
public static void main(String[] args)
{

System.out.println("Name: dishwarya K");
System.out.println("USN: 1BM23CS018");
  Shape s rect = new Rectangle(5,10);
   rect.printArea();
```



```
Shape tri = new Triangle (4,21);
tri.printArea();

Shape circle = new Circle (7);
circle.printArea();
}
}
```

O/P

```
Rectangular area : 50.0
Triangular area : 42.0
Circular area : 49.0
```

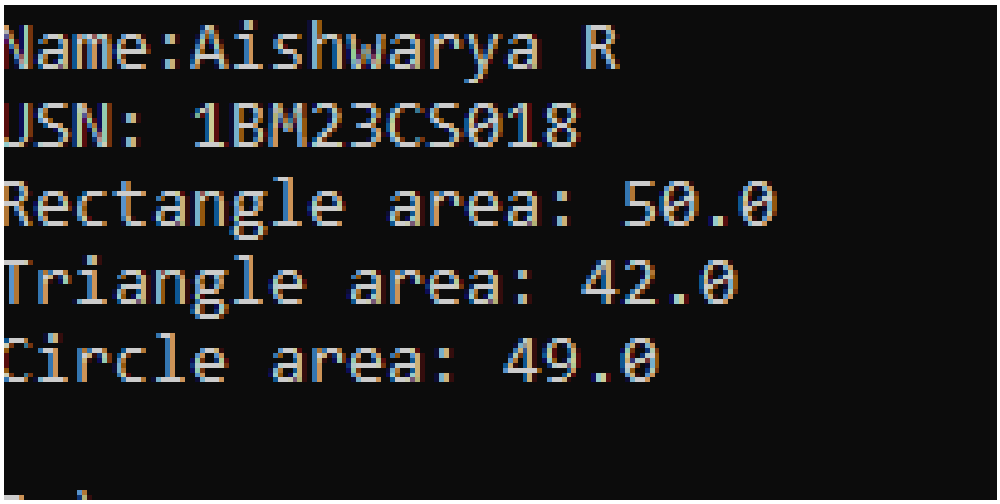
23/10/24

code:

```
import java.util.Scanner;

class Book
{
String name,author;
double price;
int num_page,book_count;


void get_details()
{
Scanner s1=new Scanner(System.in);
System.out.println("Name: ");
name=s1.nextLine();

System.out.println("Author: ");
author=s1.nextLine();

System.out.println("Price: ");
price=s1.nextDouble();

System.out.println("Page number: ");
num_page=s1.nextInt();

}
public String toString()
{
String name,author,price,num_page;
name="Name: "+this.name+"\n";
author = "Author: " + this.author + "\n";

price = "Price: " + this.price + "\n";

num_page = "Number of pages: " + this.num_page + "\n";

return name + author + price + num_page;
}
}

class dontrun
{
public static void main(String[] args)
{
System.out.println("Name: Aisharya r");
System.out.println("USN: 1BM23CS018");

Scanner s2=new Scanner(System.in);
int n;
System.out.println("Number of books: ");

n=s2.nextInt();
```

```
Book[] b=new Book[n];
for(int i=0;i<n;i++)
{
b[i]=new Book();
b[i].get_details();


}
for(int i=0;i<n;i++)

{
System.out.println(b[i].toString());
}
}
}
```

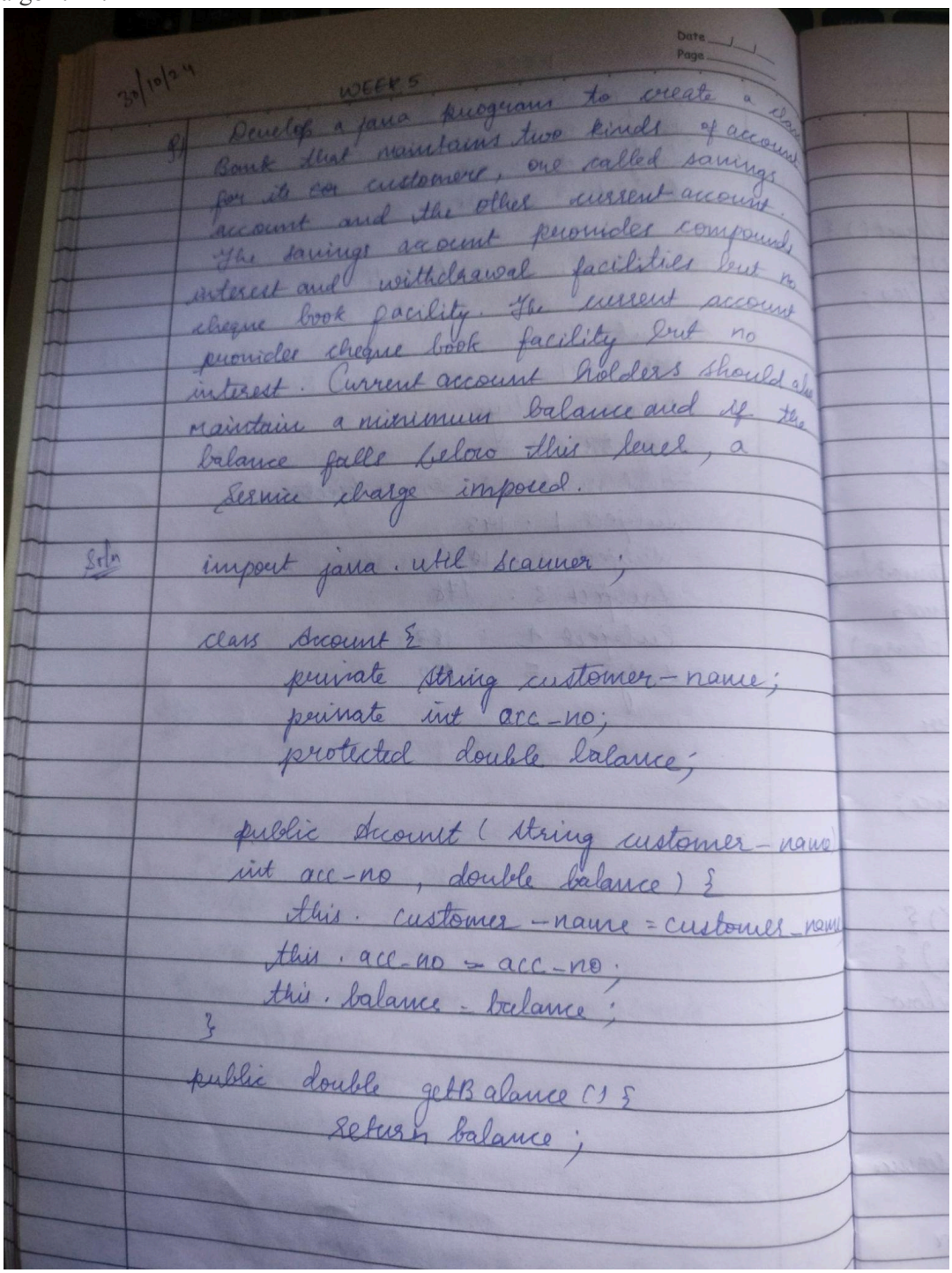
output:

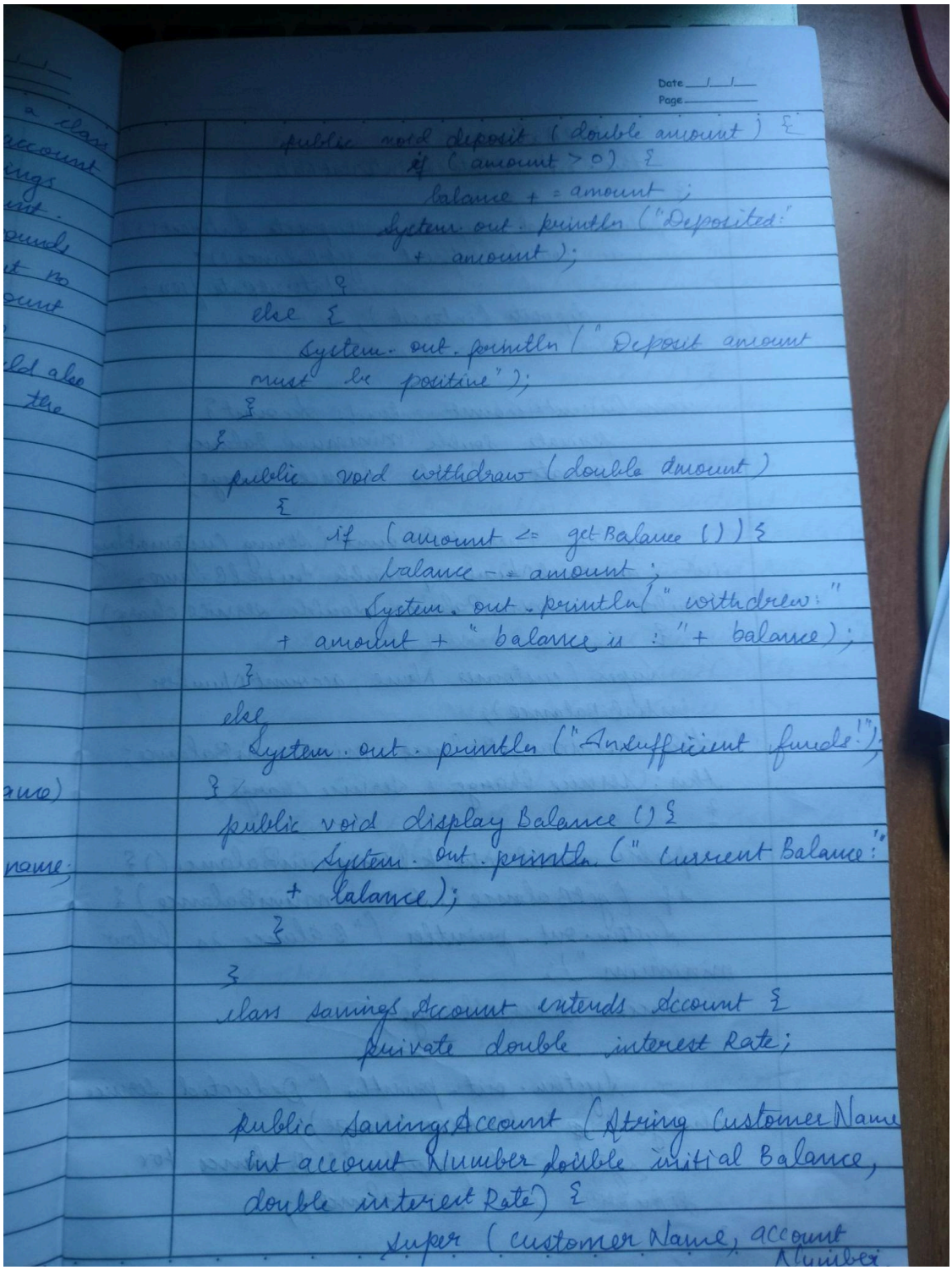


program5

bank account

algorithm:





```
public void deposit (double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited:
            + amount);
        }
        else {
            System.out.println("Deposit amount
        must be positive");
        }
    }
    public void withdraw (double amount)
        {
            if (amount <= getBalance()) {
                balance -= amount;
                System.out.println("withdrew:"
        + amount + "balance is :" + balance);
        }
        else
            System.out.println("Insufficient funds!");
    }
    public void displayBalance() {
            System.out.println("Current Balance:"
        + balance);
        }
    }
    class savingsAccount extends Account {
            private double interestRate;

        public SavingsAccount (String customerName
        int accountNumber, double initialBalance,
        double interestRate) {
            super (customerName, accountNumber,
```

```
        initialBalance);
        this.interestRate = interestRate;
    }
    public void computeAndDepositInterest() {
        double interest = getBalance() *
                          interestRate / 100;
        deposit(interest);
    }
}
class CurrentAccount extends Account {
    private double minimumBalance;
    private double serviceCharge;

    public CurrentAccount(String customerName,
int accountNumber, double initialBalance,
double minimumBalance, double serviceCharge)
    {
        super(customerName, accountNumber,
        initialBalance);
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }
    public void checkMinimumBalance() {
        if (getBalance() < minimumBalance) {
            System.out.println("Balance is below
minimum");
            balance -= serviceCharge;

            System.out.println("Deducted service
charge:" + serviceCharge);
            System.out.println("Balance for
deduction is:" + balance);
    }
```

```
    }
}
public class Bank {
    public static void main (String [] args)
{   Scanner sc = new Scanner (System.in);
    System.out.println ("Enter Customer name");
    String name = sc.nextLine ();
    System.out.println ("Enter accno.");
    int acc_no = sc.nextInt ();
    System.out.println ("Enter initial
    balance:");
    double balance = sc.nextDouble ();
    System.out.println ("Enter interest rate:");
    double minimum balance = sc.nextDouble ();
    System.out.println ("Enter interest rate :")
    Double interest_rate = sc.nextDouble ();
    System.out.println ("Enter service charge");
    double service_charge = sc.nextDouble ();
    System.out.println ("Enter choice : \n
    1. current acc \n 2. Savings acc");
    int ch = sc.nextInt ();
    System.out.println ("Customer name is:" +
    name +" \n Account number :" +
    acc_no");

    Switch (ch) {
        case (1): System.out.println ("Account
        is current_type");
        current Account ca = new CurrentAccount
        (name, acc_no, balance, minimum_
        balance, service_charge);
        do { System.out.println ("Enter choice ;
        \n 1. deposit \n 2. withdraw \n 3. display
        balance");
```

```
int c = sc.nextInt();
ca.checkMinimumBalance();
if(c==1){
    System.out.println("Enter
amount to be deposited:");
double amt = sc.nextDouble();
    ca.deposit(amt); }
else if(c==2){
    System.out.println("Enter
amount to withdraw:");
double amt = sc.nextDouble();
    ca.withdraw(amt); }

else if(c==3){
    ca.displayBalance(); }
else
    System.exit(0);
}
while(true);
```

o/p

```
Enter customer name
Ahmed
Enter accno.
03465
Enter initial Balance:
15000
Enter minimum balance:
10000
Enter interest rate:
3
Enter service charge:
2.75
Enter choice:
```

ance 1);
ln("Enter
();
}
("Enter
();
}

Date / /
Page

1: Current acc
2: Savings acc
2
Customer name is: Ahmed
Account number: 3465
Account is savings type
Enter choice:
1. Deposit
2. Withdraw
3. display balance

2
~~Customer name is~~:
Enter amount to withdraw:
3000
Withdrew: 3000.0 balance is: 12000.0

4/12/24

code:

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

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

   public void deposit(double amount) {
      if (amount > 0) {
         balance += amount;
         System.out.println("Amount deposited successfully.");
      } else {
         System.out.println("Invalid deposit amount.");
      }
   }

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

class SavAcct extends Account {
   private double interestRate;

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

   public void computeAndDepositInterest() {
      double interest = balance * interestRate / 100;
      balance += interest;
      System.out.println("Interest of " + interest + " has been deposited.");
   }

   public void withdraw(double amount) {
      if (amount > 0 && amount <= balance) {
         balance -= amount;
         System.out.println("Withdrawal successful. New balance: " + balance);
      } else {
         System.out.println("Invalid withdrawal amount or insufficient balance.");
      }
   }
}
```

```
class CurAcct extends Account {
    private static final double MINIMUM_BALANCE = 500.0;
    private static final double PENALTY = 50.0;

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

    public void checkAndImposePenalty() {
        if (balance < MINIMUM_BALANCE) {
            balance -= PENALTY;
            System.out.println("Penalty of " + PENALTY + " imposed due to insufficient balance.");
        }
    }

    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrawal successful. New balance: " + balance);
            checkAndImposePenalty();
        } else {
            System.out.println("Invalid withdrawal amount or insufficient balance.");
        }
    }
}

public class Bank {
    public static void main(String[] args) {
        SavAcct savingsAccount = new SavAcct("Alice", "SA12345", 1000.0, 5.0);
        CurAcct currentAccount = new CurAcct("Bob", "CA54321", 600.0);

        // Savings account operations
        System.out.println("Savings Account Operations:");
        savingsAccount.deposit(500);
        savingsAccount.displayBalance();
        savingsAccount.computeAndDepositInterest();
        savingsAccount.withdraw(300);
        savingsAccount.displayBalance();

        // Current account operations
        System.out.println("\nCurrent Account Operations:");
        currentAccount.deposit(200);
        currentAccount.displayBalance();
        currentAccount.withdraw(100);
        currentAccount.displayBalance();
        currentAccount.withdraw(700);
        currentAccount.displayBalance();
    }
}
```

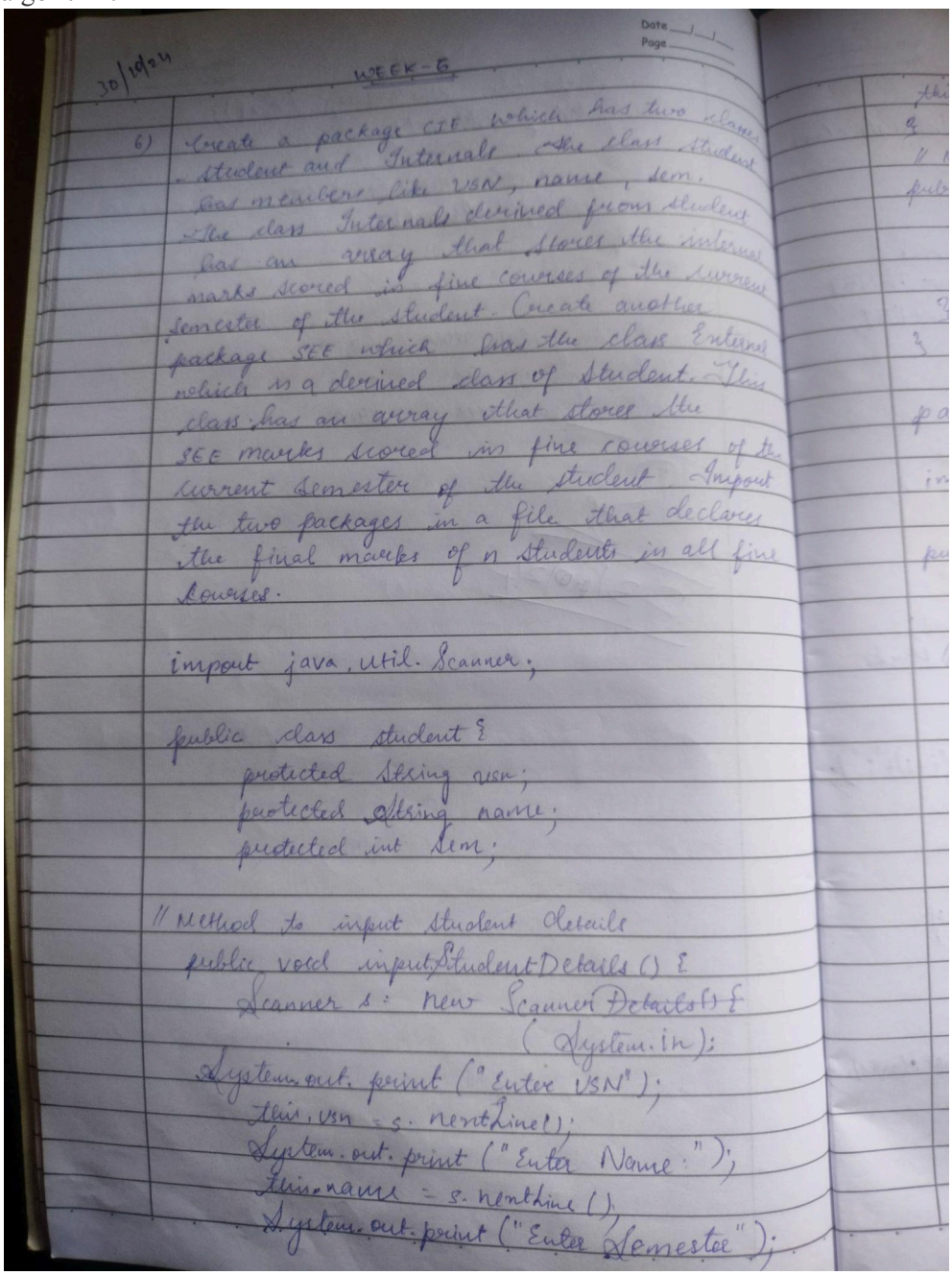
output

```
PS D:\1BM23CS018 JAVA> javac  Bank.java
PS D:\1BM23CS018 JAVA> java  Bank
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
PS D:\1BM23CS018 JAVA> java  Bank
Choose Account Type:
1. Savings Account
2. Current Account
2
Current Account Selected
Deposited: 500.0
Withdrawn: 1800.0
Balance: 700.0
PS D:\1BM23CS018 JAVA>
```

program6

packages

algorithm:



30/10/24

WEEK-6

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

```
import java.util.Scanner;

public class student {
    protected String usn;
    protected String name;
    protected int sem;

// Method to input student details
public void inputStudentDetails() {
    Scanner s = new Scanner Details() {
                    (System.in);
System.out.print("Enter USN");
    this.usn = s.nextLine();
System.out.print("Enter Name:");
    this.name = s.nextLine();
System.out.print("Enter Semester");
```

```
        this.sem = s.nextInt();
    }
    // Method to display student details
    public void displayStudentDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}

package CIE;

import java.util.Scanner;

public class Internals extends Student {
    protected int marks[] = new int[5];

    // Method to input internal marks
    public void inputCIEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter Internal Marks for
            5 courses:");
        for(int i=0; i<5; i++) {
            System.out.print("Course" + (i+1) + ": ");
            marks[i] = s.nextInt();
        }
    }

    // Method to display internal marks
    public void displayCIEmarks()
        System.out.println("Internal Marks:");
        for(int i=0; i<5; i++) {
            System.out.println("Course" + (i+1) +
                ": " + marks[i]);
        }
    }
}
```

```
package SEE;
import CIE.Internals; // Import Internals class from CIE package
import java.util.Scanner;

public class External extends Internals {
    protected int externalMarks[] = new int[5];
    protected int finalMarks[] = new int[5];

    // constructor to initialize arrays
    public External() {
        externalMarks = new int[5];
        finalMarks = new int[5];
    }

    // Method to input external marks
    public void inputSEEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter External Marks for 5 courses:");
        for (int i=0; i<5; i++) {
            System.out.print("Course" + (i+1) + ":");
            externalMarks[i] = s.nextInt();
        }
    }

    // Method to calculate final marks (sum of internal and external)
    public void calculateFinalMarks() {
        for(int i=0; i<5; i++) {
            finalMarks[i] = marks[i] + externalMarks[i];
        }
    }
```

Internals class

```
ernals {
ew int[5];
int[5];

];
;

in);
l Marks

(i+1)+":");

of

nal Marks[i];
```

Date __/__/__
Page _____

```
// Method to display final marks
public void displayFinalMarks() {
    displayStudentDetails();
    displaySEEmarks();

System.out.println("External Marks:");
for(int i=0; i<5; i++) {
    System.out.println("Course"+(i+1)+
    ":"+externalMarks[i]);
}
System.out.println("Final Marks:");
for(int i=0; i<5; i++) {
    System.out.println("Course"+(i+1)+
    ":"+finalMarks[i]);
}
}
{
import SEE.Externals;
import java.util.Scanner;

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

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

// Array to store Student data
Externals[] students = new Externals[n];

// Loop to input data for each student
for(int i=0; i<n; i++) {
```

```
        Student[i] = new External();

// Input student details, internals marks
and external marks.

        Students[i].inputStudentDetails();
        Students[i].inputCIEmarks();
        Students[i].inputSEEmarks();
        Student[i].calculateFinalMarks();

// Display the final results of all students
        for(int i=0; i<n; i++)
            Students[i].displayFinalMarks();
        System.out.println();
    }
}
```

O/P

```
Enter the marks of the students :)

Enter details for student 1:
Enter USN : 001
Enter Name : Aliya
Enter semester : 2
Enter internal marks for 5 subjects
        Subject 1 : 95
        Subject 2 : 87
        Subject 3 : 79
        Subject 4 : 94
        Subject 5 : 82
Enter SEE marks for 5 subjects :
        Subject 1 : 98
```

WEEK-5

Date / /
Page

Subject 2 : 91
Subject 3 : 99
Subject 4 : 89
Subject 5 : 98

Final marks for Students

Student 1 :
USN : 001
Name : Aliya
Semester : 2
Final marks for 5 subjects :
Subject 1 : 193
Subject 2 : 183
Subject 3 : 178
Subject 4 : 183
Subject 5 : 180

code:

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

public class FinalMarks {
   public static void main(String[] args) {
      int[] internalMarks = {20, 18, 22, 19, 25}; // Example internal marks
      int[] seeMarks = {60, 70, 65, 55, 75};      // Example SEE marks

      Internals internals = new Internals("1RV22CS001", "Alice", 5, internalMarks);
      External external = new External("1RV22CS001", "Alice", 5, seeMarks);

      System.out.println("Final Marks for Student:");
      System.out.println("USN: " + internals.usn);
      System.out.println("Name: " + internals.name);
      System.out.println("Semester: " + internals.sem);


}
     System.out.println("Course-wise Marks:");
      for (int i = 0; i < 5; i++) {
         int finalMarks = internals.internalMarks[i] + (external.seeMarks[i] / 2);
         System.out.println("Course " + (i + 1) + ": " + finalMarks);
      }
   }

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;
   }
}

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;
    }
}
```

output:

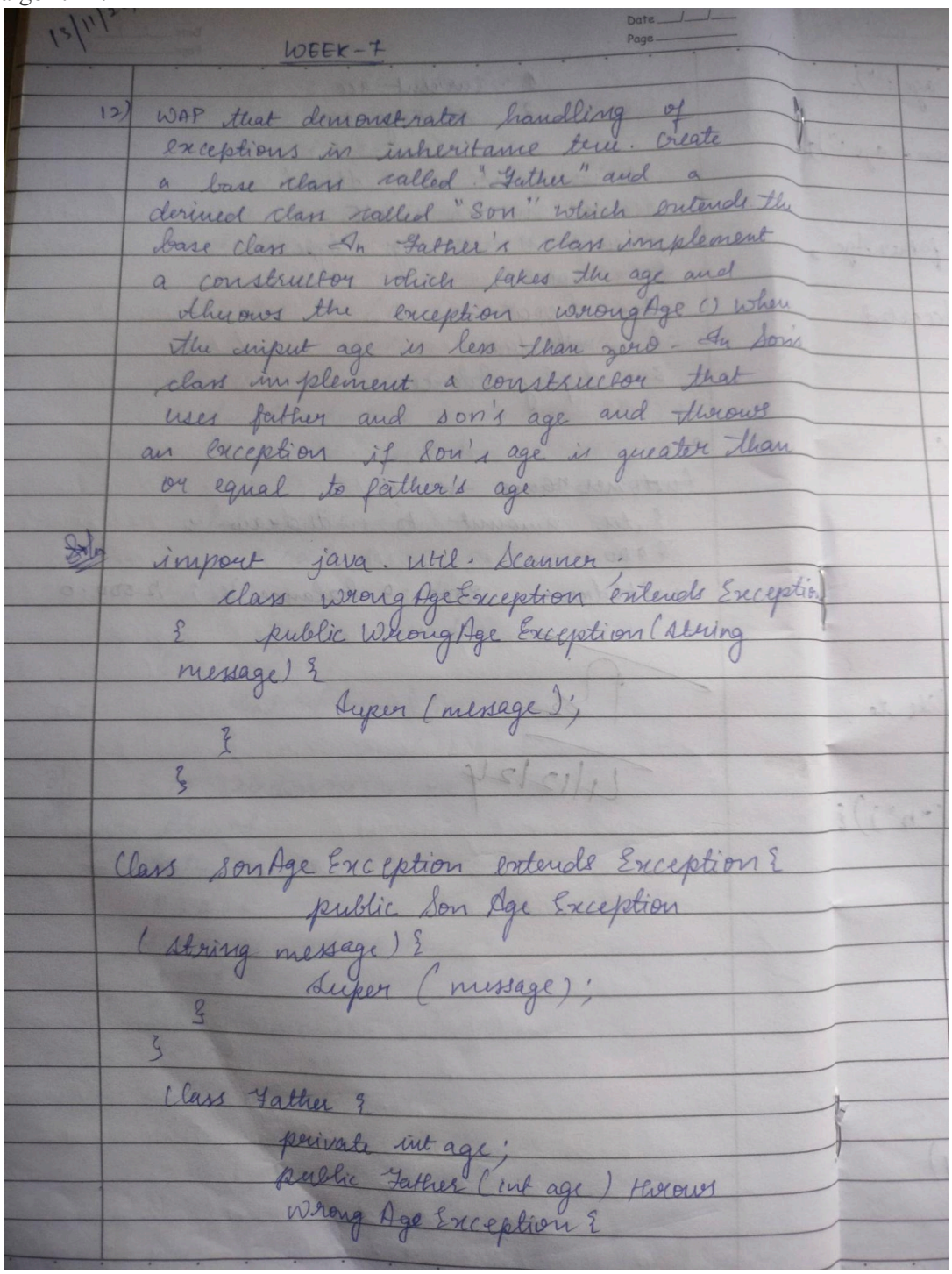
```
Enter number of students: 3

Enter details for student 1:
Enter USN: 1BM23CS006
Enter Name: Aaryan
Enter Semester: 3
Enter Internal Marks for 5 subjects:
Subject 1: 45
Subject 2: 49
Subject 3: 42
Subject 4: 48
Subject 5: 46
Enter SEE Marks for 5 subjects:
Subject 1: 44
Subject 2: 48
Subject 3: 50
Subject 4: 48
Subject 5: 49

Enter details for student 2:
Enter USN: 1BM23CS006
Enter Name: Avinash
Enter Semester: 4
Enter Internal Marks for 5 subjects:
Subject 1: 37
Subject 2: 39
Subject 3: 41
Subject 4: 43
Subject 5: 45
```

```
Subject 2: 39
Subject 3: 41
Subject 4: 43
Subject 5: 45
Enter SEE Marks for 5 subjects:
Subject 1: 45
Subject 2: 45
Subject 3: 45
Subject 4: 40
Subject 5: 40

Enter details for student 3:
Enter USN: 1BM23CS003
Enter Name: Chandan
Enter Semester: 5
Enter Internal Marks for 5 subjects:
Subject 1: 38
Subject 2: 39
Subject 3: 41
Subject 4: 41
Subject 5: 47
Enter SEE Marks for 5 subjects:
Subject 1: 44
Subject 2: 45
Subject 3: 43
Subject 4: 27
Subject 5: 50
```

program7

exceptions

algorithm:



WEEK-7

12) WAP that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and a derived class called "Son" which extends the base class. In Father's class implement a constructor which takes the age and throws the exception wrongAge() when the input age is less than zero. In Son class implement a constructor that uses father and son's age and throws an exception if Son's age is greater than or equal to father's age.

```
import java.util.Scanner;
class WrongAgeException extends Exception
{   public WrongAgeException(String
message){
        super(message);
    }
}

Class SonAgeException extends Exception{
        public SonAgeException
(String message){
        super(message);
    }
}
Class Father {
        private int age;
        public Father(int age) throws
        WrongAgeException{
```

```
if (age < 0) {
    throw new WrongAgeException
    (" : Wrong Age");
}
this.age = age;
}
public int get () {
    return age;
}
}
class Son extends Father {
    private int SonAge;
    public Son (int fatherAge,
    int SonAge)
throws WrongAgeException, SonAgeException {
    super (fatherAge);
    if (SonAge >= fatherAge) {
        throw new SonAgeException (
"Son's age cannot be greater
    than or equal to father's age");
}
this.SonAge = SonAge;
}
public int getSonAge () {
    return SonAge;
}
}

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

```
System.out.print("Enter father's age :");
int fatherAge : sc.nextInt();
    System.out.print("Enter Son's age :");
int SonAge : sc.nextInt();
try {
        Son son = new Son(fatherAge,
        SonAge);
        System.out.println("Accepted
Successfully");
}
catch (WrongAge Exception e) {
            System.out.println(
            e.getMessage());
}
    catch (SonAge Exception e) {
            System.out.println(
            e.getMessage());
}
    System.out.println("would you like to
re enter details (Y/N)");
String input = sc.next();
        if (input.equalsIgnoreCase("n")){
            break;
    }
}
}
}
```

o/p

```
Enter father's age : 38
Enter son's age : 6
Accepted successfully
would you like to re enter details (Y/N)
N
```

code:

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

class Father {
    protected int age;

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

class Son extends Father {
    private int sonAge;

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

public class ExceptionHandlingInheritance {
    public static void main(String[] args) {
        try {
            Father father = new Father(45);
            System.out.println("Father's age: " + father.age);

            Son son = new Son(45, 20);
            System.out.println("Son's age: " + son.sonAge);
        } catch (WrongAgeException e) {
            System.out.println("Exception: " + e.getMessage());
        }

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

        try {
            Son invalidSon = new Son(40, 45);
        } catch (WrongAgeException e) {
            System.out.println("Exception: " + e.getMessage());
```

```
    }
  }
}
```

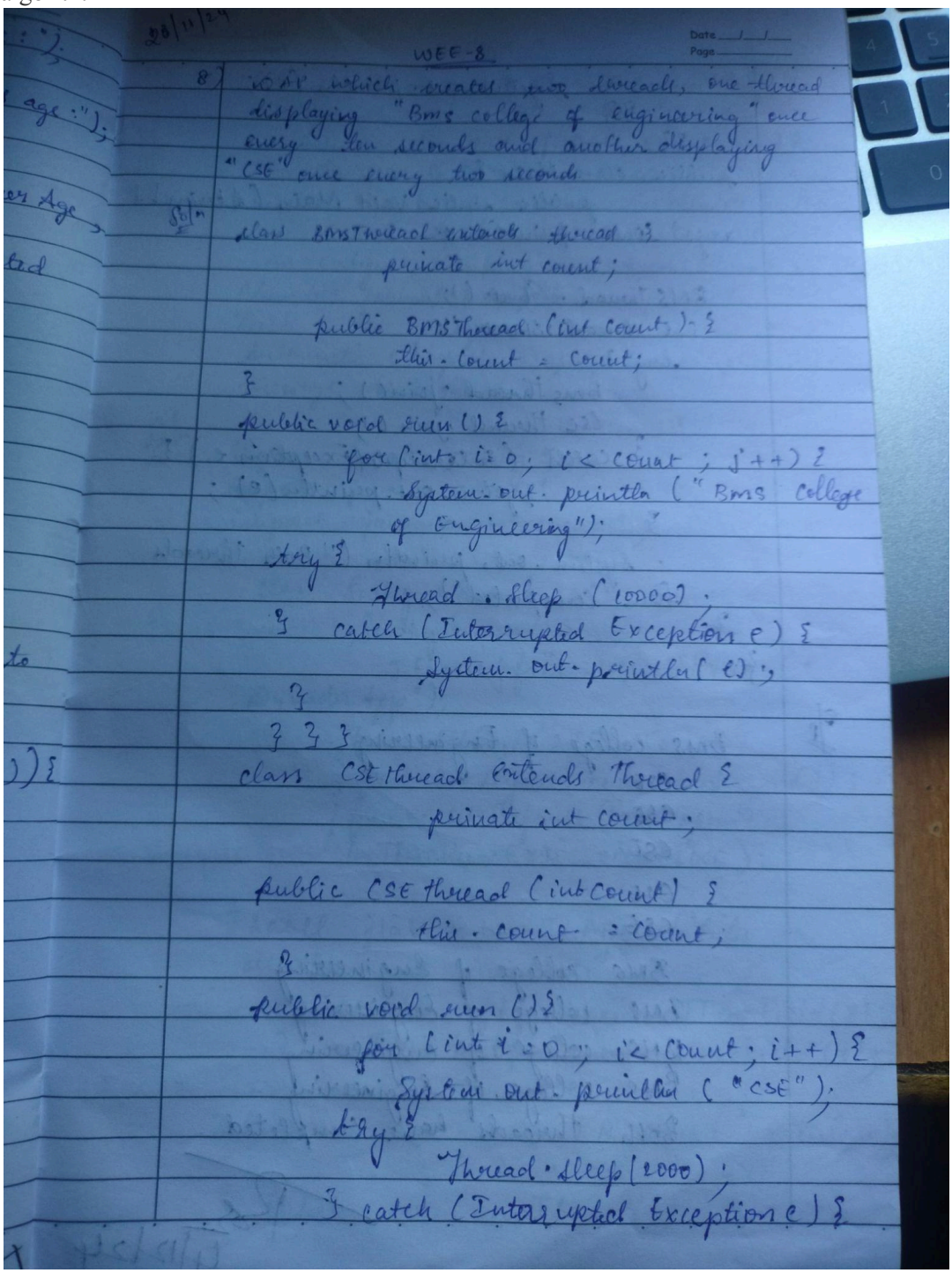
output:

```
PS D:\1BM23CS018 JAVA> javac FatherSon.java
PS D:\1BM23CS018 JAVA> java FatherSon
Enter Father's Age: 54
Enter Son's Age: 21
Father's Age: 54
Son's Age: 21
PS D:\1BM23CS018 JAVA> java FatherSon
Enter Father's Age: 27
Enter Son's Age: 29
Father's Age: 27
Exception: Son's Age Cannot be Greater than or Equal to Father's Age
PS D:\1BM23CS018 JAVA>
```

program8

threads

algorith:



```
            System.out.println(e);
        }
    }
public class Main {
        public static void main(String[] args) int printCount = 5;

    BMS Thread.start();
    CSE Thread.start();
      try {
          bms Thread.join();
          cse thread.join();
      } catch (InterruptedException e) {
            System.out.println(e);
      }
        System.out.println("Both Threads have Completed");
    }
}
```

```
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
BMS college of Engineering
BMS college of Engineering
BMS college of Engineering
BMS college of Engineering
Both Threads have completed.
```

4/12/24

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();

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

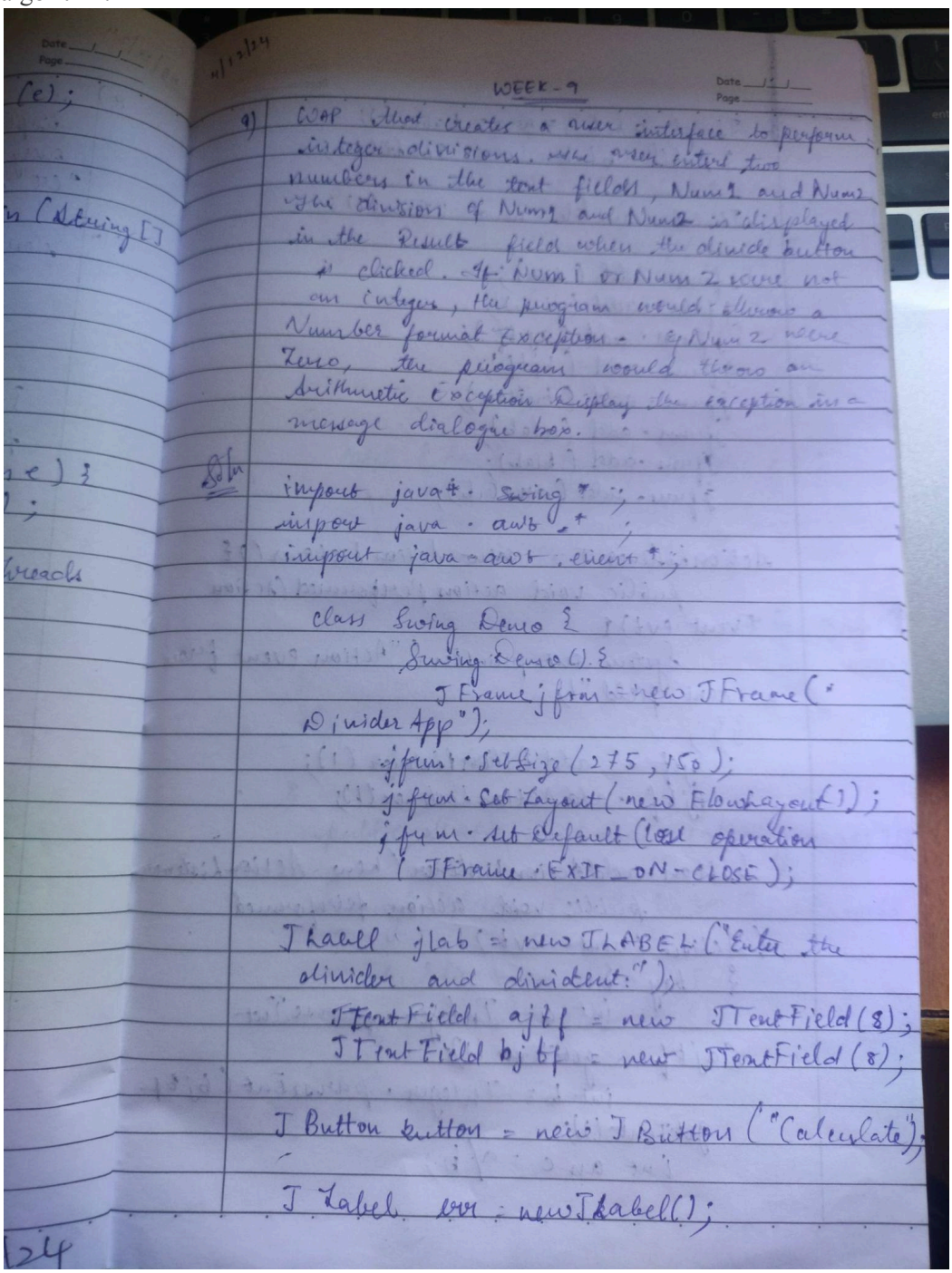
output:

```
PS D:\1BM23CS018 JAVA> javac ThreadExample.java
PS D:\1BM23CS018 JAVA> java ThreadExample
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
PS D:\1BM23CS018 JAVA>
```

program9

Interface to perform integer division

algorithm:



WEEK-9

9) GUI 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 ArithmeticException. Display the exception in a message dialog box.

```
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 the
dividend and divident:");
    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();

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

ActionListener J = 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 anc = a/b;
```

```
        a.lab.setText("\n A = "+a);
        b.lab.setText("\n B = "+b);
        anslab.setText("\n Ans = "+ans);
    }
    catch(NumberFormatException e){
            alab.setText(" ");
            b.lab.setText(" ");
            anslab.setText(" ");
            err.setText("Enter only
            Integers!.");
    }
    catch(ArithmeticException e){
            alab.setText(" ")
            blab.setText(" ");
            anslab.setText(" ");

    err.setText("B should be Non Zero!");
    }
    }};
        jfrm.setVisible(true);
    }
    public static void main(String args[]){

    SwingUtilities.invokeLater(new Runnable
    (){
        public void run(){
            new SwingDemo();
        }
    });
}
```

Date __/__/__
Page

4/12/24

10)

Q1 Enter the divider and dividend :
10 2
A=10 B=2 Ans = 5.

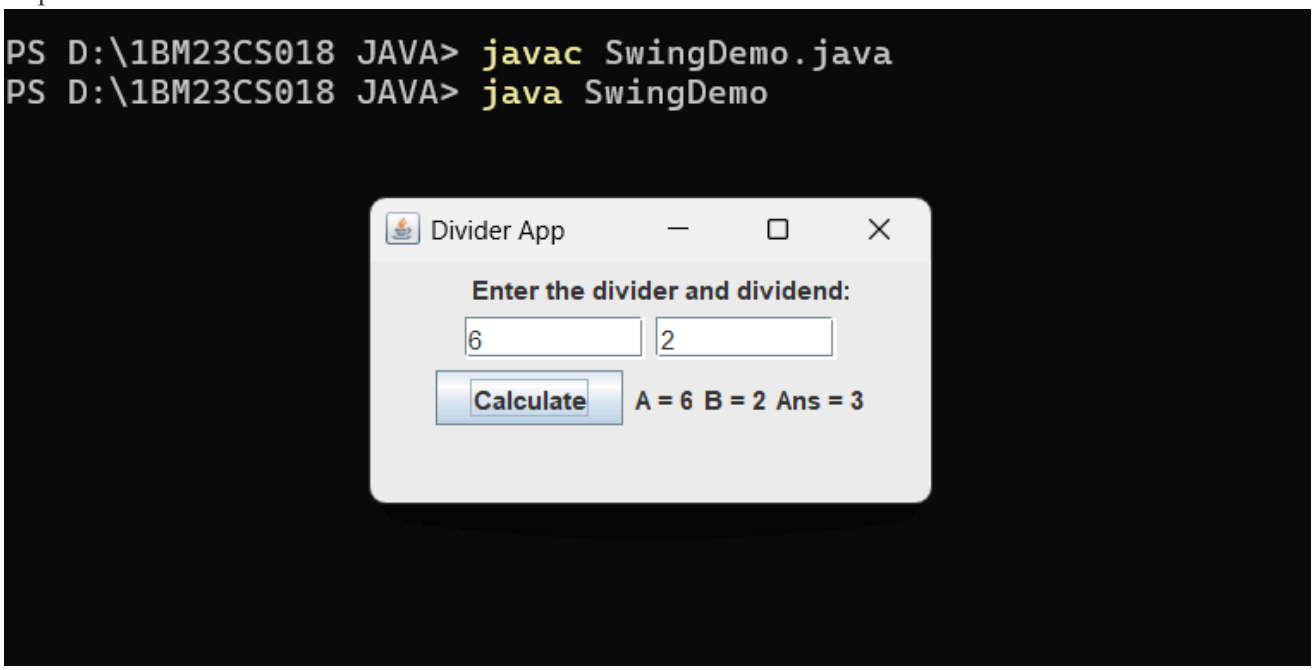
Enter the divider and dividend
10 0
B should be Non Zero !

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

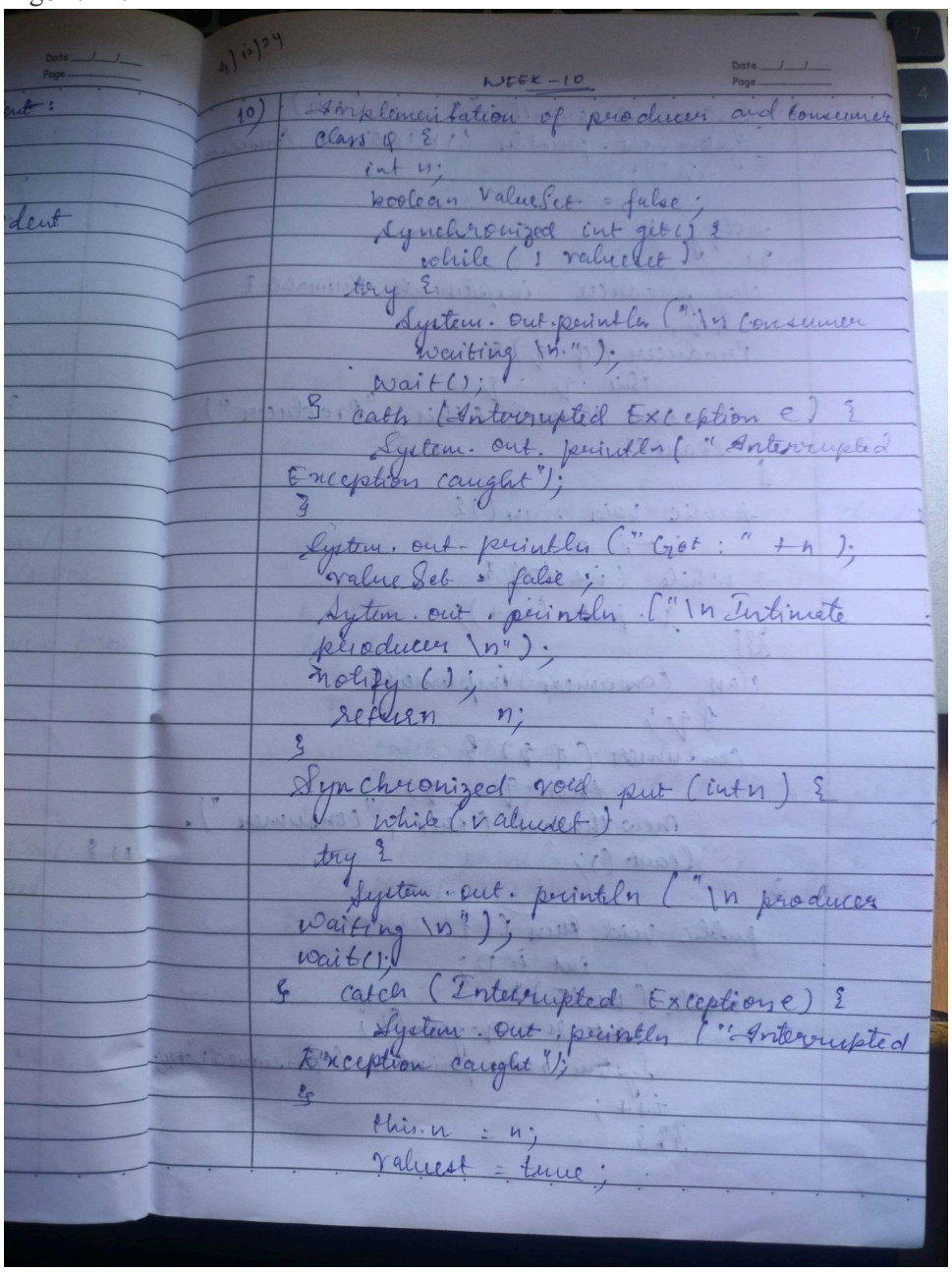
output:



program10

Interprocess Communication and Deadlock

Algorithm:



```
System.out.println("put: "+n);
System.out.println("\nIntimate Consumer
    -in");

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");
    }}

Dead lock :

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

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

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

o/p

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.

Interprocess communication

```
class Q{
    int n;
    synchronized int get(){
        System.out.println("Got : " + n);
        return n;
    }
    synchronized void put(int n){
        this.n = n;
        System.out.println("Put : " + n);
    }
}
class Producer implements Runnable{
    Q q;
    Producer(Q q){
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run(){
        int i = 0;
        while(true){
            q.put(i++);
        }
    }
}
class Consumer implements Runnable{
    Q q;
    Consumer(Q q){
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run(){
        while(true){
            q.get();
        }
    }
}
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");
    }
}
```

o/p

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 producer
Consumed : 2
put : 3
Intimate consumer
producer waiting
Got : 3
Intimate producer
Consumed : 3
put : 4
Intimate consumer
Got : 4
Intimate producer
Consumed : 4

Deadlock :- Racing Thread entered B.bar
Main Thread entered A.foo
Racing Thread trying to call
A.last()
Main Thread() trying to call
B.last()

4/12/24

code:

```
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
System.out.println("\nConsumer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("\nIntimate Producer\n");
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
System.out.println("\nProducer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\nIntimate Consumer\n");
notify();
```

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

```
public static void main(String args[]) {

Q q = new Q();

new Producer(q);

new Consumer(q);

System.out.println("Press Control-C to stop.");

}

}
```

```
    Put: 1

    Got: 1

    Put: 2

    Got: 2

    Put: 3

    Got: 3

    Put: 4

    Got: 4
    Put: 5

    Got: 5
```

```
class A {

synchronized void foo(B b) {

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

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

try {

Thread.sleep(1000);

} catch(Exception e) {

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

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

b.last();

}

void last() {

System.out.println("Inside A.last");

}
```

```
}
class B {

synchronized void bar(A a) {

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

System.out.println(name + " entered B.bar");

try {

Thread.sleep(1000);

} catch(Exception e) {

System.out.println("B Interrupted");

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

a.last();

}
void last() {

System.out.println("Inside A.last");

}

}

class Deadlock implements Runnable {

A a = new A();

B b = new B();

Deadlock() {
Thread.currentThread().setName("MainThread");

Thread t = new Thread(this, "RacingThread");

t.start();

a.foo(b); // get lock on a in this thread.

System.out.println("Back in main thread");

}
public void run() {

b.bar(a); // get lock on b in other thread.

System.out.println("Back in other thread");

}

public static void main(String args[]) {
```

```
new Deadlock();
}
}
```

MainThread entered A.foo

RacingThread entered B.bar

MainThread trying to call B.last()

Inside A.last

Back in main thread

RacingThread trying to call A.last()

Inside A.last

Back in other thread.

output: