VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

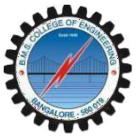
SRUJAN K R (1BM23CS340)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019 Sep

2024-Jan 2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by **SRUJAN K R(1BM23CS340),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

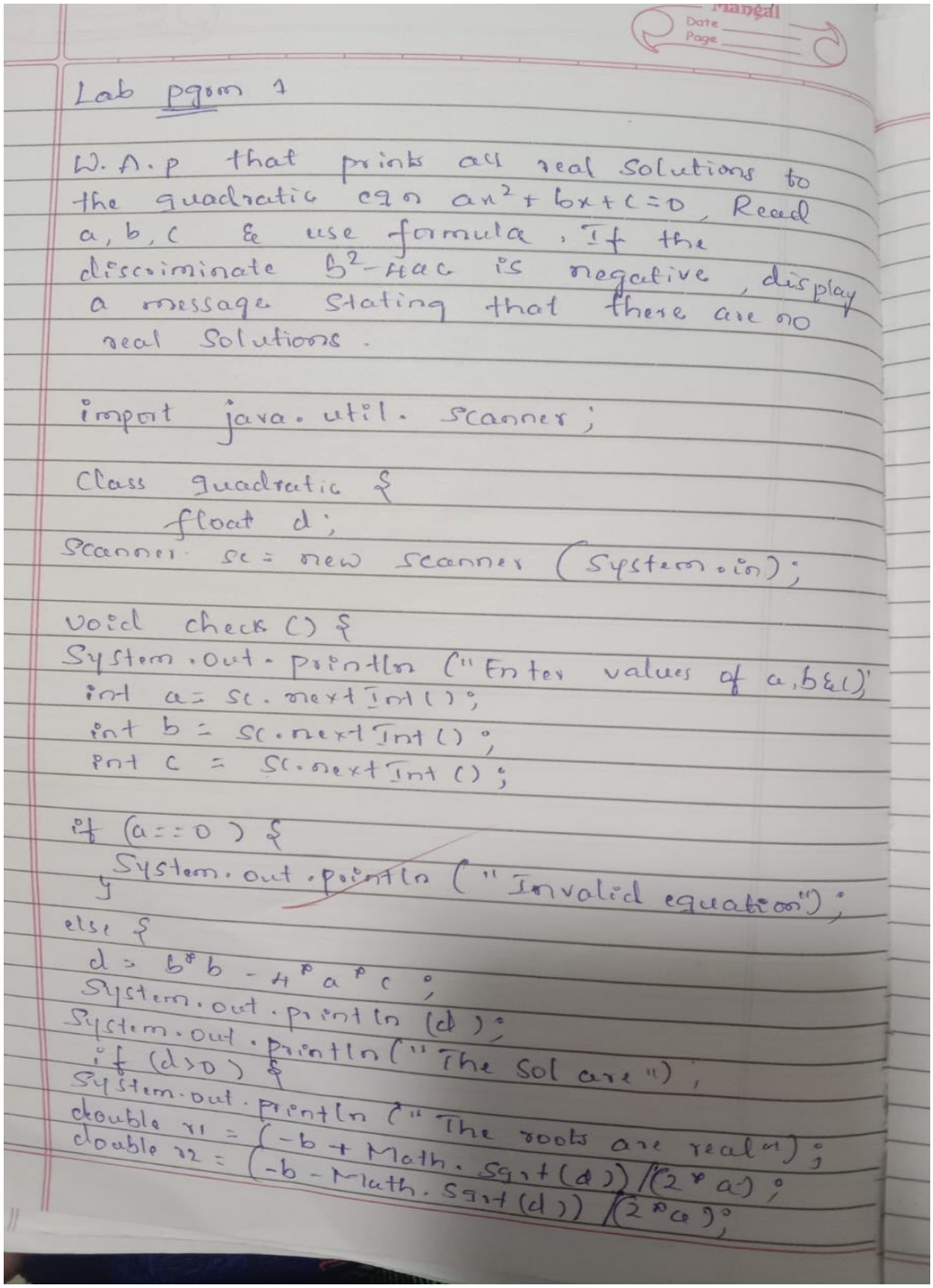
INDEX

Github Link : https://github.com/Srujan4812/1BM23CS340-OOJ

LABORATORY 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.

OBSERVATION :



Lab pgm 1

W.A.p that prints all real solutions to the quadratic eqn $ax^2+bx+c=0$, Read a, b, c & use formula. If the discriminate b^2-4ac is negative, display a message stating that there are no real solutions.

```
import java.util.Scanner;

Class Quadratic {
    float d;
Scanner sc = new Scanner (System.in);

void check () {
System.out.println ("Enter values of a,b&c");
int a = sc.nextInt();
int b = sc.nextInt();
int c = sc.nextInt();

if (a==0) {
  System.out.println ("Invalid equation");
}
else {
  d = b*b - 4*a*c;
System.out.println (d);
System.out.println ("The Sol are");
if (d>0) {
System.out.println ("The roots are real");
double r1 = (-b + Math.sqrt(d)) /(2*a);
double r2 = (-b - Math.sqrt(d)) /(2*a);
```

```
System.out.println("The roots are "+r1+" "
+r2);
if(d==0){
System.out.println("roots are equal");
double r=(-b)/(2*a);
System.out.println(r);
}
if(d<0){
System.out.println("The roots are imaginary");
double r1=Math.sqrt(-d)/(2*a);
double r2=Math.sqrt(-b)/(2*a);
System.out.println(r2+"+i"+" "
+r1+"i"+" "+r2+"-i"+r1);
}
}
}
}

class main{
public static void main(String args[]){
Quadratic q1=new quadratic();
q1.check();
}
}
```

o/p

(i) Enter coefficients of a, b, c
3
4
5

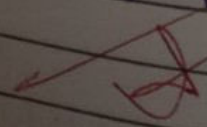
Roots are imaginary

(ii) Enter coefficients of a b c
1
0
-4

Roots are real and distinct
x1 : 2.0 x2 : -2.0

(iii) Enter Co-efficients of a, b, c
4
-4
1

Roots are real & equal x1 = x2 =



CODE :

```
import java.util.Scanner;

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

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

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

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

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



        if (discriminant > 0) {

            double root1 = (-b + Math.sqrt(discriminant)) / (2 * a);
            double root2 = (-b - Math.sqrt(discriminant)) / (2 * a);
            System.out.println("The equation has two real and distinct
solutions:");
            System.out.println("Root 1: " + root1);
            System.out.println("Root 2: " + root2);
```

```
        } else if (discriminant == 0) {

            double root = -b / (2 * a);
            System.out.println("The equation has one real solution (a double root):");
            System.out.println("Root: " + root);
        } else {

            System.out.println("The equation has no real solutions.");
        }
        scanner.close();
    }
}
```

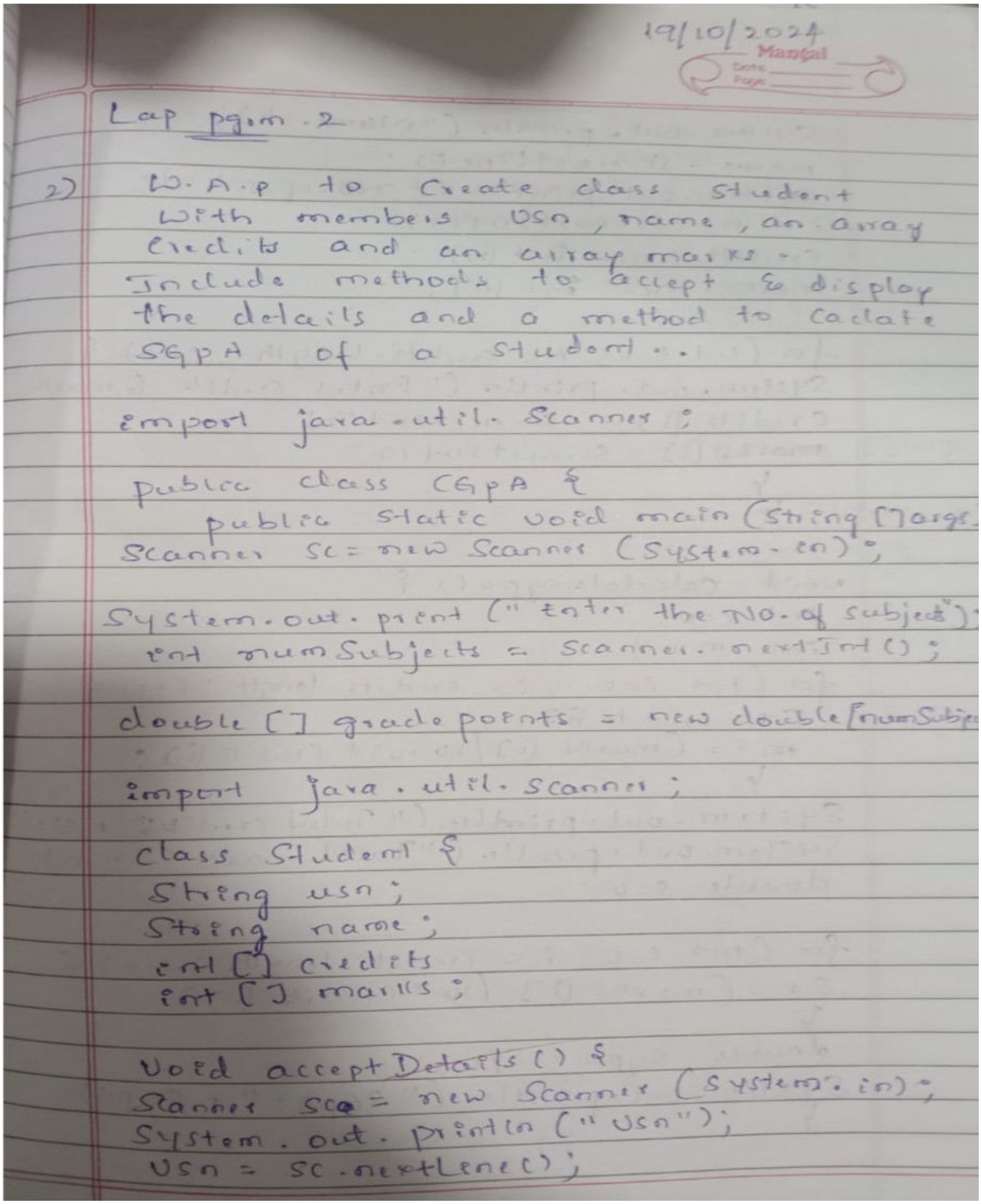
OUTPUT :

```
Enter the coefficient a: 1
Enter the coefficient b: -3
Enter the coefficient c: 2
The equation has two real and distinct solutions:
Root 1: 2.0
Root 2: 1.0
PS D:\3rd sem\OOJ JAVA\Git-hub> java QuadraticEquationSolver
Enter the coefficient a: 1
Enter the coefficient b: 2
Enter the coefficient c: 1
The equation has one real solution (a double root):
Root: -1.0
```

LABORATORY PROGRAM – 2

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

OBSERVATION :



19/10/2024

Lap pgm. 2

2) W.A.p to Create class Student With members usn, name, an array Credits and an array marks. Include methods to accept & display the details and a method to calcate SGPA of a student..

```
import java.util.Scanner;

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

System.out.print (" enter the NO. of subjects");
int numSubjects = scanner.nextInt();

double [] gradepoints = new double[numSubj

import java.util.Scanner;

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

Void acceptDetails () {
Scanner sc = new Scanner (System.in);
System.out.println ("usn");
usn = sc.nextLene();
```

o/p

Enter no. of Subjects : 2
Enter the grade points for Subjects: 9
Enter the credits for Subjects 1: 4
Enter the grade points for subject 2: 8
Enter the credits for Subject 2: 2
Your Cgpa is : 8.666

CODE :

```
import java.util.Scanner;
class Student {
    String usn;
    String name;
    double[] credits;
    double[] marks;
    int numSubjects
    Student(int numSubjects) {
        this.numSubjects = numSubjects;
        credits = new double[numSubjects];
        marks = new double [numSubjects];
    }
    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = scanner.nextLine();
        System.out.print("Enter Name: ");
        name = scanner.nextLine();
        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i+1) + ": ");
            credits[i] = scanner.nextDouble();
            System.out.print("Enter marks for subject " + (i+1) + ": ");
```

```
            marks[i] = scanner.nextDouble();}}
    public void displayDetails() {
        System.out.println("\nStudent Details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Subjects Details: ");
        for (int i = 0; i < numSubjects; i++) {
            System.out.println("Subject " + (i+1) + ": Credits = " +
credits[i] + ", Marks = " + marks[i]);}   }
    public void calculateSGPA() {
        double totalCredits = 0;
        double totalGradePoints = 0;

        for (int i = 0; i < numSubjects; i++) {
            double gradePoint = getGradePoint(marks[i]);
            totalGradePoints += gradePoint * credits[i];
            totalCredits += credits[i];
        }
        double sgpa = totalGradePoints / totalCredits;
        System.out.println("SGPA: " + sgpa);}
    private double getGradePoint(double marks) {
        if (marks >= 90) return 10;
        else if (marks >= 80) return 9;
        else if (marks >= 70) return 8;
```

```
        else if (marks >= 60) return 7;
        else if (marks >= 50) return 6;
        else if (marks >= 40) return 5;
        else return 0;  }
public class StudentMain {
    public static void main(String[] args) {
        Student student = new Student(3);
        student.acceptDetails();
        student.displayDetails();
        student.calculateSGPA();
    }
}
```

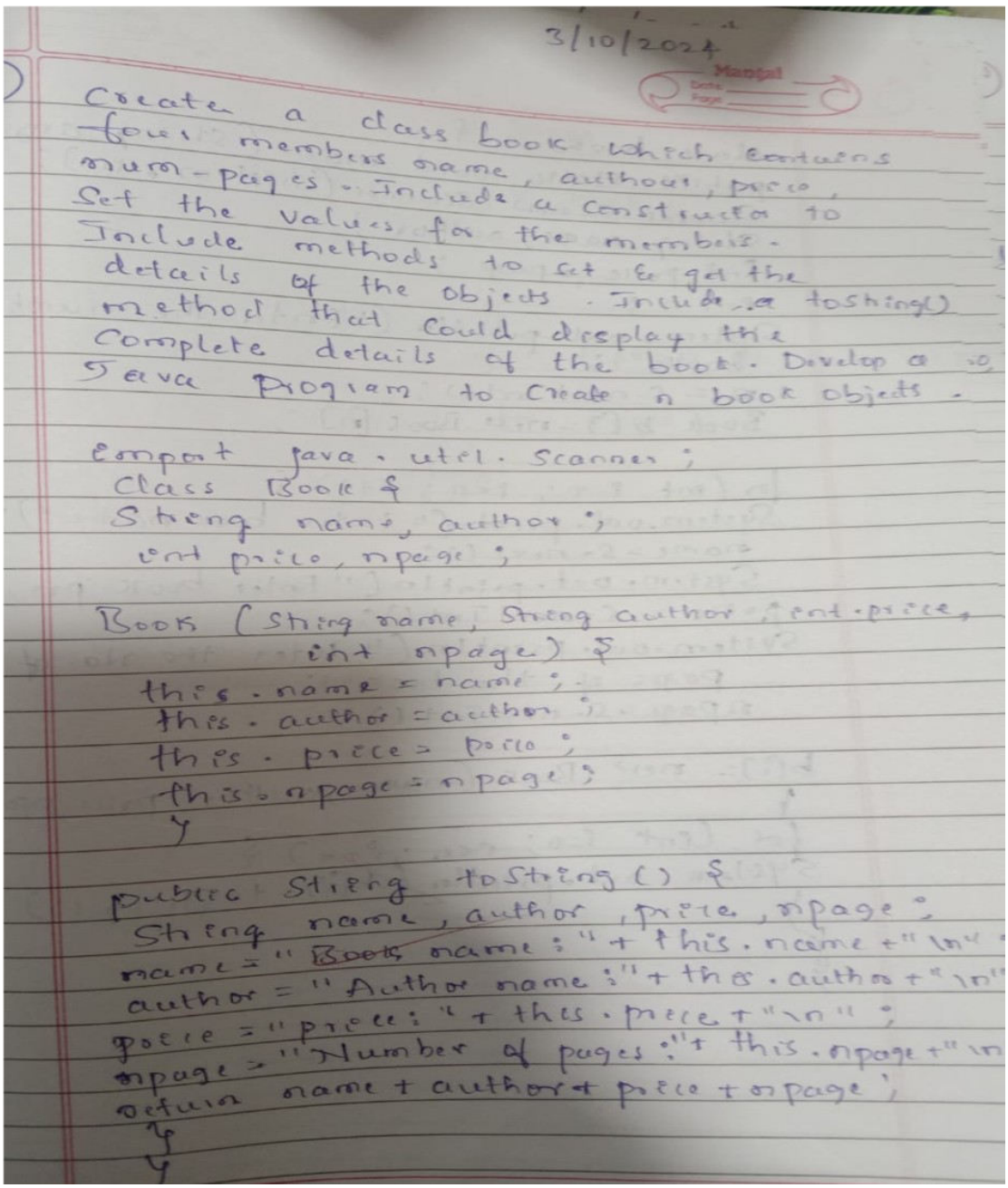
OUTPUT :

```
Enter USN: 123
Enter Name: Sushanth Rai
Enter credits for subject 1: 3
Enter marks for subject 1: 80
Enter credits for subject 2: 4
Enter marks for subject 2: 90
Enter credits for subject 3: 3
Enter marks for subject 3: 80

Student Details:
USN: 123
Name: Sushanth Rai
Subjects Details:
Subject 1: Credits = 3.0, Marks = 80.0
Subject 2: Credits = 4.0, Marks = 90.0
Subject 3: Credits = 3.0, Marks = 80.0
SGPA: 9.4
```

LABORATORY PROGRAM – 3

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

OBSERVATION :



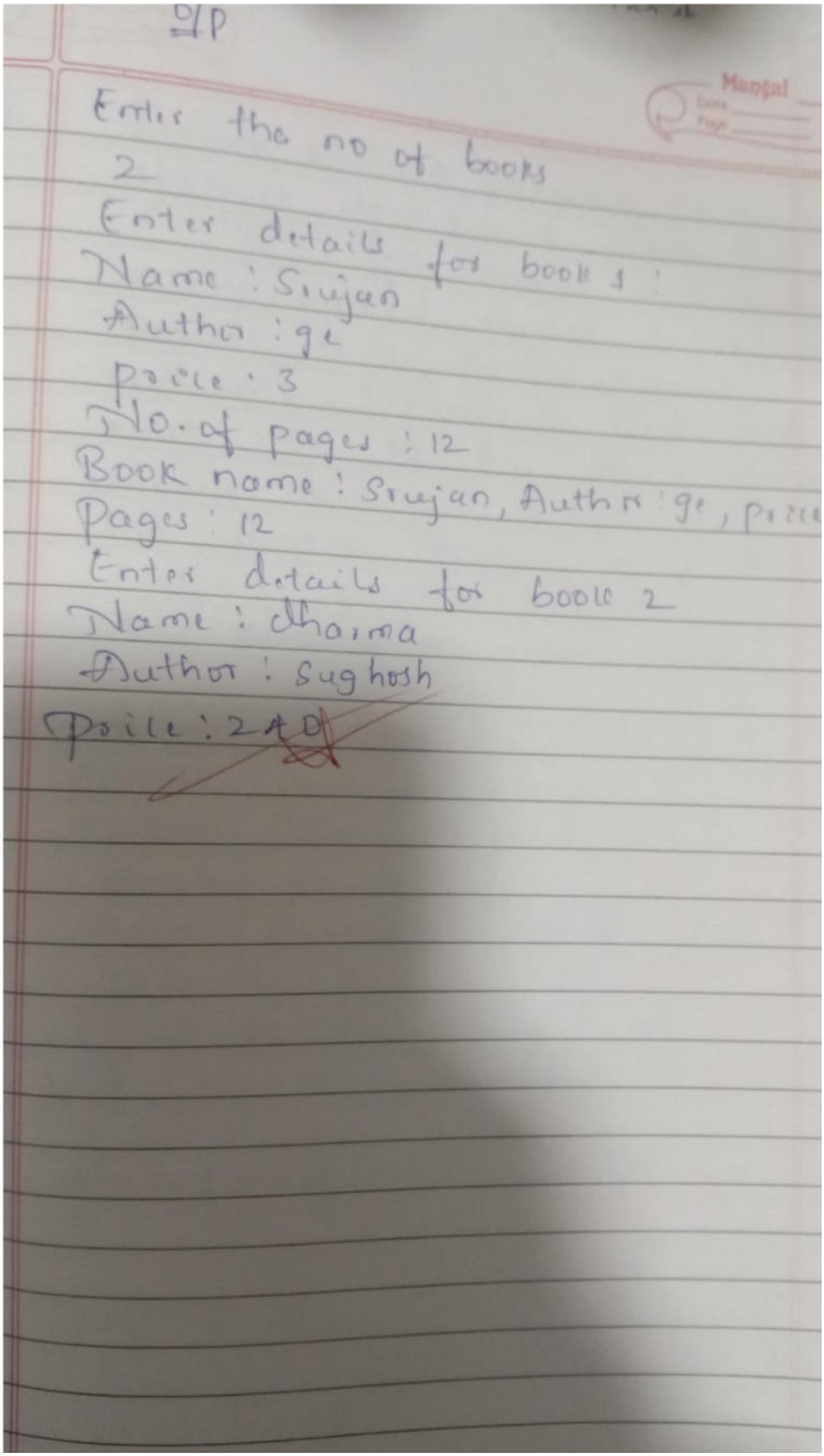
3/10/2024

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 & get the details of the objects. Include a toString() method that could display the complete details of the book. Develop a Java Program to create n book objects.

```
import java.util.Scanner;
Class Book {
String name, author;
int price, npage;

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

Public String toString () {
String name, author, price, npage;
name = "Book name:" + this.name + "\n"
author = "Author name:" + this.author + "\n"
price = "price:" + this.price + "\n";
npage = "Number of pages:" + this.npage + "\n
return name + author + price + npage;
}
}
```

```
class bookrun {
public static void main (String args[])
Scanner s=new Scanner (System.in);
String name, author ;
int price, npage ;
int n ;
System.out.println ("Enter the number of
 books");
n = s.nextInt ();
Book b[] = new Book [n] ;

for (int i=0; i<n ; i++) {
System.out.print ("Enter book name");
 name = s.next ();
 System.out.println ("Enter book price");
price = s.nextInt ();
System.out.println ("Enter the no. of
   pages of the book");
 npage = s.nextInt ();

b[i]= new Book (name, author, price, npage);
}
for (int i=0; i<n ; i++) {
System.out.println (b[i].toString ());
}
}
```

O/P



Enter the no of books
2
Enter details for book 1 :
Name : Srujan
Author : ge
price : 3
No. of pages : 12
Book name : Srujan, Author : ge, Price
Pages : 12
Enter details for book 2
Name : dharma
Author : Sughosh
Price : 240

CODE :

```
import java.util.Scanner;
class Book {
    private String name;
    private String author;
    private double price;
    private int numPages;
    public Book(String name, String author, double price, int numPages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numPages = numPages;}
    public String getName() {
        return name;}
    public void setName(String name) {
        this.name = name;  }
    public String getAuthor() {
        return author;}
    public void setAuthor(String author) {
        this.author = author;}
    public double getPrice() {
        return price;}
    public void setPrice(double price) {
        this.price = price;  }
    public int getNumPages() {
        return numPages;  }
```

```
    public void setNumPages(int numPages) {
        this.numPages = numPages;}
    @Override
    public String toString() {
        return "Book Name: " + name + "\nAuthor: " + author + "\nPrice: "
+ price + "\nNumber of Pages: " + numPages;  }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of books: ");
        int n = sc.nextInt();
        sc.nextLine();
        Book[] books = new Book[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for book " + (i + 1) + ":");
            System.out.print("Enter Book Name: ");
            String name = sc.nextLine();
            System.out.print("Enter Author Name: ");
            String author = sc.nextLine();
            System.out.print("Enter Price: ");
            double price = sc.nextDouble();
            System.out.print("Enter Number of Pages: ");
            int numPages = sc.nextInt();
            sc.nextLine();
            books[i] = new Book(name, author, price, numPages);
        }
        System.out.println("\nDetails of all books:");
        for (int i = 0; i < n; i++) {
```

```
            System.out.println("\nBook " + (i + 1) + " Details:");
            System.out.println(books[i].toString());
        }
        sc.close();
    }
}
```

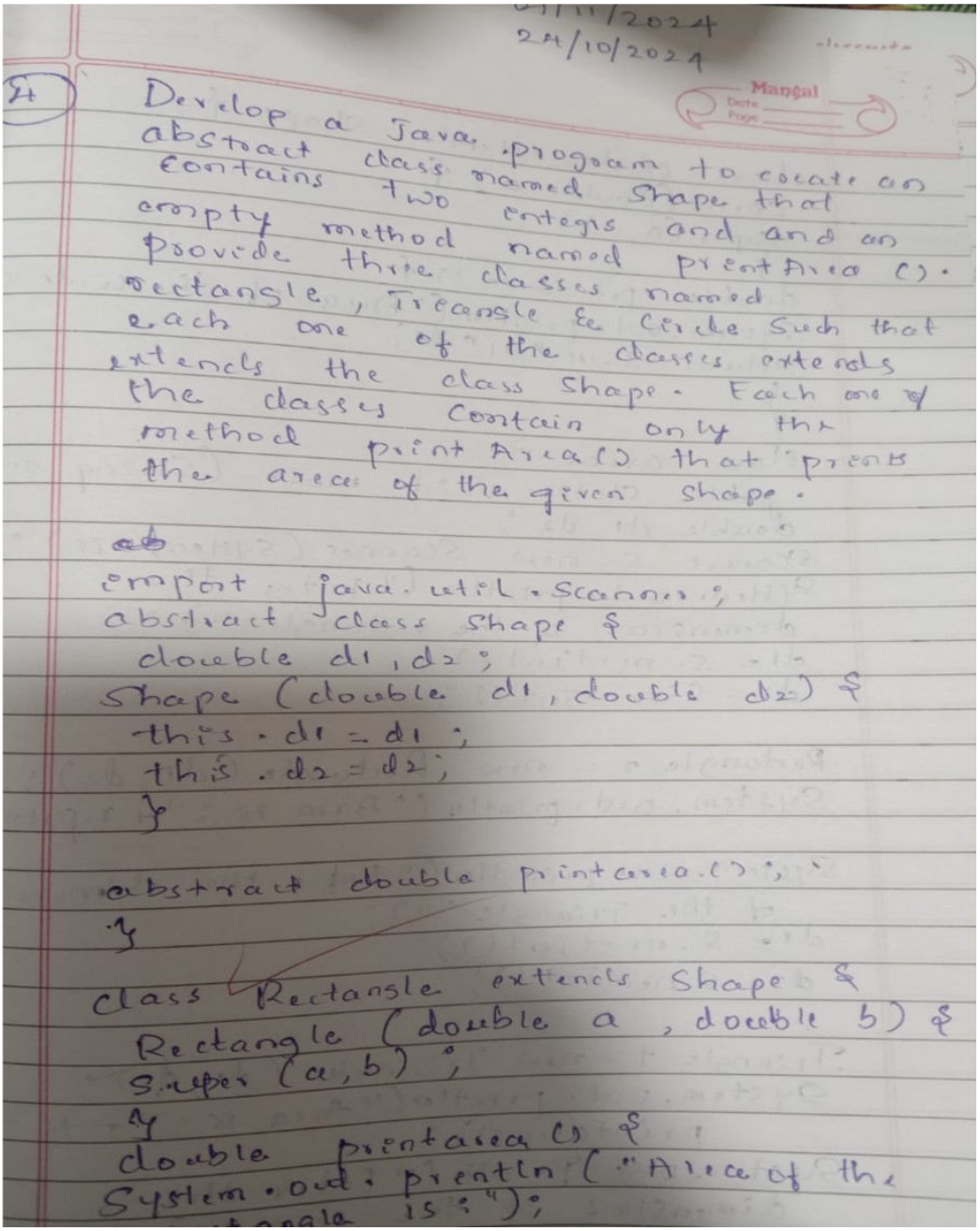
OUTPUT :

```
Enter the number of books: 2
Enter details for book 1:
Enter Book Name: Book 1
Enter Author Name: Author 1
Enter Price: 123
Enter Number of Pages: 13
Enter details for book 2:
Enter Book Name: Book
Enter Author Name: Author 2
Enter Price: 321
Enter Number of Pages: 35

Details of all books:

Book 1 Details:
Book Name: Book 1
Author: Author 1
Price: 123.0
Number of Pages: 13

Book 2 Details:
Book Name: Book
Author: Author 2
Price: 321.0
Number of Pages: 35
```

LABORATORY PROGRAM – 4

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

OBSERVATION :



24/10/2024

4 Develop a Java program to create an abstract class named Shape that contains two integers and and an empty method named printArea(). Provide three classes named Rectangle, Triangle & Circle such that each one of the classes extends the class Shape. Each one of the classes contain only the method printArea() that prints the area of the given shape.

```
import java.util.Scanner;
abstract class Shape {
  double d1, d2;
Shape (double d1, double d2) {
  this.d1 = d1;
  this.d2 = d2;
  }

abstract double printarea();
}

class Rectangle extends Shape {
  Rectangle (double a, double b) {
  super(a,b);
  }
  double printarea() {
  System.out.println("Area of the
                      is :");
```

```
class circle extends shape {
circle(double a) {
super(a, 1);
}
double printarea() {
System.out.println("Area of the circle is");
return 3.14 * d1 * d2;
}
}

class areas {
public static void main (String args[])
double d1, d2;
Scanner s = new Scanner (System.in);
System.out.println("Enter the
dimensions of the rectangle:");
d1 = s.nextInt();
d2 = s.nextInt();

Rectangle r = new Rectangle (d1, d2);
System.out.println("Area is :" + r.printarea());
System.out.println("Enter the dimensions
of the Triangle:");
d1 = s.nextInt();
d2. s.nextInt();

Triangle t = new Triangle (d1, d2);
System.out.println("Area is :" + t.
printarea());
System.out.println("Enter the
dimensions of the circle:");
d1 = s.nextInt();
```

Circle c = new Circle (d);
System.out.println ("Area is: " + c.printcircle());
}
}

~~Enter~~
output

Enter dimensions for rectangle:
3 4
Area of rectangle
12.0

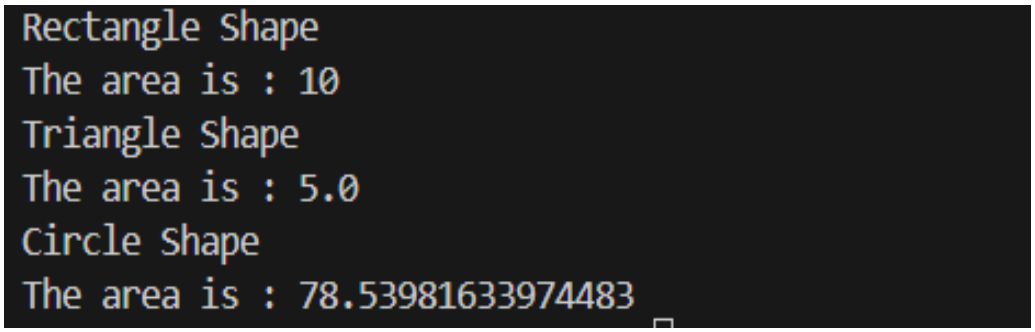
Enter dimensions of triangle:
2 3 4
Area of triangle
6.0

CODE :

```
abstract class Shape {
   int d1;
   int d2;
   Shape(int d1, int d2) {
      this.d1 = d1;
      this.d2 = d2;
   }
   abstract void printArea();
}
class Rectangle extends Shape {
   Rectangle(int l, int b) {
      super(l, b);
   }
   void printArea() {
      System.out.println("Rectangle Shape");
      System.out.println("The area is : " + d1 * d2);}}
class Triangle extends Shape {
   Triangle(int b, int h) {
      super(b, h);
   }
   void printArea() {
      System.out.println("Triangle Shape");
      System.out.println("The area is : " + 0.5 * d1 * d2);
   }
}
```

```
class Circle extends Shape {
    Circle(int r) {
        super(r, 0);
    }
    void printArea() {
        System.out.println("Circle Shape");
        System.out.println("The area is : " + (Math.PI * d1 * d1));}}
class ShapeMain {
    public static void main(String[] args) {
        Shape shape;
        shape = new Rectangle(5, 2);
        shape.printArea();
        shape = new Triangle(5, 2);
        shape.printArea();
        shape = new Circle(5);
        shape.printArea();
    }
}
```

OUTPUT :



LABORATORY 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.

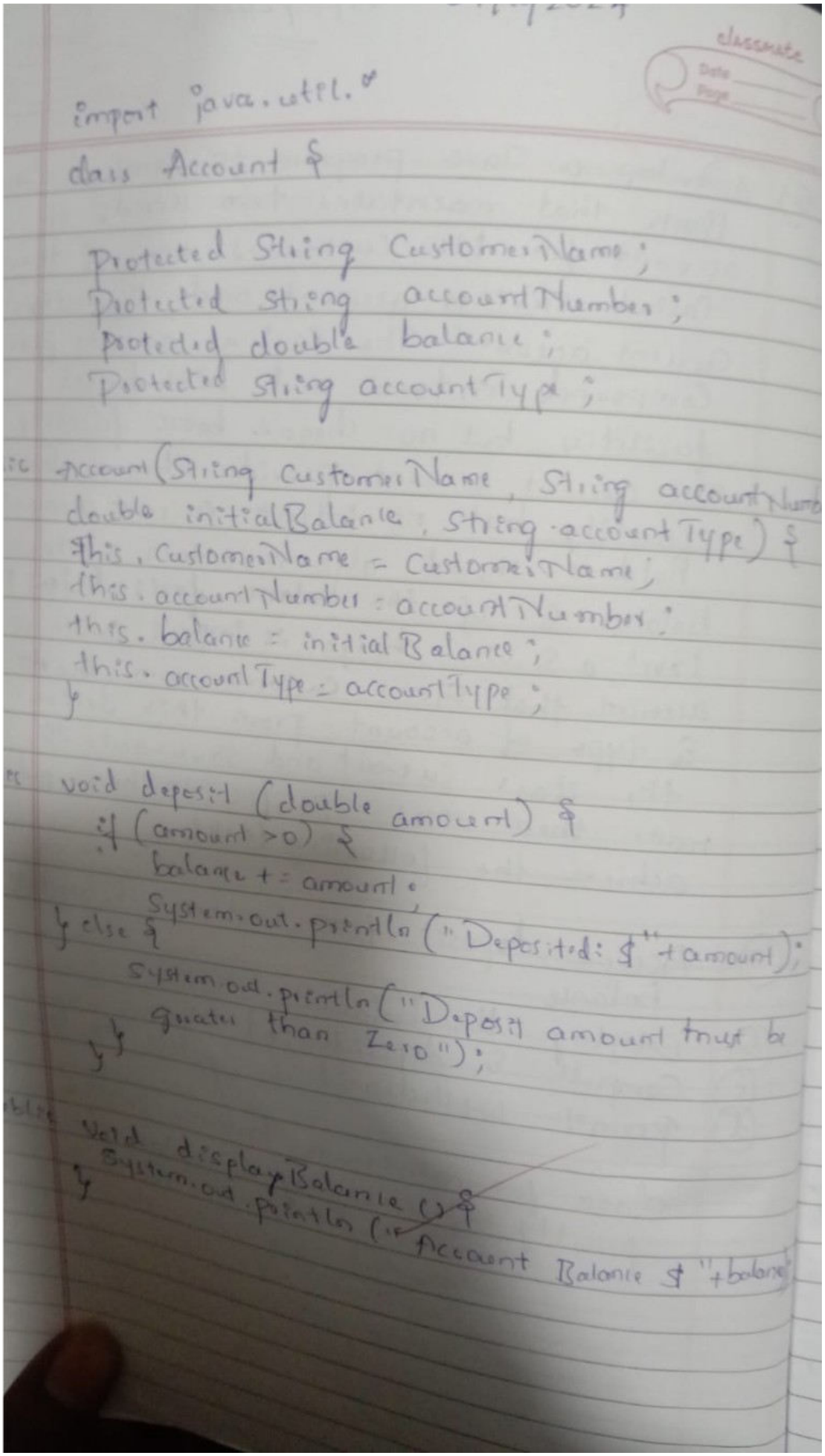
OBSERVATION :

07/11/2024

5) Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one of them called savings account and the other current account. The savings account provide compound interest and withdrawl 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 & if the balance falls below this level, a service charge is imposed. Create a class account that stores customer name, acc no. & type of account. From this derive the class cur-acct and sav-acct to make them more specific to their order to achieve the following tasks.

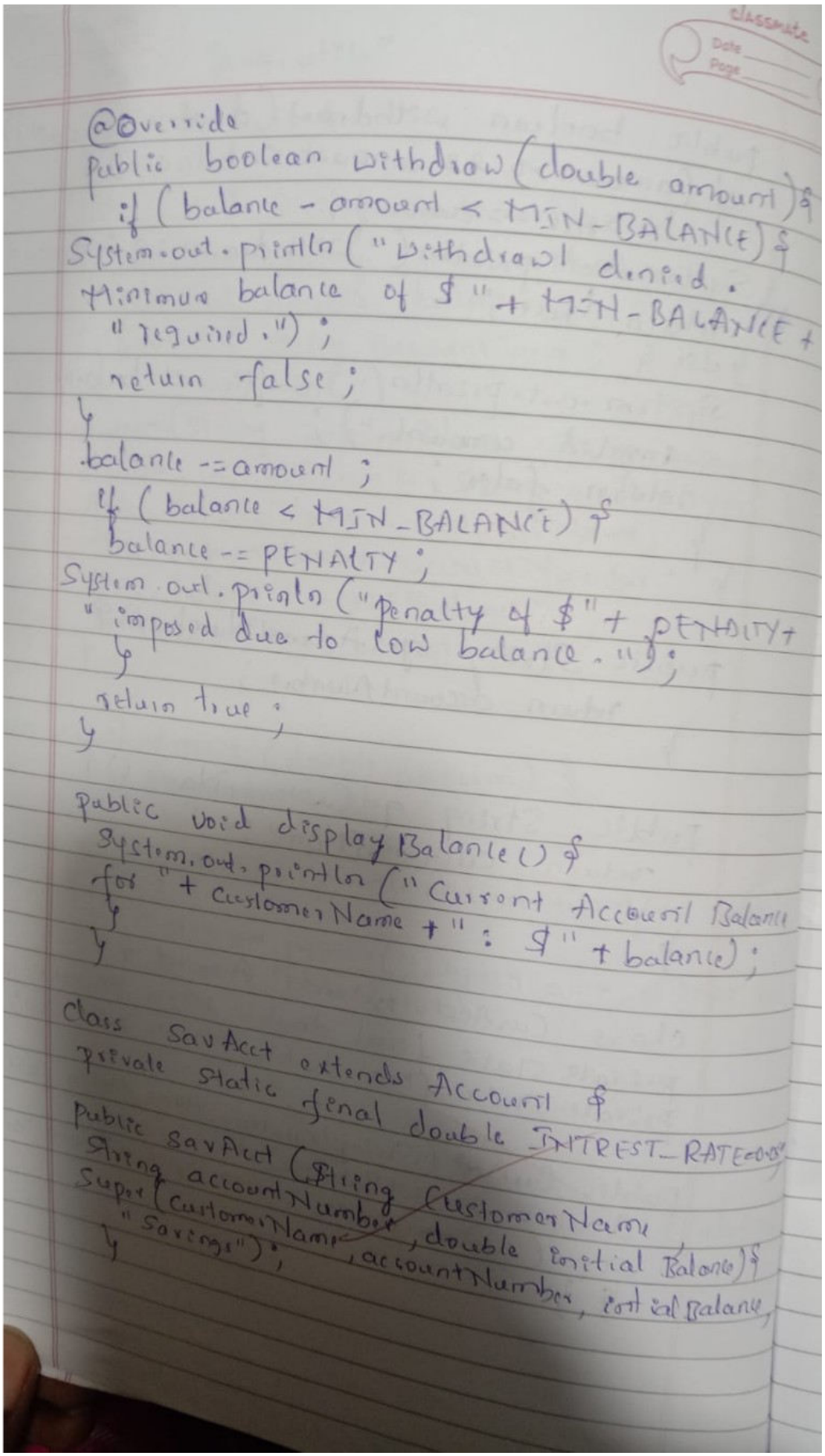
1) Accept deposit from customer & update the balance
2) Display the balance
3) Compute & deposit interest
4) Permit withdrawl & update the balance

Check for minimum balance, impose penalty if necessary & update the balance



```
import java.util.*
class Account {

    Protected String CustomerName;
    Protected String accountNumber;
    protedid double balance;
    Protected String accountType;

ic  Account(String CustomerName, String accountNumb
    double initialBalance, String accountType) {
    this.CustomerName = CustomerName;
    this.accountNumber = accountNumber;
    this.balance = initialBalance;
    this.accountType = accountType;
    }

es  void deposit (double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Deposited: $" + amount);
        } else {
            System.out.println("Deposit amount must be
            greater than Zero");
        }

blic void displayBalance() {
        System.out.println("Account Balance $" + balance);
    }
```

```
Public boolean withdraw (double amount) {
  if (amount > 0 && amount <= balance) {
    balance -= amount;
System.out.println("Withdrawn: $" + amount);
  return true;
} else {
  System.out.println("Insufficient balance or
    invalid amount.");
  return false;
  }
}

public String getAccountNumber() {
    return accountNumber;
  }

public String getCustomerName() {
  return CustomerName;
}
}

class CurAcct extends Account {
private static final double MIN-BALANCE = 10
private static final double PENALTY = 50.00;

Public CurAcct (String CustomerName, String acct
  double initialBalance) {
  super(customerNam, accNO, initialBalanc, "C
}}
```



```
@Override
public boolean withdraw(double amount){
  if (balance - amount < MIN_BALANCE){
System.out.println("Withdrawal denied.
Minimum balance of $" + MIN_BALANCE +
  " required.");
  return false;
}
balance -= amount;
 if (balance < MIN_BALANCE){
  balance -= PENALTY;
System.out.println("Penalty of $" + PENALTY +
" imposed due to low balance.");
  }
  return true;
}

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

class SavAcct extends Account {
private static final double INTREST_RATE=0.05;
public SavAcct(String customerName,
String accountNumber, double initialBalance){
super(customerName, accountNumber, initialBalance,
  "Savings");
  }
```

```
Public void ComputeAndDepositInterest() {
    double interest = balance * INTEREST_RATE;
    balance += interest;
System.out.println("Interest of $" + interest +
"has been credited to your account.");
}

@Override
public boolean Withdraw (double amount) {
    if (balance >= amount) {
        balance -= amount;
        return true;
} else {
    System.out.println("Insufficient balance
        to withdraw.");
    return false;
}
}

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

Class Bank {
    public static void main (String[] args) {
Account CurrentAccount = new CurAcct("John Doe",
    "CA12345", 1500.00);
Account SavingsAccount = new SavAcct("Jane Smith"
    "SA54321", 2000.00);

    CurrentAccount.displayBalance();
    SavingsAccount.displayBalance();
```

```
CurrentAccount.deposit(500);
CurrentAccount.withdrawl(100);
CurrentAccount.displayBalance();

((SavAcct)SavingsAccount).computeAndDepositInterest();
SavingsAccount.displayBalance();

SavingsAccount.withdraw(500);
SavingsAccount.displayBalance();

CurrentAccount.withdraw(1100);
CurrentAccount.displayBalance();
}
}
```

O/P

```
Current Account Balance for John Doe: $1500.0
Savings Account Balance for Janesmith: $2000.0
Deposited: $500.0
Current Account Balance for John Doe: $1900.0
Interest of $100.0 has been credited to your account
Savings Account Balance for Janesmith: $2100.0
Savings Account Balance for Janesmith: $1600.0
Withdrawl denied. Minimum balance of $1000.0
required
Current Account Balance for John Doe: $1900.0
```

CODE :

```
class Account{
   String customer_name;
   long account_no;
   String typeofAccount;
   double balance;
   Account(String customer_name,long account_no,String
typeAccount,double balance){
      this.customer_name=customer_name;
      this.account_no=account_no;
      this.typeofAccount=typeAccount;
      this.balance=balance;    }
   void deposites(double amount){
      this.balance+=amount;
      System.out.println("Amount of "+amount+" has been debited");  }
   void displayBalance(){
      System.out.println("The Balance Of The "+account_no+" and Name
"+customer_name+" is :"+balance);    }
   void withdraw(double amount){
      if(balance<amount){
         System.out.println("Insufficient Balance");    }else{
         this.balance-=amount;
         System.out.println("Amount of "+amount+" succesfully
withdrwn");}}}
class SavingAccount extends Account{
   double compound_interest=0.04;
   SavingAccount(String customername,long account_no){
```

```
      super(customername,account_no,"SAVINGS",0);}
   void compoundInterest(){
      double interest=balance*0.04;
      balance+=interest;
      System.out.println("Intereset deposited");}  }
   class CurrentAccount extends Account{
      boolean chequebook=true;
      double minimum_Balance=5000;
      double service_charge=50;
      CurrentAccount(String customername,long account_no){
         super(customername, account_no,"CURRENT", 5000);  }
      void withdraw(int amount){
         if(balance>amount){
            this.balance-=amount;
            System.out.println("Amounte of "+amount+" withdrawed
Succesfully");
            imposePenalty();    }else{
            System.out.println("Insuffescient  Balance");}}
      void imposePenalty(){
         if(balance<minimum_Balance){
            balance-=service_charge;
            System.out.println("Penalty Added");}    }  }
   public class Bank {
       public static void main(String[] args) {
         SavingAccount savingAccount=new
SavingAccount("sushanth",123456789 );
         savingAccount.displayBalance();
```

```
        savingAccount.withdraw(500);
        savingAccount.deposites(1000);
        savingAccount.deposites(1000);
        savingAccount.compoundInterest();
        savingAccount.withdraw(1000);
        savingAccount.displayBalance();
        System.out.println();
        CurrentAccount currentAccount=new
CurrentAccount("likhith",987654321 );
        currentAccount.displayBalance();
        currentAccount.withdraw(500);
        currentAccount.deposites(1000);
        currentAccount.deposites(1000);
        currentAccount.withdraw(1000);
        currentAccount.displayBalance(); }
   }
```

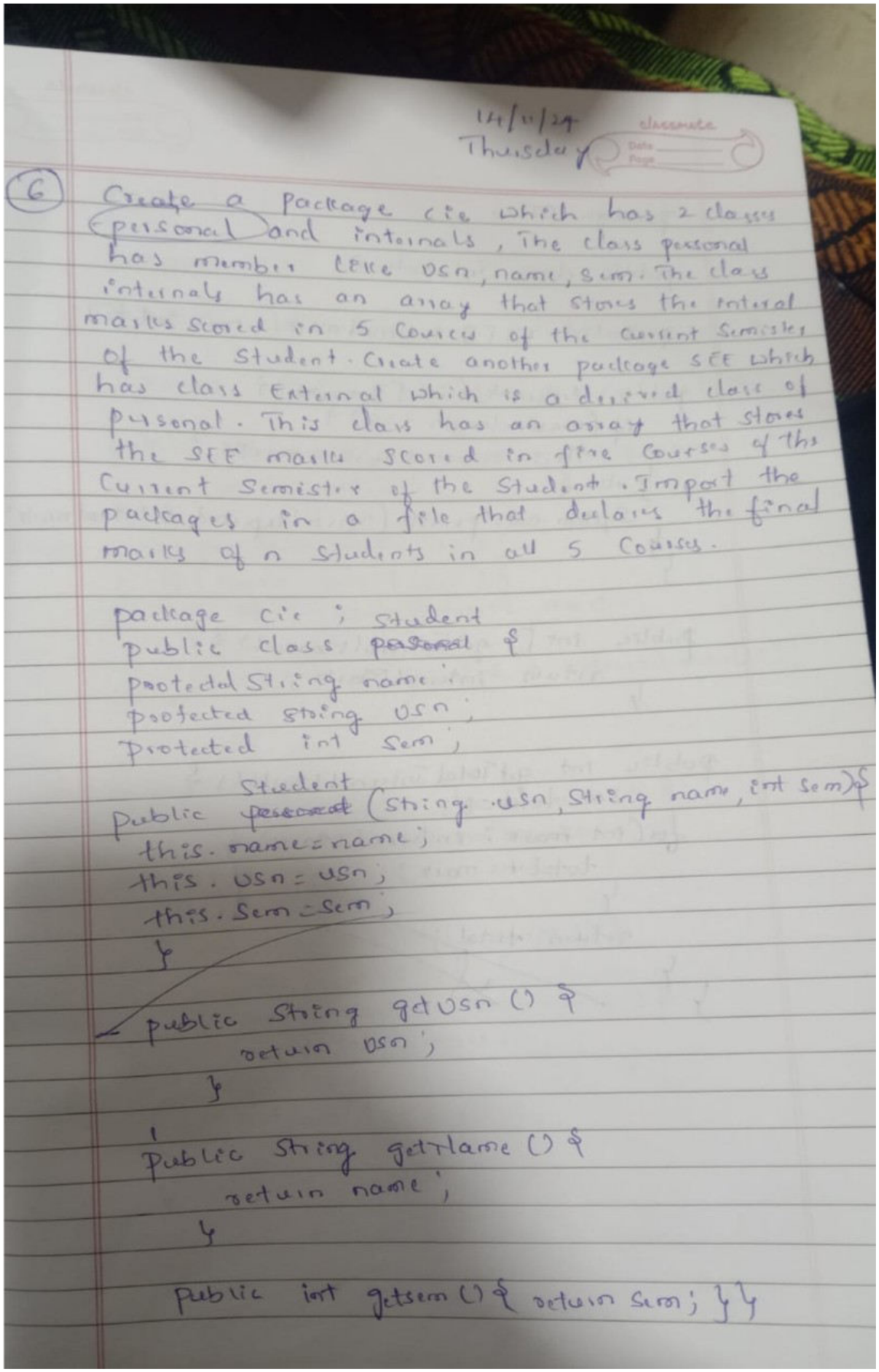
OUTPUT :

```
The Balance Of The 123456789 and Name sushanth is :0.0
Insufficient Balance
Amount of 1000.0 has been debited
Amount of 1000.0 has been debited
Intereset deposited
Amount of 1000.0 succesfully withdrwn
The Balance Of The 123456789 and Name sushanth is :1080.0

The Balance Of The 987654321 and Name likhith is :5000.0
Amounte of 500 withdrawed Succesfully
Penalty Added
Amount of 1000.0 has been debited
Amount of 1000.0 has been debited
Amounte of 1000 withdrawed Succesfully
The Balance Of The 987654321 and Name likhith is :5450.0
```

LABORATORY PROGRAM – 6

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

OBSERVATION :

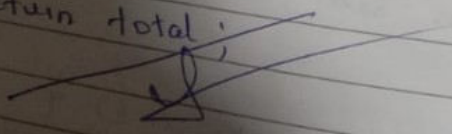


14/11/24
Thursday

(6) Create a package cie which has 2 classes personal and internals, The class personal has member like usn, name, sem. The class internals has an array that stores the internal marks scored in 5 courses of the current semister of the student. Create another package SEE which has class External which is a derived class of personal. This class has an array that stores the SEE marks scored in five courses of the current semister of the student. Import the packages in a file that declares the final marks of n students in all 5 courses.

```
package cie ; student
Public class personal {
protected String name;
protected string usn;
Protected int sem;

Public personal (String usn, String name, int sem){
this.name=name;
this.usn=usn;
this.sem=sem;
}

public String getusn () {
    return usn;
}

Public String getName () {
    return name;
}

Public int getsem () { return sem; } }
```

```
package CIE;

public class Internals {

protected int [] internalmarks = new int [5];

public internals (int [] marks) {
    if (marks.length == 5) {
        this.internalmarks = marks;
    } else {
        System.out.println ("Error: Expected 5 internal ma
    }
}

public int [] getInternalMarks () {
    return internalMarks;
}

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



```
package SEE;
import CIE.Student;
public class External extends Student {
    private int[] externalMarks = new int[5];
    public External(String usn, String name, int sem,
            int marks[]) {
        super(usn, name, sem);
        if(marks.length == 5) {
            this.externalMarks = marks;
        } else {
            System.out.println("Error: Expected 5 external marks");
        }
    }
    public int[] getExternalMarks() {
        return externalMarks;
    }
    public int getTotalEMarks() {
        int total = 0;
        for(int mark : externalMarks) {
            total += mark;
        }
        return total;
    }
}
```

```
import CIE.*;
import SEE.*;
import java.util.Scanner;

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

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

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

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

int [] internalMarks = new Int [5];
Sout ("Enter 5 marks");
for (int i=0; i<5; i++) {
  System.out.print("Course" + (i+1) + ":");
internal Marks [i] = scanner.nextInt();
}

int [] externalMarks = new int [5];
Sout ("Enter exter marks for 5 courses");
for (int i=0; i<5; i++) {
Sout ("Course" + (i+1) + ":");
externalMarks [i] = scanner.nextInt();
}

External Student = new External (usn, name, sem,
                                 externalMarks);
Internals internals = new Internals (internalMarks);
```

```
int totalInternalMarks = internals.getTotalInternalMarks();
int totalExternalMarks = student.getTotalExternalMarks();
int finalMarks = totalInternalMarks + totalExternalMarks;

System.out.println("Stud Name" + student.getName());
System.out.println("USN" + student.getUsn());
System.out.println("Semester" + student.getSem());

System.out.print("Interal marks");
    for (int i=0; i<5; i++) {
    System.out.print(internals.getInternalMarks()[i]+" ");
    }

System.out.println();

System.out.print("External Marks:");
    for (int i=0; i<5; i++) {
    System.out.print(student.getExternalMarks()[i]+" ");
    }
System.out.println();

System.out.println("Total Internal Marks:" +
                    totalInternalMarks);
System.out.println("Total External Marks:" +
                    totalExteralMarks);
System.out.println("Final Marks:" + finalMarks);

scanner.close();
}
}
```

CODE :

File : CIE/Student.java

```
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;
    }
}
```

File : CIE/Internal.java

```
package cie;
public class Internals {
    public int[] internalMarks = new int[5];
    public Internals(int[] marks) {
        if (marks.length == 5) {
            System.arraycopy(marks, 0, internalMarks, 0, 5);
        } else {
            System.out.println("Error: Please provide marks for exactly 5
courses.");
        }
    }
}
```

File : SEE/External.java

```
package see;
import cie.Student;
public class External extends Student {
    public int[] externalMarks = new int[5];
    public External(String usn, String name, int sem, int[] marks) {
        super(usn, name, sem);
        if (marks.length == 5) {
            System.arraycopy(marks, 0, externalMarks, 0, 5);
        } else {
            System.out.println("Error: Please provide marks for exactly 5
courses.");
        }
    }
}
```

File : FinalMarrks.java

```
import cie.*;
import see.*;
import java.util.Scanner;
public class FinalMarks {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of students: ");
        int n = sc.nextInt();
        Student[] students = new Student[n];
        Internals[] internals = new Internals[n];
```

```
        External[] externals = new External[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for student " + (i + 1) + ":");
            System.out.print("USN: ");
            String usn = sc.next();
            System.out.print("Name: ");
            String name = sc.next();
            System.out.print("Semester: ");
            int sem = sc.nextInt();
            System.out.println("Enter internal marks for 5 courses:");
            int[] internalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = sc.nextInt();
            }
            System.out.println("Enter SEE marks for 5 courses:");
            int[] externalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                externalMarks[j] = sc.nextInt();
            }
            students[i] = new Student(usn, name, sem);
            internals[i] = new Internals(internalMarks);
            externals[i] = new External(usn, name, sem, externalMarks);
        }
        System.out.println("\nFinal Marks of Students:");
        for (int i = 0; i < n; i++) {
            System.out.println("Student: " + students[i].name + " (USN: " +
students[i].usn + ")");
```

```
            for (int j = 0; j < 5; j++) {
                int finalMarks = internals[i].internalMarks[j] +
externals[i].externalMarks[j] / 2;
                System.out.println("Course " + (j + 1) + ": " + finalMarks);
            }
        }
        sc.close();
    }
}
```

OUTPUT :

```
Enter details for student 1:
USN: 1RV23CS001
Name: John
Semester: 5
Enter internal marks for 5 courses:
18 19 20 17 16
Enter SEE marks for 5 courses:
70 60 80 90 50
Final Marks of Students:
Student: John (USN: 1RV23CS001)
Course 1: 53
Course 2: 49
Course 3: 60
Course 4: 62
Course 5: 41
```

LABORATORY 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.

OBSERVATION :

```
public int getAge() {
    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
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();
```

CODE :

```
class WrongAgeException extends Exception {
   public WrongAgeException(String message) {
      super(message); } }
class Father {
   int fatherAge;
   public Father(int age) throws WrongAgeException {
      if (age < 0) {
         throw new WrongAgeException("Father's age cannot be
negative!");
      }
      this.fatherAge = age;
      System.out.println("Father's Age: " + fatherAge); } }
class Son extends Father {
   int sonAge;
   public Son(int fatherAge, int sonAge) throws WrongAgeException {
      super(fatherAge);
      if (sonAge < 0) {
         throw new WrongAgeException("Son's age cannot be negative!");
      }
      if (sonAge >= fatherAge) {
         throw new WrongAgeException("Son's age cannot be greater than
or equal to father's age!");
      }
      this.sonAge = sonAge;
      System.out.println("Son's Age: " + sonAge); } }
public class ExceptionMain {
```

```
public static void main(String[] args) {
    java.util.Scanner sc = new java.util.Scanner(System.in);
    try {
        System.out.print("Enter Father's Age: ");
        int fatherAge = sc.nextInt();
        System.out.print("Enter Son's Age: ");
        int sonAge = sc.nextInt();
        Son son = new Son(fatherAge, sonAge);
    } catch (WrongAgeException e) {
        System.out.println("Exception: " + e.getMessage());
    } catch (Exception e) {
        System.out.println("Unexpected Exception: " + e);} } }
```

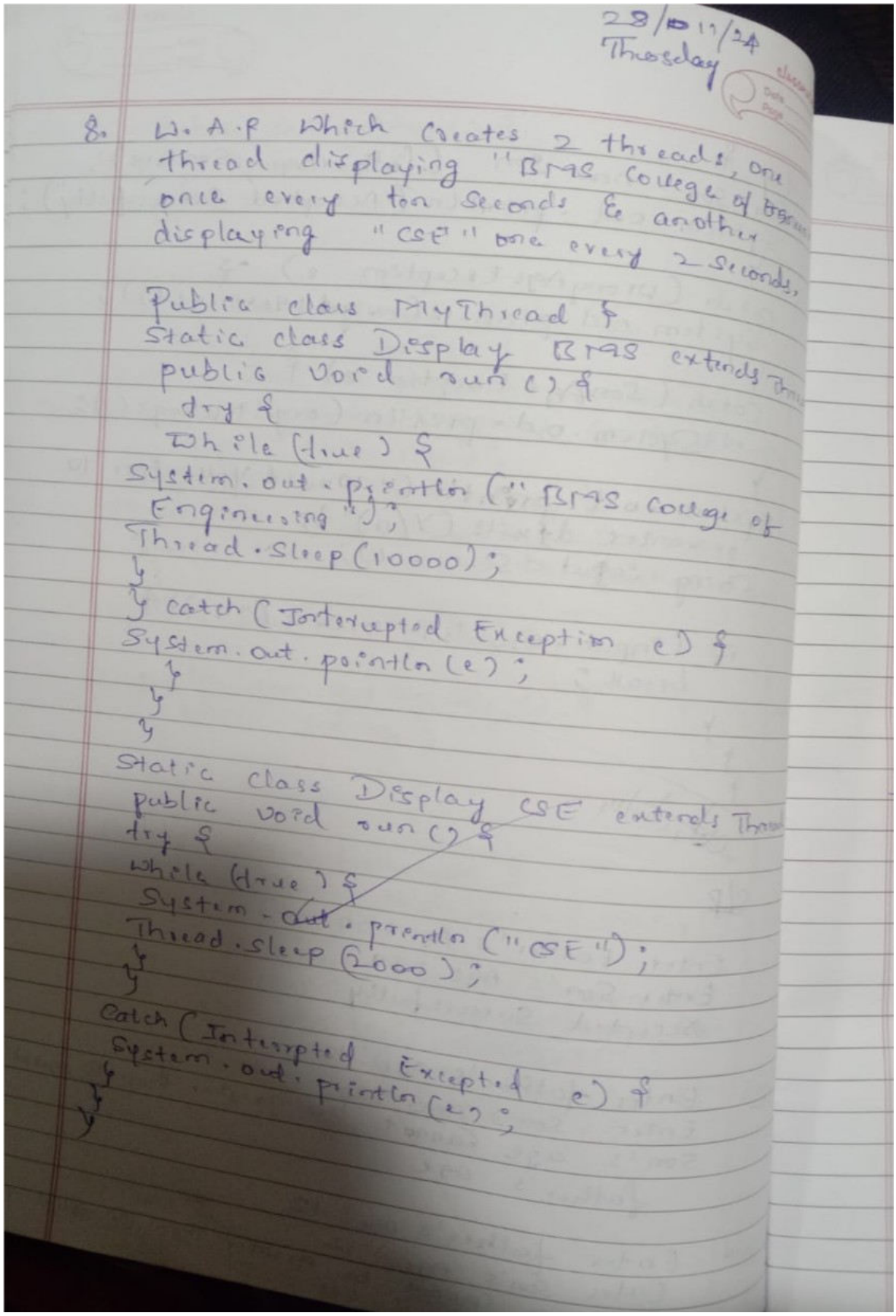
OUTPUT :

```
PS D:\3rd sem\OOJ JAVA\Git-hub> java ExceptionMain
Enter Father's Age: 40
Enter Son's Age: -10
Father's Age: 40
Exception: Son's age cannot be negative!
PS D:\3rd sem\OOJ JAVA\Git-hub> java ExceptionMain
Enter Father's Age: 40
Enter Son's Age: 50
Father's Age: 40
Exception: Son's age cannot be greater than or equal to father's age!
PS D:\3rd sem\OOJ JAVA\Git-hub> java ExceptionMain
Enter Father's Age: -40
Enter Son's Age: 20
Exception: Father's age cannot be negative!
```

LABORATORY PROGRAM – 8

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

OBSERVATION :



28/11/24
Thursday

8. W.A.P which creates 2 threads, one thread displaying "BMS College of Engineering" once every ten seconds & another displaying "CSE" once every 2 seconds.

```
Public class MyThread {
Static class Display BMS extends Thread
public void run () {
  try {
  While (true) {
System.out.println ("BMS college of
Engineering");
Thread.sleep (10000);
}
} catch (Interrupted Exception e) {
System.out.println (e);
  }
}
}
Static class Display CSE extends Thread
public void run () {
try {
while (true) {
System.out.println ("CSE");
Thread.sleep (2000);
}
}
Catch (Interrupted Exception e) {
System.out.println (e);
}
}
```

```
public static void main (String [] args){
Thread threadBMS = new Display BMS();
Thread thread CSE = new Display CSE();

threadBMS.start();
threadCSE.start();
}
}
```

O/P

BMS College of engineering
CSE
CSE
CSE
CSE
BMS College of engineering
CSE
CSE
CSE
CSE
BMS College of engineering

ctrl+c

C:\Users\BMSCE\Desktop\thread>java Mythr

CODE :

```
class BmsThread extends Thread{
   public void run(){
      try{
         while (true) {
            System.out.println("BMS college of Engineering");
            Thread.sleep(10000);
         }
      }catch(InterruptedException e){
         System.out.println(e);
      }
   }
}
 class CseThread implements Runnable{
   Thread t;

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

```
}
public class DemoThread  {
   public static void main(String[] args) {
      BmsThread b=new BmsThread();
      Runnable cse=new CseThread();
      Thread t1=new Thread(cse);
      b.start();
      t1.start();;
   }
}
```

OUTPUT :

```
BMS college of Engineering
Cse
Cse
Cse
Cse
Cse
BMS college of Engineering
Cse
Cse
Cse
Cse
Cse
BMS college of Engineering
Cse
Cse
Cse
Cse
Cse
```

LABORATORY PROGRAM – 9

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

OBSERVATION :

Week 9

(9) W.A.p that creates a user interface to Perform integer division that user enters numbers in the text files n1 & n2. The ÷ of n1 & n2 it displays result in field when ÷ button is clicked if n1 & n2 not an integer it should throw a no format exception if n1 & n2 were zero the program should throw AE.

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

public class main demo extends
  Frame implements Action listner {

Textfield n1, n2;
Button dResult;
Label outResult;
String out = "";
double resultNum;
int flag = 0;

Public Maindemo() {
 SetLayout (new FlowLayout());
  dResult = new Button("Result");
label n1 = new label("n1:", Label.RIGHT);
Label n2 = new label("n2", Label.RIGHT);
n1 = new TextField(5);
n2 = new TextField(5);
outResult = new label();
add(n1);
add(n2);
add(Result);
n1.addActionListner(this);
```

```
public void paint (Graphics g){
Font Customfont = new font ("Serif",);
g.SetFont (Customfont);

 if (flag==0){
g.drawString (out, outResult.getX());
}
 else {
 g.drawString(out,100,200);
 flag=0;
 }}
 public Static void main(String() a){

Maindemo dm= new Main demo();
dm.SetSize (new Dimension (800, 400));
dm.SetTitle ("Division of Integers");
dm.SetVisible (true);
}
}
```

Output

Num: 1 [100] Num2; [20] Reject

Result = 100/20 = 50

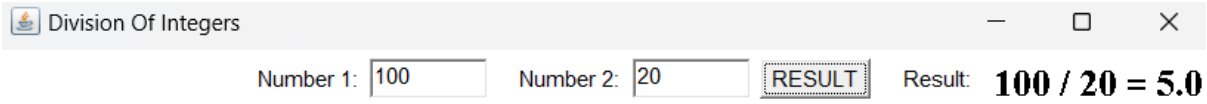
CODE :

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

public class Maindemo extends Frame implements ActionListener {
   TextField num1, num2;
   Button dResult;
   Label outResult;
   String out = "";
   double resultNum;
   int flag = 0;
   public Maindemo() {
      setLayout(new FlowLayout());
      dResult = new Button("RESULT");
      Label number1 = new Label("Number 1:", Label.RIGHT);
      Label number2 = new Label("Number 2:", Label.RIGHT);
      num1 = new TextField(5);
      num2 = new TextField(5);
      outResult = new Label("Result:", Label.RIGHT);
      add(number1);
      add(num1);
      add(number2);
      add(num2);
      add(dResult);
```

```
    add(outResult);
    num1.addActionListener(this);
    num2.addActionListener(this);
    dResult.addActionListener(this);
    addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent we) {
            System.exit(0);
        }       });
}
public void actionPerformed(ActionEvent ae) {
    int n1, n2;
    try {
        if (ae.getSource() == dResult) {
            n1 = Integer.parseInt(num1.getText());
            n2 = Integer.parseInt(num2.getText());
            if (n2 == 0) {
                throw new ArithmeticException("Division by zero"):}
            resultNum = (double) n1 / n2;
            out = n1 + " / " + n2 + " = " + resultNum;
        }
    } catch (NumberFormatException e1) {
        flag = 1;
        out = "Number Format Exception! Please enter valid integers.";
    } catch (ArithmeticException e2) {
        flag = 1;
        out = "Divide by 0 Exception! " + e2.getMessage();}
```

```
        repaint();}
    public void paint(Graphics g) {
        Font customFont = new Font("Serif", Font.BOLD, 20);
    g.setFont(customFont);
        if (flag == 0) {
            g.drawString(out, outResult.getX() + outResult.getWidth() + 10,
outResult.getY() +20);
        } else {
            g.drawString(out, 100, 200);
            flag = 0;
        } }
    public static void main(String[] args) {
        Maindemo dm = new Maindemo();
        dm.setSize(new Dimension(800, 400));
        dm.setTitle("Division Of Integers");
        dm.setVisible(true);
    }
}
```

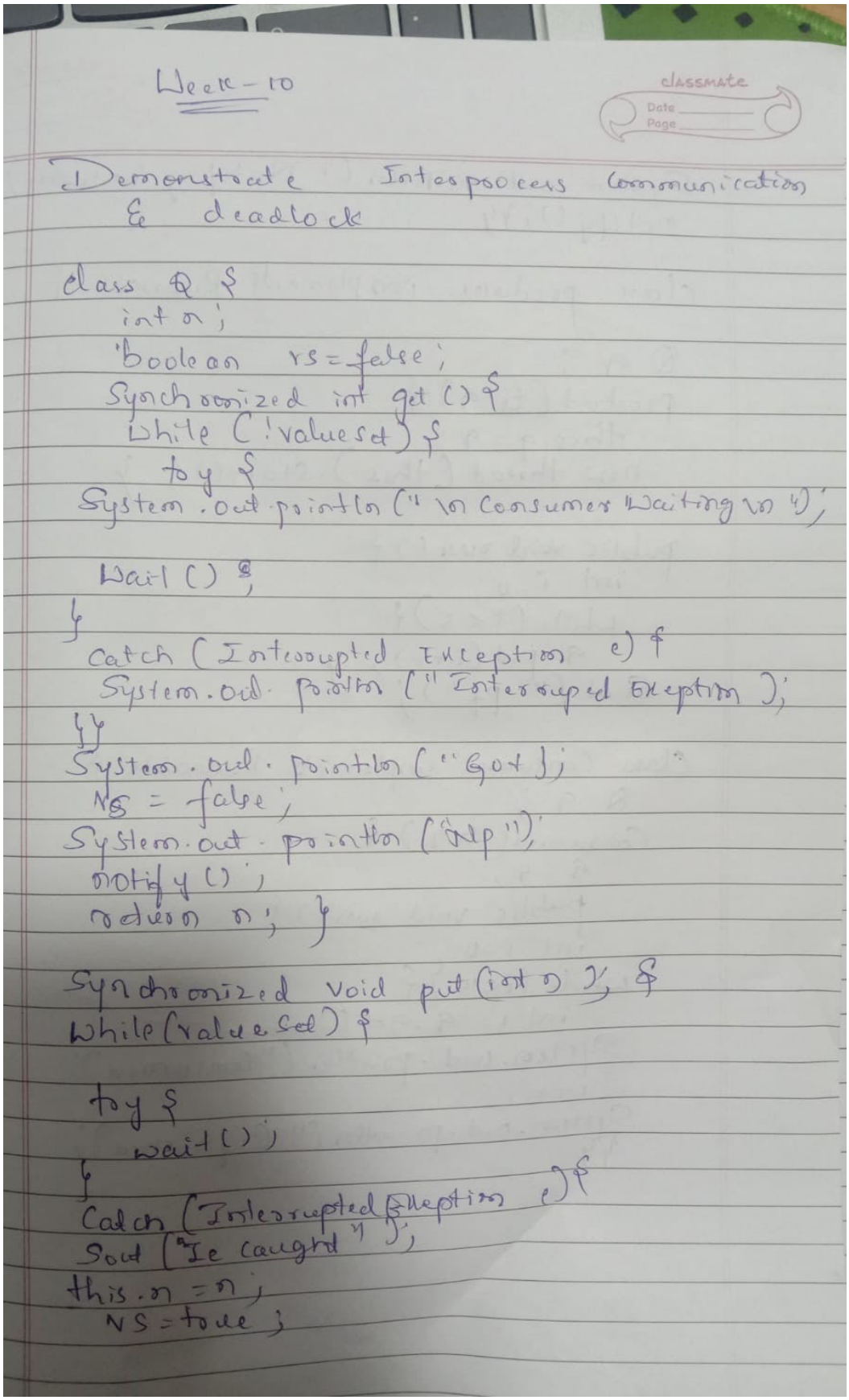
OUTPUT :


Division Of Integers
Number 1:
100
Number 2:
20
RESULT
Result:
100 / 20 = 5.0

LABORATORY PROGRAM – 10

Demonstrate Inter process Communication and deadlock.

OBSERVATION :



Week - 10

1. Demonstrate Interprocess communication & deadlock

```
class Q {
    int n;
    boolean vs = false;
    Synchronized int get() {
    While (!valueSet) {
        try {
System.out.println("\n Consumer Waiting \n");

    wait();
}
    catch (InterruptedException e) {
    System.out.println("Interrupted Exception");
}}
System.out.println("Got");
    VS = false;
System.out.println("\nP");
    notify();
    return n; }

Synchronized void put(int n); {
While(valueSet) {

    try {
        wait();
}
    Catch (InterruptedException e){
    Sout("Ie caught");
this.n = n;
    VS = true;
```

```
System.out.println("Notifying Consumer");
notify();}}

class producer implements Runnable {

Q q;
producer(Q q) {
   this.q = q;
   new Thread(this).start(); }

public void run() {
   int i=0;
   while(i<5){
      q.put(i++); }
   Sout("pf");}

Class Consumer implements Runnable {
   Q q;
   Consumer(Q q) {
      q=q;
      public void run() {
      int i=0;
   while(i<15) {
      int r = q.get();
   System.out.println("Consumed");
      i++;
   System.out.println("C. finished");
   }}
```

```
class pcfixed {
    psvm(String a[]){
      Q q = new Q();
      new Producer(q);
      new Customer(q);
    System.out.println("press ctrl-c to stop");}
```

o/p

```
press Ctrl-c to stop
put : 0

Notifying Consumer

producer waiting

Got : 0

Notifying producer
put : 1
Consumed : 0

Notifing Consumed
producer waiting

Got : 1

Notifying producer
Consumed : 1
put : 2
Notifying Consumer
producer waiting
Got : 2
Notifiying producer,
```

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("\nNotifying 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("\nNotifying 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++); }
        System.out.println("Producer finished."); } }
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++; }
        System.out.println("Consumer finished.");
    } }
```

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

OUTPUT :

```
Press Control-C to stop.
Put: 0

Notifying Consumer


Producer waiting

Got: 0

Notifying Producer

Put: 1
Consumed: 0

Notifying Consumer


Producer waiting

Got: 1

Notifying Producer

Consumed: 1
Put: 2

Notifying Consumer


Producer waiting

Got: 2

Notifying Producer
```

Deadlock

```
class A {
Synchronized void foo (B b) {
String n = Thread.CurrentThread().getName();
Sout (name + " entered A.foo");

try {
   Thread.sleep (1000);
  }
  catch (Exception e) {
   Sout ("A interrupted");
  }
Sout (name + " trying to call B.last()");
   b.last();
 }
 Synchronized void last() {
  Sout ("Inside A.last");
   }
 }

class B {
 Synchronized void bar (A a) {
String n = Thread.CurrentThread().getName();
 Sout ("B.bar");

 try {
  Thread.Sleep (1000);
 }
 Catch ("B Interrupted");
  }
```

```
Sout ("Trying to Call A.last()");
  a.last();
}
Synchronized void last(){
Sout ("Inside B.last");

Class Deadlock Implements Runnable {
  A a = new A();
  B b = new B();

  Deadlock(){

  Thread t = new Thread (this, "RacingThread");
    t.start();
    a.foo(5);

  }

  Public void run(){
      b.bar(a);
    Sout ("Back in other thread");

  }
Public Static void main(String args[]){
    new Deadlock(); }}
```

output

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

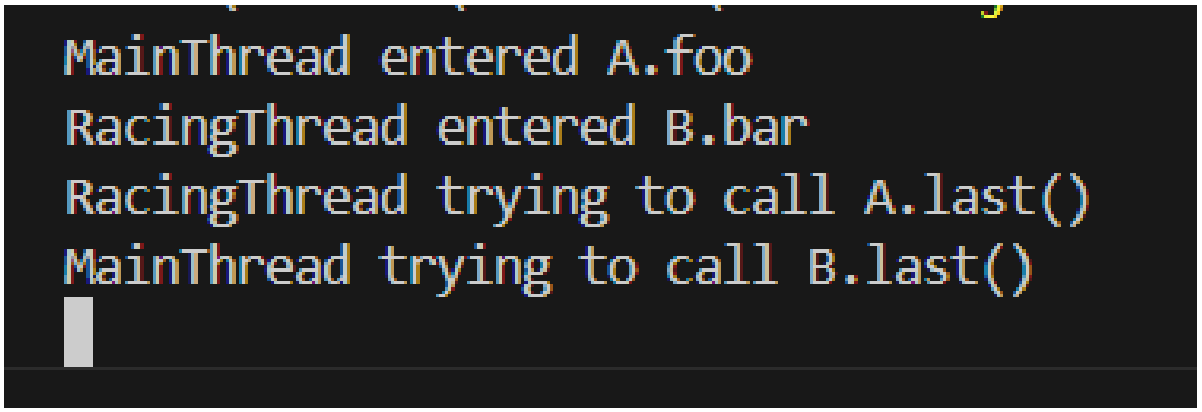
CODE :

```
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();
    }
```

}

OUTPUT :


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