VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

ATHARV BORIKAR(1BM23IC015)

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

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

BENGALURU-560019
Sep-2024 to Jan-2025

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

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by **ATHARV BORIKAR (1BM23IC015),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

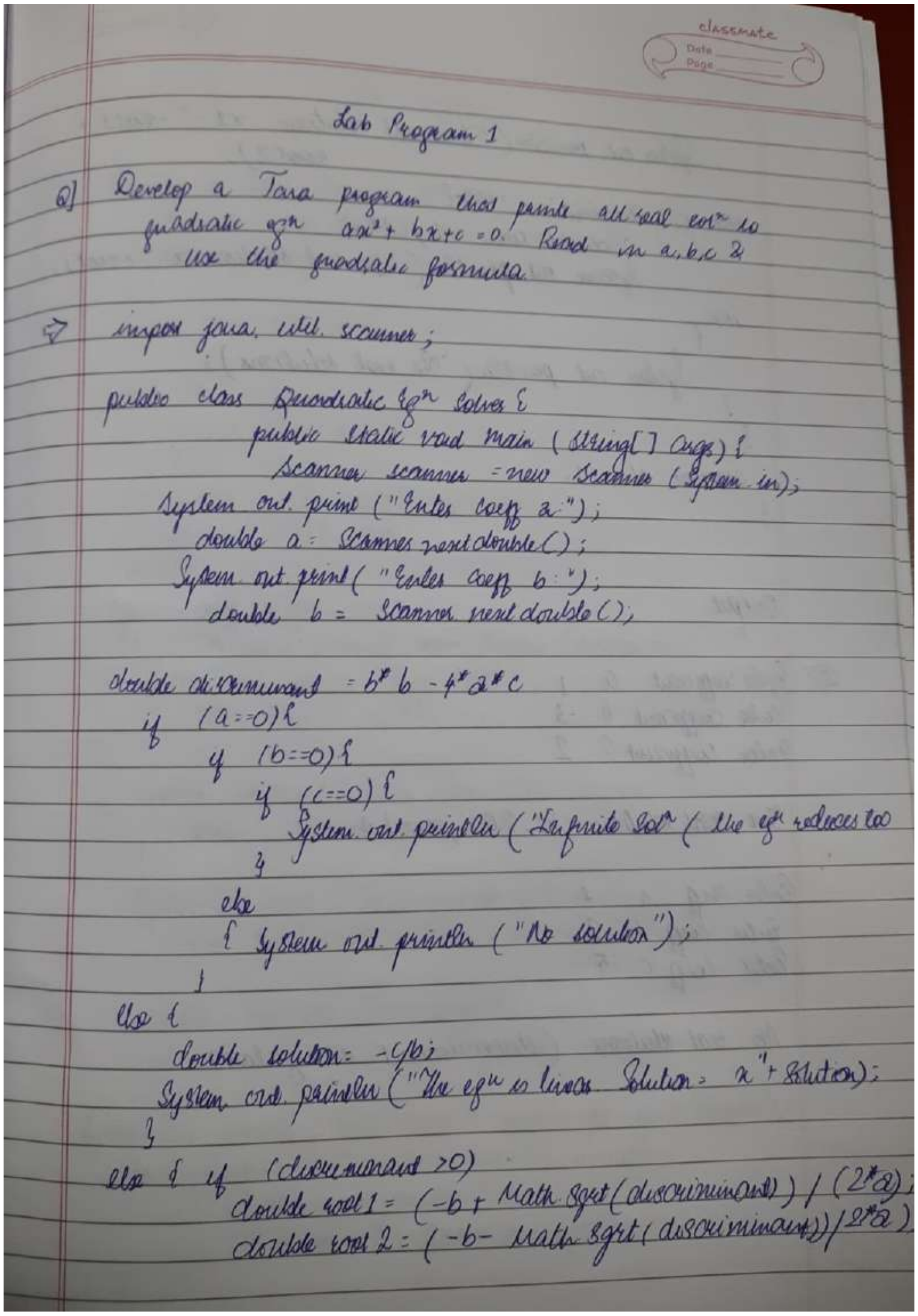
Dr. Seema Patil Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/Ath007-dev/Lab-Programs

Program 1
Implement Quadratic Equation

Algorithm:



Lab Program 1

Q] Develop a Java program that prints all real sol^n to quadratic eq^n $ax^2+bx+c=0$. Read in a,b,c & use the quadratic formula.

```
⇨ import java.util.scanner;

public class QuadraticEqnSolver {
        public static void main (String[] args) {
          Scanner scanner = new Scanner (System.in);
   System.out.print ("Enter coeff a:");
     double a = Scanner.nextDouble();
    System.out.print( "Enter coeff b:");
      double b = Scanner.nextDouble();

double discriminant = b*b - 4*a*c
   if (a==0){
       if (b==0){
          if (c==0){
            System.out.println ("Infinite sol^n (the eq^n reduces too
          }
       else
       { System.out.println ("No solution");
      }
 else {
    double solution = -c/b;
   System.out.println ("The eq^n is linear Solution = x" + Solution);
  }
else { if (discriminant >0)
       double root1 = (-b + Math.sqrt(discriminant)) / (2*a);
       double root2 = (-b - Math.sqrt(discriminant))/(2*a)
System.out.println("Two real solutions : x1 = " +root1 +
                                         root2);
else if( discriminant ==0) {
        double root = -b/(2*a)
        System.out.println("One real solution : x= " +root);
else
  {
    System.out.println("No real solutions);
  }
}
scanner.close();
}
```

Output:

```
Enter coefficient a : 1
Enter coefficient b : -3
Enter coefficient c : 2

Two real solutions x1 = 2.0 , x2 = 1.0

Enter coeff a : 1
Enter coeff b : 2
Enter coeff c : 5

No real solutions (discriminant is negative)
```

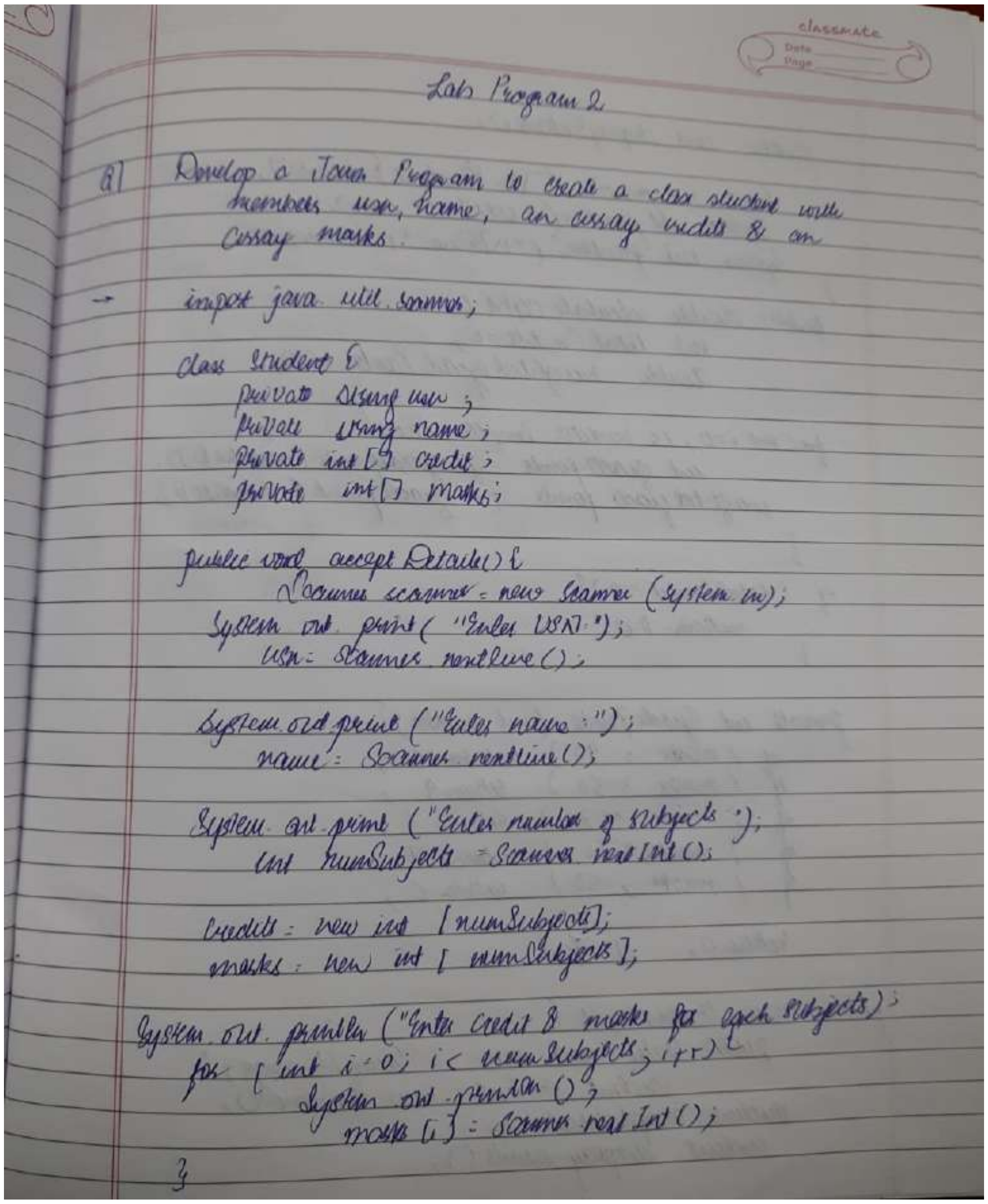
Code:

```
import java.util.Scanner;
class Quadratic
{ int a, b, c;
   double r1, r2, d;
   void getd()
   {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the coefficients of a,b,c");
      a = s.nextInt(); b = s.nextInt(); c = s.nextInt();
   }
   void compute()
   {
      while(a==0)
      {
         System.out.println("Not a quadratic equation");
         System.out.println("Enter a non zero value for a:");
         Scanner s = new Scanner(System.in);
         a = s.nextInt();
      }
      d = b*b-4*a*c;
      if(d==0)
      {
         r1 = (-b)/(2*a);
         System.out.println("Roots are real and equal");
         System.out.println("Root1 = Root2 = " + r1);
      }
      else if(d>0)
      {
         r1 = ((-b)+(Math.sqrt(d)))/(double)(2*a);
         r2 = ((-b)-(Math.sqrt(d)))/(double)(2*a);
         System.out.println("Roots are real and distinct");
         System.out.println("Roo1 = " + r1 + " Root2 = " + r2);
      }
      else if(d<0)
      {
         System.out.println("Roots are imaginary");
         r1 = (-b)/(2*a);
         r2 = Math.sqrt(-d)/(2*a);
         System.out.println("Root1 = " + r1 + " + i"+r2);
         System.out.println("Root1 = " + r1 + " - i"+r2);
      }
   }
}
class QuadraticMain
```

```
{
   public static void main(String args[])
   {
      Quadratic q = new Quadratic();
      q.getd();
      q.compute();
   }
}
```

<u>Program 2</u>
SGPA Calculation

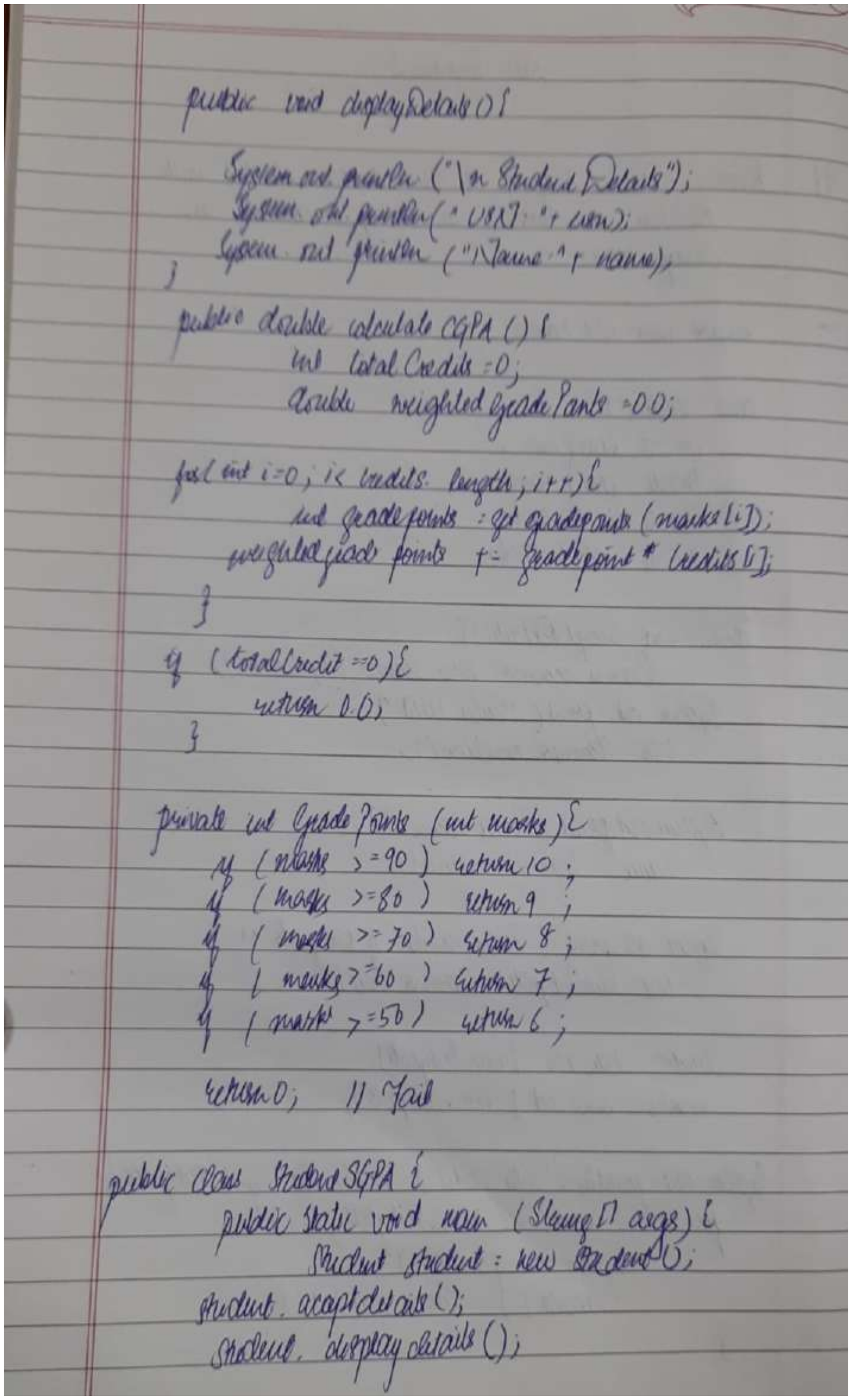
Algorithm:



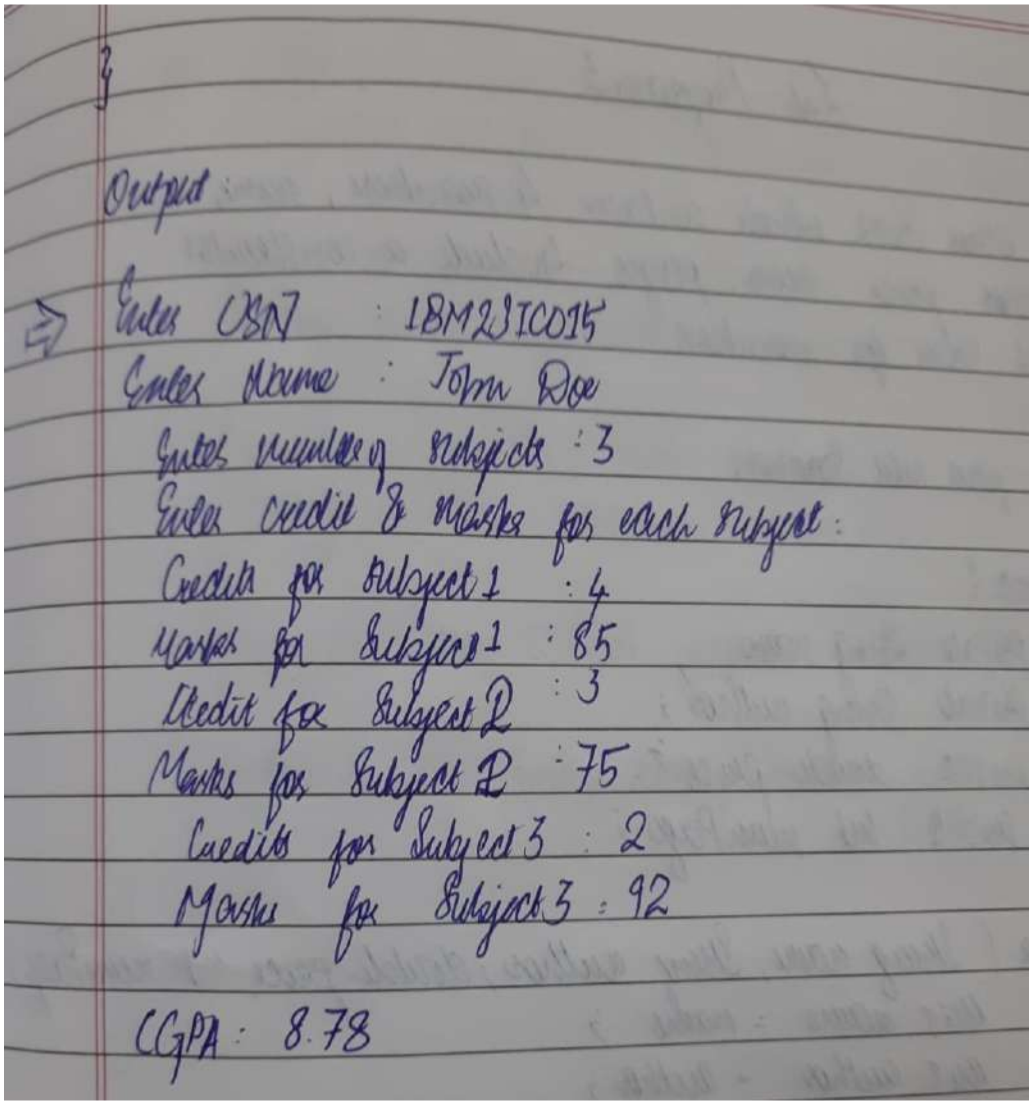
Lab Program 2

Q] Develop a Java Program to create a class student with members usn, name, an array credits & an array marks.

```
→ import java.util.Scanner;

class Student {
      private String usn ;
      private String name ;
      private int[] credit ;
      private int[] marks;

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();

  System.out.print ("Enter number of subjects :");
     int numSubjects = Scanner.nextInt();

  credits = new int [numSubjects];
  marks = new int [ numSubjects ];

System.out.println ("Enter credit & marks for each subjects);
  for ( int i=0; i< numSubjects; i++) {
        System.out.println();
           marks [i] = Scanner.nextInt();

  }
```



```
public void displayDetails(){

    System.out.println("\n Student Details");
    System.out.println(" USN: "+ usn);
    System.out.println("\Name:"+ name);
}

public double calculate CGPA (){
        int total Credits =0;
        double weighted GradePoints =0.0;

for(int i=0; i< credits.length; i++){
        int gradepoints = get gradepoints (marks[i]);
    weightedgrade points += gradepoint * credits[i];
 }
if (totalCredit ==0){
    return 0.0;
 }

private int GradePoints (int marks){
    if (marks >=90) return 10;
    if (marks >=80) return 9;
    if (marks >=70) return 8;
    if (marks >=60) return 7;
    if (marks >=50) return 6;

    return 0;   // Fail

public class StudentSGPA {
    public static void main (String[] args){
            Student student = new Student();
    student.acceptdetails();
    student.displaydetails();
```



```
}
Output :

⇨ Enter USN      : 1BM23IC015
  Enter Name     : John Doe
  Enter number of subjects : 3
  Enter credit & marks for each subject :
  Credits for Subject 1 : 4
  Marks for Subject 1   : 85
  Credit for Subject 2  : 3
  Marks for Subject 2   : 75
  Credits for Subject 3 : 2
  Marks for Subject 3   : 92

  CGPA : 8.78
```

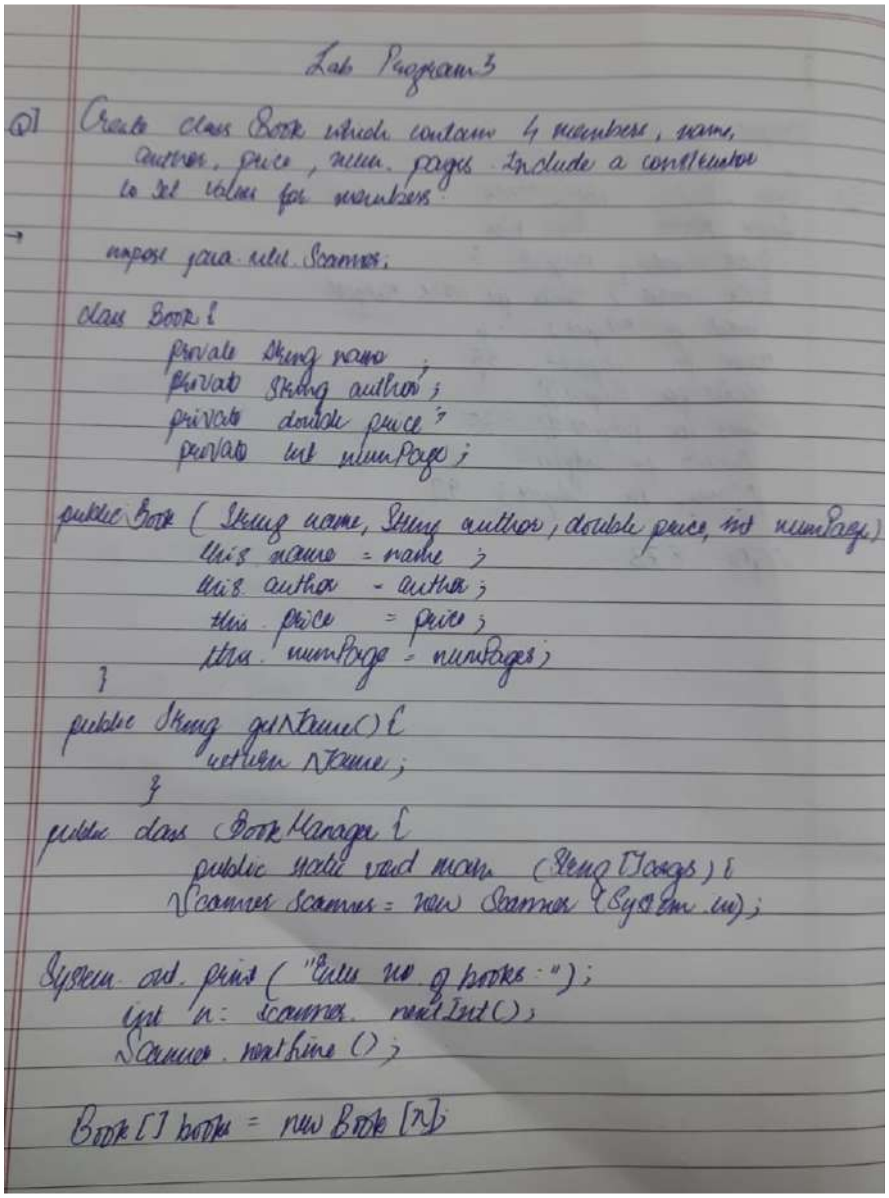
Code:

```
import java.util.Scanner;
class Subject
{
int subjectMarks;

   int credits;
   int grade;
}
class Student
{
   Subject subject[];
   String name;
   String usn;
   double SGPA;
   Scanner s;
   Student()
```

```
        int i;
        subject = new Subject[9];
        for(i=0;i<9;i++)
            subject[i] = new Subject();
        s = new Scanner(System.in);
    }
    void getStudentDetails()
    {
        System.out.print("Enter your Name: ");
        name = s.next();
        System.out.print("Enter your USN: ");
        usn = s.next();
    }
    void getMarks()
    {
        for(int i=0;i<9;i++)
        {
            System.out.print("Enter marks for subject "+(i+1)+" :");

            subject[i].subjectMarks = s.nextInt();
            System.out.print("Enter credits for subject "+(i+1)+" :");

            subject[i].credits = s.nextInt();
            subject[i].grade = (subject[i].subjectMarks/10) + 1;
            if(subject[i].grade==11)
                subject[i].grade = 10;
            if(subject[i].grade<=4)
                subject[i].grade = 0;

        }
    }
    void computeSGPA()
    {
        int effectiveScore = 0;
        int totalCredits = 0;
        for(int i=0;i<9;i++)
        {
            effectiveScore += (subject[i].grade*subject[i].credits);
            totalCredits += subject[i].credits;
        }
        SGPA = (double)effectiveScore/(double)totalCredits;
    }
}
class Student_SGPA
{
    public static void main(String args[])
    {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        System.out.println("Name: "+s1.name);
        System.out.println("USN: "+s1.usn);
        System.out.println("SGPA: "+s1.SGPA);
    }
}
```

Program 3
Display Book Details

Algorithm:

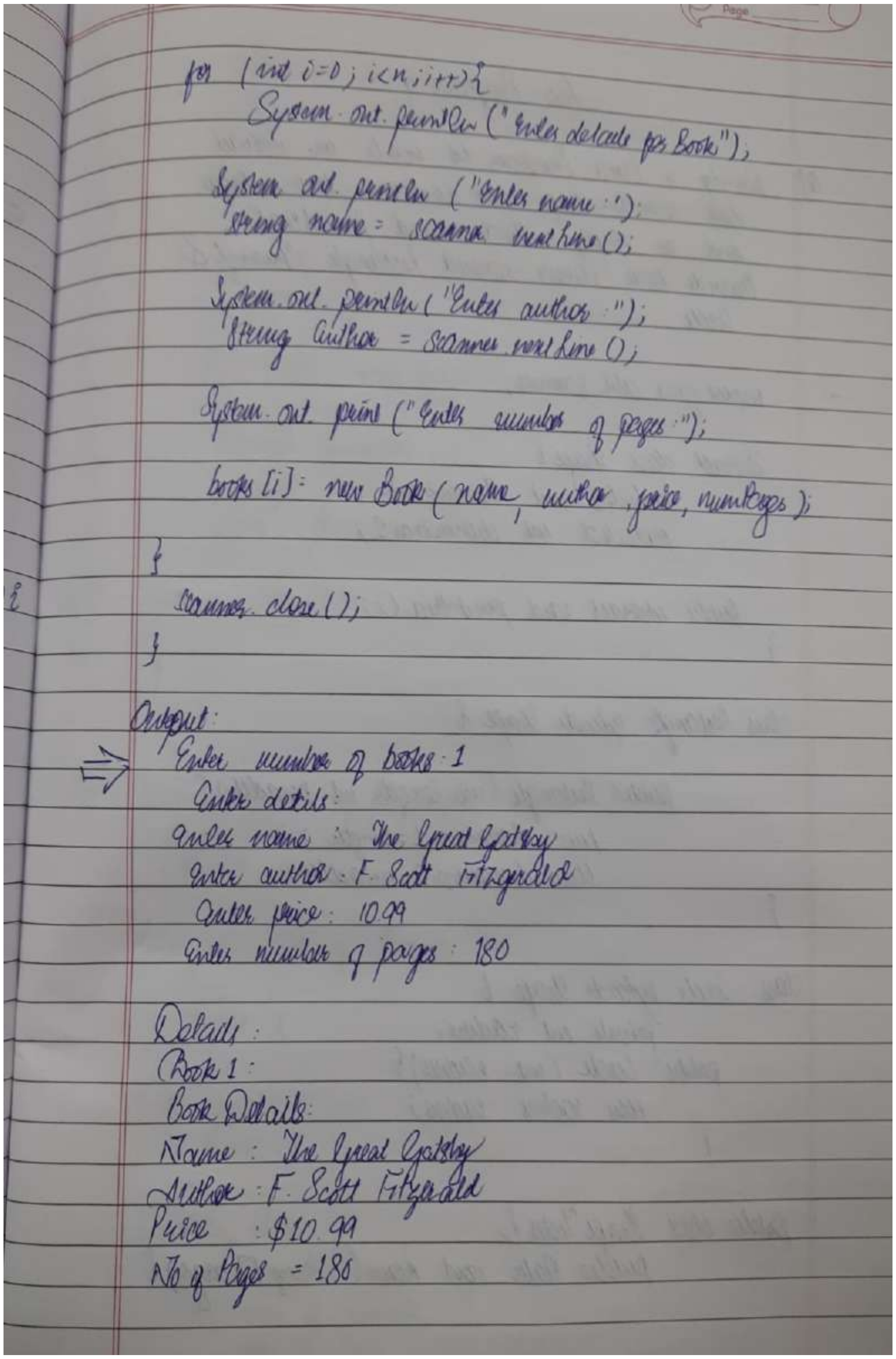


Lab Program 3

Q1 Create class Book which contain 4 members, name, author, price, num. pages Include a constructor to set values for members

→

```
import java.util.Scanner;

class Book {
        private String name ;
        private String author;
        private double price;
        private int numPage;

public Book ( String name, String author, double price, int numPage)
        this.name = name ;
        this.author = author;
        this.price = price;
        this.numPage = numPages;
  }
public String getName(){
        return Name;
    }
public class BookManager {
        public static void main (String [] args) {
        Scanner scanner = new Scanner (System.in);

System.out.print ( "Enter no. of books : ");
    int n = scanner. nextInt();
    Scanner. nextLine ();

  Book [] books = new Book [n];
```



```
for (int i=0; i<n;i++){
    System.out.println("Enter details for Book");

  System.out.println("Enter name :");
  String name = scanner.nextLine();

  System.out.println("Enter author :");
  String Author = Scanner.nextLine();

  System.out.print("Enter number of pages :");

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

}
 scanner.close();
}
```

Output:

```
⇒ Enter number of books: 1
   Enter details:
  Enter name : The Great Gatsby
  Enter author: F. Scott Fitzgerald
  Enter price: 10.99
  Enter number of pages : 180

Details :
Book 1 :
Book Details:
Name : The Great Gatsby
Author : F. Scott Fitzgerald
Price  : $10.99
No of Pages = 180
```

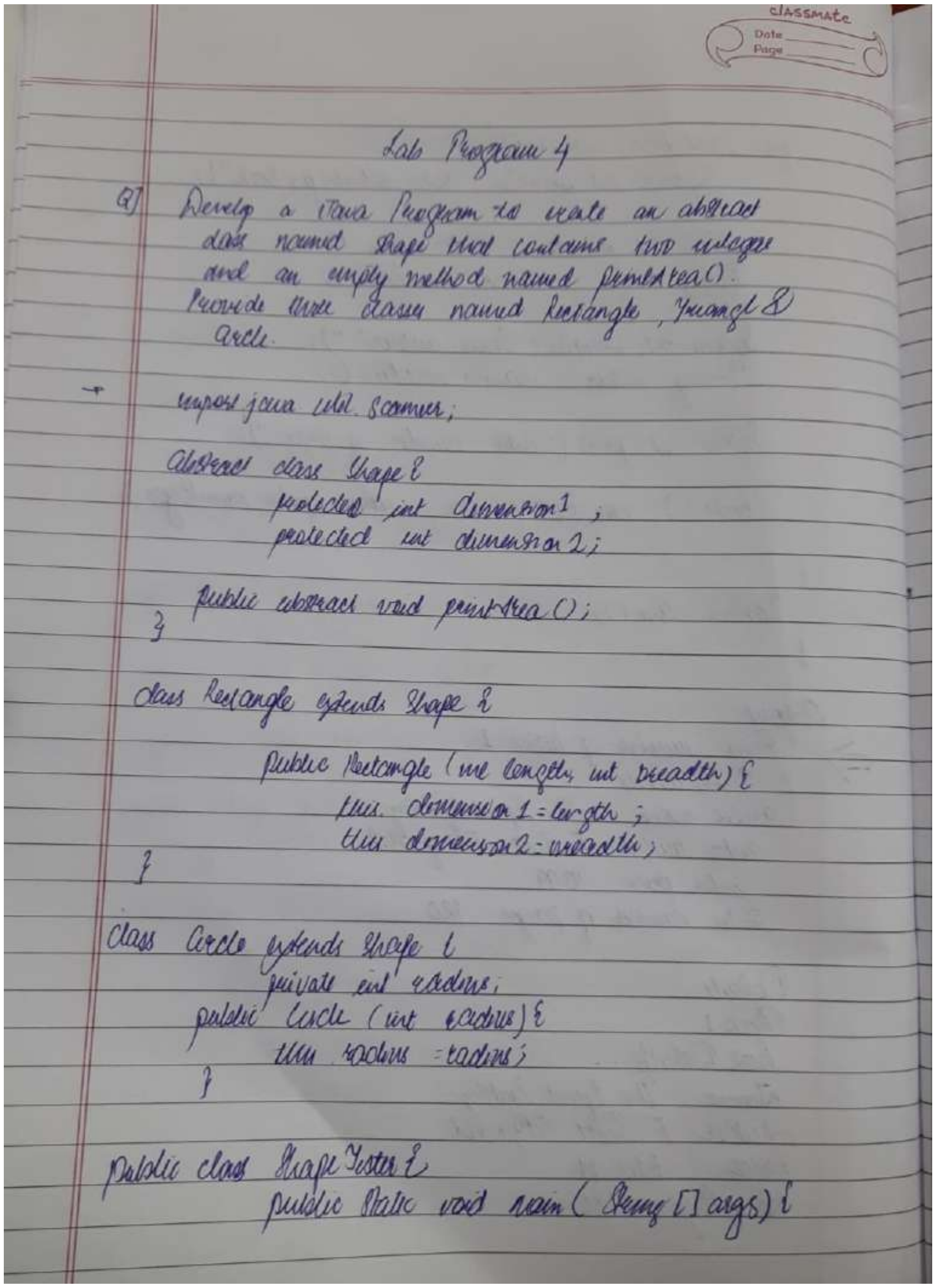
Code:

```
import java.util.Scanner  ;

class Main{
    public static void main(String args[]){
        int n ;
        System.out.print("Enter the number of books:") ;
        Scanner sc  = new Scanner(System.in) ; n = sc.nextInt() ;
        sc.nextLine() ;
        Book books[] = new Book[n];
        for(int i = 0 ; i<n ; i++){
            System.out.print("Enter the book name: ") ;
            String name = sc.nextLine() ;

            System.out.print("Enter the author name: ") ;
            String author = sc.nextLine()  ;
            System.out.print("Enter the price of the book: ") ;
            int price = sc.nextInt() ;
            System.out.print("Enter the number of pages in the book: ") ;
            int numPages = sc.nextInt() ;
            sc.nextLine() ;

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

        System.out.println("");
        for(int i = 0 ; i<n ; i++){
            System.out.println(books[i].toString()) ;
        }
        System.out.println("ATHARV BORIKAR" )  ;
        System.out.print("1BM23IC015") ;
        sc.close();
    }
}


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

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

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

Program 4

Using Abstract Class Shape

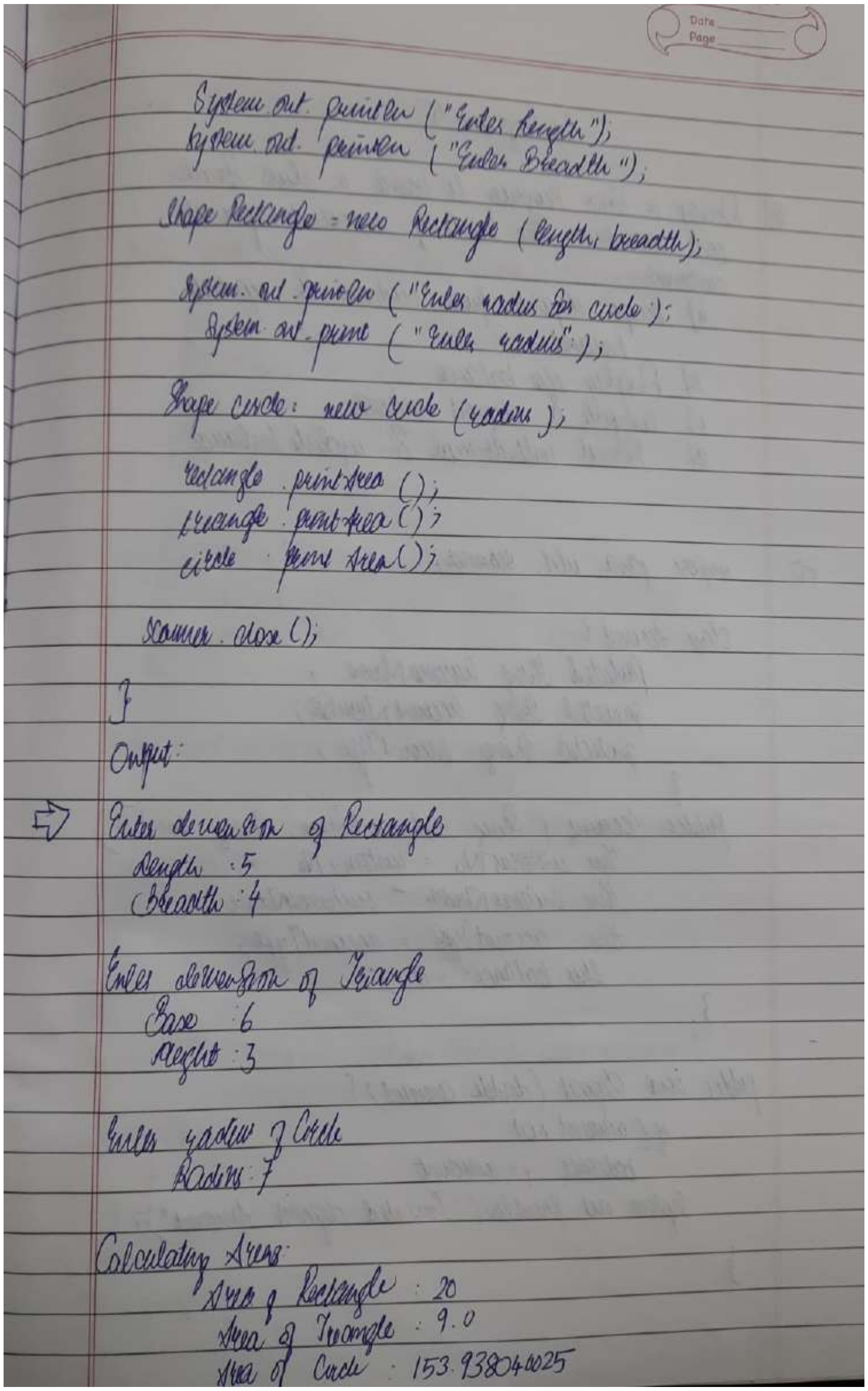
Algorithm:



Lab Program 4

Q] Develop a Java Program to create an abstract class named Shape that contains two integer and an empty method named printArea(). Provide three classes named Rectangle, Triangle & Circle.

```
→ import java.util.Scanner;

abstract class Shape{
        protected int dimension1 ;
        protected int dimension2;

    public abstract void printArea();
}

class Rectangle extends Shape {

        public Rectangle (int length, int breadth) {
            this.dimension1 = length ;
            this.dimension2 = breadth;
}

class Circle extends Shape {
        private int radius;
    public Circle (int radius) {
        this.radius = radius;
    }

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



```
System.out.println ("Enter length");
System.out.println ("Enter Breadth");

Shape Rectangle = new Rectangle (length, breadth);

System.out.println ("Enter radius for circle");
System.out.print ("Enter radius");

Shape circle = new Circle (radius);

rectangle.printArea ();
triangle.printArea ();
circle.printArea ();

scanner.close ();

}
```

Output:

```
Enter dimension of Rectangle
  Length : 5
  Breadth : 4

Enter dimension of Triangle
  Base : 6
  Height : 3

Enter radius of Circle
  Radius : 7

Calculating Areas:
  Area of Rectangle : 20
  Area of Triangle : 9.0
  Area of Circle : 153.93804025
```

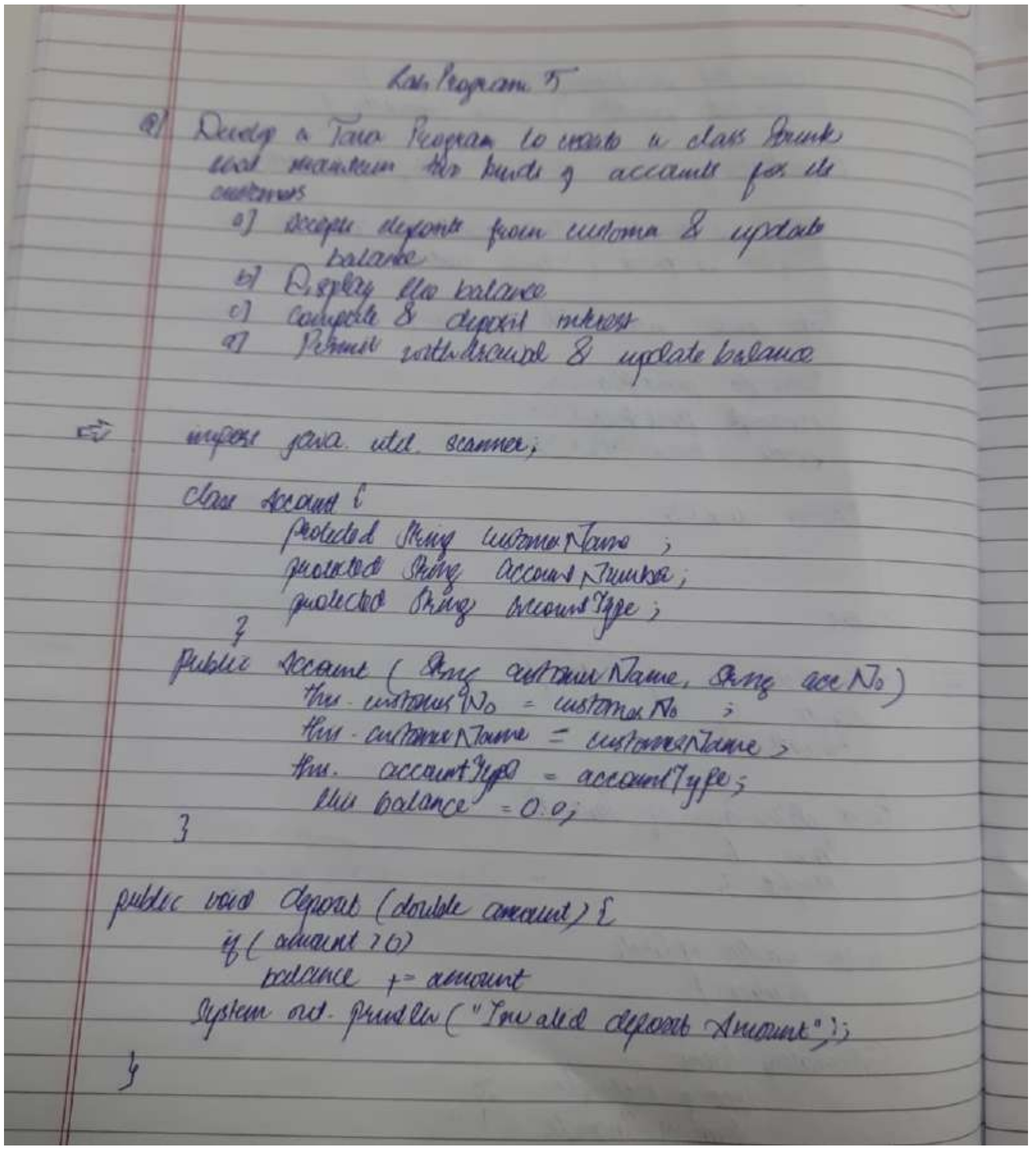
Code:

```
import java.util.Scanner ;

class Main{
    public static void main(String[] args){
        Rectangle ob2  = new Rectangle() ;
        Triangle ob1 = new Triangle() ;
        Circle ob3 = new Circle() ;
        ob2.printArea() ;
        ob1.printArea() ;
        ob3.printArea() ;
        System.out.println("ATHARV BORIKAR " )  ;
        System.out.print("1BM23IC015") ;
    }
}

abstract class Shape{
    Scanner sc = new Scanner(System.in) ;
    int dimension1 , dimension2  ;
    abstract void printArea();
}

class Rectangle extends Shape{

    Rectangle(){
        System.out.println("Enter the dimensions of the rectangle(Length and Breadth): " ) ;
        dimension1 = sc.nextInt() ;
        dimension2 = sc.nextInt() ;
    }

    void printArea(){
        System.out.print("The area of the rectangle is = ") ;
        System.out.println(dimension1*dimension2) ;
    }
}

class Triangle extends Shape{
    Triangle(){
        System.out.println("Enter the dimensions of the triangle(base and height): " ) ;
        dimension1 = sc.nextInt() ;
        dimension2 = sc.nextInt() ;
    }

    void printArea(){
        System.out.print("The area of the Triangle is = ") ;
        System.out.println(0.5*dimension1*dimension2) ;
    }
}
class Circle extends Shape{
    Circle(){
        System.out.println("Enter the dimension of the circle(radius): ")  ;
        dimension1 = sc.nextInt() ;

    }

    void printArea(){
        System.out.print("The area of the Circle is = ") ;
        System.out.println(3.1415926535897*dimension1*dimension1) ;
    }
}
```

<u>Program 5</u>

Bank Account Storage

Algorithm:



Lab Program 5

Q) Develop a Java Program to create a class Bank that maintain the bank of accounts for the customers

a] Accept deposits from customer & update balance

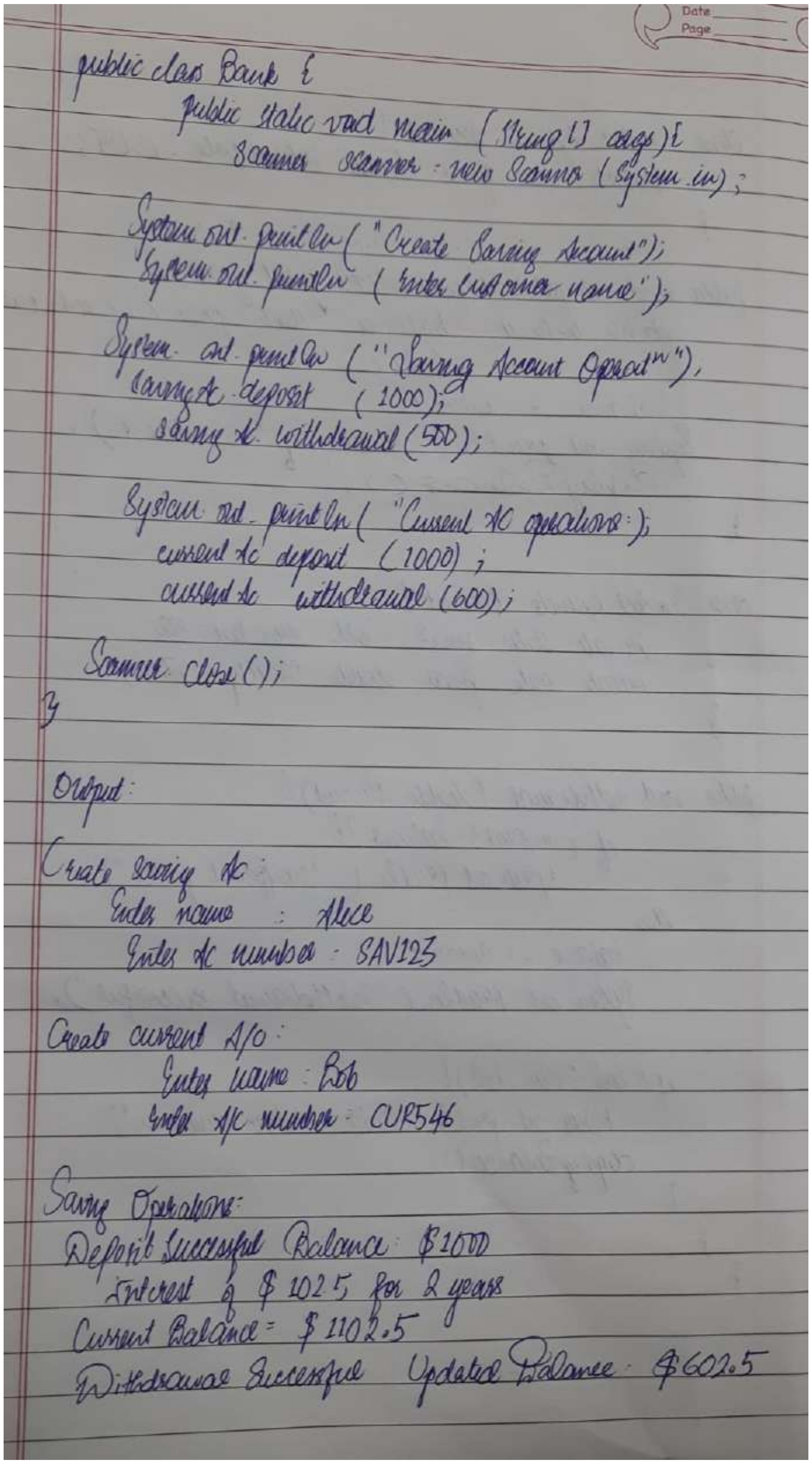
b] Display the balance

c] Compute & deposit interest

d] Permit withdrawal & update balance

```
import java.util.scanner;

class Account {
        protected String customerName ;
        protected String accountNumber;
        protected String accountType ;
    }
public Account ( String customerName, String accNo)
        this.customerNo = customerNo ;
        this.customerName = customerName ;
        this. accountType = accountType;
         this balance = 0.0;
    }

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

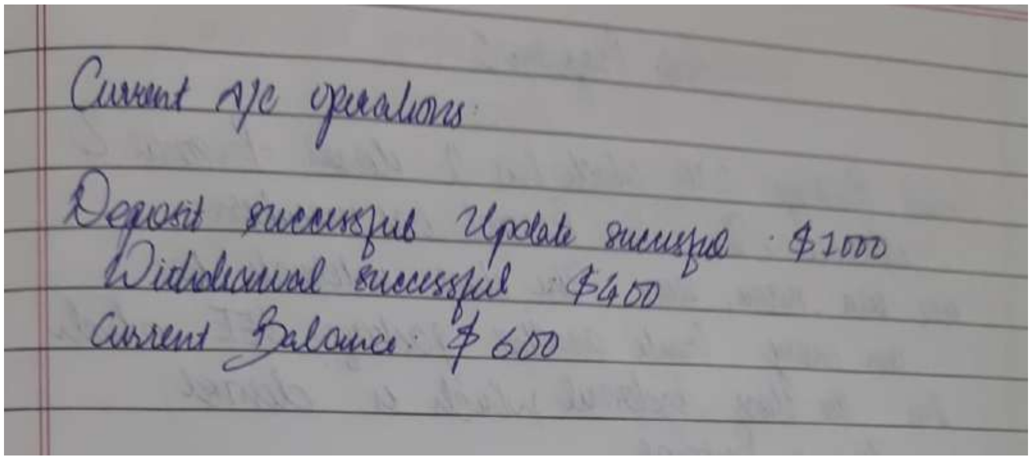
```
class SavAcct extends Accounts {
    private static final double interest rate = 0.05;
 }

public void compute & Deposit Interest ( int years) {
    double interest = balance * Math. pow (1+ int_rate)-
                              balance;
    balance += interest;
    System. out. println( "Interest of " + interest);
    displayBalances ();
}

class CurAcct extends Account {
        private static final double min_bal = 500 ;
        private static final double penalty = 50.0;
  }

public void withdrawal (double Amount) {
        if (amount > balance ) {
            System.out.println ("Insufficient funds");
    else
        balance -= amount;
        System.out. println ("Withdrawal successful");

    if (bal < min_bal) {
        System.out. println ("Balance below minimum");
        displayBalance ();
    }
  }
}
```



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

    System.out.println ("Create Saving Account");
    System.out.println ( "Enter customer name");

    System.out.println ("Saving Account Opened");
     savingAc.deposit  (1000);
      saving Ac.withdrawal (500);

     System.out.println( "Current AC operations:);
       current Ac deposit  (1000) ;
       current Ac  withdrawal (600);

   Scanner.close();
}
```

Output:

```
Create saving Ac:
   Enter name   : Alice
   Enter Ac number : SAV123

Create current A/c:
       Enter name : Bob
       Enter A/c number : CUR546

Saving Operations:
 Deposit Successful Balance : $1000
   Interest of $ 102.5 for 2 years
 Current Balance = $ 1102.5
 Withdrawal Successful  Updated Balance : $602.5
```



Current A/c operations

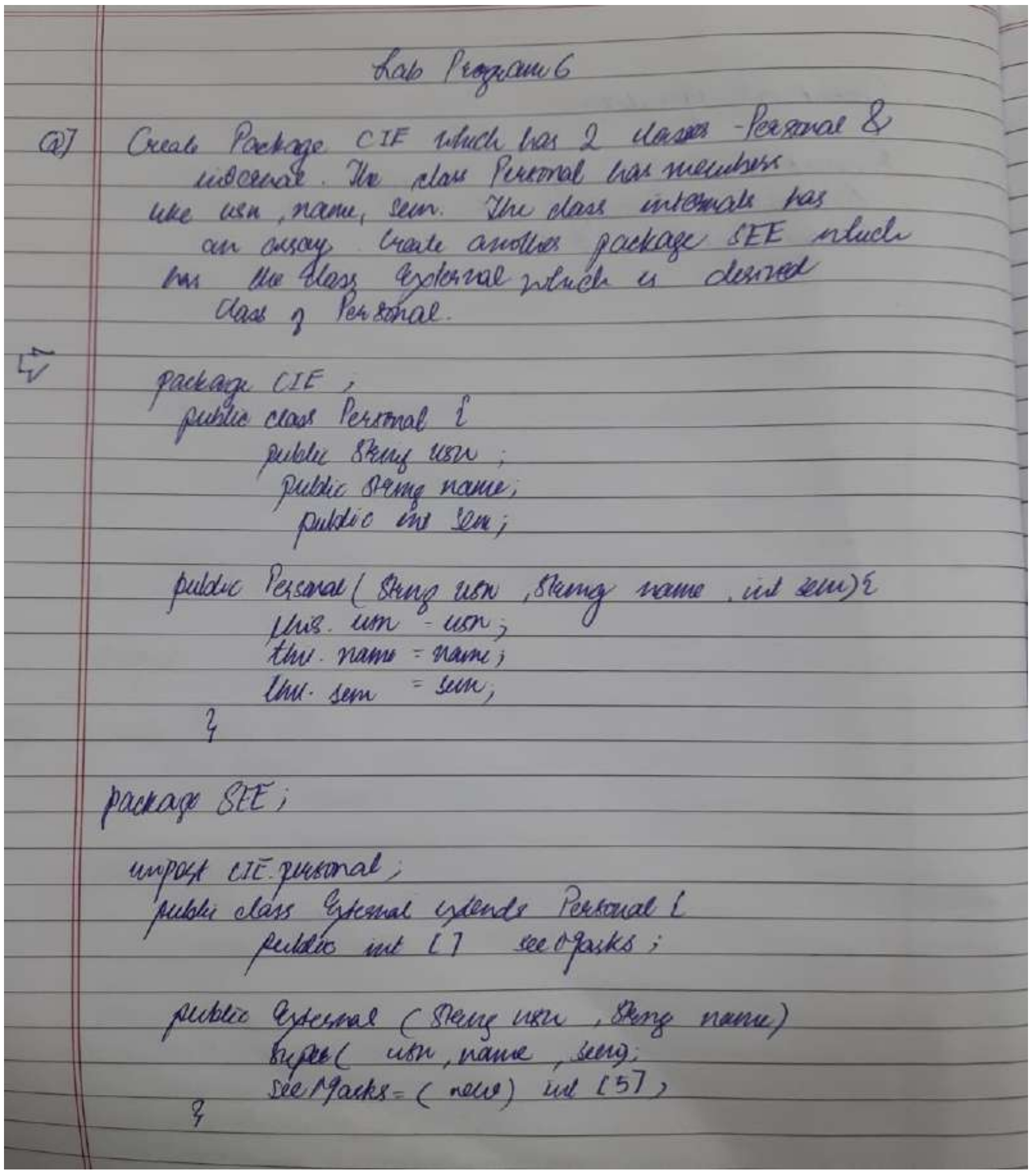
Deposit successful Update successful $1000
Withdrawal successful $400
Current Balance $600

Code:

```
import java.util.Scanner;

public class Bank {
    static Scanner sc = new Scanner(System.in);
    Account ob1;

    void createAccount() {
        String customer;
        int account;
        String type;
        int initBal;

        System.out.print("Enter the customer name: ");
        customer = sc.nextLine();
        System.out.print("Enter account Number: ");
        account = sc.nextInt();
        sc.nextLine();  // Consume the newline
        System.out.print("Enter Account type (Savings or Current): ");
        type = sc.nextLine();
        System.out.print("Enter the initial Balance: ");
        initBal = sc.nextInt();

        if (type.equals("Savings")) {
            ob1 = new Savings(customer, account, initBal);
        } else {
            ob1 = new Current(customer, account, initBal);
        }
    }

    public static void main(String[] args) {
        Bank bank = new Bank();
        bank.createAccount();
        while (true) {
            System.out.println("------------------MENU----------------");
            System.out.println("1. Deposit     2. Withdraw");
            System.out.println("3. Compute interest");
            System.out.println("4. Display account details");
            System.out.println("5. exit " ) ;
            int choice = sc.nextInt();

            switch (choice) {
                case 1:
                    bank.ob1.deposit();
                    break;
                case 2:
                    bank.ob1.withdraw();
                    break;
                case 3:
                    if (bank.ob1 instanceof Savings) {
                        ((Savings) bank.ob1).computeInterest();
                    } else {
                        System.out.println("Interest computation is only available for Savings accounts.");
                    }
                    break;
                case 4:
                    bank.ob1.display();
                    break;
                case 5:
                    break ;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
            if(choice == 5) break ;
        }
    }
}

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

    Account(String customer, int accountNum, int bal) {
        customerName = customer;
        accountNumber = accountNum;
        balance = bal;
    }

    void deposit() {
        System.out.print("Enter the amount to deposit: ");
        int amt = Bank.sc.nextInt();
        balance += amt;
        System.out.println("Deposited: " + amt + ", New Balance: " + balance);
    }

    void withdraw() {
        System.out.print("Enter the amount to withdraw: ");
        int amt = Bank.sc.nextInt();
        if (balance - amt < 0) {
            System.out.println("Insufficient Balance to withdraw the given amount.");
        } else {
            balance -= amt;
            System.out.println("Amount of " + amt + " withdrawn successfully. Current Balance is " +
balance);
        }
    }

    void display() {
        System.out.println("The Balance in the account is " + balance);
    }
}

class Savings extends Account {
    double interestPercent;

    Savings(String customer, int accountNum, int bal) {
        super(customer, accountNum, bal);
        System.out.print("Enter the interest percentage on the account: ");
        interestPercent = Bank.sc.nextDouble();
    }

    void computeInterest() {
        balance += balance * (interestPercent / 100);
        System.out.println("Amount after applying interest is: " + balance);
    }
}

class Current extends Account {
    int minBalance = 1000;

    Current(String customer, int accountNum, int bal) {
        super(customer, accountNum, bal);
    }

    void withdraw() {
        System.out.print("Enter the amount to withdraw: ");
        int amt = Bank.sc.nextInt();
        if (balance - amt < minBalance) {
            System.out.println("Insufficient Balance to maintain the minimum required.");
        } else {
            balance -= amt;
            System.out.println("Amount of " + amt + " withdrawn successfully. Current Balance is " +
balance);
        }
    }
}
```

Program 6
Creating Packages CIE and SEE

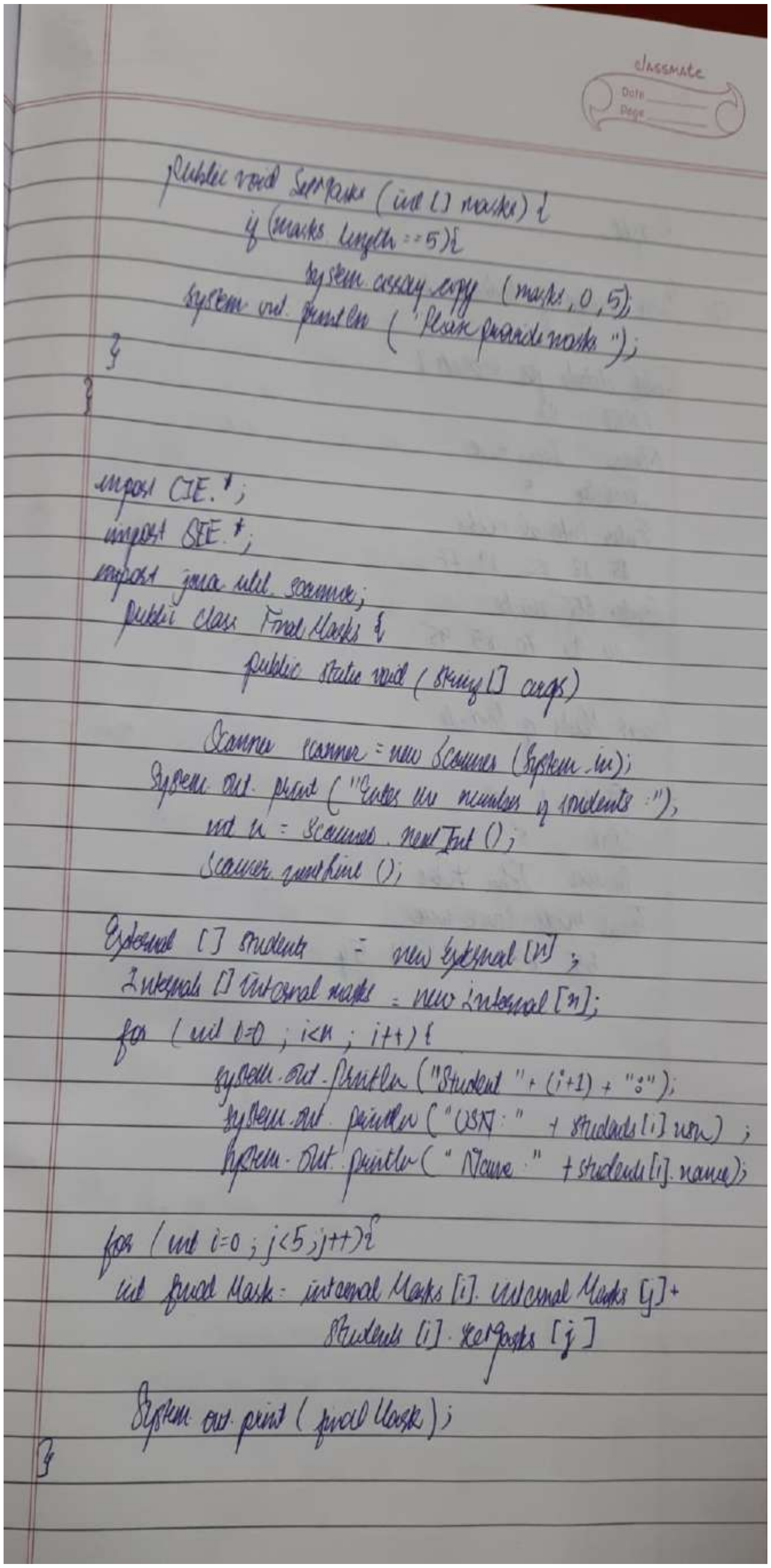
Algorithm:



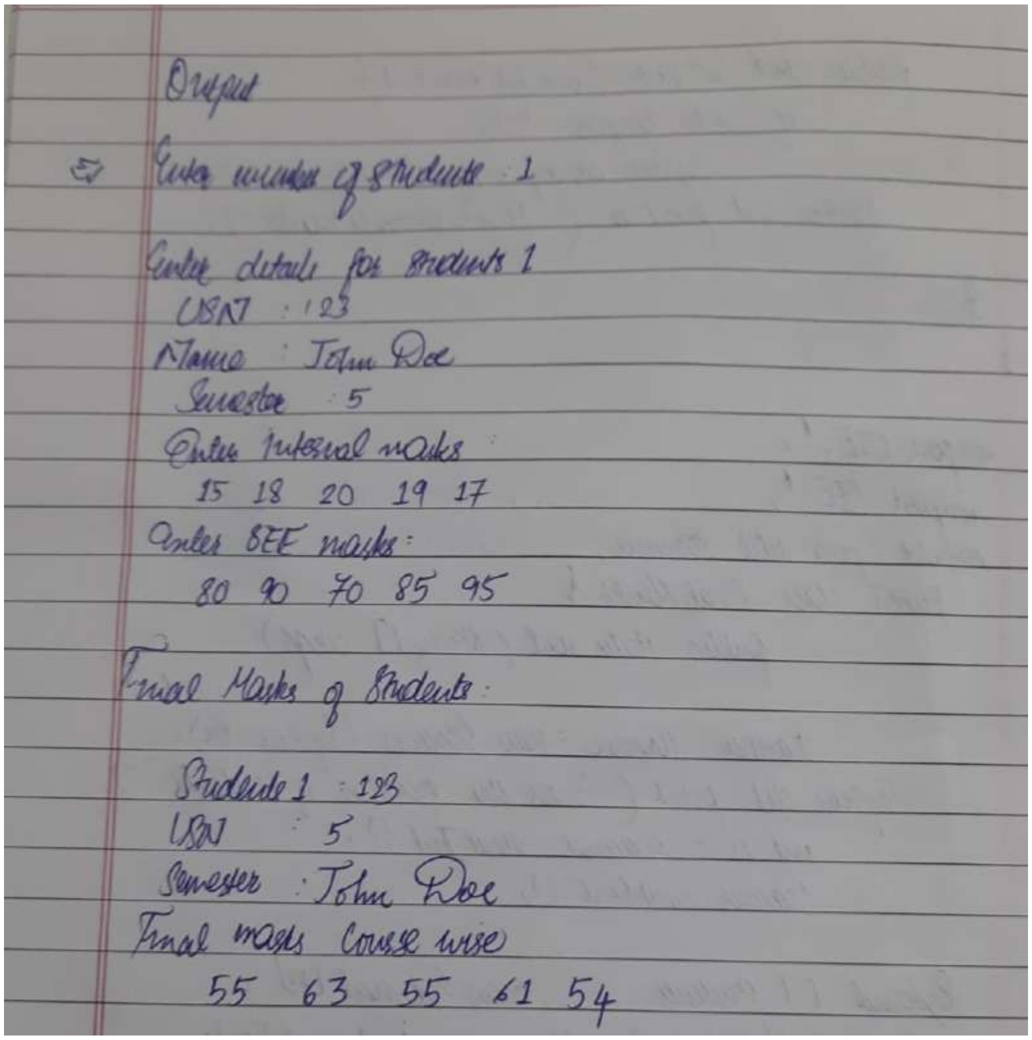
Lab Program 6

Q] Create Package CIE which has 2 classes - Personal & internal. The class Personal has members like usn, name, sem. The class internals has an array. Create another package SEE which has the class External which is derived class of Personal.

```
package CIE ;
 public class Personal {
        public String usn ;
         public String name;
          public int sem;

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

package SEE;

 import CIE.personal;
  public class External extends Personal {
          public int [ ]  seeMarks ;

   public External ( String usn , String name)
          super( usn , name , sem);
          seeMarks = ( new) int [5] ;
   }
```



```
public void setMarks (int [] marks) {
        if (marks.length ==5){
                System.arraycopy (marks,0,5);
    System.out.println ("Please provide marks");
  }
}


import CIE.*;
import SEE.*;
import java.util.Scanner;
 public class FinalMarks {
              public static void (String[] args)

         Scanner scanner = new Scanner (System.in);
   System.out.print ("Enter the number of students :");
         int n = Scanner.nextInt ();
         Scanner.nextLine ();

External [] students    =  new External [n] ;
 Internal [] internal marks = new Internal [n];
 for (int i=0 ; i<n ; i++) {
            System.out.println ("Student "+ (i+1) + ":");
            System.out.println ("USN : "  + students[i].usn) ;
            System.out.println (" Name : " + students[i].name);

for (int j=0 ; j<5;j++){
 int finalMark = internalMarks [i]. internalMarks [j]+
                        Students [i]. seeMarks [j]

  System.out.print (finalMark);

}
```



```
Output

⇨ Enter number of Students : 1

Enter details for students 1
 USN : 123
Name : John Doe
 Semester : 5
 Enter internal marks :
  15 18 20 19 17
Enter SEE marks :
  80 90 70 85 95

Final Marks of Students :

 Student 1 : 123
 USN : 5
 Semester : John Doe
Final marks Course wise
  55 63 55 61 54
```

Code:

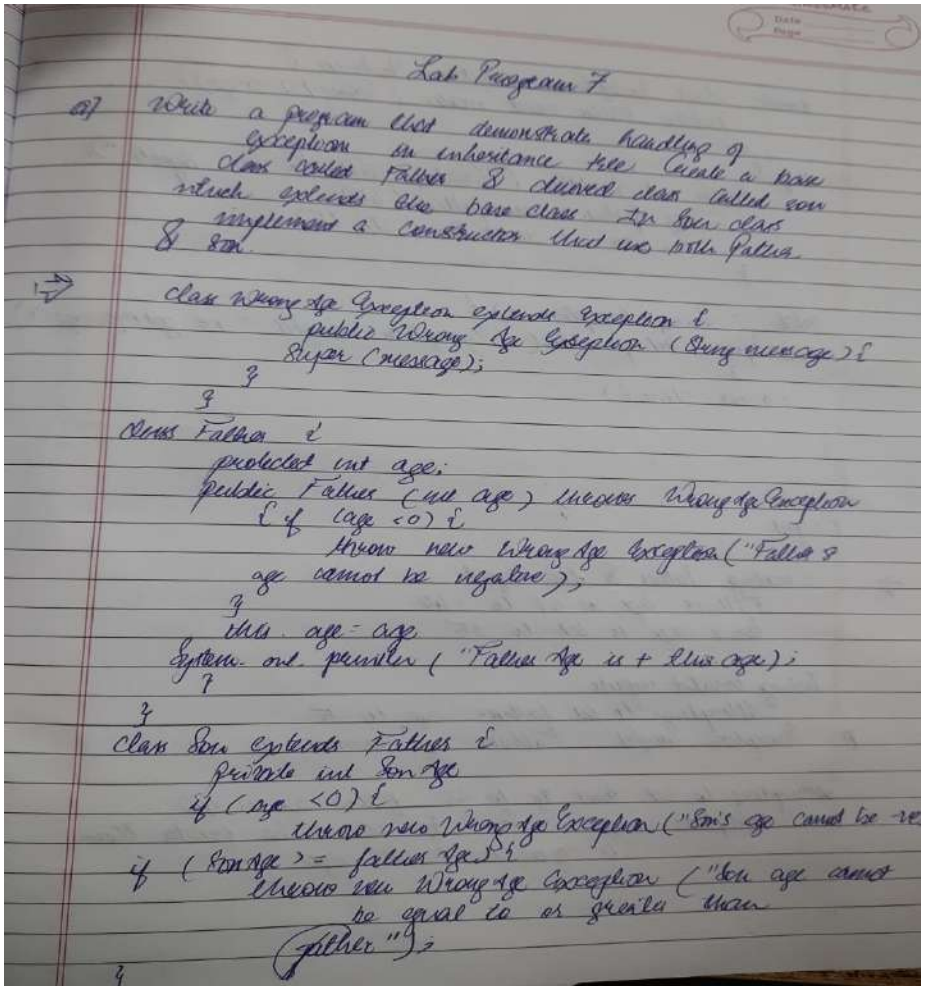
```
package CIE;

import java.util.Scanner;

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

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

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


package CIE;

import java.util.Scanner;

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

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



package SEE;

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

public class Externals extends Internals {
    protected int[] marks = new int[5];      // SEE marks
    protected int[] finalMarks = new int[5]; // Final marks

    // Constructor to initialize the marks arrays
    public Externals() {
        marks = new int[5];
        finalMarks = new int[5];
    }
    // Method to input SEE marks
    public void inputSEEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter SEE marks for 5 subjects:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter SEE marks for subject " + (i + 1) + ": ");
            marks[i] = s.nextInt();
        }
    }

    // Method to calculate final marks (internal + external)
    public void calculateFinalMarks() {
        for (int i = 0; i < 5; i++) {
            finalMarks[i] = marks[i] + this.marks[i]; // Final marks = internal + external
        }
    }

    // Method to display final marks
    public void displayFinalMarks() {
        displayStudentDetails();  // Display student details (inherited from Student)
        System.out.println("Final Marks:");
        for (int i = 0; i < 5; i++) {
            System.out.println("Subject " + (i + 1) + ": " + finalMarks[i]);
        }
    }
}
```

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

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

        // Input number of students
        System.out.print("Enter number of students: ");
        int n = s.nextInt();
        s.nextLine(); // Consume newline

        Externals[] students = new Externals[n];

        // Input details for each student
        for (int i = 0; i < n; i++) {
            students[i] = new Externals();
            System.out.println("\nEnter details for student " + (i + 1) + ":");
            students[i].inputStudentDetails();
            students[i].inputCIEmarks();
            students[i].inputSEEmarks();
            students[i].calculateFinalMarks();
        }

        // Display final marks for each student
        System.out.println("\nDisplaying final marks for all students:");
        for (int i = 0; i < n; i++) {
            students[i].displayFinalMarks();
        }

        s.close();
    }
}
```

Program 7

Handling Exceptions in Inheritance Tree

Algorithm:

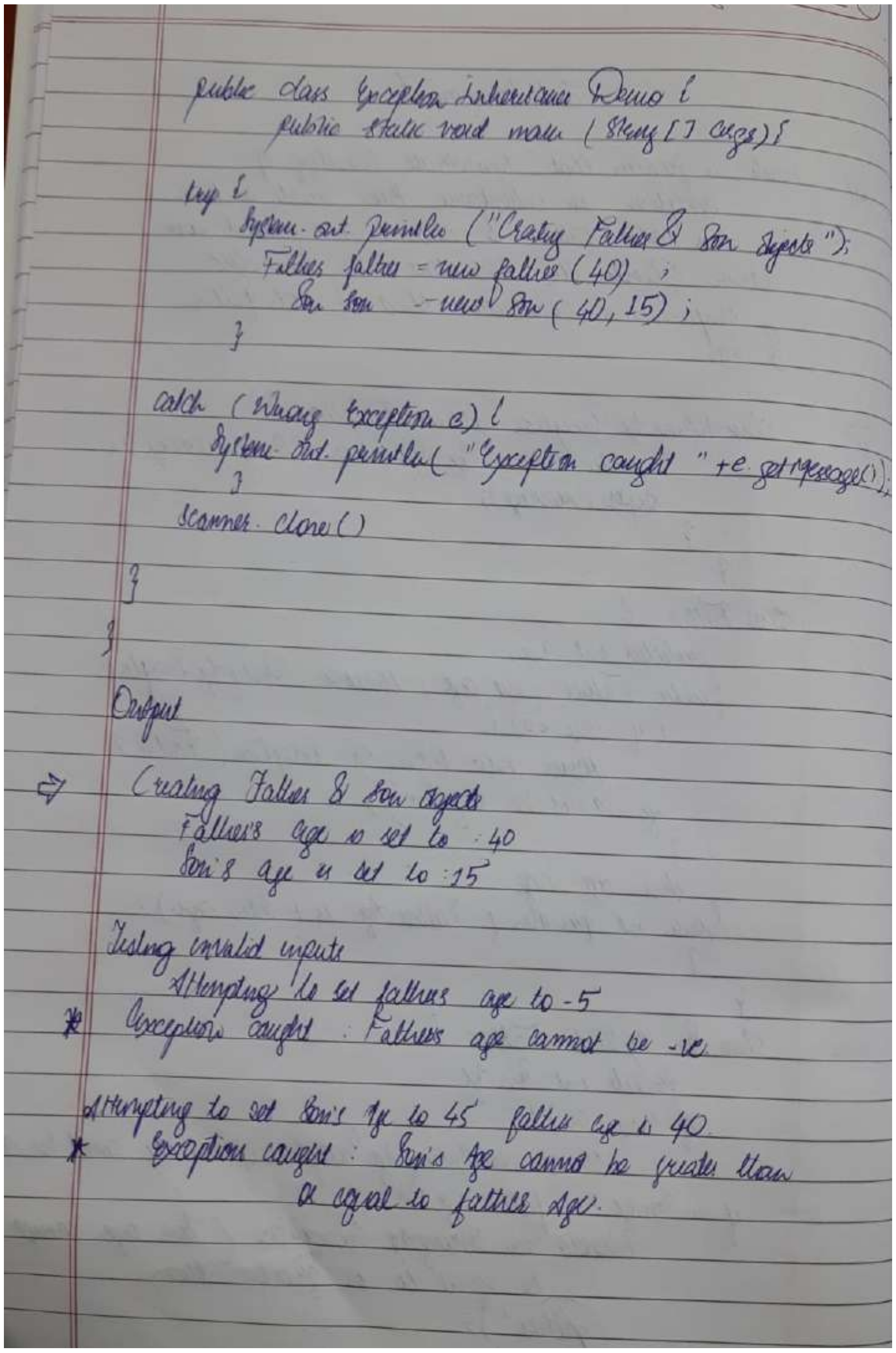


Lab Program 7

Q] Write a program that demonstrate handling of exception in inheritance tree. Create a base class called Father & derived class called son which extends the base class. In Son class implement a constructor that use both father & son.

⇨

```
class WrongAgeException extends Exception {
        public WrongAgeException (String message) {
        Super (message);
    }
  }
class Father {
    protected int age;
    public Father (int age) throws WrongAgeException
        { if (age <0) {
                throw new WrongAgeException ("Father's
        age cannot be negative");
        }
        this. age = age.
    System. out. println ( "Father Age is + this.age);
    }
 }
class Son extends Father {
    private int SonAge
    if ( age <0) {
            throw new WrongAgeException ("Son's age cannot be -ve
  if ( SonAge >= fatherAge) {
        throw new WrongAgeException ("Son age cannot
                be equal to or greater than
            father");
 }
```



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

try {
   System.out.println ("Creating Father & Son objects");
     Father father = new father (40) ;
       Son son  = new Son ( 40, 15) ;
   }

catch (WrongException e) {
  System.out.println( "Exception caught " +e.getMessage());
   }
Scanner.close()

 }
}
```

Output

```
⇨  Creating Father & Son objects
      Father's age is set to : 40
      Son's age is set to : 15

  Testing invalid inputs
      Attempting to set fathers age to -5
*  Exception caught : Fathers age cannot be -ve.

Attempting to set Son's Age to 45 father age is 40.
*   Exception caught : Son's Age cannot be greater than
              or equal to father Age.
```

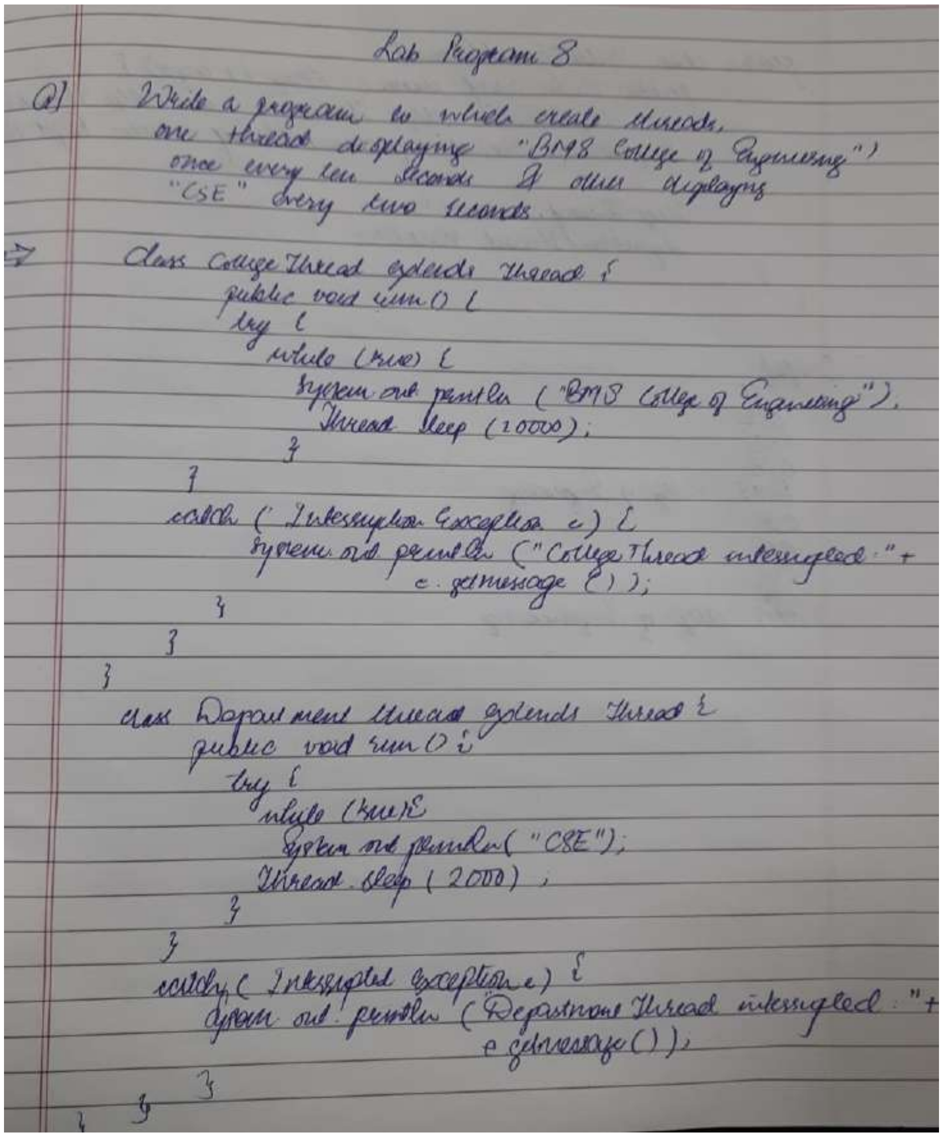
Code:

```
import java.util.Scanner;

class WrongAge extends Exception {
   public WrongAge() {
      super("Age Error");
   }

   public WrongAge(String message) {
        super(message);
    }
}

class InputScanner {
    Scanner s = new Scanner(System.in);
}

class Father extends InputScanner {
    int fatherAge;

    public Father() throws WrongAge {
        System.out.print("Enter Father's age: ");
        fatherAge = s.nextInt();

        if (fatherAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }

    public void display() {
        System.out.println("Father's age: " + fatherAge);
    }
}

class Son extends Father {
    int sonAge;

    public Son() throws WrongAge {
        super();

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

        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to father's age");
        } else if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }

    public void display() {
        System.out.println("Son's age: " + sonAge);
        super.display();  // This calls the Father's display method
    }
}
```

```
public class Exception_Handling{
    public static void main(String[] args) {
        try {
            System.out.println("ATHARV BORIKAR  1BM23IC015");
            Son son = new Son();
            son.display();
        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

Program 8
Threads Creation

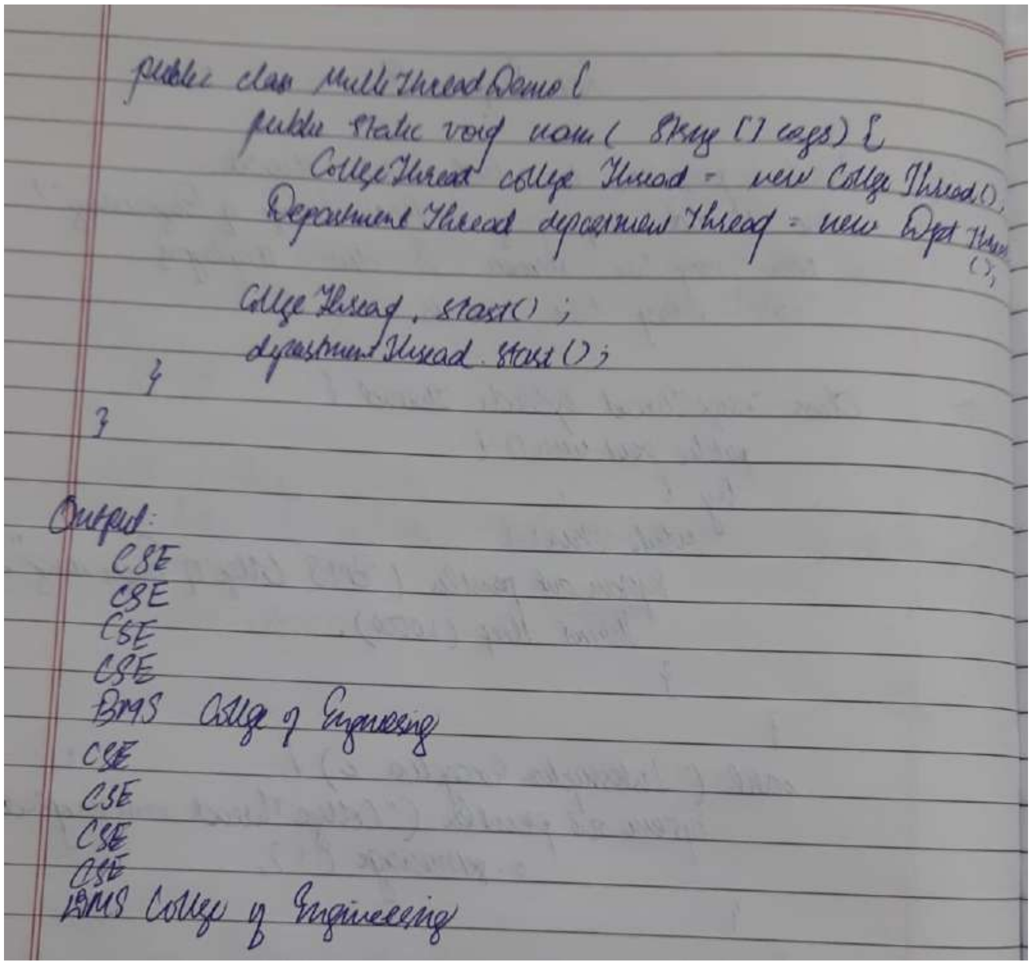
Algorithm:



Lab Program 8

Q] Write a program to which create threads, one thread displaying "BMS College of Engineering") once every ten seconds & other displaying "CSE" every two seconds

```
⇒ class CollegeThread extends Thread {
        public void run() {
        try {
            while (true) {
                System.out.println ("BMS College of Engineering").
                  Thread.sleep (10000);
            }
        }
        catch (InterruptedException e) {
            System.out.println ("CollegeThread interrupted: "+
                        e.getMessage ());
        }
     }
  }
  class DepartmentThread extends Thread {
     public void run() {
        try {
          while (true){
            System.out.println( "CSE");
          Thread.sleep ( 2000) ;
          }
        }
        catch( InterruptedException e) {
          System.out.println ("DepartmentThread interrupted: "+
                        e.getMessage());
        }
     }
  }
```



```
public class MultiThreadDemo {
        public static void main( String [] args) {
            CollegeThread collegeThread = new CollegeThread();
        DepartmentThread departmentThread = new DeptThread();

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

Output:

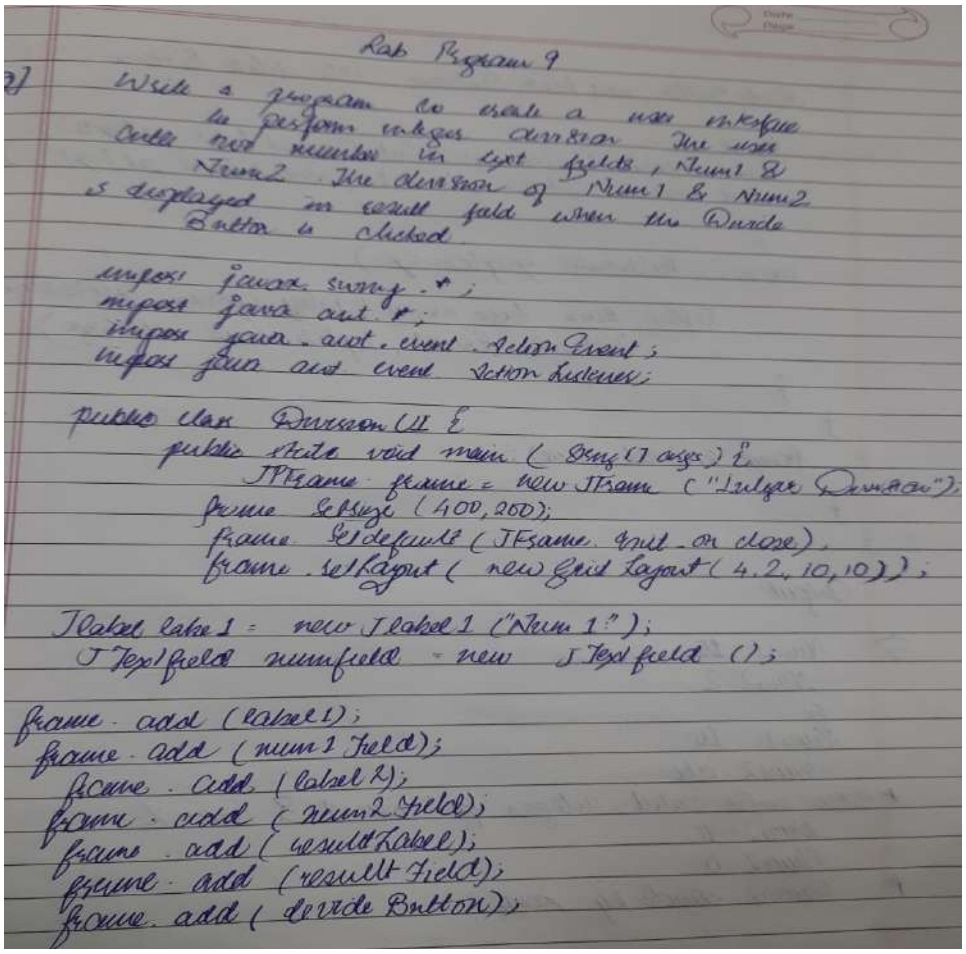
```
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
```

Code:

```
public class Threads {
    static class BMSDisplayThread extends Thread {
        public void run() {
            try {
                while (true) {
                    System.out.println("BMS College of Engineering");
                    Thread.sleep(10000);  // Sleep for 10 seconds
                }
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }

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

    public static void main(String[] args) {
        BMSDisplayThread bmsThread = new BMSDisplayThread();
        CSEDisplayThread cseThread = new CSEDisplayThread();
        System.out.println("ATHARV BORIKAR");
        System.out.println("1BM23IC015");
        bmsThread.start();
        cseThread.start();
    }
}
```

<u>Program 9</u>
User Interface Creation

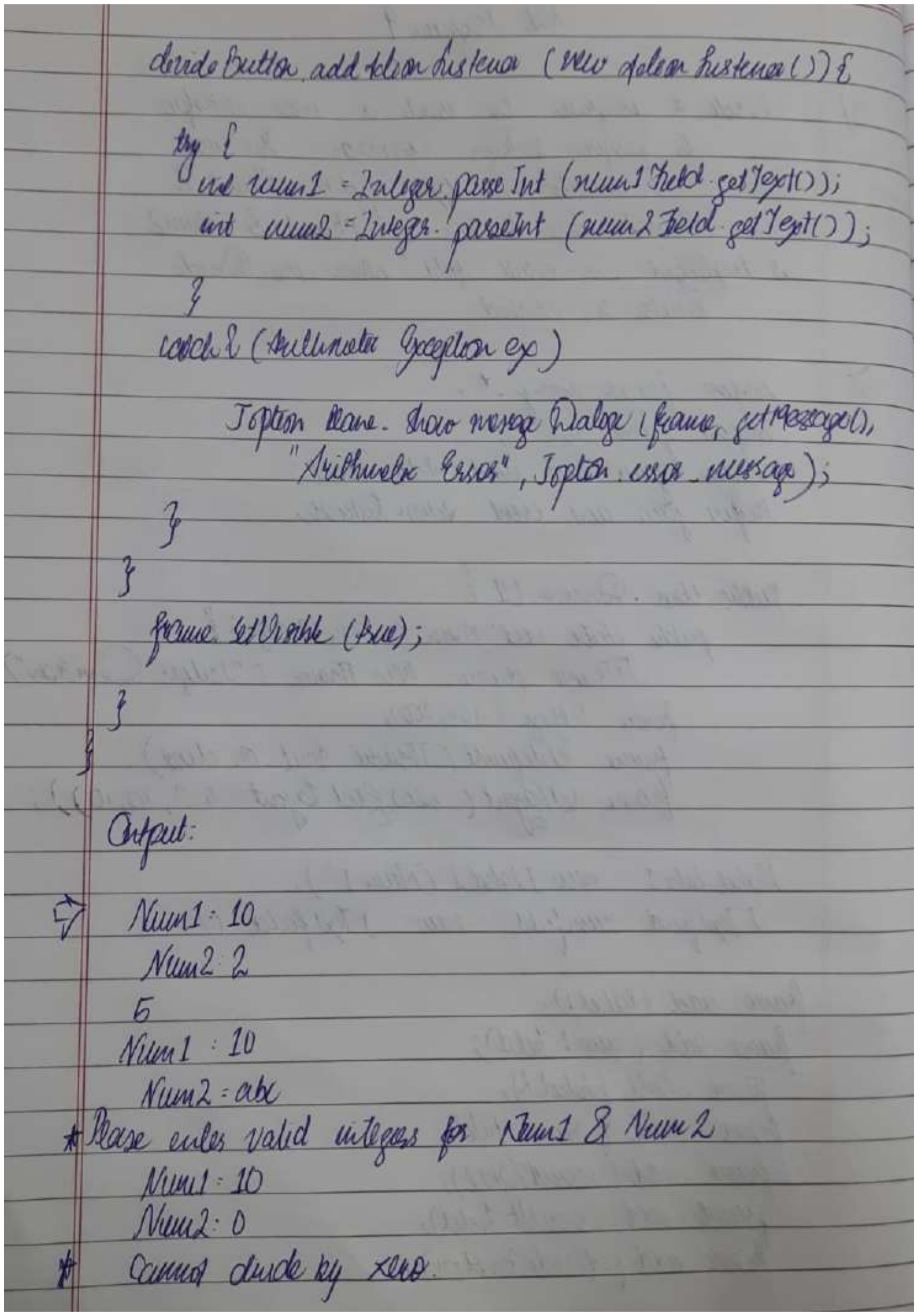
Algorithm:



Lab Program 9

2] Write a program to create a user interface to perform integer division. The user enters two numbers in text fields, Num1 & Num2. The division of Num1 & Num2 is displayed in result field when the Divide Button is clicked.

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

public class DivisionUI {
    public static void main (String[] args) {
        JFrame frame = new JFrame ("Integer Division");
    frame.setSize (400, 200);
    frame.setDefault (JFrame.exit_on_close).
    frame.setLayout ( new GridLayout (4, 2, 10, 10));

Jlabel label1 = new Jlabel1 ("Num 1:");
 JTextfield numfield = new JTextfield ();

frame.add (label1);
 frame.add (num1Field);
  frame.add (label2);
 frame.add (num2Field);
  frame.add (resultLabel);
  frame.add (resultField);
  frame.add (devide Button);
```



```
divideButton.addActionListener (new ActionListener()) {

  try {
    int num1 = Integer.parseInt (num1Field.getText());
    int num2 = Integer.parseInt (num2Field.getText());

  }
  catch { (ArithmeticException ex )

       JOptionPane.showMessageDialog (frame, getMessage(),
          "Arithmetic Error", JOption.error_message);
  }
}
  frame.setVisible (true);

}
}
```

Output:

```
Num1: 10
 Num2: 2
 5
Num1 : 10
 Num2 = abc
* Please enter valid integers for Num1 & Num2
 Num1: 10
 Num2: 0
* Cannot divide by zero.
```

Code:

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

class SwingDemo {
   SwingDemo() {
      JFrame jfrm = new JFrame("Divider App");
      jfrm.setSize(275, 200);
      jfrm.setLayout(new FlowLayout());

      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JLabel jlab = new JLabel("Enter the divisor and dividend:");

      JTextField ajtf = new JTextField(8);
      JTextField bjtf = new JTextField(8);

      JButton button = new JButton("Calculate");

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

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

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

               int ans = a / b;

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

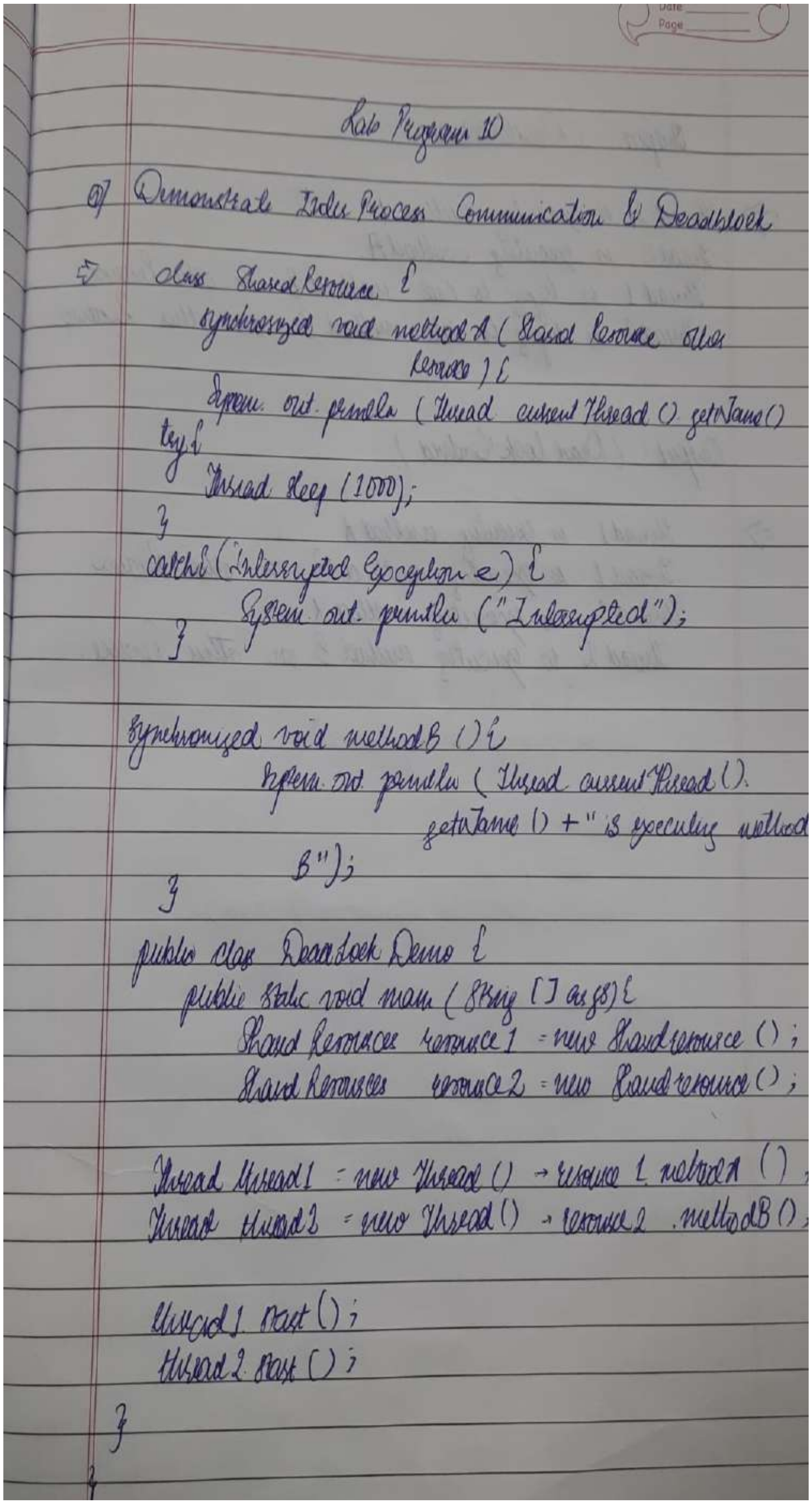
```
            }
        }
    });

    jfrm.setVisible(true);
  }

  public static void main(String args[]) {
    SwingUtilities.invokeLater(new Runnable() {
      public void run() {
        new SwingDemo();
      }
    });
  }
} import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SwingDemo {
  SwingDemo() {
    JFrame jfrm = new JFrame("Divider App");
    jfrm.setSize(275, 200);
    jfrm.setLayout(new FlowLayout());

    jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JLabel jlab = new JLabel("Enter the divisor and dividend:");

    JTextField ajtf = new JTextField(8);
    JTextField bjtf = new JTextField(8);

    JButton button = new JButton("Calculate");

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

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

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

                    int ans = a / b;

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

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

Program 10 a)

Demonstrating IPC

Algorithm:

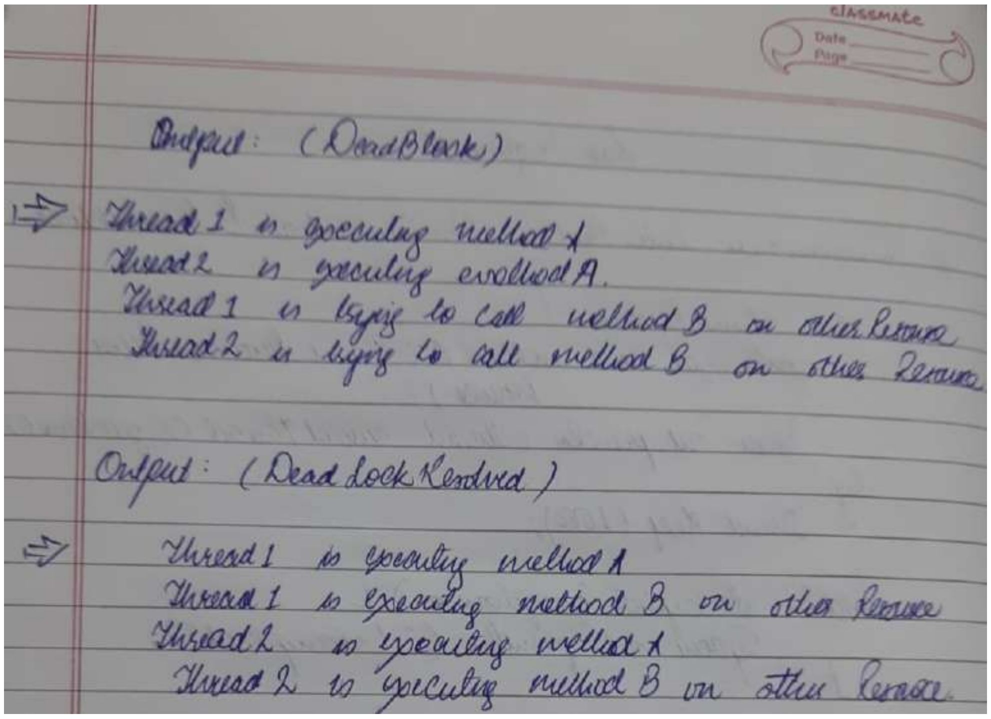


Date
Page

Lab Program 10

Q) Demonstrate Inter Process Communication & Deadblock

⇨
```
class Shared Resource {
    synchronized void method A ( Shared Resource other
                        Resource ) {
        System. out. println ( Thread current Thread () getName()
try {
    Thread sleep (1000);
}
catch ( Interrupted Exception e) {
        System. out. println ("Interrupted");
   }

synchronized void method B () {
        System. out. println ( Thread current Thread ().
                    getName () + " is executing method
            B");
   }

public class Deadlock Demo {
    public Static void main ( String [] args) {
        Shared Resources resource 1 = new Shared resource () ;
        Shared Resources resource 2 = new Shared resource () ;

Thread thread1 = new Thread () → resource 1 method A () ;
Thread thread2 = new Thread () → resource 2 .methodB () ;

thread1 start () ;
thread2 start () ;

}
}
```

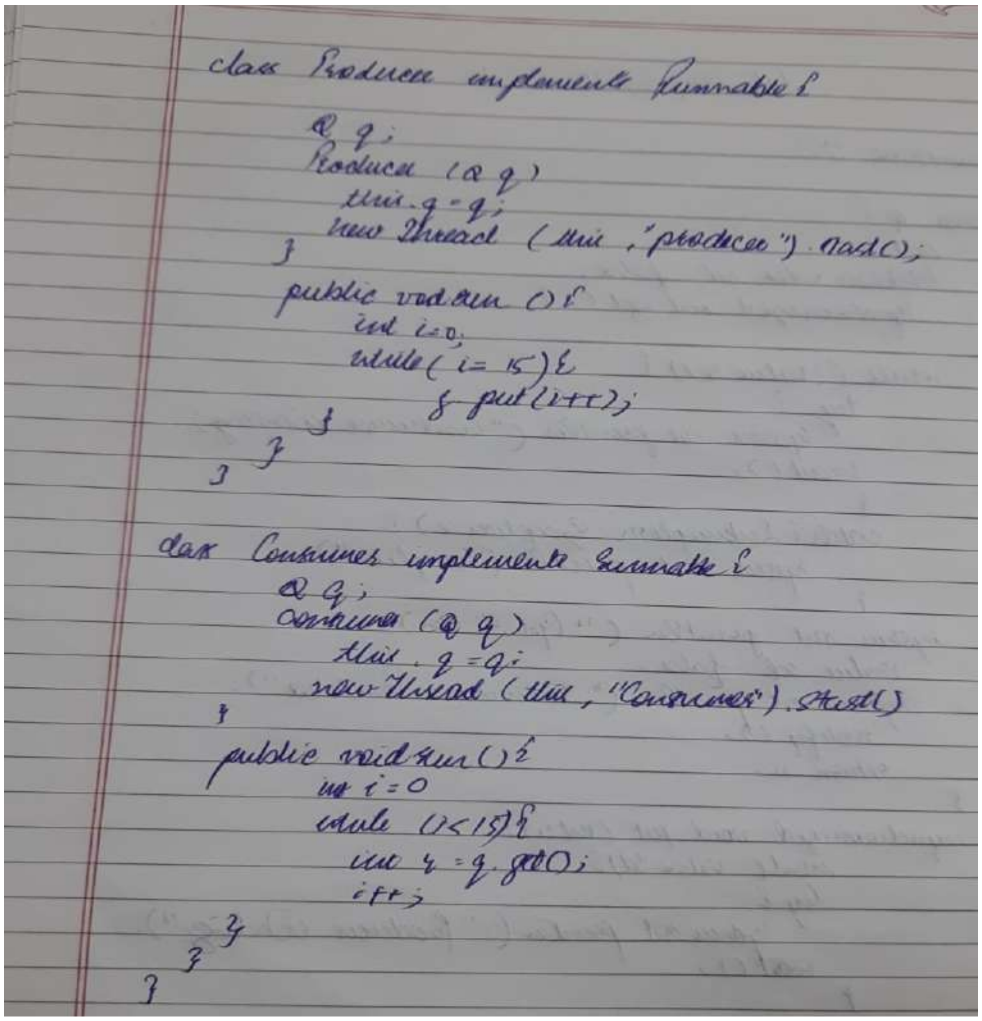


Output: (DeadBlock)

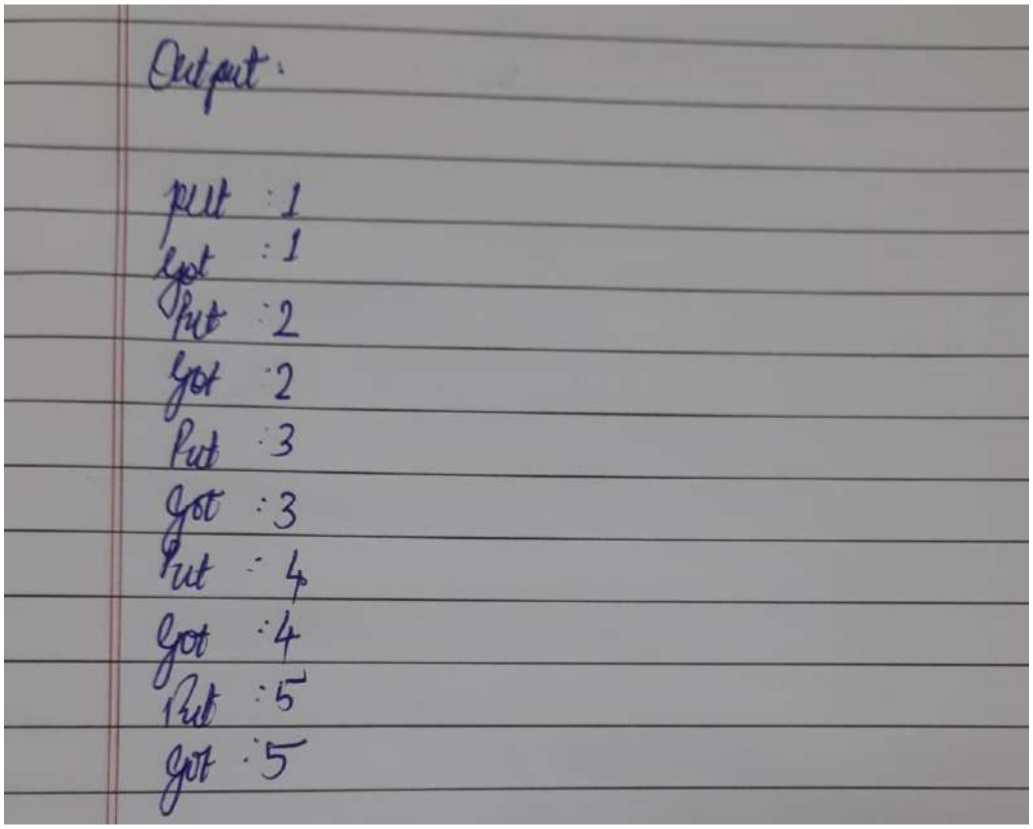
⇨ Thread 1 is executing method A
Thread 2 is executing method A.
Thread 1 is trying to call method B on other Resource
Thread 2 is trying to call method B on other Resource

Output: (Dead lock Resolved)

⇨ Thread 1 is executing method A
Thread 1 is executing method B on other Resource
Thread 2 is executing method A
Thread 2 is executing method B on other Resource.



```
class Producer implements Runnable {
      Q q;
      Producer (Q q)
        this.q = q;
        new Thread (this, "producer").start();
    }
    public void run () {
        int i=0;
        while( i= 15) {
               q.put(i++);
        }
    }
}

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



```
Output:

put :1
got :1
Put :2
Got :2
Put :3
Got :3
Put :4
Got :4
Put :5
Got :5
```

Code:

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

   synchronized int get() {
      while(!valueSet)
         try {
            System.out.println("\nConsumer waiting\n");
            wait();
         } catch(InterruptedException e) {
            System.out.println("InterruptedException caught");
         }
      System.out.println("Got: " + n);
      valueSet = false;
      System.out.println("\nIntimate Producer\n");
      notify();
      return n;
   }

   synchronized void put(int n) {
      while(valueSet)
         try {
            System.out.println("\nProducer waiting\n");
```

```
                wait();
            } catch(InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);
        System.out.println("\nIntimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
    public void run() {
        int i = 0;
        while(i<15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;
    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }
    public void run() {
        int i=0;
        while(i<15) {
            int r=q.get();
            System.out.println("consumed:"+r);
            i++;
        }
    }
}
class IPC {
    public static void main(String args[]) {
        Q q = new Q();
        System.out.println("ATHARV BORIKAR 1BM23IC015");
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }

}
```

<u>Program 10 b)</u>
Demonstrating Deadlock

Algorithm:

Code:

```
import java.util.*;
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();
    }
}
```