VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

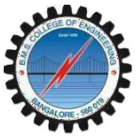
SUSHANTH RAI (1BM24CS425)

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 **SUSHANTH RAI(1BM24CS425),** 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

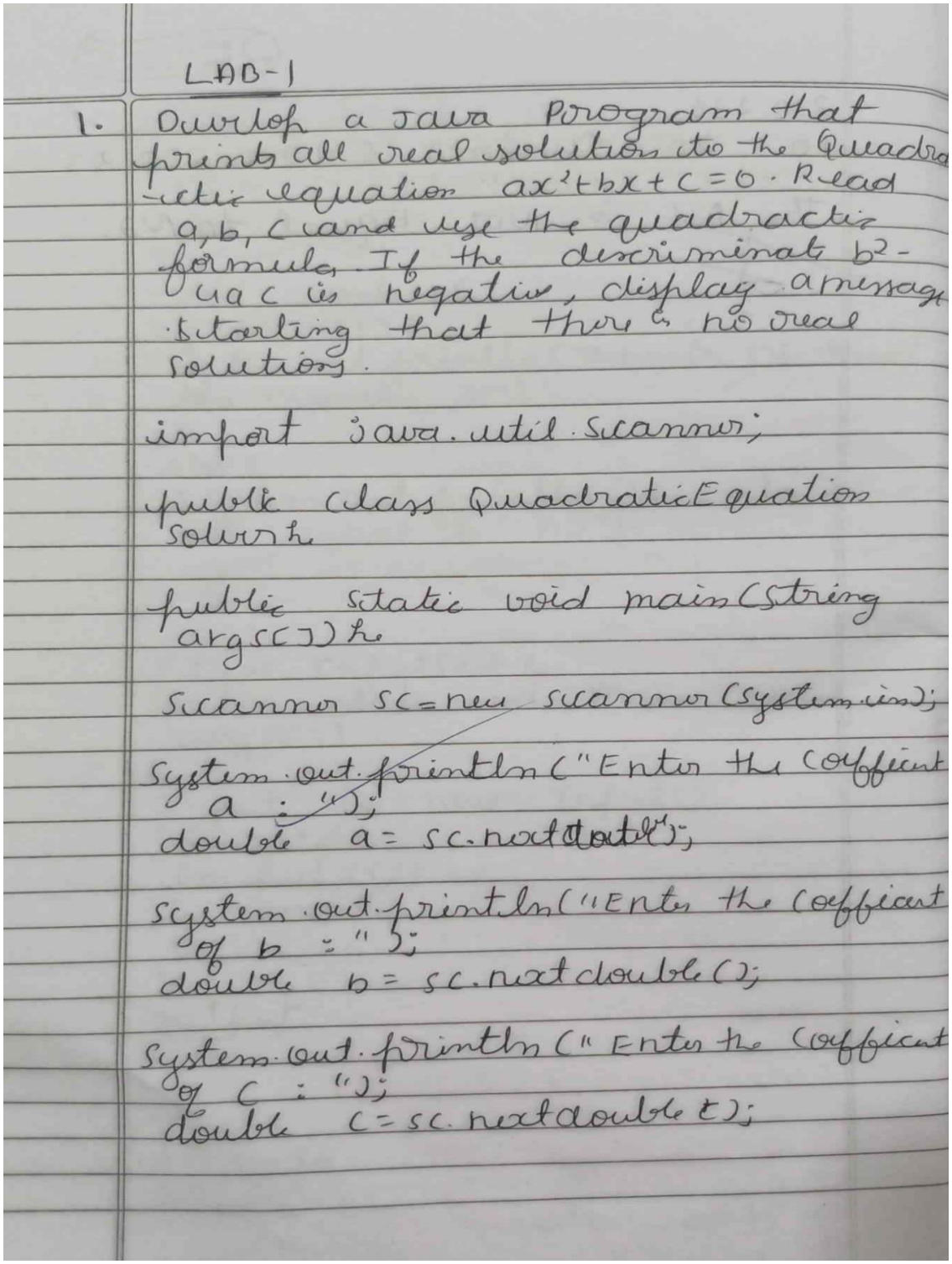
INDEX

Github Link : https://github.com/sushanthraiurwa/1BM24CS425-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-1

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

```
import java.util.Scanner;

public class QuadraticEquationSolver{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Coefficient a : ");
double a=sc.nextDouble();

System.out.println("Enter the Coefficant of b : ");
double b=sc.nextdouble();

System.out.println("Enter the Coefficant of c : ");
double c=sc.nextdouble();
```

```
double discriminant = b*b - 4*a*c;
if (discriminant > 0) {
double root1 = (-b + Math.sqrt(discriminant)
 / (2*a);

double root2 = (-b + Math.sqrt(discrimi
 / (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("There is no real or
 distinct solutions");
}
Scanner.close();
}
}
```

output:

* Case 1:

Enter the coefficient a: 10
Enter the coefficient b: 20
Enter the coefficient c: 30

The Equation has no real solutions.

* Case 2:-

Enter the coefficent a: 1
Enter the coefficent b: -3
Enter the coefficent c: 2

The Equation has two real and distinct solutions:

Root 1 : 2.0
Root 2 : 1.0

* Case 3:

Enter the coefficent a: 1
Enter the coefficent b: 2
Enter the coefficent c = 1

The Equation has one real solution (a double root):
Root : -1.0.

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();
    }
}
```

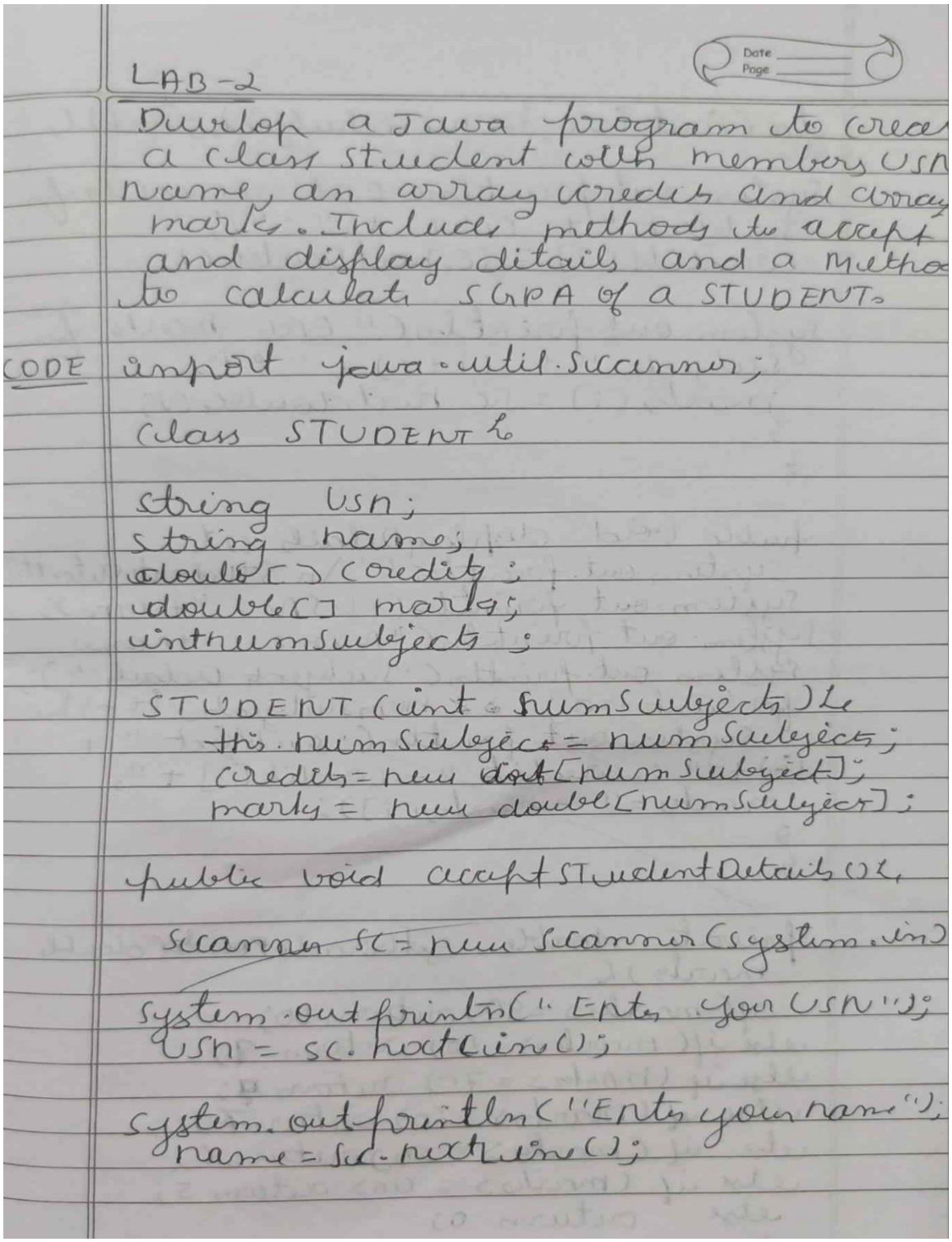
OUTPUT :

```
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 :



LAB-2

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

CODE

```
import java.util.Scanner;

class STUDENT {

String USN;
String name;
double[] credit;
double[] marks;
int numsubjects;

STUDENT(int numSubjects){
    this.numSubjects = numSubjects;
    credits = new double[numSubjects];
    marks = new double[numSubjects];

public void acceptStudentDetails(){

Scanner sc = new Scanner(System.in)

System.out.println("Enter your USN");
USN = sc.nextLine();

System.out.println("Enter your name");
name = sc.nextLine();
```

```
for(int i=0; i<numSubjects; i++){
System.out.println("Enter credits for
Subject " + (i+1) + " :");
credits[i] = sc.nextDouble();

System.out.println("Enter marks for
Subject " + (i+1) + " :");
marks[i] = sc.nextdouble();
}
}

public void displayDetails(){
System.out.println("\n Student details
System.out.println("USN : "+usn);
System.out.println("Name : "+name);
System.out.println("Subject Details:");
for(int i=0; i<numSubjects; i++){
System.out.println("subject " +
(i+1) + " : credits " + credits[i] + ",
Marks = " + marks[i]);
}
}.

private double getGradePoints(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 void calculateSGPA(){
    double totalCredits = 0;
    double totalGrade = 0;

for(int i = 0; i < numSubjects; i++){

    double Gradepoint = getGradepoint(marks
                                      [i]
totalgradepoints += Gradepoint *
    marks(credits[i];
totalCredits += Credits[i];
}

    double sgpa = totalGradpoint/totalCred
system.out.println("SGPA: " +sgpa);


public class StudentMain{
public Static void main(String args[]
{

Student Stud = new Student(3);

Student acceptDetails();
student.DisplayDetails();
Student.CalculateSGPA();
}
}
```

Multiple Student object

Output:

Enter USN : 125
Enter Name : Sushanth Rai
Enter credit for sub 1 : 3
Enter marks for subject : 80
Enter credit for sub 2: 4
Enter marks for sub 2 : 90
Enter credit for sub 3 : 3
Enter marks for sub 3 : 80

Student Details :
USN : 125
Name : Sushanth

Subject Details :
Subject 1: Credits = 3.0, marks = 80.0
subject 2: Credits = 4.0, marks = 80.0
Subject 3: Credits = 3.0, marks = 80.0.
SGPA 9.4.

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();
    }
}
```

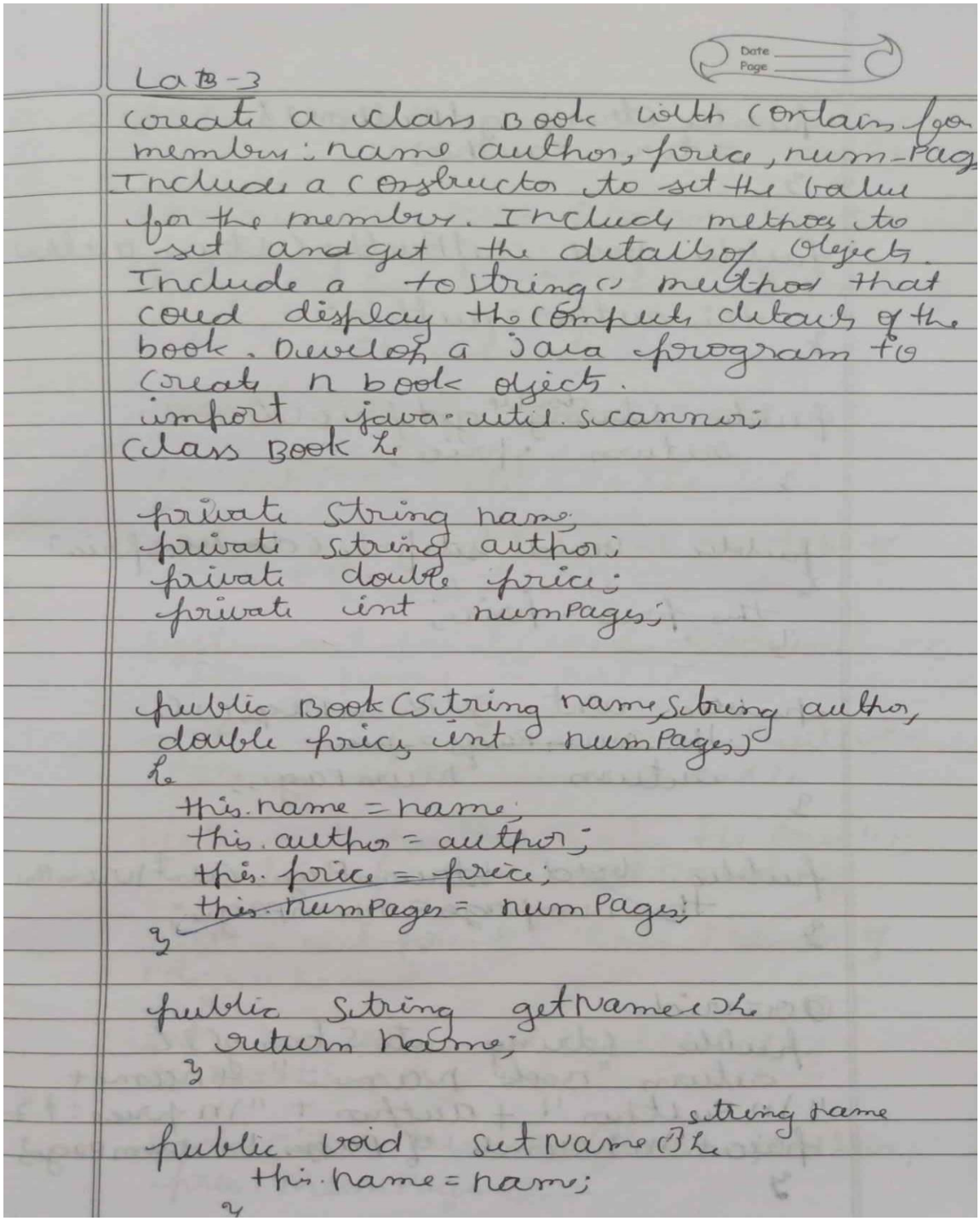
OUTPUT :

```
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 :



Lab-3

Create a class Book with contains four members: name author, price, num_Pages. Include a Constructor to set the value for the member. Include method to set and get the details of Objects. Include a toString() method that could display the complete details of the book. Develop a Java program to Create n book objects.

```
import java.util.Scanner;
class Book {

private String name;
private String author;
private double price;
private int numPages;

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

public String getName(){
  return name;
}

public void setName(String name
  this.name = name;
}
```

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

public void setAuthor(String author)
{
    this.author = author;
}

public String double getPrice() {
    return price;
}

public void setPrice(double price)
{
    this.price = price;
}

public int getNumPages() {
    this.numPages = numPages
    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.println("Enter the Number of books");
int n = sc.nextInt();

Book[] books = new Book[n];

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

System.out.println("Enter the details of books" + (i+1) + ":");

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

System.out.println("Enter Author name");
String Author = sc.nextLine();

System.out.println("Enter the 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);
```

```
}

// Display Details of all books.
System.out.println("Details of all books");

for(int i=0;i<n;i++){
  System.out.println("Book "+(i+1)+" Details");
  System.out.println(books[i].toString());
}
sc.close();
}
}
```

output:-

```
Enter the number of Books=2
Enter the details of book 1.
Enter the Name: book1
Enter author Name: Author1
Enter price: 123
Enter Number of Pages: 12

Enter details of book 2
Enter the Name: Book2
Enter the author Name: Author2
Enter price: 321
Enter Number of Pages: 35.

Details of books
```

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();
    }
}
```

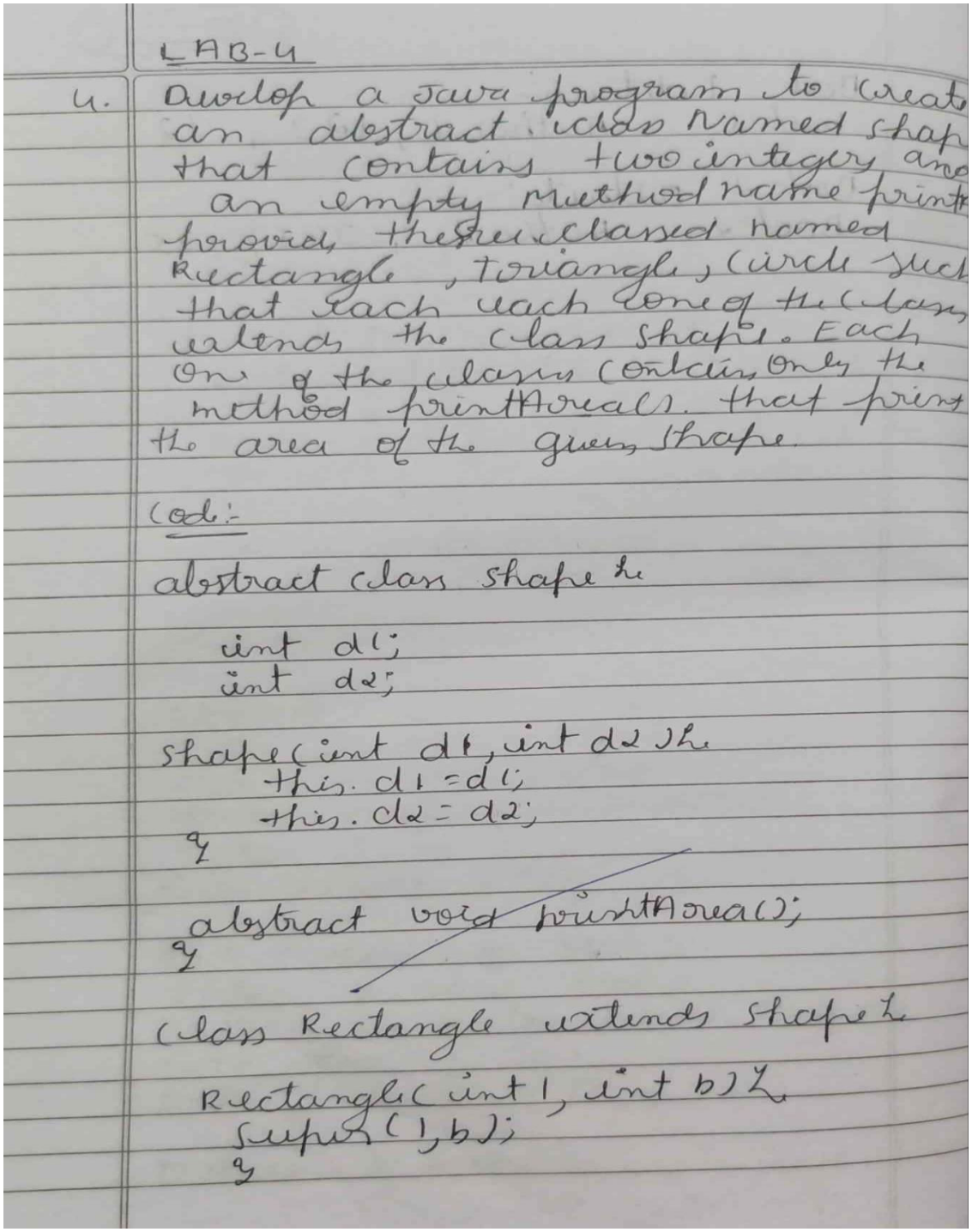
OUTPUT :

```
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 :



LAB-4

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

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("The 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:-

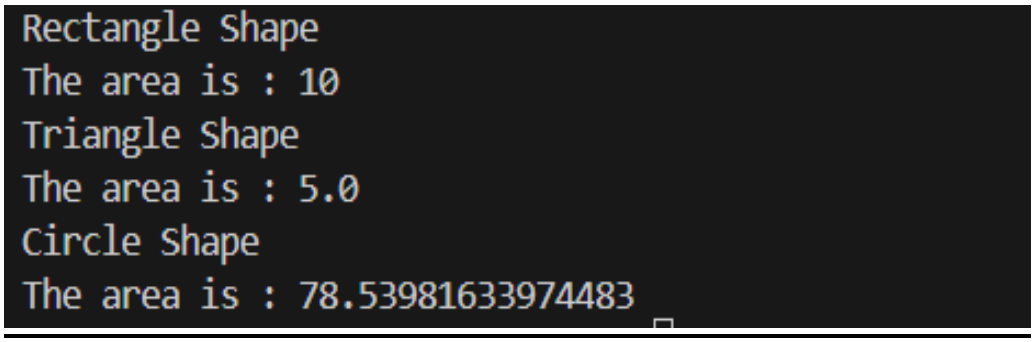
Rectangle Shape
the area is: 10
Triangle Shape
the area is: 5.0
the Circle Shape
the area is: 78.53981

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();
    }
}
```

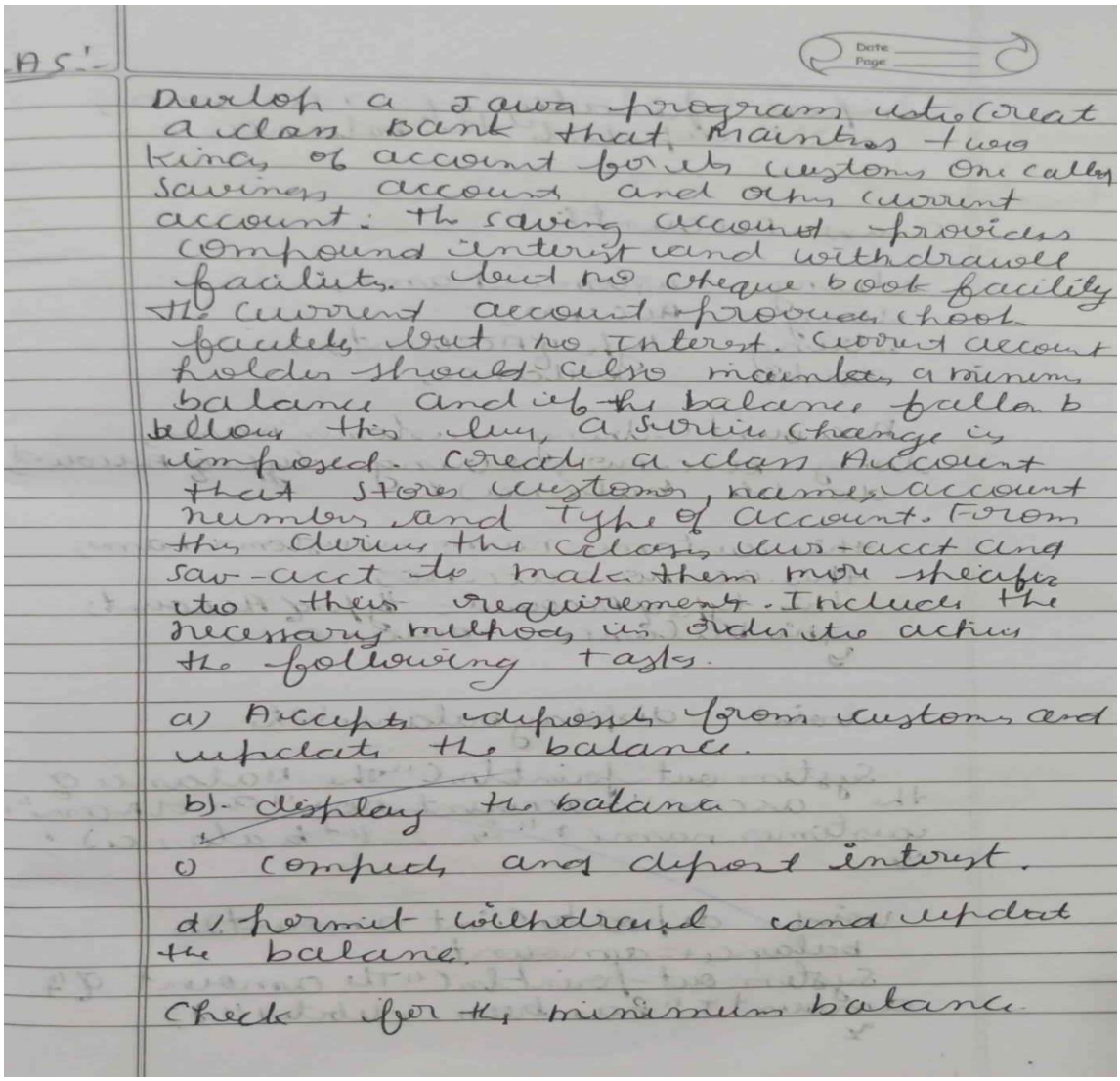
OUTPUT :



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 :



impose penalty if necessary and update the balance.

```
class Account{
    string customer_name;
    long Acc_no;
    string typeofAccount;
    double balance;

    Account(string customer_name,
    long acc_no, string typeofAccou
      double balance){

        this.customer_name = customer_name;
        this.acc_no = acc_no;
        this.typeofAccount = typeofAccount;
        this.balance = balance
    }

    void displaybalance(){

      System.out.println("The Balance of
the accno"+account_no+"and Name
customer name+"is :"+balance);
    }

    void deposit(int amount){
      balance+=amount;
      System.out.println("The amount of
amount+"has been debited");
    }
```

```
void withdraw(int amount){
if(balance >= amount){

System.out.println("the Insufficient
Balance");
}
else{
  this.balance -= amount;
  System.out.println("Amount of "
+ amount + "has been successfully
withdrawn");

Class SavingAccount extends Account{
  double compounded_Interst = 0.04;

SavingAccount(String customName,
long accno){
  super(customName, accno, "SAVING
", 0);
}

void compoundInterst(){
  double interstamount = balance * compon-
                         Interst;
  balance += interstamount;
  System.out.println("Interst
deposited");
}
}
```

```
class currentAccount extends Acco
{
  boolean checkqumbook = true;
  double minimum_balance = 5000;
  double servia_charges = 50;

currstAccount (String customsName,
long accno){

  this.customsname = customs.Nam;
  this.accno = accno;
  b.balance = 5.
  super(customsName, accno, "curren
  5000);
  }

void withdrawal(int amount)
  {
    if (balance > amount)
    {
      this.balance -= amount;
      System.out.println("The amount
  + amount + "withdrawa sucasfully");
    }
    if (balance <= minimum_balance){
      imposbpenality();
    }.
    else {
      System.out.println("Insufficient
    Balance");
  }
}
```

```
void imposfinanlty(){
if{
 balance -= servicecharge;
System.out.println(" finalety Added");
}
}
}
public class Bank{

public static void main(String args
{
SavingAccount SavingAccount = new
SavingAccount("Sushanth Rai", 12345678
);
sa.displabalance();
sa.withdraw();
sa.deposit(1000);
sa.deposit(1000);
sa.compoundInterest();
sa.withdraw(1000);
sa.displayBalance();
System.out.println();

CurrentAccount CA = new CurrentAccou
("Likhith", 987654321);
CA.displaybalance();
CA.withdraw(500);
CA.deposit(1000);
CA.deposit(1000);
CA.withdraw(1008);
CA.displayBalance();
}
```

Output :-

The Balance of the 12345678 9 and Name Sushanth is 0.0

Insufficient balance

Amount of 1000.0 has been debited.
Amount of 1000.0 has been debited

Interest deposited

Amount of 1000.0 successfully drawn
The balance of the 123456789 and Name Sushanth is 1080.0

The balance of the 987654321 and Name Likhith is 5000.00

Amount of 500 withdrawn successfully.
Penalty Added
Amount of 1000.0 has been debited
Amount of 1000.0 has been debited
Amount of 1000 withdrawn successfully
The balance of the 987654321 and Name Likhith is 5450.0

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(); }
   }
```

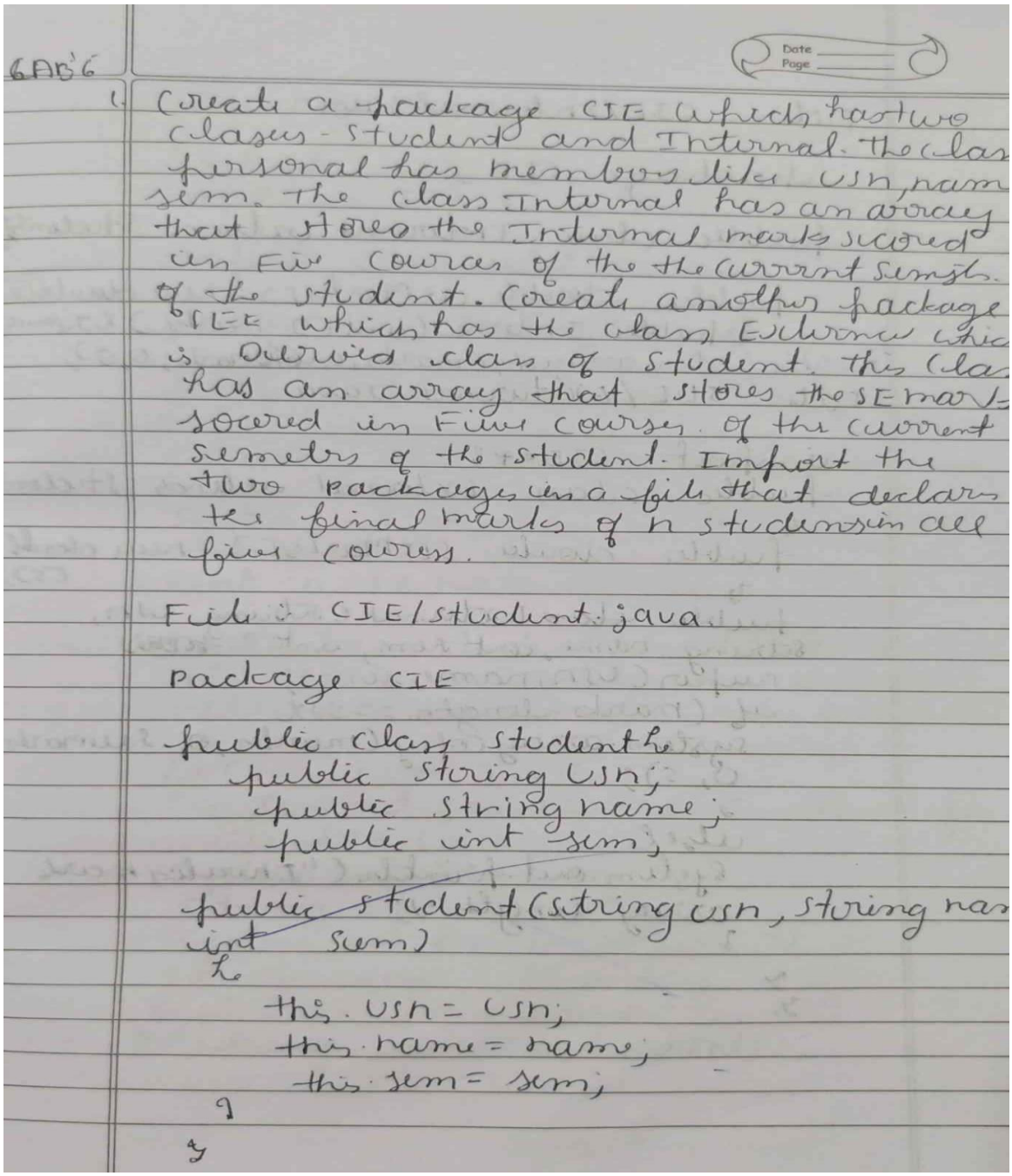
OUTPUT :

```
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 :



LAB 6

1. Create a package CIE which has two classes - Student and Internal. The class Personal has members like usn, name, sem. The class Internal has an array that stores the Internal marks scored in five courses of the the current semsth. 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 semesters of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

File : CIE/student.java

```
package CIE

public class student {
    public String USN;
    public String name;
    public int sem;

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

File : CIE/Internal.java

```
package CIE;

public class Internal extends Student
{
    public double cieMarks[] = new double[5];
    public Internal(int[] marks ) {if(marks.length
==5){ System.arraycopy(marks, 0, cieMarks, 0, 5);
```

File : SEE/External.java

```
import CIE.*;
public class External extends Student
{
    public double seeMarks[] = new double[5];

    public External(String usn,
    String name, int sem, int[] marks)
    {
     super(usn, name, sem);
     if (marks.length == 5){
      System.arraycopy(marks, 0, seeMarks,
      0, 5);
      }
      else{
       System.out.println("Invalid marks
       array length");
       }

    }
}
```

```
package Main;

import CIE.*;
import SEE.*;

import java.util.Scanner;

public class PackageMain {

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

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

Student[] students= new Student[n];
Internal[] internals = new Internal[n];
External[] externals = new External[n];

for(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("Sem:");
int String sem = sc.nextInt();
```

```
System.out.println("Enter internal
 marks for 5 courses");
int marksInt= new int[5];
for(int j=0; j<5; j++)
 {
   Int marksInter[j] = SC.nextInt();
 }

System.out.println("Enter SEE marks
 for 5 courses");
 int[] externalmarks = new Int[5];
for(int j=0; j<n; j++){
   externalmarks[j] = SC.nextInt();
 }

 student[i] = new student(usn, name, sem);
 internal[i] = new Internal(internalmarks
 External[i] = new External(usn,name,sem,
}
System.out.println("Final marks of
 students");
for(int i=0; i<n; i++){
 System.out.println("student "+(i+1)+
"("+students[i].usn + " - " + student[i].
 name + " " + student[i].sem + " ):");

System.out.print("Final marks (5 courses)
for(int j=0; j<5; j++){
 int finalmarks = internal[i].internalmarks
 + (external[i].externalmarks[j]/2.5[j]
  System.out.println("Final marks+" ");
}
System.out.println();
}
```

Output :-

Enter the number of users 2
Enter the details for student 1:
Enter the USN 24BE415
Enter the Name Sushanth Rai
Enter the Sem 4.

Enter Internal marks for 5 courses :-
10 20 30 40 50

Enter External marks for 5 courses.
10 20 30 40 50

Enter the details for student 2
Enter the USN 420
Enter the Name Uday.
Enter the Sem 4.

Enter Internal marks for 5 courses.
10 20 30 40 50.

Enter External marks for 5 courses.
10 20 30 40 50.

Final marks of student.
student 1 : sushanth - 24BE415 - 4.
Final Marks 15 30 45 60 75

student 2 : Uday - 420 - 4
Final marks 15 30 45 60 75.

25/11/24

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();
    }
}
```

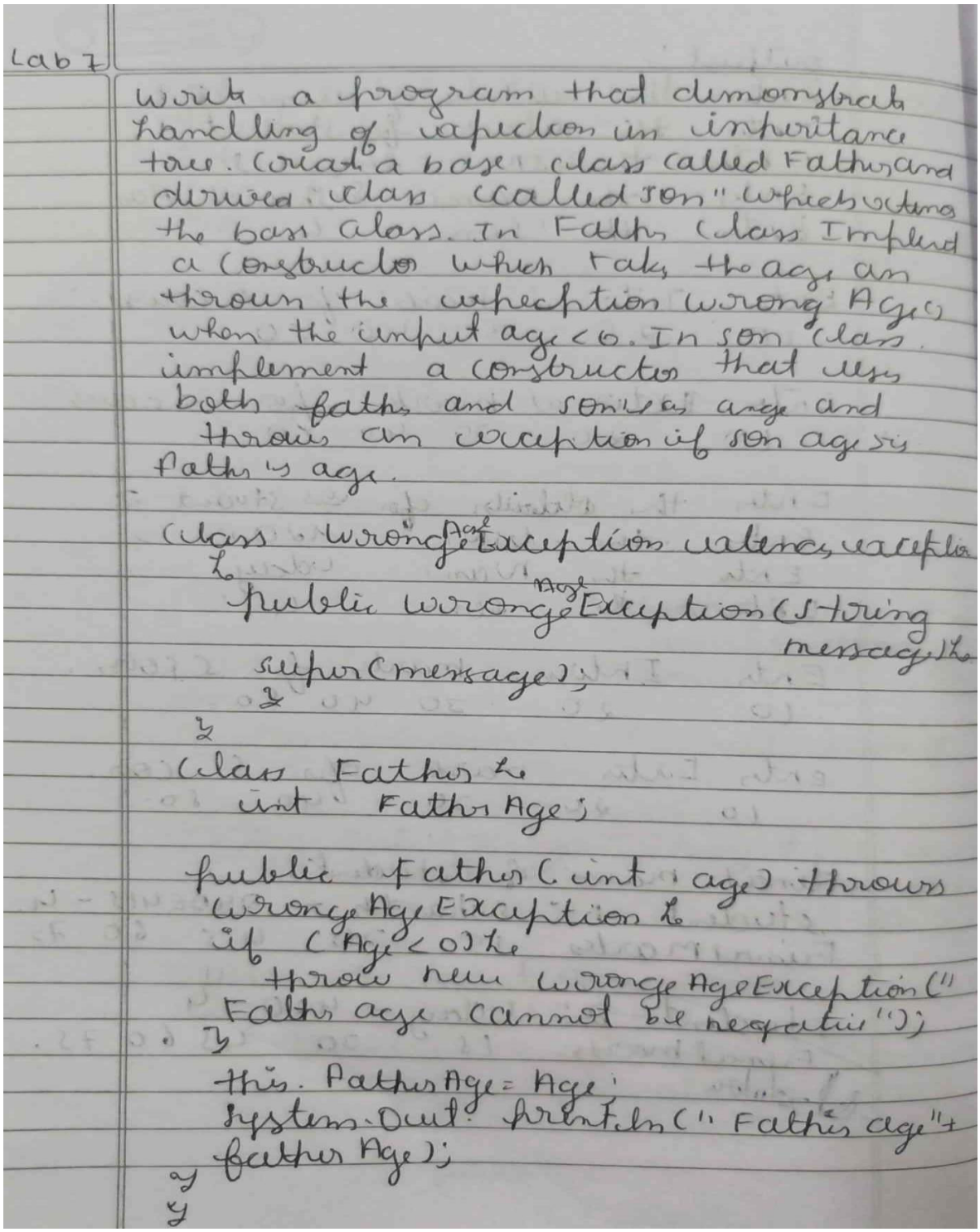
OUTPUT :

```
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 :



class Son extends Father {

int SonAge;

public Son(int FatherAge, int SonAge)
throws WrongAgeException {
super(FatherAge);
if (SonAge >= FatherAge)
{
throw new WrongAgeException("
Sons Age cannot be greater than or
equal to Father Age");
if (SongAge < 0) {
throw new WrongAgeException
("Age cannot be negative!");
}
this.SonAge = SonAge;
System.out.println("Son's age "+SonAge);
}
}

~~class~~ public class ExceptionInheritance
{
public static void main(String Args[])
{
try{
System.out.println("Enter Father Age");
Father F1 = new Father();
int f1Age = SC.nextInt();
F1.FatherAge = F1Age;

```
System.out.println("Enter Son Age");
int s1=sc.nextInt();
Son s1 = new Son(s1Age);
}
Catch(WrongAgeException e){
System.out.println("Exception:"+
e));
}
}
}
```

output:

> Enter the Fathes Age -20.

Exception : Fathes Age cannot be negative

> Enter the Fathes Age 40
Enter the Son Age -10

Exception : Son Age cannot be negat

> Enter the Fathes Age 40
Enter the Son Age 50.

Exception: Son Age cannot be greate than Fathes Age.

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);} } }
```

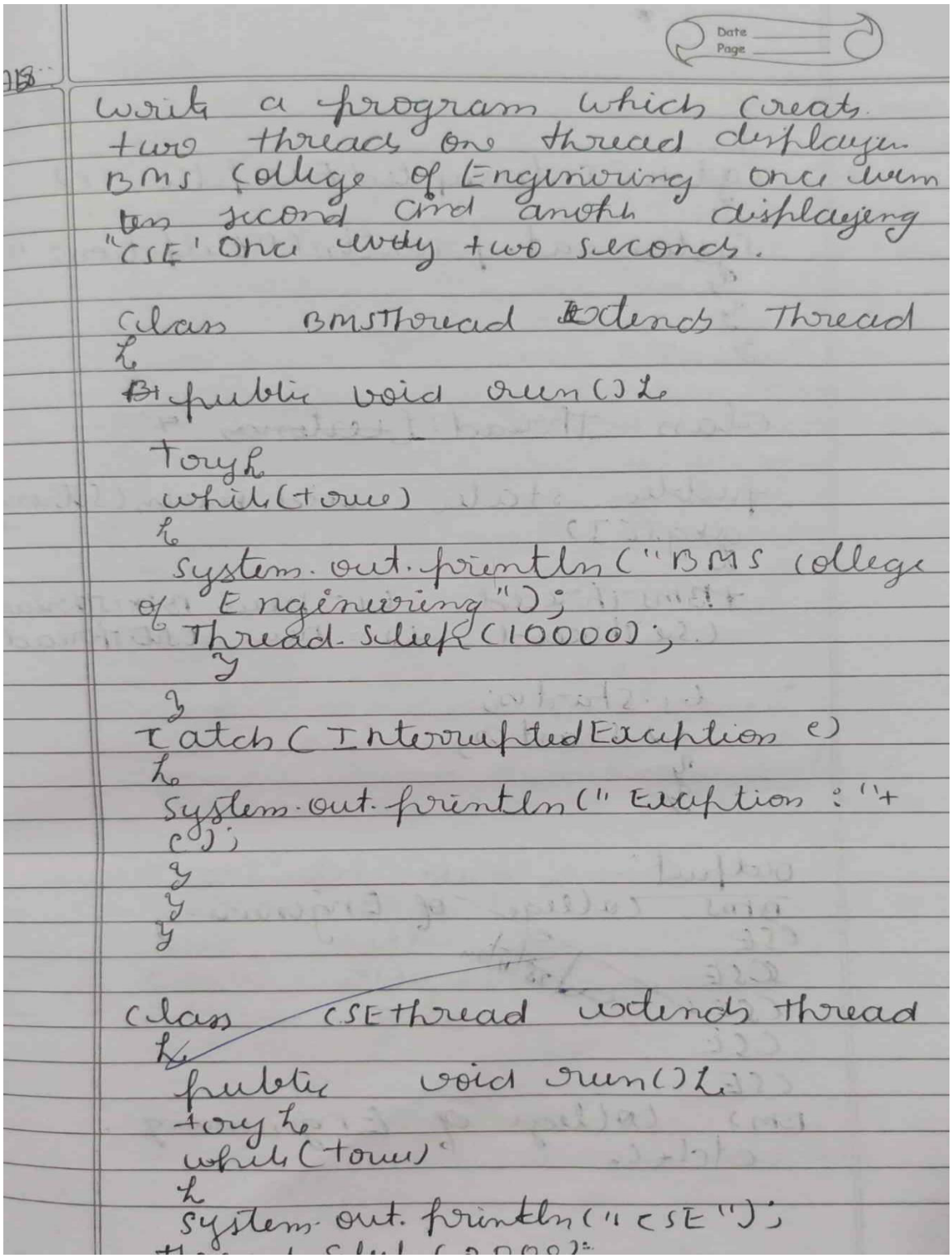
OUTPUT :

```
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 :



```
Write a program which creats
two threads one thread displayes
BMS College of Engineering once every
ten second and anoth displayeng
"CSE" once evdy two seconds.

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("Exception :"+
e);
}
}
}

class CSEthread extends thread
{
public void run()
try
while(true)
{
System.out.println("CSE");
```

```
}
}
catch(InterruptedException e)
{
System.out.println("Exception: "+e
}
}
}

class Thread1 {~~extends~~ {

public static void main(String
args[])
{
BMSThread t1 = new BMSThread
CSEThread t2 = new CSEThread

t1.start();
t2.start();
}
}
```

output:-

BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering.
ctrl+c.

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();;
   }
}
```

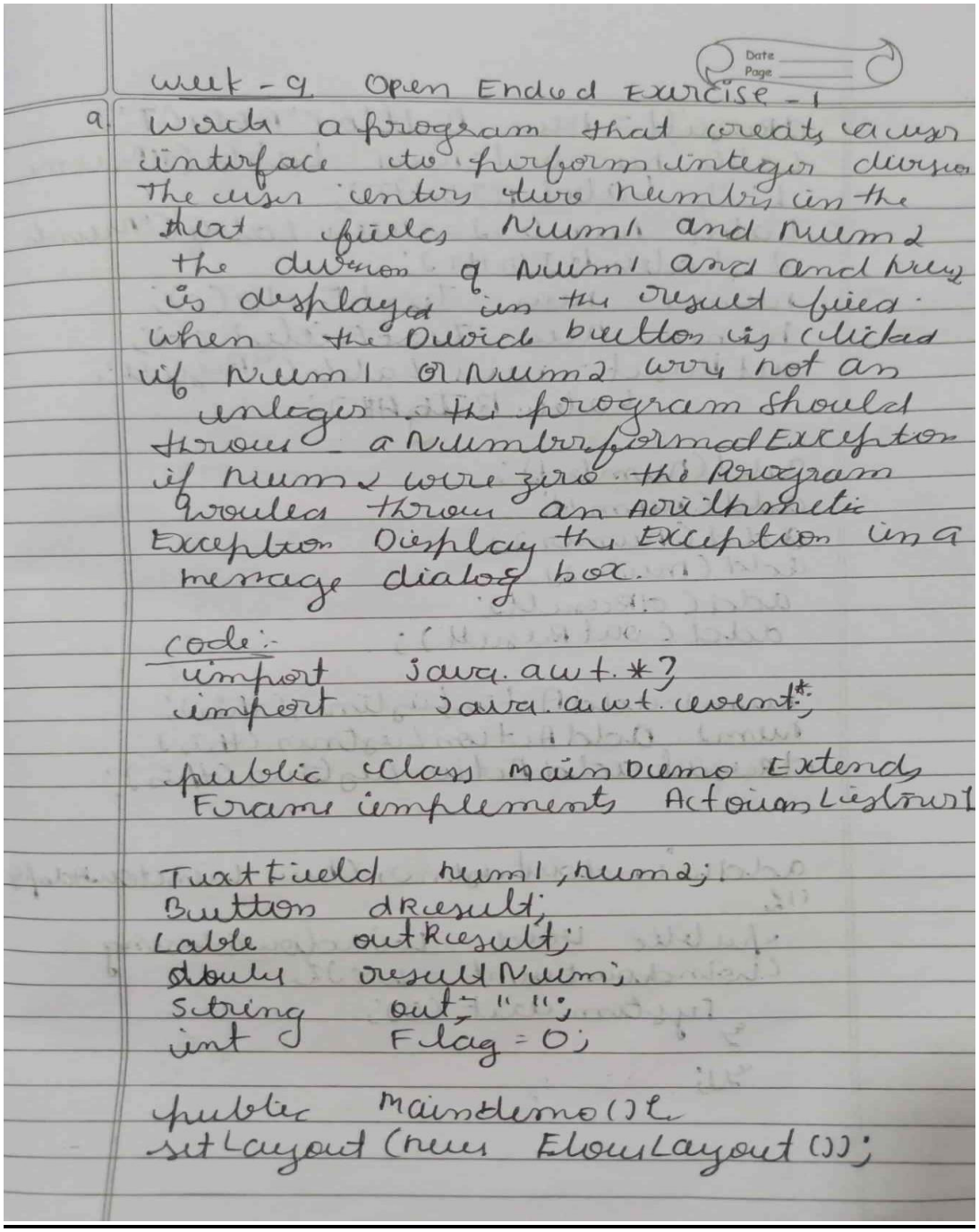
OUTPUT :

```
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 Open Ended Exercise - 1

9. Write a program that creats a user interface to perform integer divisions The user enters two numbers in the text fields Num1 and Num2 the division of Num1 and and Num2 is displayed in the result field when the Divide button is clicked if Num1 or Num2 were not an integer. the program should throws a NumberformadException if Num2 were zero the Program would throw an Arithmetic Exception Display the Exception in a message dialog box.

Code:

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

public class MainDemo Extends
Frame implements ActionListener

TextField num1,num2;
Button dResult;
Lable outResult;
double resultNum;
String out="";
int Flag=0;

public MainDemo(){
setLayout(new FlowLayout());
```

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

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

num1.addActionListner(this);
num2.addActionListner(this);
dResult.addActionListner(this);


addwindowlistner(new windowAdapt
(){
  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");
}
result=(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){
if (flag == 0){
  g.drawString (out, outResult.
  getX() + outResult.getWidth()+
  10, outResult.get(Y) + 50);
}
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("Divn of Integers");
    dm.setVisible(true);
  }
}.
```

output:

Num : 1 [100] Numbr [20] Result:

Result = 100 / 20 = 5.0.

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) {
```

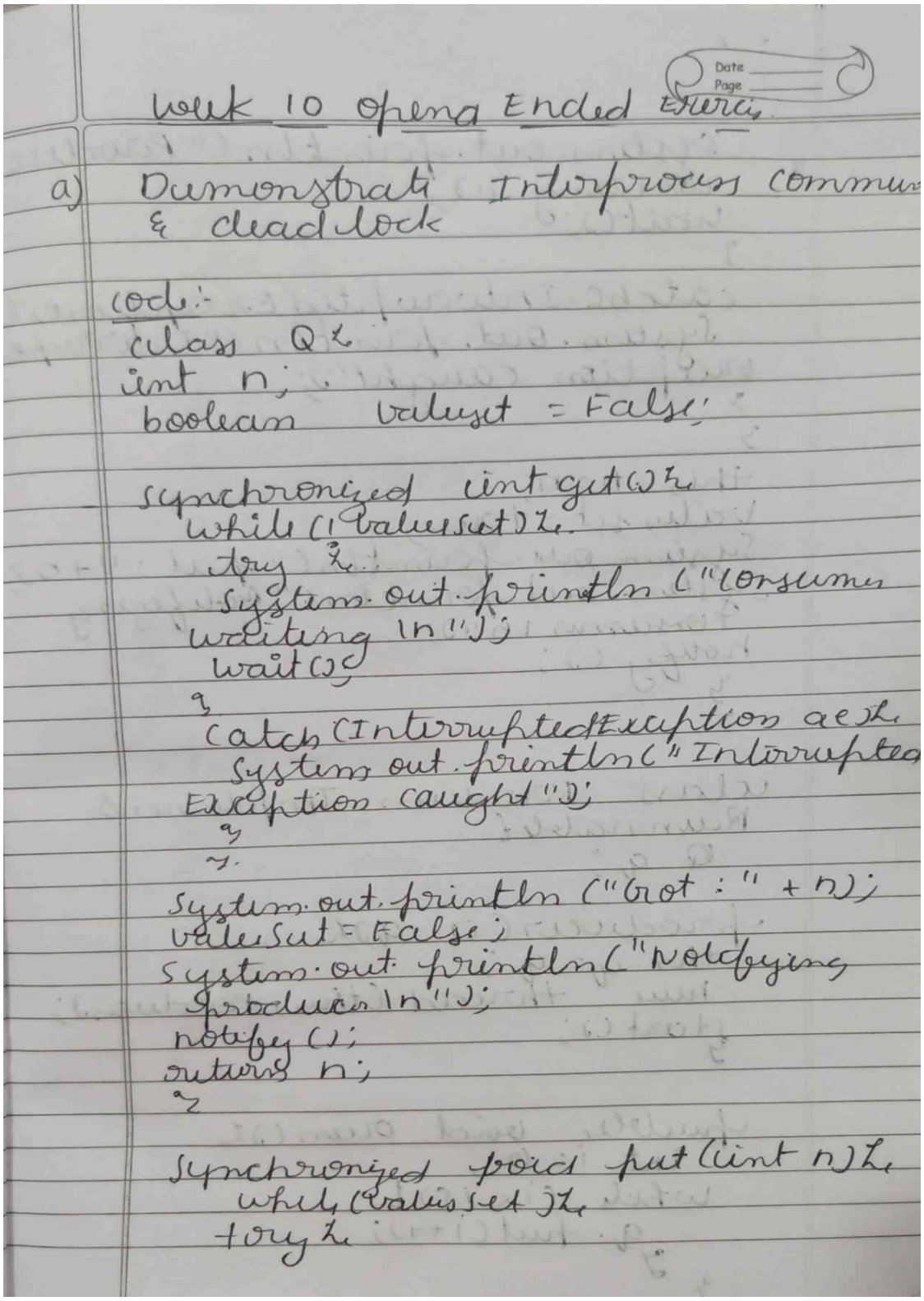
```
        Font customFont = new Font("Serif", Font.BOLD, 20);
    g.setFont(customFont);
        if (flag == 0) {
            g.drawString(out, outResult.getX() + outResult.getWidth() + 10,
outResult.getY() +20);
        } else {
            g.drawString(out, 100, 200);
            flag = 0;
        } }
    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 :

LABORATORY PROGRAM – 10

Demonstrate Inter process Communication and deadlock.

OBSERVATION :



Week 10 Open Ended Ques.

a) Demonstrati Interprocess commun & deadlock

code:-

```
class Q {
int n;
boolean valueset = False;

synchronized int get() {
  while (!valueset) {
    try {
    System.out.println ("Consume
    waiting in ");
    wait();
    }
    catch (InterruptedException ae) {
      System.out.println ("Interrupted
    Exception caught ");
    }
  }
System.out.println ("Got : " + n);
valueSet = False;
System.out.println ("Notifying
  Producer in ");
notify();
return n;
}

synchronized void put (int n) {
  while (valueset) {
  try {
```

```
System.out.println("Produce
Waiting ();
wait();
}
catch( InterruptedException e){
System.out.println("Interrupte
Exception caught");
}
this.n=n;
valueSet=true;
System.out.println("put: "+n);
System.out.println("Notifying
Consumer");
notify();
}
}

class Producer implements
Runnable{
Q q;

Producer(Q q){
this.q=q;
new thread(this, Producer);
start();
}

public void run(){
int i=0;
while(i<15){
q.put(i++);
}
```

```
class consumer implement Runnal
  Q q;
  Consumer(Q q){
  this.q = q;
  new Thread(this, "Consumer").sta
}

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 Contro
  to stop");
}
}
```

output:-

put: 0
Notifying consum.

producer waiting
count = 0

Notifying producer.
put = 1

Notifying consumer.

producer waiting
consume: 0
count: 1

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 :

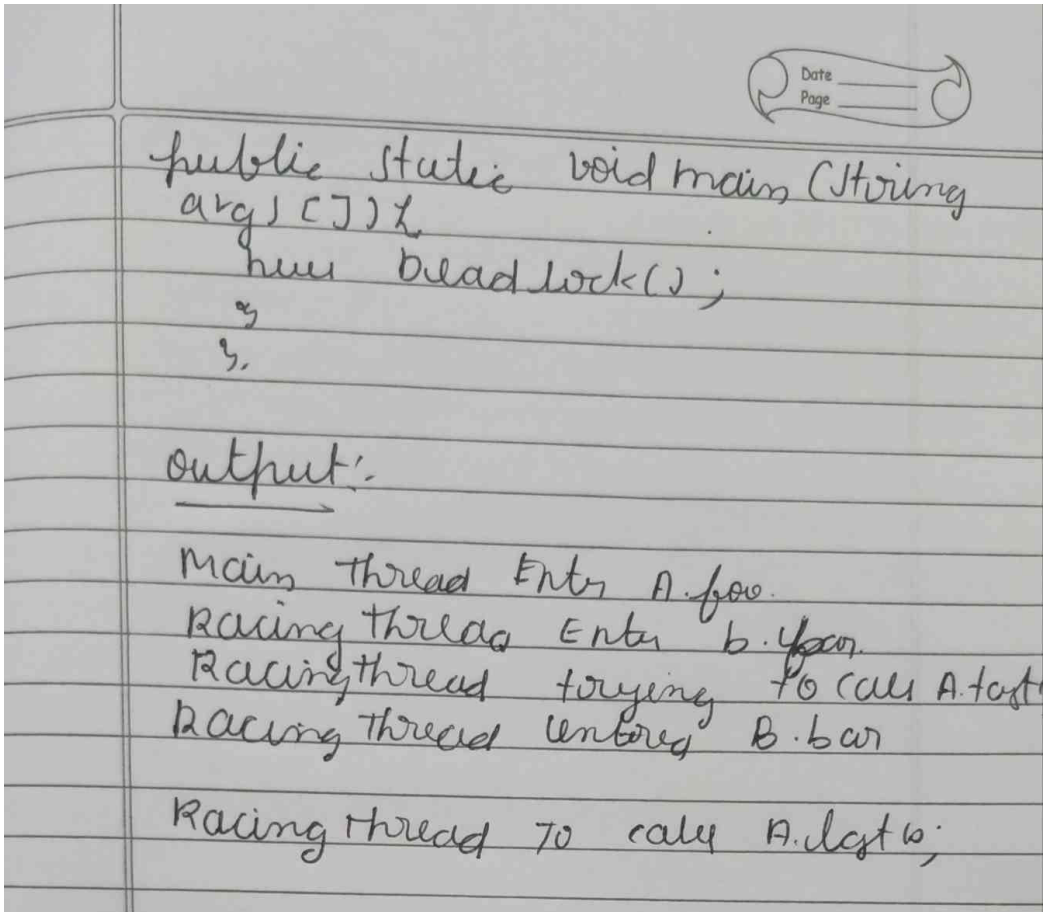
```
Press Control-C to stop.
Put: 0

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
```

ii) Deadlock Implementation :-

Code :-

```
class A {
  Synchronized void foo (B b) {
  String name = thread.currentThread
  getName();

  System.out.println("name " + name
  A.foo");
  try {
    Thread.sleep(1000);
  }
  catch (Exception e) {
    System.out.println("A Interrupt
  }
  System.out.println("name " +
    try 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 interru
}
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(
    "Main thread");
    Thread t = new Thread(this, "Racer
    t.start                    thread");
    a.foo(b);
    System.out.println("Back in
main thread");
  }
  public void run(){
      b.bar(a);
      System.out.println("Back in oth
    } thread");
```



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()
```