VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Shreya Jaganatha Gowda
(1BM23IC061)

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 **Shreya Jaganatha Gowda (1BM23IC061),** 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.Prasad G R Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

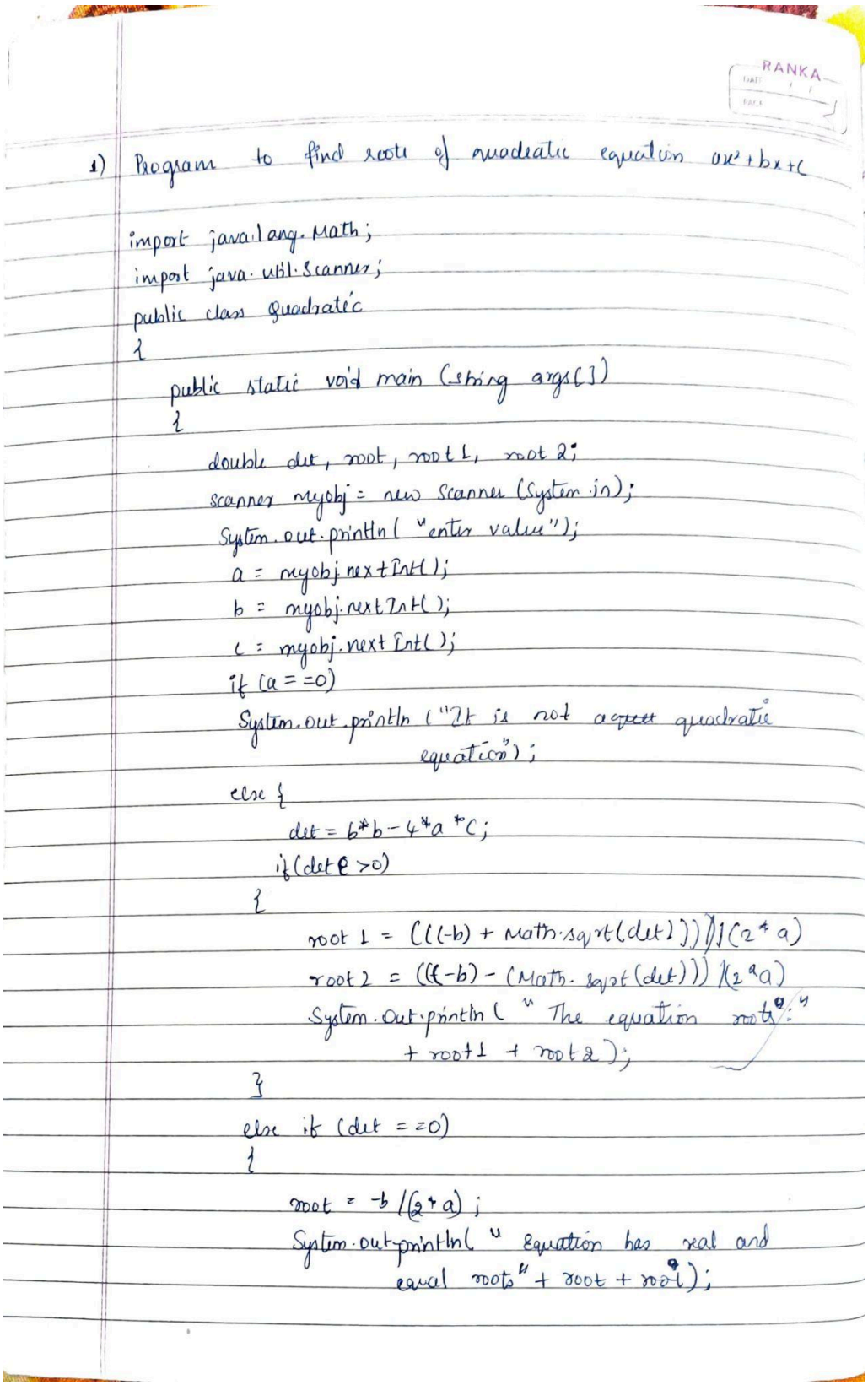
Index

Sl. No.	Date	Experiment Title	Page No.
1		Quadratic Equation	
2			
3			
4			
5			
6			
7			
8			
9			
10			

Github Link:
(You should provide your github link which contains all lab programs)

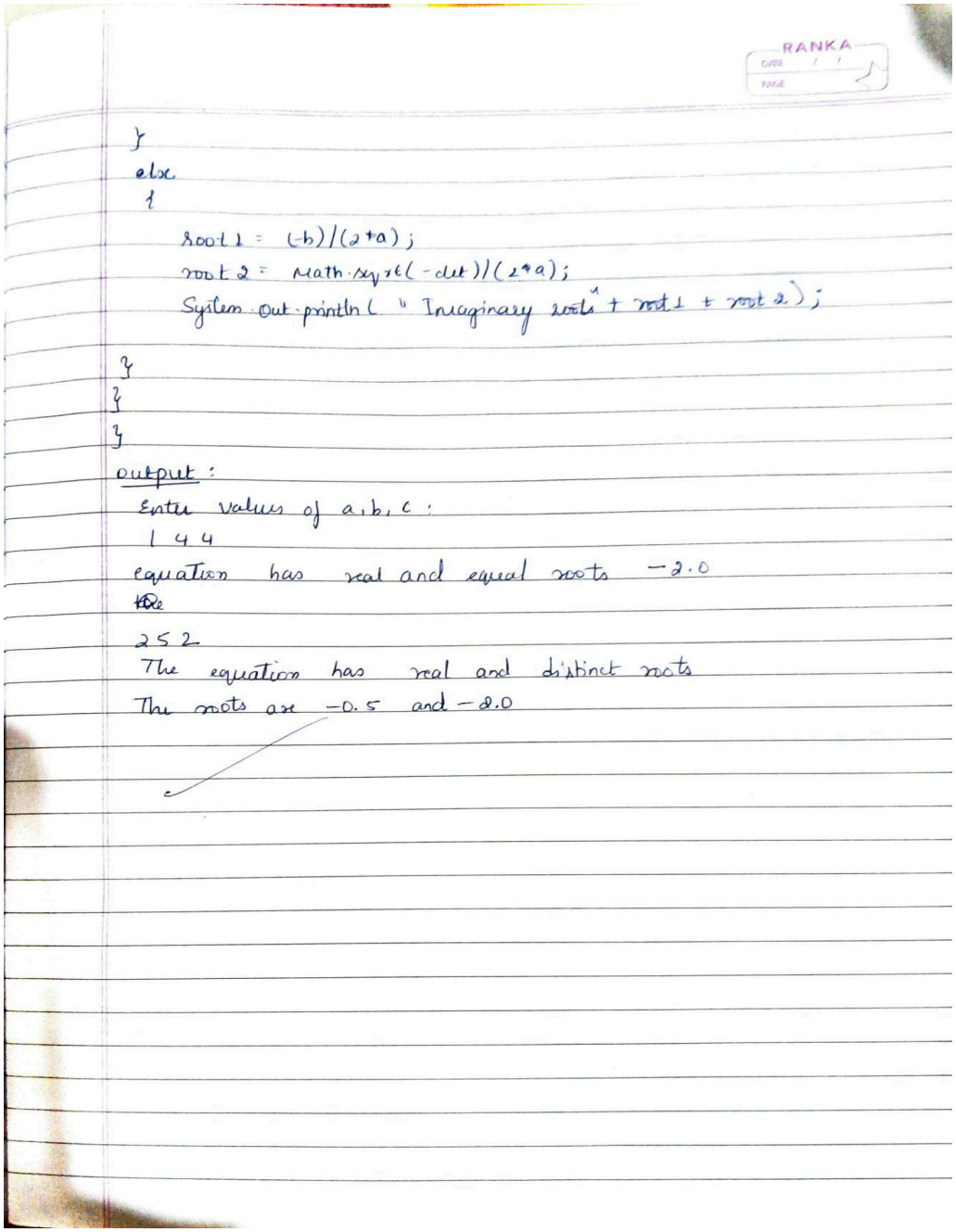
Program 1
Implement Quadratic Equation

Algorithm:



1) Program to find roots of quadratic equation ax^2+bx+c

```
import java.lang.Math;
import java.util.Scanner;
public class Quadratic
{
   public static void main (String args[])
   {
      double det, root, root1, root2;
      Scanner myobj = new Scanner (System.in);
      System.out.println( "enter value");
      a = myobj.nextInt();
      b = myobj.nextInt();
      c = myobj.nextInt();
      if (a==0)
      System.out.println ("It is not a quadratic
                           equation");
      else {
          det = b*b-4*a*c;
          if(det>0)
          {
              root1 = ((-b) + Math.sqrt(det)))/(2*a)
              root2 = ((-b) - (Math.sqrt(det)))/(2*a)
              System.out.println( " The equation roots:"
                    + root1 + root2);
          }
          else if (det==0)
          {
              root = -b/(2*a);
              System.out.println( " Equation has real and
                    equal roots" + root + root);
```



```
}
else
{
    root1 = (-b)/(2*a);
    root2 = Math.sqrt(-det)/(2*a);
    System.out.println("Imaginary roots" + root1 + root2);
}
}
}
```

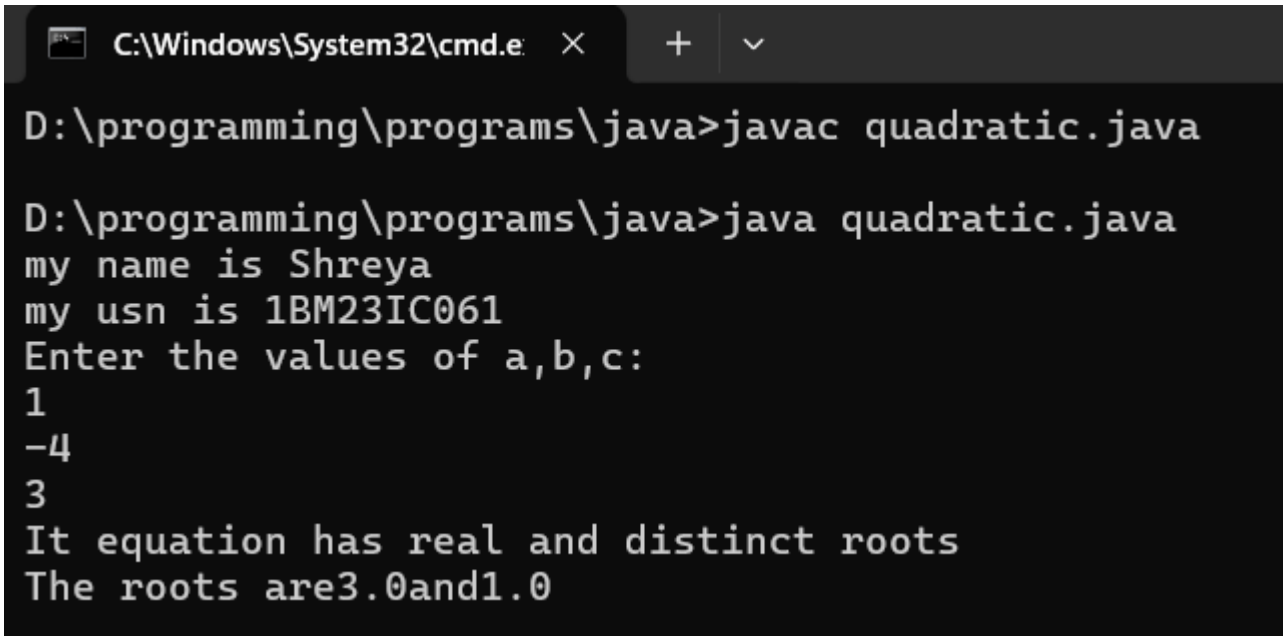
output:

```
Enter values of a,b,c:
1 4 4
equation has real and equal roots -2.0
2 5 2
The equation has real and distinct roots
The roots are -0.5 and -2.0
```

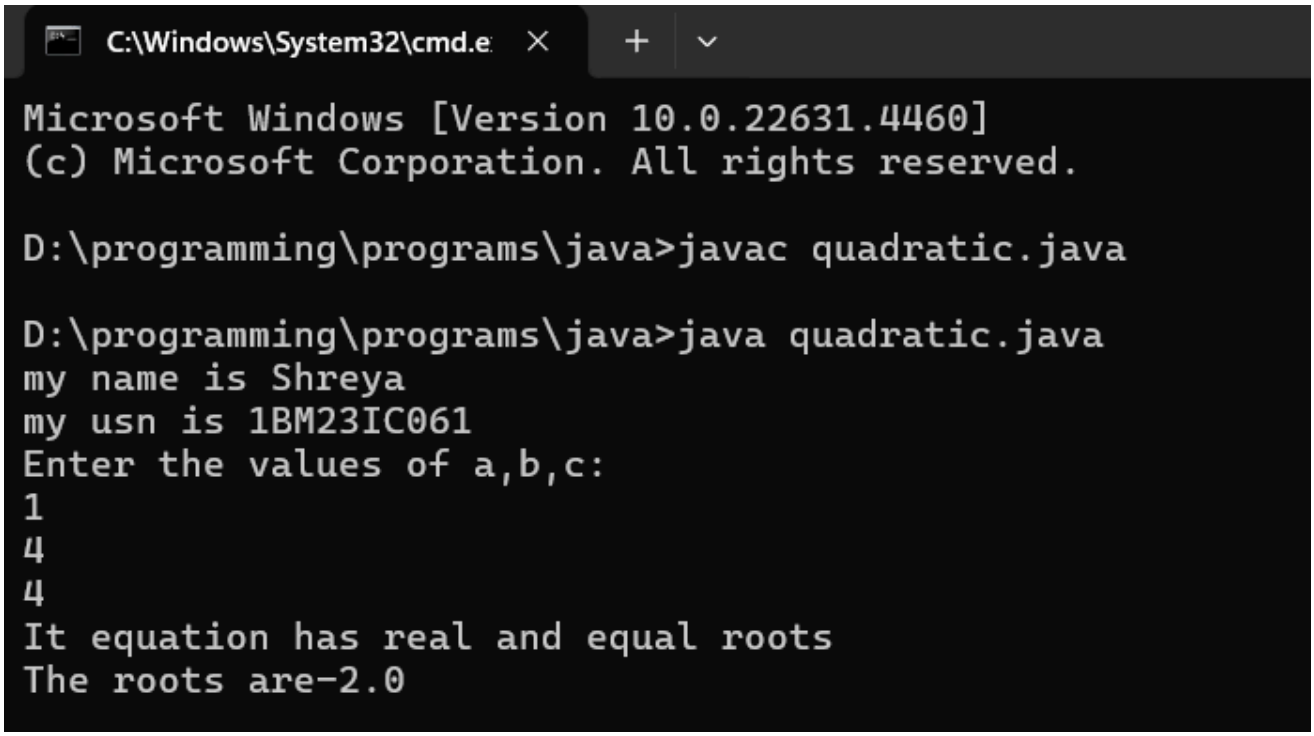
Code:

```
import java.util.Scanner;
import java.lang.Math;
```

```
class quadratic
{
 public static void main(String args[])
{
  int a,b,c;
  double det,root,root1,root2;
  Scanner myobj=new Scanner(System.in);
System.out.println("my name is Shreya");
System.out.println("my usn is 1BM23IC061");
  System.out.println("Enter the values of a,b,c:");
  a=myobj.nextInt();
  b=myobj.nextInt();
  c=myobj.nextInt();

  if(a==0)
  {
  System.out.println("It is not a quadratic equation");
  }
  else
  {
    det=(b*b)-(4*a*c);
    if(det>0)
    {
      root1=(((-b)+(Math.sqrt(det)))/(2*a));
      root2=(((-b)-(Math.sqrt(det)))/(2*a));
      System.out.println("It equation has real and distinct roots");
      System.out.println("The roots are"+root1+"and"+root2);
    }
    else if(det==0)
    {
     root=-b/(2*a);
     System.out.println("It equation has real and equal roots");
     System.out.println("The roots are"+root);
    }
    else
    {
     root1=(-b)/(2*a);
     root2=Math.sqrt(-det)/(2*a);
     System.out.println("It equation has imaginary roots");
     System.out.println("The root 1 is"+root1+"+i"+root2);
           System.out.println("The root 2 is"+root1+"-i"+root2);
    }
  }
}
```

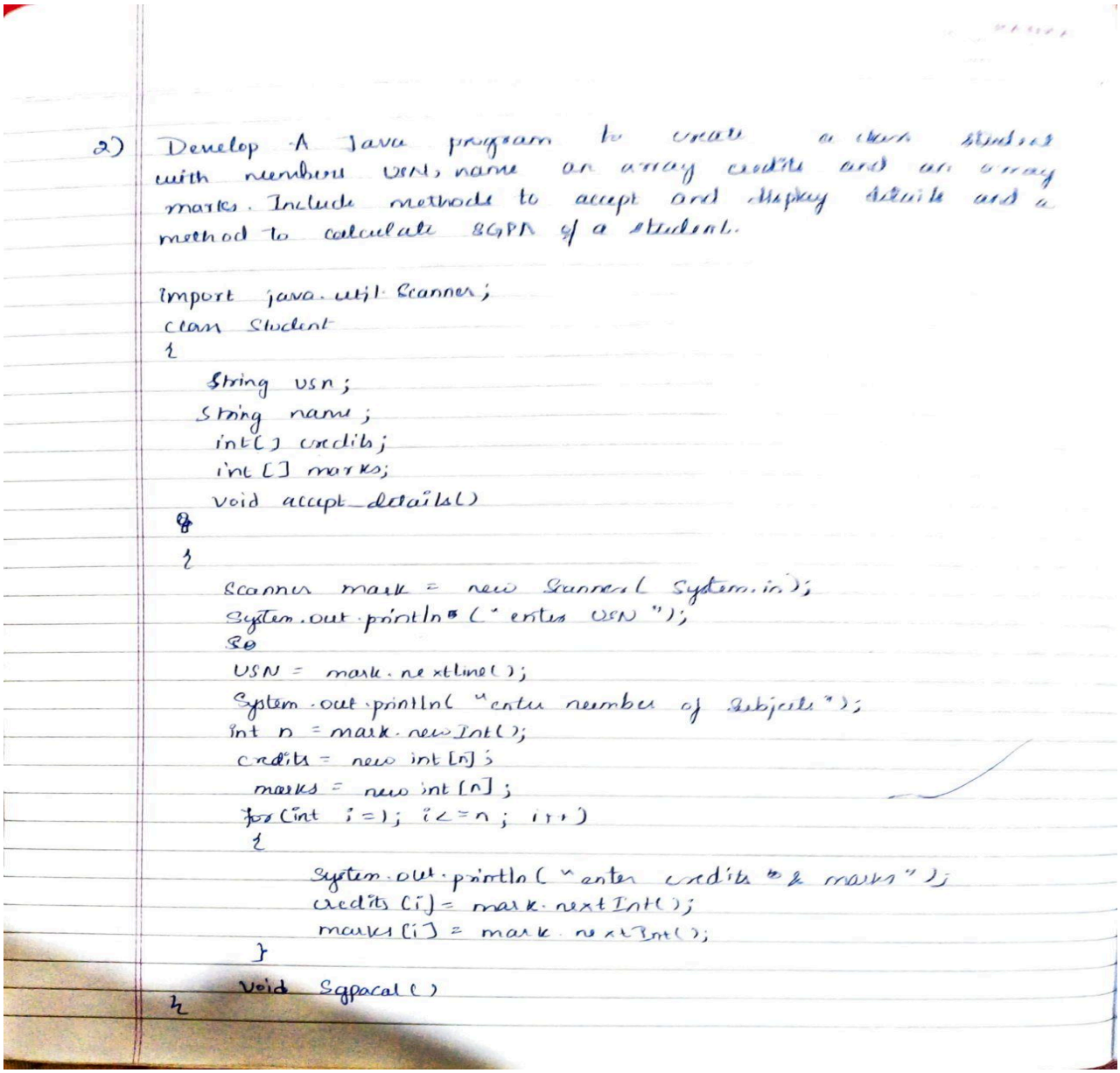



```
D:\programming\programs\java>javac quadratic.java

D:\programming\programs\java>java quadratic.java
my name is Shreya
my usn is 1BM23IC061
Enter the values of a,b,c:
1
0
16
It equation has imaginary roots
The root 1 is0.0+i4.0
The root 2 is0.0-i4.0

D:\programming\programs\java>

D:\programming\programs\java>
```


C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.
D:\programming\programs\java>javac quadratic.java
D:\programming\programs\java>java quadratic.java
my name is Shreya
my usn is 1BM23IC061
Enter the values of a,b,c:
1
4
4
It equation has real and equal roots
The roots are-2.0

Program 2
Calculate the SGPA of a student

Algorithm:



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.

```
import java.util.Scanner;
class Student
{
    String usn;
    String name;
    int[] credits;
    int[] marks;
    void accept_details()
    {
        Scanner mark = new Scanner( System.in );
        System.out.println(" enter USN ");
        USN = mark.nextline();
        System.out.println("enter number of subjects");
        int n = mark.nextInt();
        credits = new int[n];
        marks = new int[n];
        for(int i=1; i<=n; i++)
        {
            System.out.println("enter credits & marks");
            credits[i] = mark.nextInt();
            marks[i] = mark.nextInt();
        }
        void Sgpacal()
    }
```

```
double Sgpa;
int cred=0, mark=0;
for cred=0, mark=0;
for (int i=0; i<n; i++)
{
    cred += credits[i];
    mark += marks[i];
}
System.out.println("Total credits "+cred);
System.out.println("Total marks: " + mark);
for(int i=0; i<n; i++)
{
    double sgpa = ((marks[i]/10.0)) * credits[i];
}
System.out.println("SGPA is : " + sgpa);
void displaydetails()
{
    System.out.println("Name: "+mark.name);
    System.out.println("USN:" + mark.usn);
    for (int i=1; i<=n; i++)
    {
        System.out.println("Credits for " +i+ credits[i]);
        System.out.println("Marks for " +i+ marks[i]);
    }
    public static void main(String [] args)
    {
        Student obj = new Student();
        obj.acceptdetails();
        obj.displaydetails();
        obj.Sgpacal();
    }
}
}
```

Code:

```
import java.util.*;

class Stud_det {
    int m[] = new int[8];
    int c[] = new int[8];
    int p[] = new int[8];
    int g, sum;
    String name, usn;
    double sgpa;
    Scanner s = new Scanner(System.in);
    void getdetails()
    {
        System.out.println("My name is archie");
        System.out.println("My usn is 1BM23CS049");
        System.out.println("Enter name:");
        name = s.next();
        System.out.println("Enter usn:");
        usn = s.next();
        for (int i = 0; i < 8; i++) {
            System.out.println("Enter marks of subject:"+(i+1));
            m[i] = s.nextInt();
            System.out.println("Enter credits for subject:"+(i+1));
            c[i] = s.nextInt();
        }
    }

    void gradepoint() {
        for (int i = 0; i < 8; i++) {
            if (m[i] >= 90 && m[i] <= 100)
                p[i] = 10;
            else if (m[i] >= 80 && m[i] < 90)
                p[i] = 9;
            else if (m[i] >= 70 && m[i] < 80)
                p[i] = 8;
            else if (m[i] >= 60 && m[i] < 70)
```

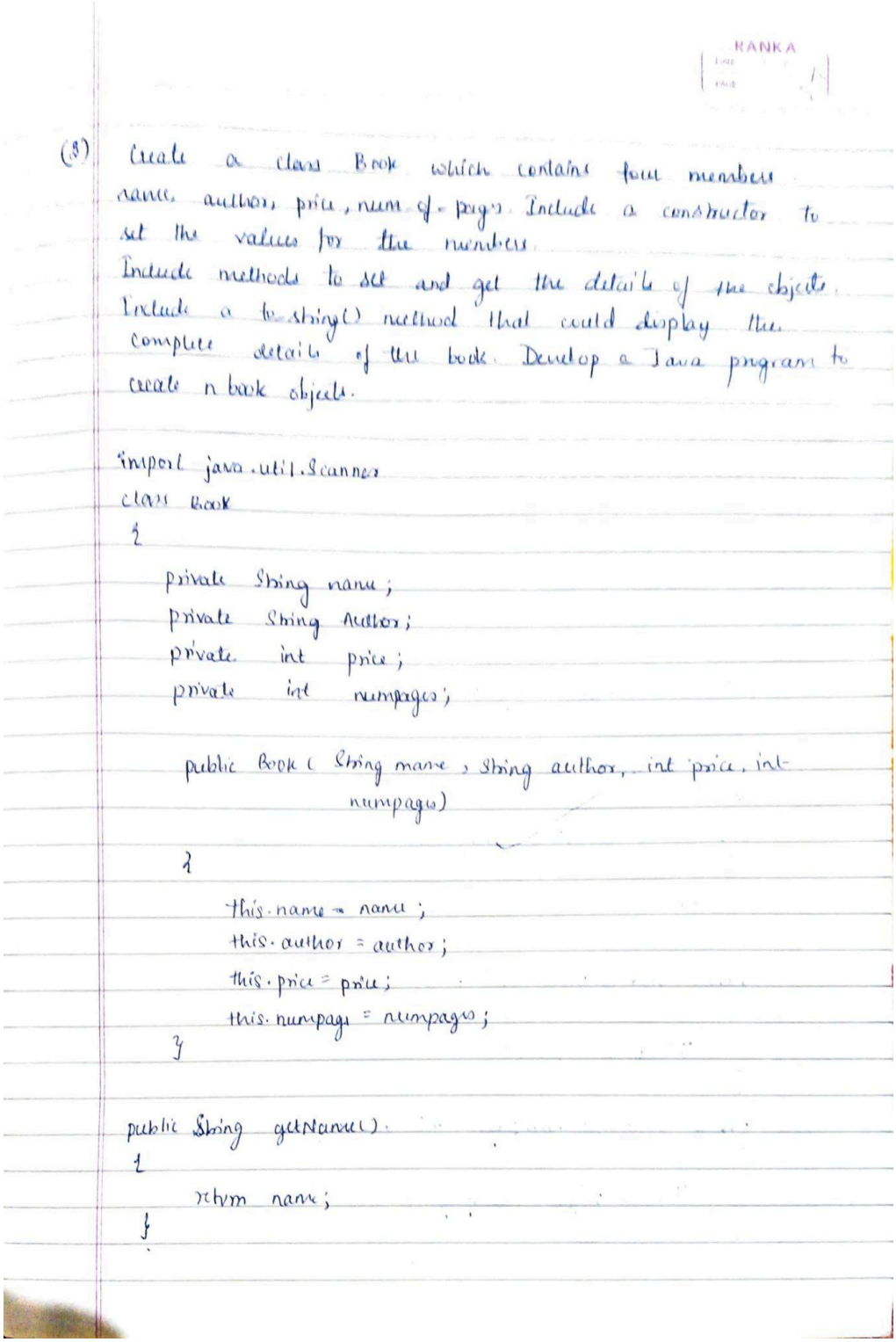
```
            p[i] = 7;
        else if (m[i] >= 50 && m[i] < 60)
            p[i] = 6;
        else if (m[i] >= 40 && m[i] < 50)
            p[i] = 5;
        else
            p[i] = 0;
      }
   }

   void calculate() {
      for (int i = 0; i < 8; i++) {
         g += c[i] * p[i];
      }
      for (int i = 0; i < 8; i++) {
         sum += c[i];
      }
      sgpa = g / sum;
   }

   void display() {
      System.out.println("Name:" + name);
      System.out.println("USN:" + usn);
      System.out.println("SGPA=:" + sgpa);
   }
}

class student {
 public static void main(String a[]){
  Stud_det s1[]=new Stud_det[3];
  for(int i=0;i<3;i++)
  {
   s1[i]=new Stud_det();
  }
  for(int i=0;i<3;i++)
  {
   System.out.println("Enter details of student:"+(i+1));
   s1[i].getdetails();
  }
  for(int i=0;i<3;i++)
  {
   s1[i].gradepoint();
   s1[i].calculate();
  }
  for(int i=0;i<3;i++)
  {
   System.out.println("Student-"+(i+1));
```

```
    s1[i].display();
  }
 }
}
```

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.

Algorithm:



(8) Create a class Book which contains four members name, author, price, num of pages. Include a constructor to set the values for the members.
Include methods to set and get the details of the objects.
Include a to_string() 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 int price;
    private int numpages;

    public Book ( String name , String author, int price, int
                numpages)

    {
        this.name = name;
        this.author = author;
        this.price = price;
        this.numpages = numpages;
    }

public String getName().
{
    return name;
}
```

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

public int getprice()
{
    return price;
}
public int getnumpages()
{
    return numpages;
}

public void setname(String name)
{
    this.name = name;
}
public void SetAuthor(String author)
{
    this.author = author;
}
public void setprice(int price)
{
    this.price = price;
}
public void set numpages(int numpages)
{
    this.numpages = numpages;
}
```

```
public String toString()
{
    return "Book Details : \n" + "Name:" + name + "\n" +
        "Author:" + author + "\n" + "price:" + price +
        "\n" + "Number of pages :" + numpages + "\n";
}
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("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("Book details :");
            System.out.println("Book name:");
            String name = sc.nextLine();
            System.out.println("Books price");
            int price = sc.nextInt();
            books[i] = new Book(name, author, price, numpages)
        }
        System.out.println("Details of all books :");
        for(Book book: books)
        {
            System.out.println(book);
        }
    }
}
```

Output:

Enter the number of books: 3
Book details:
Book name:
I came upon a light house
Author's name: Shantaru Naidu
Book's price: 200
Number of pages: 175

Book details:
Book name: Mast and more
Author's Name: Maharsia Ra
Book's price: 275
Number of pages: 250

Book details:
Book name: Two states
Boo
Author's name: Chetan Bhagat
Book's price: 175
Number of pages: 200

Details of all books
Book @ ₹184₹C6
Book @ ₹184₹C7
Book @ ₹184₹C8

Code:

```
import java.util.Scanner;
class Book
{
 private String name;
 private String author;
 private int price;
 private int numpages;

 public Book(String name, String author, int price, int numpages)
{
 this.name=name;
 this.author=author;
 this.price=price;
 this.numpages=numpages;
}
 public String getname()
{
 return name;
}
public String  getauthor()
{
 return author;
}
public int getprice()
{
 return price;
}
public int getnumpages()
{
 return numpages;
}
public void setname(String name)
{
 this.name=name;
}
public void setauthor(String author)
{
 this.author=author;
}
public void setprice(int price)
{
 this.price=price;
}
public void setnumpages(int numpages)
{
```

```
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;
}
}
public class bookse
{
public static void main(String []args)
{
Scanner sc=new Scanner(System.in);
System.out.println("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("Book details:");
System.out.println("Book name:");
 String name=sc.nextLine();
System.out.println("Authors name:");
String author=sc.nextLine();
System.out.println("Book price:");
int price=sc.nextInt();
System.out.println("Number of pages:");
int numpages=sc.nextInt();
books[i]=new Book (name,author,price,numpages);
}
System.out.println("Details of all books:");
for(Book book:books)
{
 System.out.println(book);
}}
}
```

```
D:\programming\programs\java>javac bookset.java

D:\programming\programs\java>javac bookset.java

D:\programming\programs\java>java bookset.java
Enter the number of books:
2
Book details:
Book name:
Alliegant
Authors name:
smith
Book price:
100
Number of pages:
46
Book details:
Book name:
Authors name:
two states
Book price:
456
Number of pages:
865
Details of all books:
Book@71b1176b
Book@6193932a
```

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.

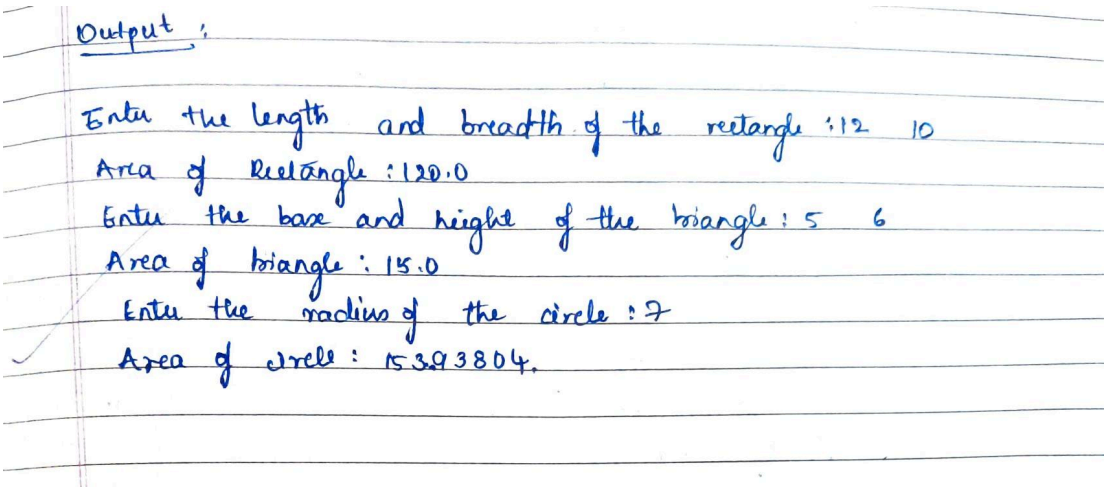
Algorithm:

(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 contain only the method printArea() that prints the area of the given Shape

```
import java.util.Scanner;
abstract class Shape
{
    int dimension1;
    int dimension2;
    abstract void printArea();
}
Class Rectangle extends Shape
{
    Rectangle ( int length ,int breadth)
    {
        this.dimension1 = length;
        this.dimension2 = breadth;
    }
    Void printArea()
    {
        double area = dimension1 * dimension2;
        System.out.println ("Area of rectangle = " +area);
    }
}
class Triangle extends Shape
{
    Triangle ( int base, int height)
    {
```

```
        this.dimension1 = base;
        this.dimension2 = height;
    }
    void printArea()
    {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of triangle = "+area);
    }
}
class circle extends shape
{
    circle (int radius)
    {
        this.dimension1 = radius;
        this.dimension2 = 0;
    }
    void printArea()
    {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of circle : " +area);
    }
}
public class Shapes
{
    public static void main (String []args)
    {
        Scanner sc = new Scanner (System.in);
        System.out.println("Enter the length and breadth of
                            the rectangle");
```

```
int length = sc.nextInt();
int breadth = sc.nextInt();
Shape rectangle = new Rectangle (length, breadth);
rectangle.printArea();
System.out.println("Enter the base and height of the triangle:")
int base = sc.nextInt();
Int height = sc.nextInt();
Shape triangle = new triangle (base, height);
triangle.printArea();
System.Out.println("Enter radius of the circle");
int radius= sc.nextInt();
shape circle = new circle (radius);
circle.printArea();
}
}
```



Code:

```
import java.util.Scanner;

abstract class Shape
{
   int dimension1;
   int dimension2;

   abstract void printArea();
}


class Rectangle extends Shape
{
   Rectangle(int length, int breadth)
   {
      this.dimension1 = length;
      this.dimension2 = breadth;
   }
   void printArea()
   {
      double area = dimension1 * dimension2;
      System.out.println("Area of Rectangle: " + area);
   }
}


class Triangle extends Shape
 {
```

```
    Triangle(int base, int height)
    {
        this.dimension1 = base;
        this.dimension2 = height;
    }

    void printArea()
    {
        double area = 0.5 * dimension1 * dimension2;
        System.out.println("Area of Triangle: " + area);
    }
}

class Circle extends Shape
{
    Circle(int radius)
    {
        this.dimension1 = radius;
        this.dimension2 = 0;
    }

    void printArea()
    {
        double area = Math.PI * dimension1 * dimension1;
        System.out.println("Area of Circle: " + area);
    }
}


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

        System.out.print("Enter the length and breadth of the rectangle: ");
        int length = sc.nextInt();
        int breadth = sc.nextInt();
        Shape rectangle = new Rectangle(length, breadth);
        rectangle.printArea();

        System.out.print("Enter the base and height of the triangle: ");
        int base = sc.nextInt();
        int height = sc.nextInt();
        Shape triangle = new Triangle(base, height);
        triangle.printArea();
```

```
        System.out.print("Enter the radius of the circle: ");
        int radius = sc.nextInt();
        Shape circle = new Circle(radius);
        circle.printArea();
    }
}
```

```
D:\programming\programs\java>javac shapes.java

D:\programming\programs\java>java shapes.java
Enter the length and breadth of the rectangle: 10
20
Area of Rectangle: 200.0
Enter the base and height of the triangle: 10
5
Area of Triangle: 25.0
Enter the radius of the circle: 7
Area of Circle: 153.93804002589985

D:\programming\programs\java>
```

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.

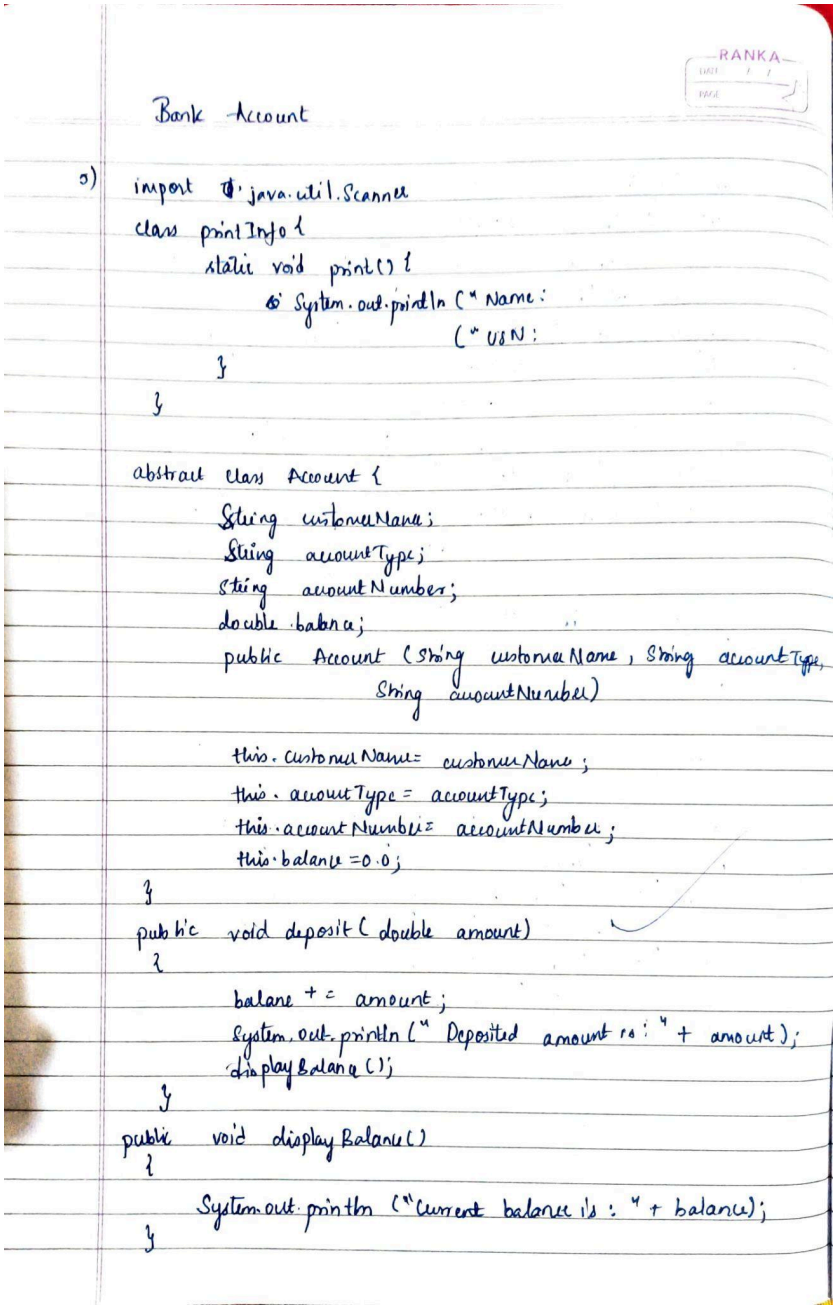
b)Display the balance.

c)Compute and deposit interest

d)Permit withdrawal and update the balance

e) Check for the minimum balance, impose penalty if necessary and update the balance.

Algorithm:



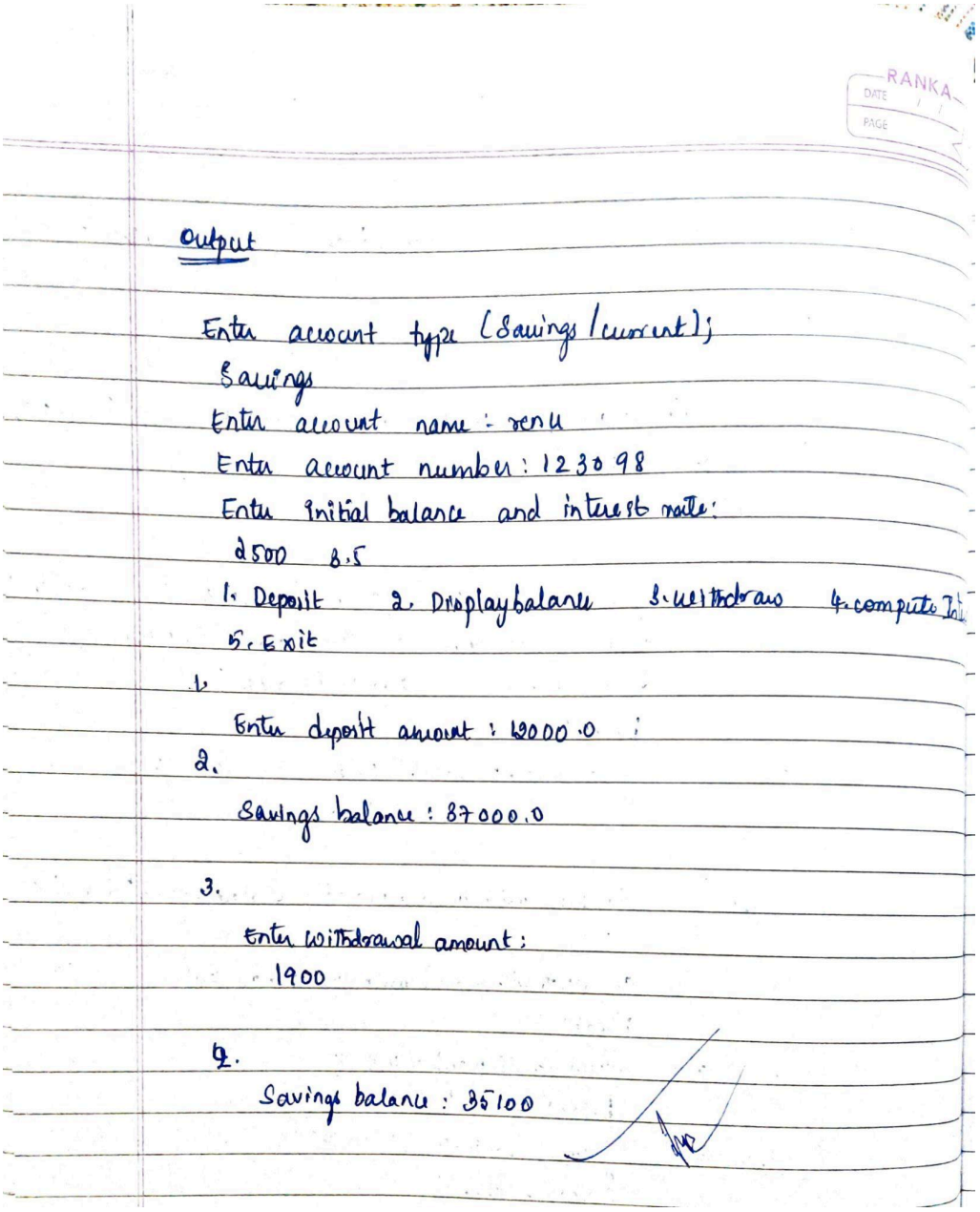
Bank Account

```
o)  import java.util.Scanner
    class printInfo {
        static void print() {
            System.out.println("Name:
                               ("USN:
        }
    }

    abstract class Account {
        String customerName;
        String accountType;
        String accountNumber;
        double balance;
        public Account (String customerName, String accountType,
                        String accountNumber)

        this.customerName= customerName;
        this.accountType = accountType;
        this.accountNumber= accountNumber;
        this.balance =0.0;
    }
    public void deposit ( double amount)
    {
        balance += amount;
        System.out.println("Deposited amount is: " + amount);
        displayBalance();
    }
    public void displayBalance()
    {
        System.out.println("Current balance is : " + balance);
    }
```

```
public abstract void withdraw (double amount);
}

class SavingsAcct extends Account
{
    double interestRate;
    public SavingsAcct (String CustomerName, String accountNumber,
                        double interestRate)
    {

        super (customerName, "savings", accountNumber);
        this.interestRate = interestRate;
    }

public void compoundDeposit()
{
    double interest = balance * (interestRate/100);
    deposit (interest);
    System.out.println (" Interest of " + interest + " deposited");
}
public void withdraw (double amount)
{
    if (amount <= balance)
    {
        balance -= amount;
        System.out.println(" Insufficient account for withdrawal");
        return;
    }
    displayBalance();
}
}
```

```
class curAcct extends Account {

    private static final double minBalance = 1000.0;
    private static final double serviceCharge = 50.0;

public curAcct ( String customerName, String accountNumber)

{
    super(customerName, "current", accountNumber);
}

public void withdraw (double amount) {
    if (amount <= balance)
    {
        balance -= amount;
        System.out.println("withdrawn amount is :" + amount);
    }
else
{ System.out.println("withdrawn ~~amount~~ is Insufficient amount
                    for withdrawn");
 return;
}
if( balance < minBalance)
 {
    balance -= serviceCharge;
    System.out.println("Enter interest rate :");
    ~~double rate~~
    double interestRate = Scanner.nextDouble();
    account = new SavAcct( customerName, accountnumber),
                    interestRate);
    }
else {
```

```
account = new CurAcct ( customerName, accountNumber);
}
while (true)
{
    System.out.println("1. Deposit \n 2. withdraw \n 3. Display
    \n Balance \n 4. Exit");
    Switch(choice)
    {
        case 1: System.out.println ("Enter amount to deposit");
                double depositAmount = Scanner.nextDouble();
                account.deposit( depositAmount);
                break;
        case 2: System.out.println("Enter amount to
                                    withdraw:");

                double withdrawAmount = Scanner.nextDouble();

                account.withdraw( withdrawAmount);
                break;
        case 3: account.displayBalance();
                break;
        case 4: System.out.println ("Exit");
                Scanner.close();
                return;

        default : System.out.println ("Try Again");
    }
}
}
```



Output

Enter account type (Savings/current);
Savings
Enter account name : renu
Enter account number : 123098
Enter initial balance and interest rate:
2500 8.5
1. Deposit 2. Display balance 3. Withdraw 4. compute 5. Exit
1.
Enter deposit amount : 29000.0
2.
Savings balance : 37000.0
3.
Enter withdrawal amount :
1900
4.
Savings balance : 35100

Code:

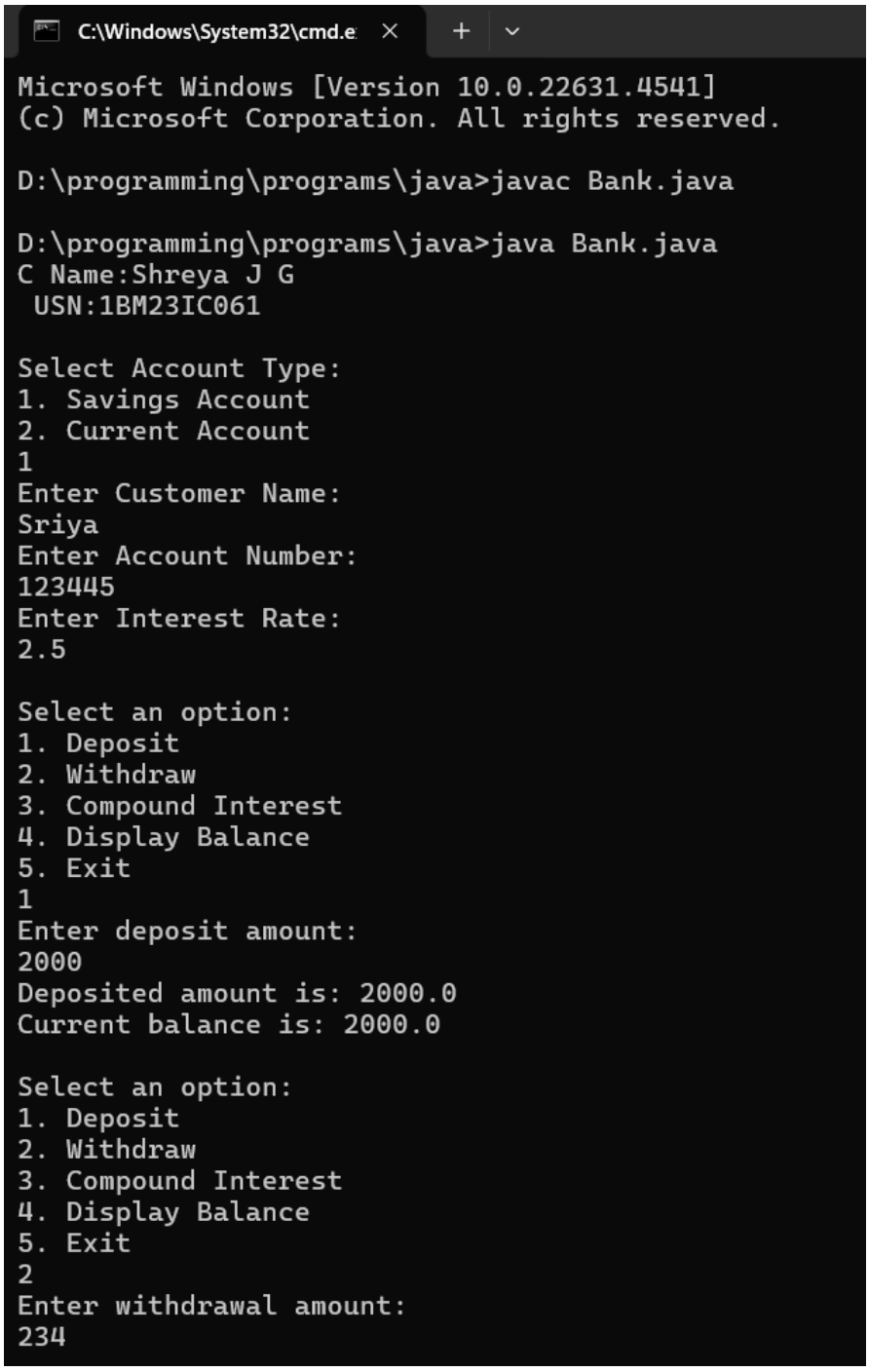
```
import java.util.Scanner;
// Main class with switch cases
public class Bank {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

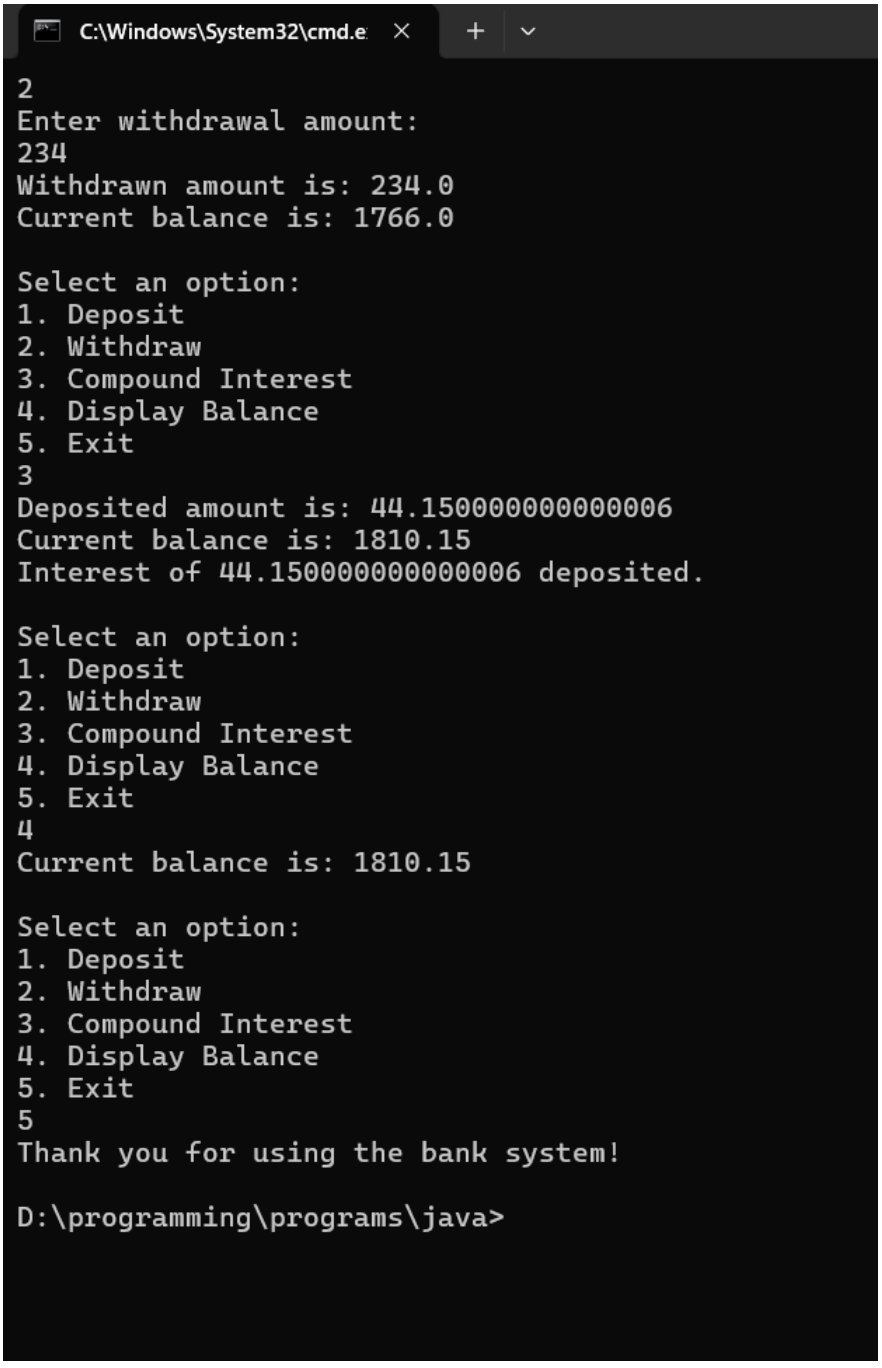
```
PrintInfo.print();

System.out.println("Select Account Type:");
System.out.println("1. Savings Account");
System.out.println("2. Current Account");
int choice = scanner.nextInt();
scanner.nextLine(); // Consume the newline

Account account = null;

if (choice == 1) {
    System.out.println("Enter Customer Name:");
    String name = scanner.nextLine();
    System.out.println("Enter Account Number:");
    String accNum = scanner.nextLine();
    System.out.println("Enter Interest Rate:");
    double rate = scanner.nextDouble();
    account = new SavAcct(name, accNum, rate);
} else if (choice == 2) {
    System.out.println("Enter Customer Name:");
    String name = scanner.nextLine();
    System.out.println("Enter Account Number:");
    String accNum = scanner.nextLine();
    account = new CurAcct(name, accNum);
} else {
    System.out.println("Invalid choice!");
    System.exit(0);
}

int option;
do {
    System.out.println("\nSelect an option:");
    System.out.println("1. Deposit");
    System.out.println("2. Withdraw");
    if (account instanceof SavAcct) {
        System.out.println("3. Compound Interest");
    }
    System.out.println("4. Display Balance");
    System.out.println("5. Exit");
    option = scanner.nextInt();

    switch (option) {
        case 1:
            System.out.println("Enter deposit amount:");
            double depositAmount = scanner.nextDouble();
            account.deposit(depositAmount);
            break;
```

```
                case 2:
                    System.out.println("Enter withdrawal amount:");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;

                case 3:
                    if (account instanceof SavAcct) {
                        ((SavAcct) account).compoundDeposit();
                    } else {
                        System.out.println("Invalid option for Current Account.");
                    }
                    break;

                case 4:
                    account.displayBalance();
                    break;

                case 5:
                    System.out.println("Thank you for using the bank system!");
                    break;

                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (option != 5);

        scanner.close();
    }
}

// Utility class for printing information
class PrintInfo {
    static void print() {
        System.out.println("C Name:Shreya J G \n USN:1BM23IC061\n");
    }
}

// Abstract Account class
abstract class Account {
    String customerName;
    String accountType;
    String accountNumber;
    double balance;

    public Account(String customerName, String accountType, String accountNumber) {
```

```
        this.customerName = customerName;
        this.accountType = accountType;
        this.accountNumber = accountNumber;
        this.balance = 0.0;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposited amount is: " + amount);
        displayBalance();
    }

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

    public abstract void withdraw(double amount);
}

// Savings Account class
class SavAcct extends Account {
    double interestRate;

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

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

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

// Current Account class
class CurAcct extends Account {
```

```
    private static final double minBalance = 1000.0;
    private static final double serviceCharge = 50.0;

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

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

        if (balance < minBalance) {
            balance -= serviceCharge;
            System.out.println("Balance below minimum! Service charge of " + serviceCharge + "
applied.");
        }
        displayBalance();
    }
}
```

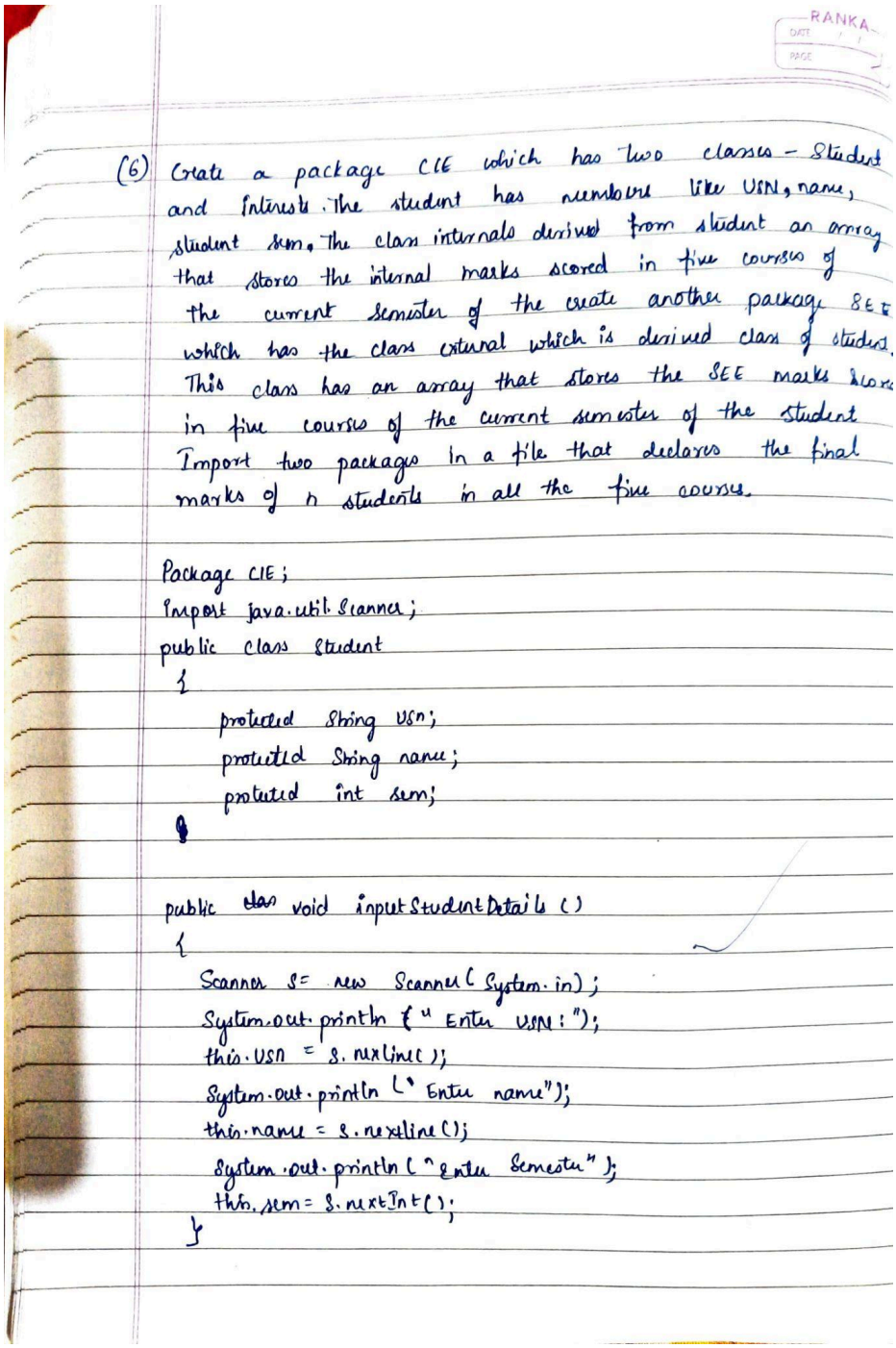

C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.4541]
(c) Microsoft Corporation. All rights reserved.
D:\programming\programs\java>javac Bank.java
D:\programming\programs\java>java Bank.java
C Name:Shreya J G
USN:1BM23IC061
Select Account Type:
1. Savings Account
2. Current Account
1
Enter Customer Name:
Sriya
Enter Account Number:
123445
Enter Interest Rate:
2.5
Select an option:
1. Deposit
2. Withdraw
3. Compound Interest
4. Display Balance
5. Exit
1
Enter deposit amount:
2000
Deposited amount is: 2000.0
Current balance is: 2000.0
Select an option:
1. Deposit
2. Withdraw
3. Compound Interest
4. Display Balance
5. Exit
2
Enter withdrawal amount:
234


C:\Windows\System32\cmd.e
2
Enter withdrawal amount:
234
Withdrawn amount is: 234.0
Current balance is: 1766.0
Select an option:
1. Deposit
2. Withdraw
3. Compound Interest
4. Display Balance
5. Exit
3
Deposited amount is: 44.150000000000006
Current balance is: 1810.15
Interest of 44.150000000000006 deposited.
Select an option:
1. Deposit
2. Withdraw
3. Compound Interest
4. Display Balance
5. Exit
4
Current balance is: 1810.15
Select an option:
1. Deposit
2. Withdraw
3. Compound Interest
4. Display Balance
5. Exit
5
Thank you for using the bank system!
D:\programming\programs\java>

Program 6

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

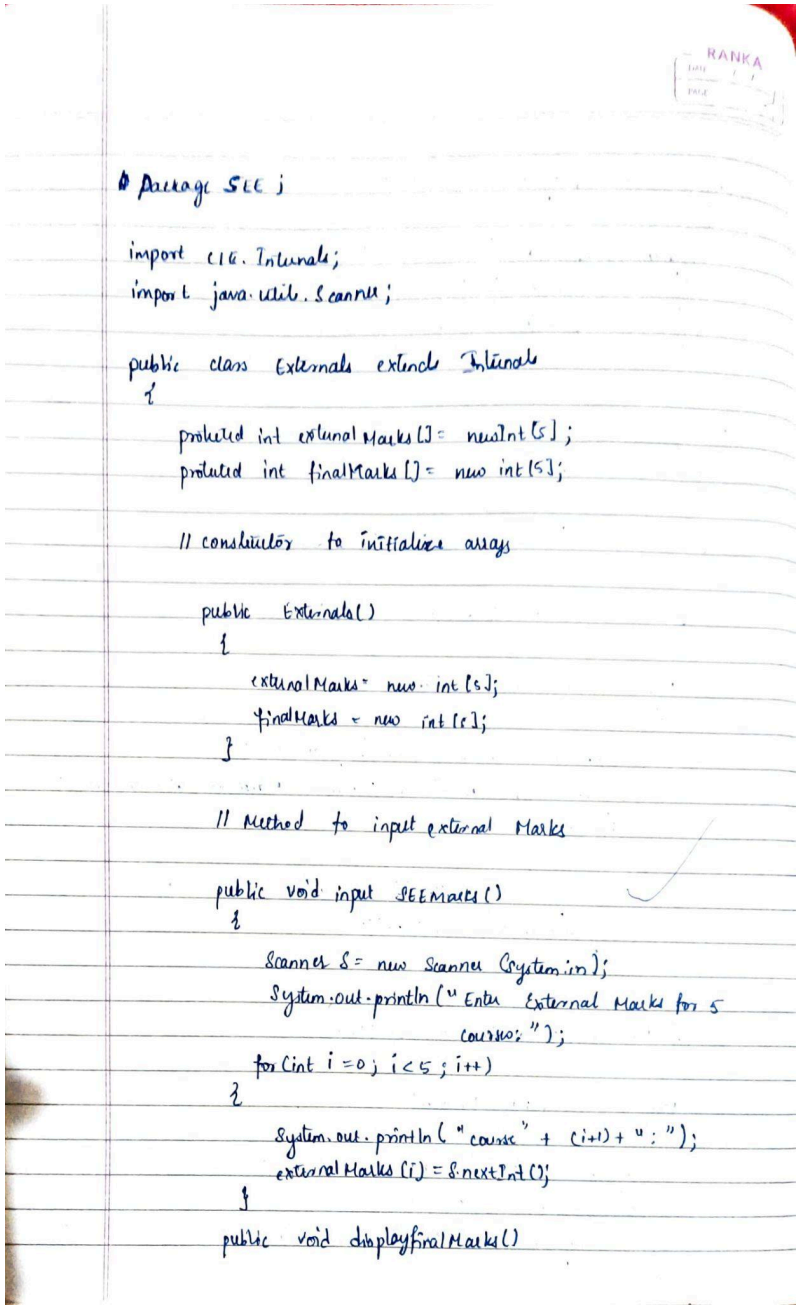
Algorithm:



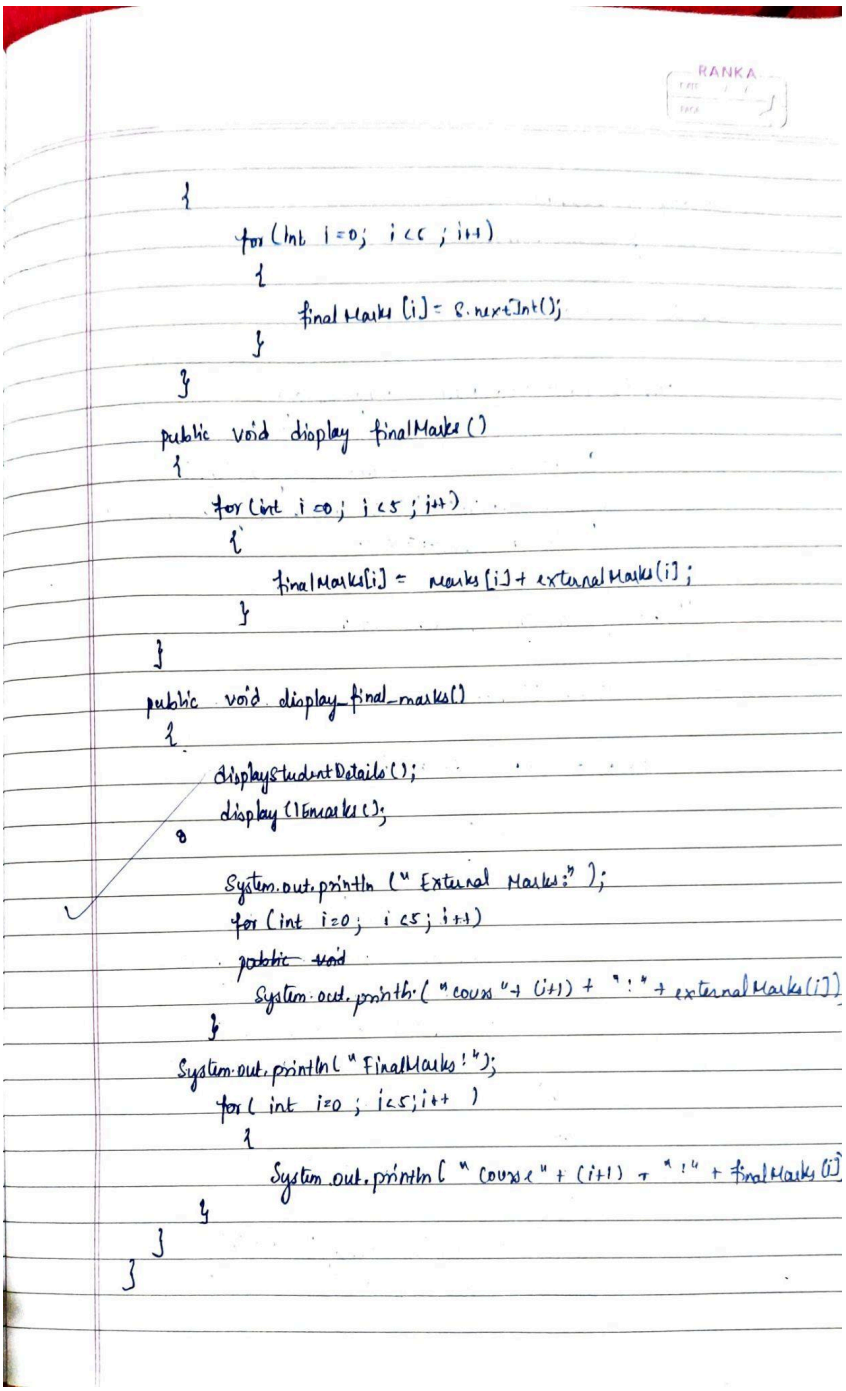
(6) Create a package CIE which has two classes - Student and Internals. The student has members like USN, name, student sem. The class internals derived from student an array that stores the internal marks scored in five courses of the current semester of the create another package SEE which has the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

```
Package CIE;
Import java.util.Scanner;
public class Student
{
    protected String USN;
    protected String name;
    protected int sem;
}

public class void inputStudentDetails()
{
  Scanner S= new Scanner( System.in);
  System.out.println("Enter USN:");
  this.USN = S.nextLine();
  System.out.println("Enter name");
  this.name = S.nextline();
  System.out.println("Enter Semester");
  this.sem = S.nextInt();
}
```

```
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 ~~static void~~ class Internals extends Student
{
    protected int marks[] = new int[5];

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

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



```
package SEE;

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

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

    // constructor to initialize arrays

    public Externals()
    {
        externalMarks = new int[5];
        finalMarks = new int[5];
    }

    // Method to input external Marks

    public void inputSEEMarks()
    {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter External Marks for 5
                            courses:");
        for(int i=0; i<5; i++)
        {
            System.out.println("course" + (i+1) + ":");
            externalMarks[i] = s.nextInt();
        }
    public void displayFinalMarks()
```



```
{
    for(int i=0; i<c; i++)
    {
        finalMarks[i] = s.nextInt();
    }
}
public void display finalMarks()
{
    for(int i=0; i<5; i++)
    {
        finalMarks[i] = marks[i] + externalMarks[i];
    }
}
public void display_final_marks()
{
    displayStudentDetails();
    display(1Emarks();

    System.out.println("External Marks:");
    for(int i=0; i<5; i++)
    public void
        System.out.println("course"+(i+1) + ":" + externalMarks[i]
    }
    System.out.println("FinalMarks!");
    for(int i=0; i<5; i++)
    {
        System.out.println("course" + (i+1) + ":" + finalMarks[i]
    }
}
}
```

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

public class Main
{
    public static void main (String [] args)
    {
        Scanner sc = new Scanner (System.in);
        System.out.println("Enter number of Students:");
          int n=sc.nextInt();

// Array to store Student data

        Externals [] students = new Externals [n];

// Loop to input data for each student

        for (int i =0 ; i<n ; i++)
        {
            students [i]= new Externals();
            students [i] = inputStudentdetails();
            students[i] = input CIE marks();
            students[i]= input SEEmarks();
            students [i] = calculate Final Marks();
        }

        for (int i =0 ; i<n; i++)
        {
            students [i] = display FinalMarks();
            System.out.println("-----");
        }
    }
}
```

Output :

```
Enter number of students : 1
Enter usn : 1BM23IC061
Enter Name : Shreya
Enter Semester : 3
Enter Internal Marks for 5 courses :
course 1 : 34
course 2 : 35
course 3 : 32
course 4 : 37
course 5 : 40
Enter External Marks for 5 courses :
course 1 : 38
course 2 : 39
course 3 : 40
course 4 : 40
course 5 : 30
USN : 1BM23IC061
Name : Shreya
Semester 3
Internal Marks :
Course 1 : 34
course 2 : 35
Course 3 : 32
Course 4 : 37
Course 5 : 40
External Marks :
Course 1 : 38
Course 2 : 39
Course 3 : 40
Course 4 : 40
Course 5 : 30
Final Marks :
Course 1 : 72
Course 2 : 74
course 3 : 72
course 4 : 77
course 5 : 70
```

Code:

```
// File: SEE/Externals.java
package SEE;

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

public class Externals extends Internals {
    protected int externalMarks[] = new int[5];  // Array to store external marks for 5 courses
    protected int finalMarks[] = new int[5];     // Array to store final marks (sum of internal and
external)

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

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

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

    // Method to display final marks
    public void displayFinalMarks() {
        displayStudentDetails();  // Display student details (from parent class)
        displayCIEmarks();        // Display internal marks (from parent class)

        System.out.println("External Marks: ");
        for (int i = 0; i < 5; i++) {
            System.out.println("Course " + (i + 1) + ": " + externalMarks[i]);
        }

        System.out.println("Final Marks: ");
        for (int i = 0; i < 5; i++) {
```

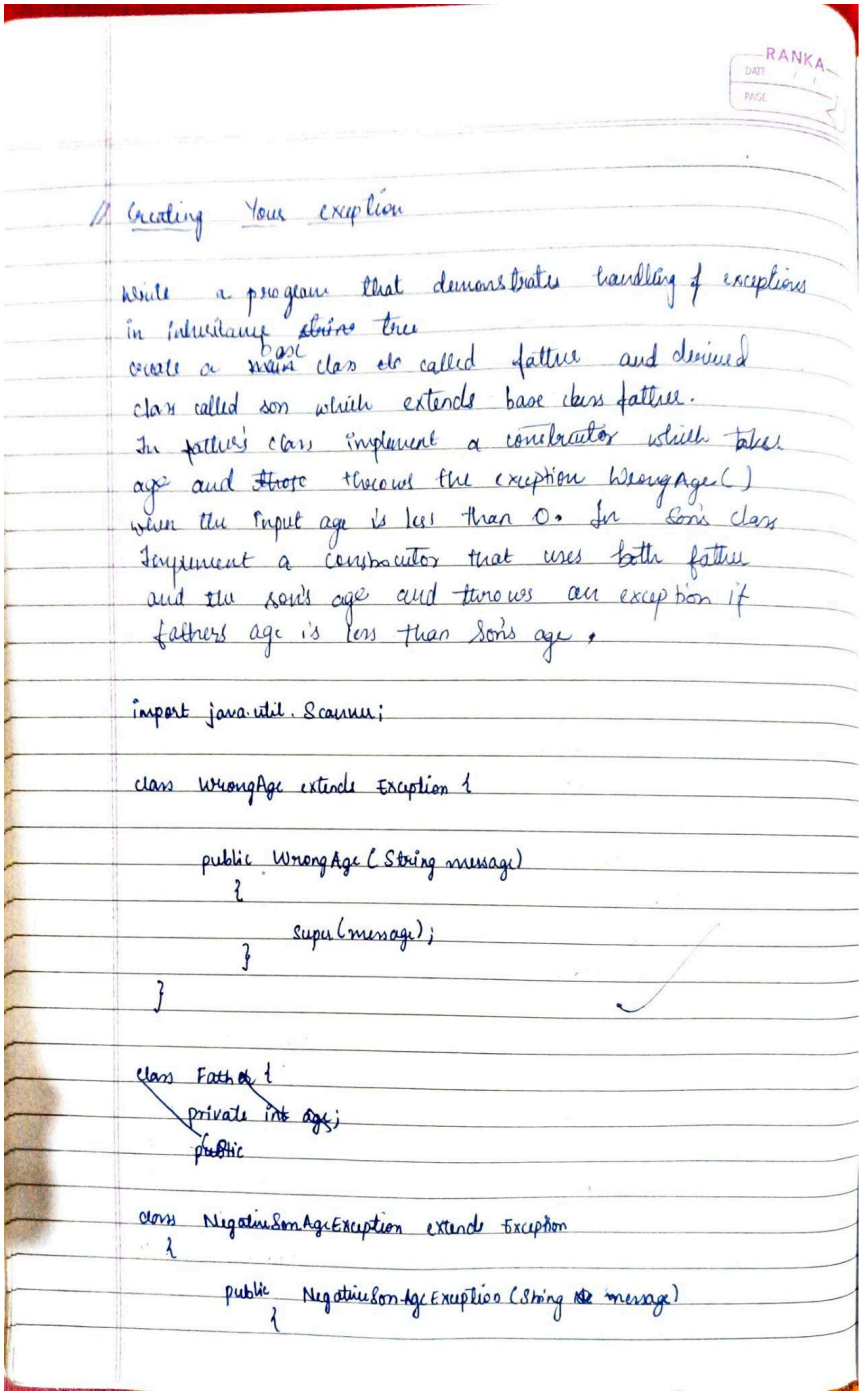
```
            System.out.println("Course " + (i + 1) + ": " + finalMarks[i]);
        }
    }
}
// File: CIE/Student.java
package CIE;

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: ");
        this.usn = s.nextLine();
        System.out.print("Enter Name: ");
        this.name = s.nextLine();
        System.out.print("Enter Semester: ");
        this.sem = s.nextInt();
    }

    // Method to display student details
    public void displayStudentDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Semester: " + sem);
    }
}
// File: Main.java
import SEE.Externals;  // Import Externals class from SEE package

import java.util.Scanner;

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

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

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

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

            // Input student details, internal marks, and external marks
            students[i].inputStudentDetails();
            students[i].inputCIEmarks();
            students[i].inputSEEmarks();
            students[i].calculateFinalMarks();
        }

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

```
C:\Users\Admin\Documents\student>javac -d . CIE/Internals.java

C:\Users\Admin\Documents\student>javac -d . CIE/Student.java

C:\Users\Admin\Documents\student>javac -d . SEE/Externals.java

C:\Users\Admin\Documents\student>javac Main.java

C:\Users\Admin\Documents\student>java Main
Enter number of students: 1
Enter USN: 1286759
Enter Name: SIRI
Enter Semester: 3
Enter Internal Marks for 5 courses:
Course 1: 34
Course 2: 35
Course 3: 32
Course 4: 37
Course 5: 40
Enter External Marks for 5 courses:
Course 1: 38
Course 2: 39
Course 3: 40
Course 4: 40
Course 5: 30
USN: 1286759
Name: SIRI
Semester: 3
Internal Marks:
Course 1: 34
Course 2: 35
Course 3: 32
Course 4: 37
Course 5: 40
External Marks:
Course 1: 38
Course 2: 39
Course 3: 40
Course 4: 40
Course 5: 30
Final Marks:
Course 1: 72
Course 2: 74
Course 3: 72
Course 4: 77
Course 5: 70
```

Program 7

Algorithm:

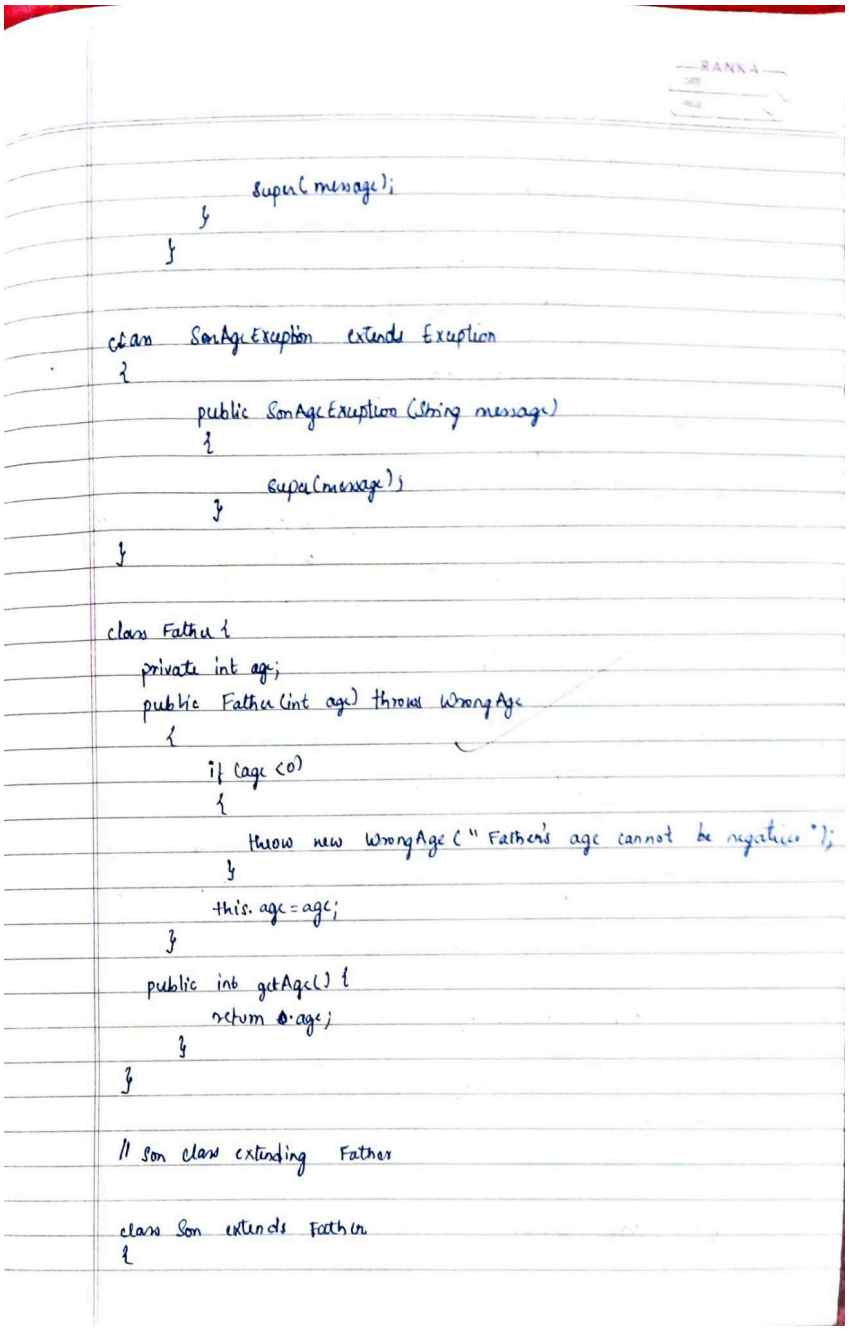


// Creating Your exception

Write a program that demonstrates handling of exceptions in inheritance trees
create a base class as called father and derived class called son which extends base class father.
In father's class implement a constructor which takes age and throws the exception WrongAge() when the input age is less than 0. In Son's class implement a constructor that uses both father and the son's age and throws an exception if fathers age is less than Son's age.

```
import java.util.Scanner;

class WrongAge extends Exception {

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

class Father {
    private int age;
    public

class NegativeSonAgeException extends Exception
{
    public NegativeSonAgeException(String message)
    {
```

```
private int SonAge;

public Son (int fatherAge, int SonAge) throws WrongAge,
        SonAgeException, NegativeSonAgeException
    {
        super(fatherAge);

        if(SonAge<0)
          {
            throw new NegativeSonAgeException("Son's Age
                                    cannot be negative");
          }

        // check if son's Age is greater than or equal to
        father's age

        if(sonAge>=fatherAge)
          {
            throw new SonAgeException("Son's age cannot
                    be greater than or equal to father's age");
          }
        this.sonAge = sonAge;
    }
    public int getSonAge()
      {
        return sonAge;
      }
}

public class FatherSon
  {
```



```
        super(message);
    }
}

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

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

// Son class extending Father

class Son extends Father
{
```

```
public static void main (String [] args)
{
    Scanner sc = new Scanner (System.in);
    while (true)
    {
        System.out.println (" Enter father's age:");
        int fatherAge = sc.nextInt();
        System.out.println (" Enter son's age");
        int sonAge = sc.nextInt();

        try {
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Accepted Successfully");
        } catch (WrongAge e) {

            System.out.println (" " + e.getMessage());
        } catch (WrongAge e)
        {
            System.out.println(e.getMessage());
        } catch (NegativeSonAgeException e)
        {
            System.out.println (" e.getMessage()");
        }
    // Ask if the user wants to re-enter details
        System.out.println (" Would you like to enter the
                                details (Y/N)?");
        String input = sc.next();
        if (input.equalsIgnoreCase ("n"))
        {
            break;
        }
        sc.close();
    }
}
```

Output:

```
Enter Father's Age : 40
Enter Son's Age : -10
Son's Age cannot be negative
Would you like to re-enter details (Y/N)?
Y
Enter Father's Age :- 35
Enter Son's Age : 20
Father's age cannot be negative
Would you like to re-enter details (Y/N)?
Y
Enter Father's Age : 40
Enter Son's Age : 40
Son's age cannot be greater than or equal to Father's age
Would you like to re-enter the details (Y/N)?
Y
Enter Father's Age : 40
Enter Son's Age : 20
Accepted Successfully
Would you like to re-enter details (Y/N)?
n
```

Code:

```
import java.util.Scanner;


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


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


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


class Father {
    private int age;


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

    public int getAge() {
        return age;
    }
}


class Son extends Father {
    private int sonAge;


    public Son(int fatherAge, int sonAge) throws WrongAge, SonAgeException,
```

```
NegativeSonAgeException {
        super(fatherAge);


        if (sonAge < 0) {
            throw new NegativeSonAgeException("Son's age cannot be negative");
        }


        if (sonAge >= fatherAge) {
            throw new SonAgeException("Son's age cannot be greater than or equal to father's age");
        }

        this.sonAge = sonAge;
    }

    public int getSonAge() {
        return sonAge;
    }
}

public class FatherSon {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (true) {
            System.out.print("Enter Father's Age: ");
            int fatherAge = sc.nextInt();
            System.out.print("Enter Son's Age: ");
            int sonAge = sc.nextInt();

            try {
                Son son = new Son(fatherAge, sonAge);  // Create Son object
                System.out.println("Accepted Successfully");
            } catch (WrongAge e) {
                System.out.println(e.getMessage());
            } catch (SonAgeException e) {
                System.out.println(e.getMessage());
            } catch (NegativeSonAgeException e) {
                System.out.println(e.getMessage());
            }

            // Ask if the user wants to re-enter details
            System.out.println("Would you like to re-enter details (Y/n)?");
            String input = sc.next();
            if (input.equalsIgnoreCase("n")) {
                break;
            }
```

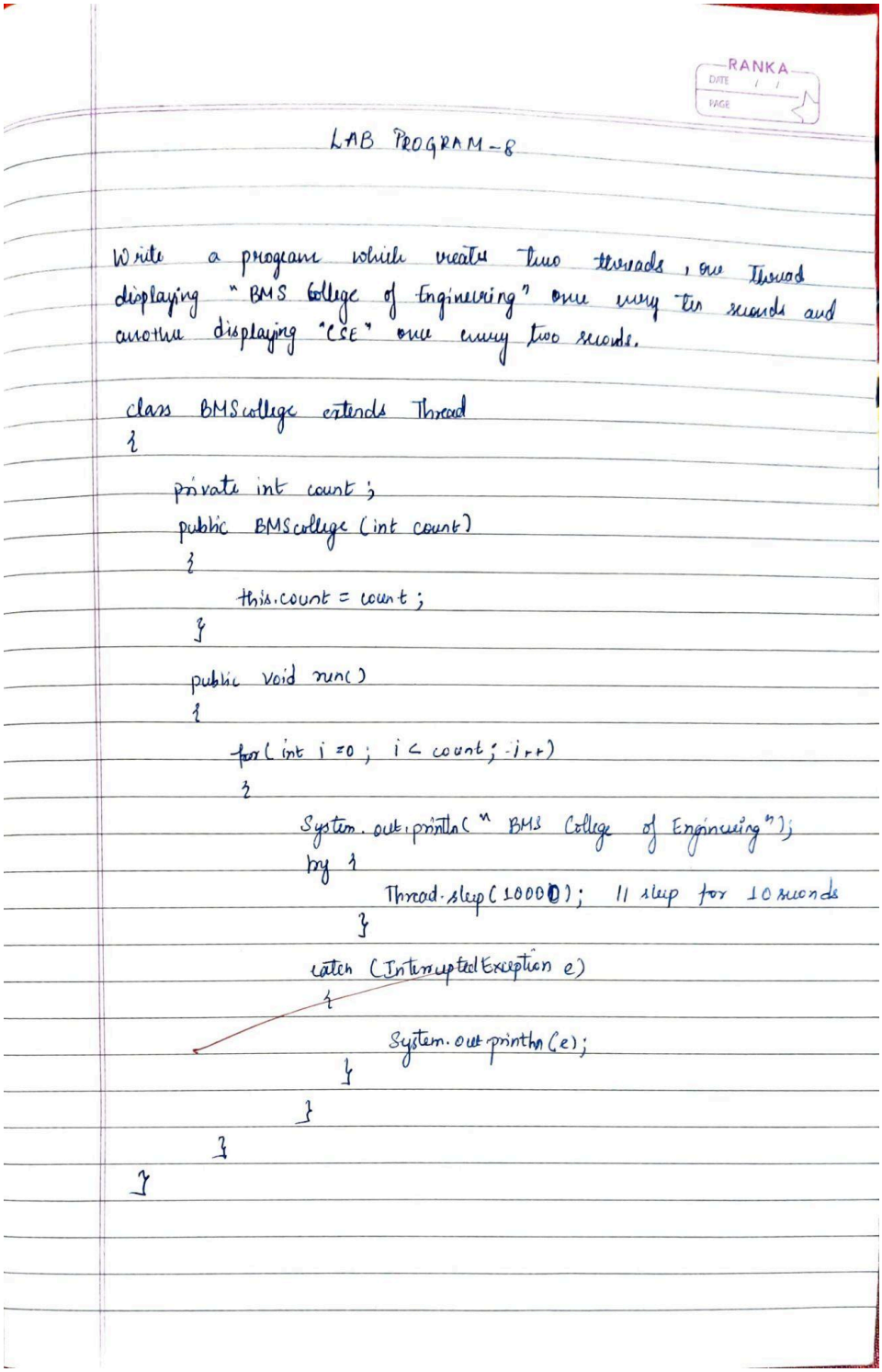
```
        }
        sc.close();
    }
}
```

```
D:\programming\programs\java>javac FatherSon.java

D:\programming\programs\java>java FatherSon.java
Enter Father's Age: 50
Enter Son's Age: -30
Son's age cannot be negative
Would you like to re-enter details (Y/n)?
y
Enter Father's Age: -84
Enter Son's Age:
40
Father's age cannot be negative
Would you like to re-enter details (Y/n)?
y
Enter Father's Age: 50
Enter Son's Age: 50
Son's age cannot be greater than or equal to father's age
Would you like to re-enter details (Y/n)?
y
Enter Father's Age: 40
Enter Son's Age: 20
Accepted Successfully
Would you like to re-enter details (Y/n)?
n

D:\programming\programs\java>
```

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 very two seconds.

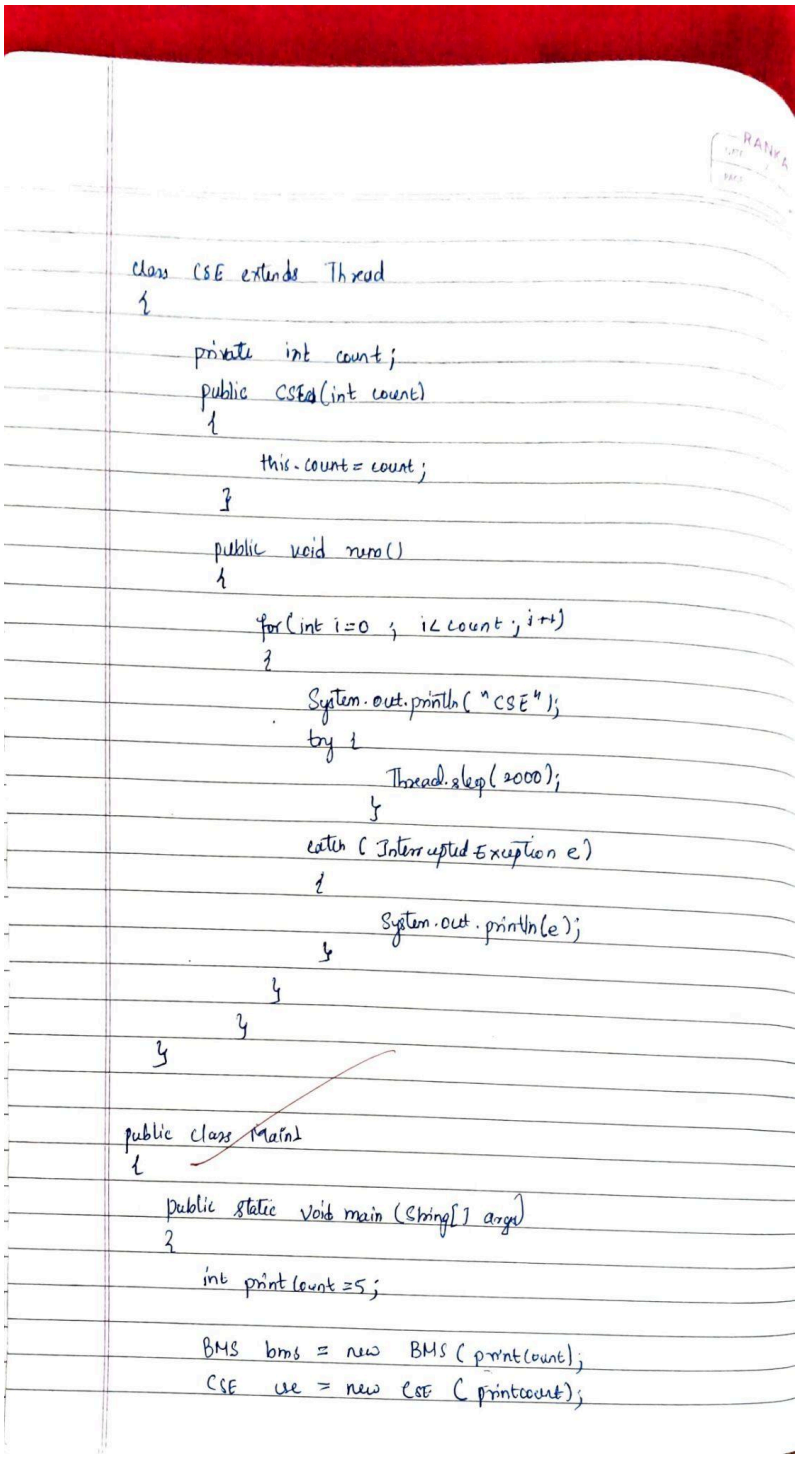
Algorithm:



LAB PROGRAM-8

Write a program which create two threads, one Thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

```
class BMScollege extends Thread
{
    private int count;
    public BMScollege (int count)
    {
        this.count = count;
    }
    public void run()
    {
        for(int i=0; i<count; i++)
        {
            System.out.println(" BMS College of Engineering");
            try {
                Thread.sleep(10000); // sleep for 10 seconds
            }
            catch (InterruptedException e)
            {
                System.out.println(e);
            }
        }
    }
}
```



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

public class Main1
{
    public static void main (String[] args)
    {
        int printcount=5;

        BMS bms = new BMS (printcount);
        CSE cse = new CSE (printcount);
```

```
bms.start();
cse.start();

try {
    bms.join();
    cse.join();
  }
catch (InterruptedException e)
  {
    System.out.println("Both have been completed");
  }
}
```

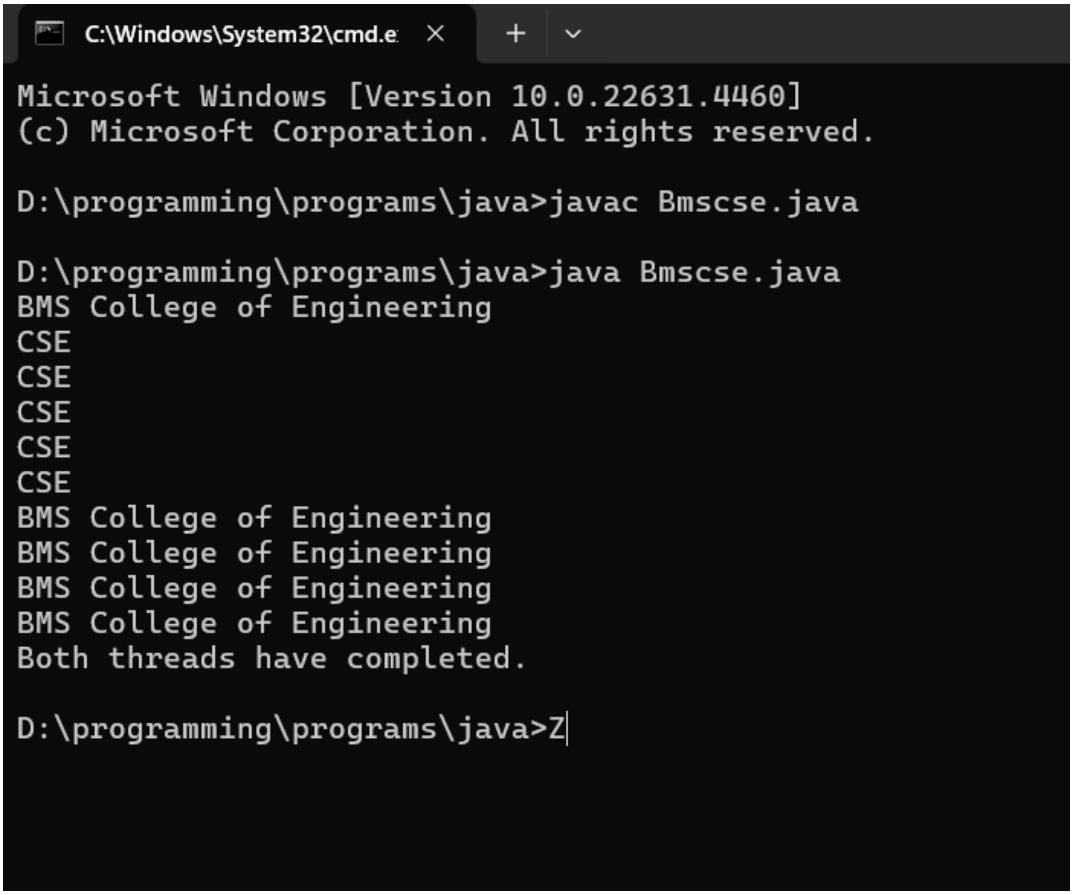
Output :

```
BMS college of Engineering
ECE
ECE
CSE
CSE
CSE
BMS College of Engineering
BMS college of Engineering
BMS college of Engineering
BMS college of Engineering
Both threads have completed.
```

Code:

```
class BMSThread extends Thread {
    private int count;

    public BMSThread(int count) {
        this.count = count;
    }

    @Override
    public void run() {
        for (int i = 0; i < count; i++) {
            System.out.println("BMS College of Engineering");
            try {
                Thread.sleep(10000); // Sleep for 10 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

class CSEThread extends Thread {
    private int count;

    public CSEThread(int count) {
        this.count = count;
    }

    public void run() {
        for (int i = 0; i < count; i++) {
            System.out.println("CSE");
            try {
                Thread.sleep(2000); // Sleep for 2 seconds
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}
public class Bmscse {
    public static void main(String[] args) {
        int printCount = 5; // Number of times each message should be printed

        // Create two threads
        BMSThread bmsThread = new BMSThread(printCount);
```

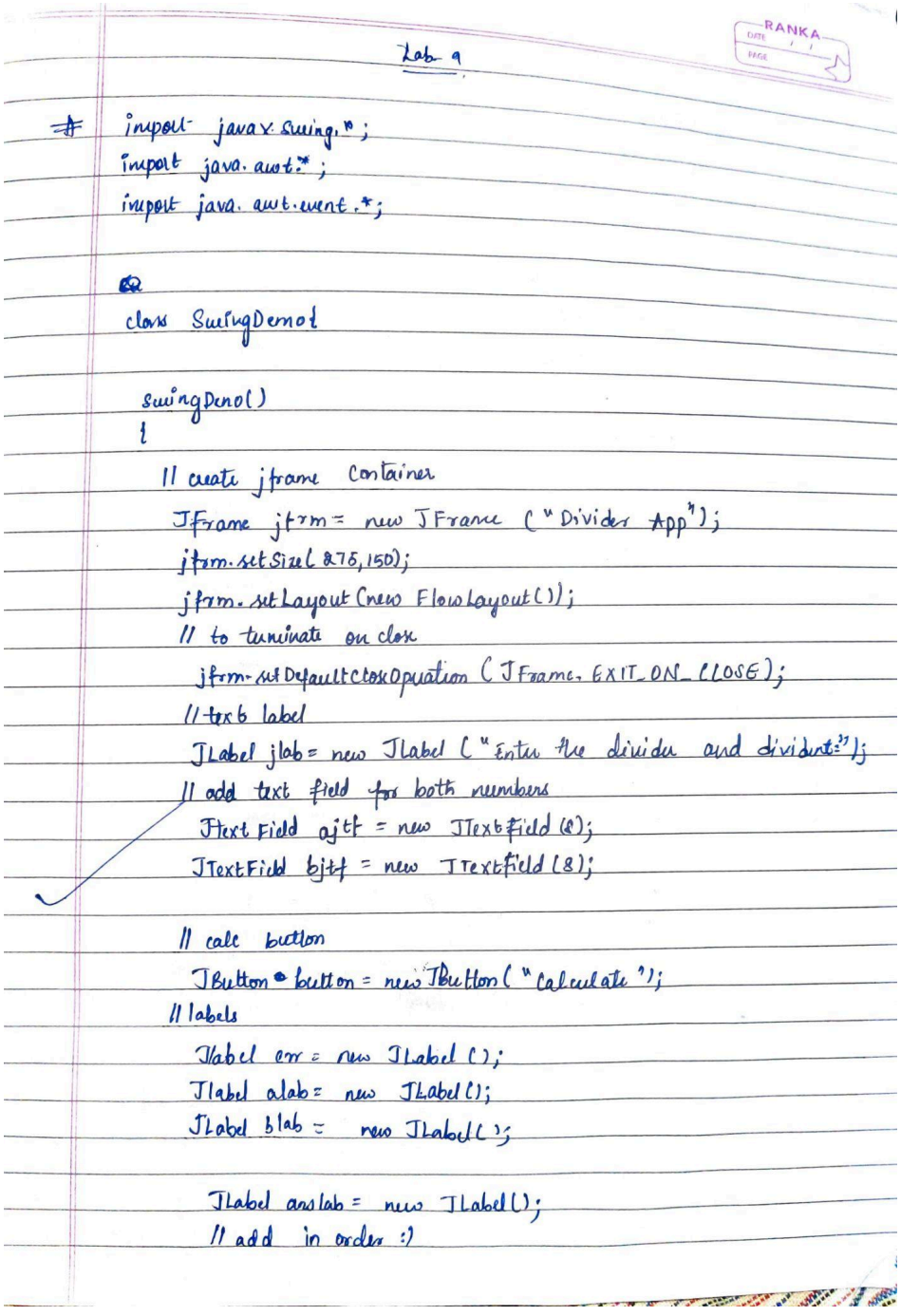
```
        CSEThread cseThread = new CSEThread(printCount);

        // Start both threads
        bmsThread.start();
        cseThread.start();

        // Wait for both threads to finish
        try {
            bmsThread.join();
            cseThread.join();
        } catch (InterruptedException e) {
            System.out.println(e);
        }

        // Print completion message
        System.out.println("Both threads have completed.");
    }
}
```



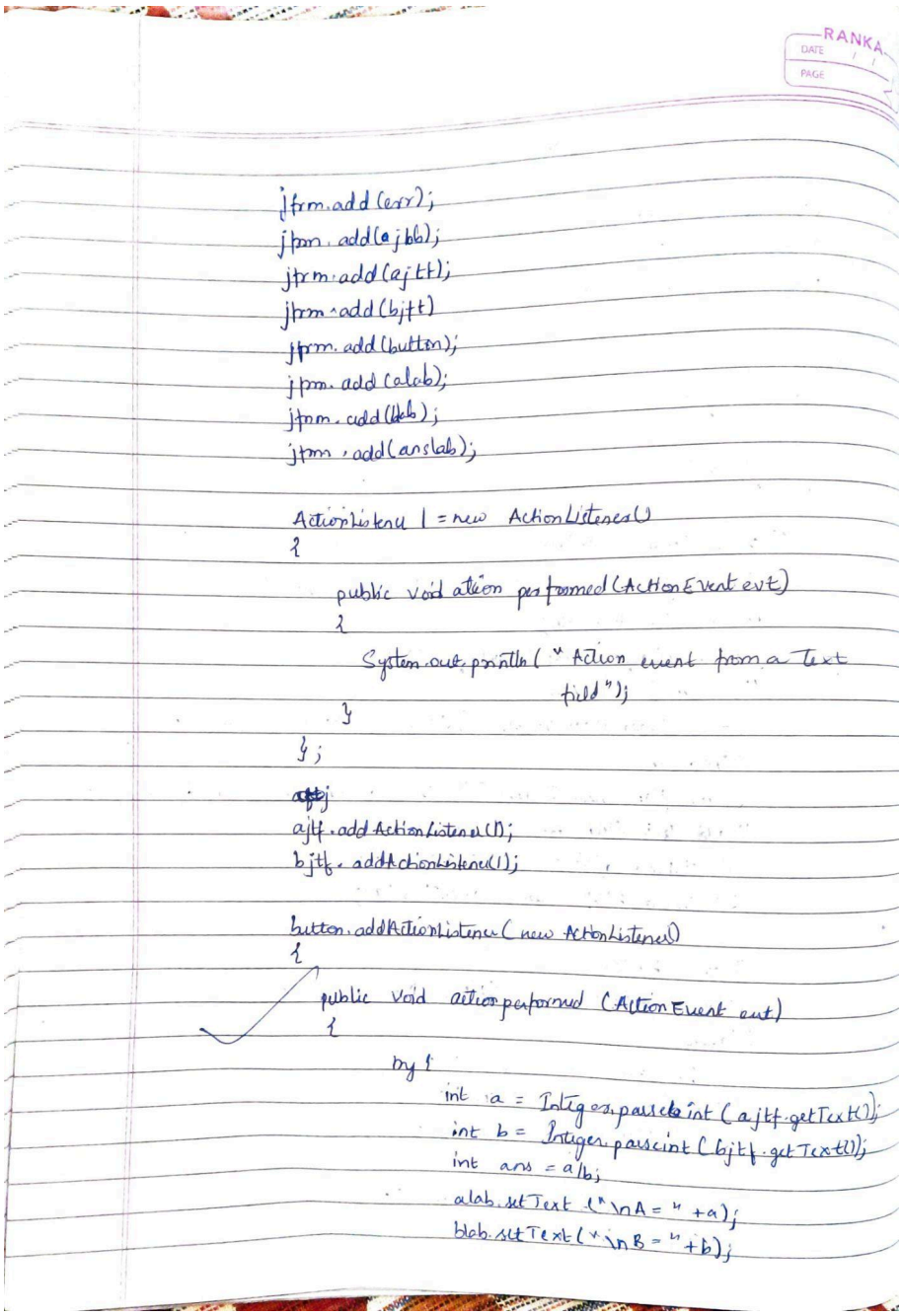
Program 9

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.

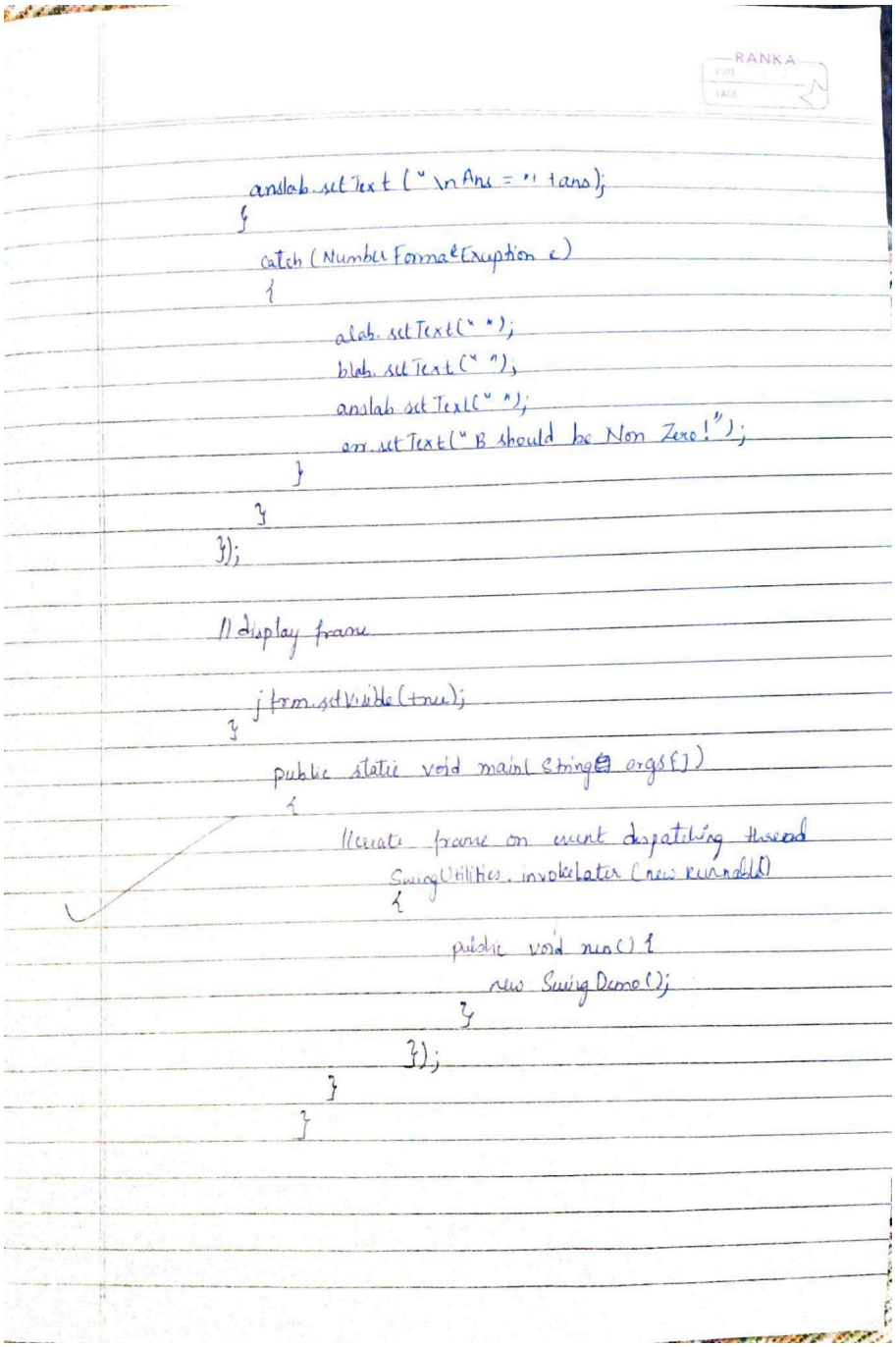
Algorithm:



```
Lab 9

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

class SwingDemo1

SwingDemo()
{
  // create jframe Container
  JFrame jfrm = new JFrame("Divider App");
  jfrm.setSize(275,150);
  jfrm.setLayout(new FlowLayout());
  // to terminate on close
   jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  // text label
  JLabel jlab = new JLabel("Enter the divider and dividant:");
  // add text field for both numbers
   JTextField ajtf = new JTextField(8);
  JTextField bjtf = new JTextField(8);

  // calc button
  JButton button = new JButton("Calculate");
 // labels
  JLabel err = new JLabel();
  JLabel alab = new JLabel();
  JLabel blab = new JLabel();

   JLabel anslab = new JLabel();
   // add in order :)
```



```
jfrm.add(err);
jfrm.add(ajbb);
jfrm.add(ajtt);
jfrm.add(bjtt)
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener l = new ActionListener()
{
    public void actionperformed(ActionEvent evt)
    {
        System.out.println("Action event from a Text
                            field");
    }
};

ajtf.addActionListener(l);
bjtf.addActionListener(l);

button.addActionListener(new ActionListener()
{
    public void actionperformed(ActionEvent evt)
    {
        try{
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());
            int ans = a/b;
            alab.setText("inA=" +a);
            blab.setText("inB=" +b);
```



```
anslab.setText("\n Ans = " + ans);
}
catch (NumberFormatException e)
{
    alab.setText(" ");
    blab.setText(" ");
    anslab.setText(" ");
    err.setText("B should be Non Zero!");
  }
 }
});

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

Code:

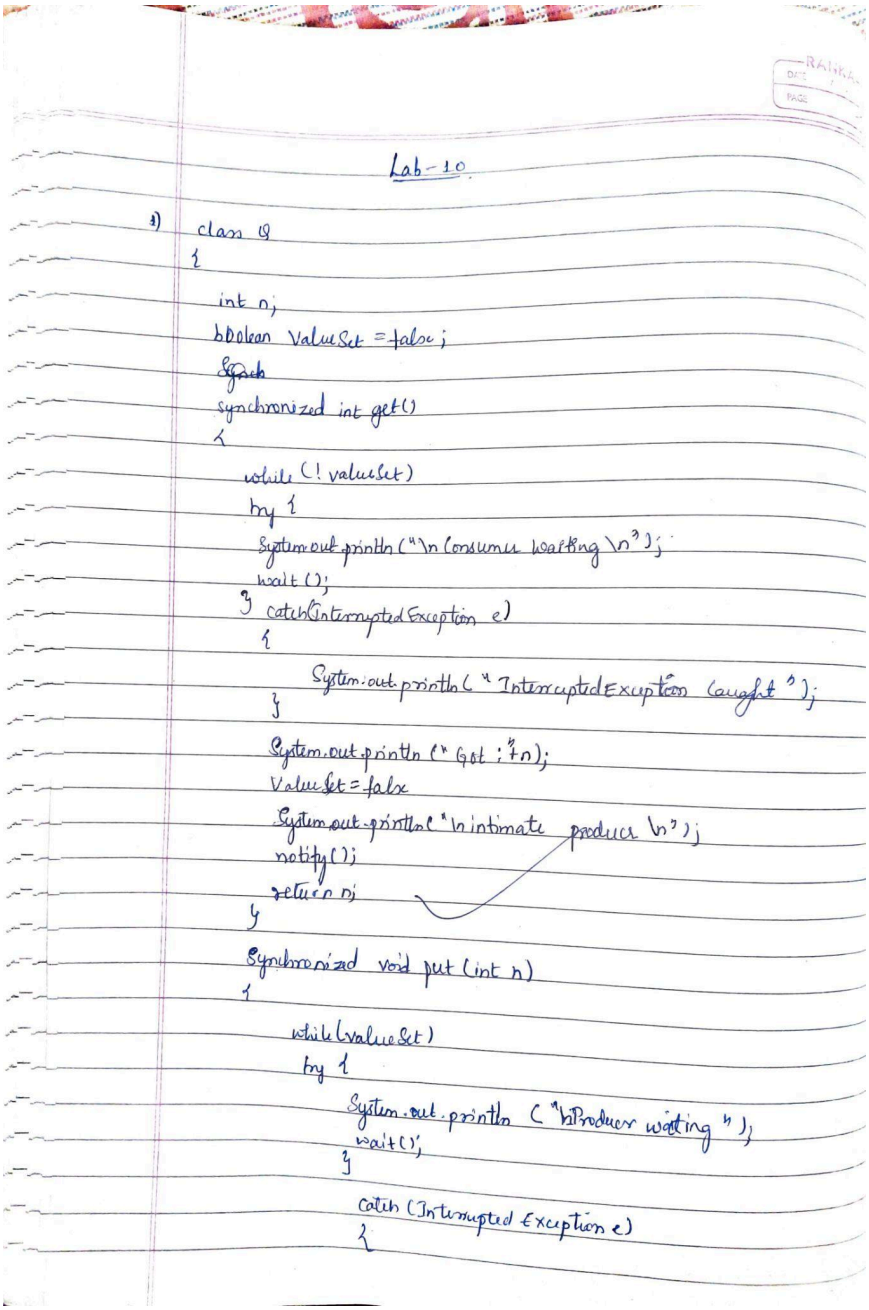
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo{
SwingDemo(){
// create jframe container
JFrame jfrm = new JFrame("Divider App");
jfrm.setSize(275, 150);
jfrm.setLayout(new FlowLayout());
// to terminate on close
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// text label
JLabel jlab = new JLabel("Enter the divider and divident:");
// add text field for both numbers
JTextField ajtf = new JTextField(8);
JTextField bjtf = new JTextField(8);
// calc button
JButton button = new JButton("Calculate");
// labels
JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();

JLabel anslab = new JLabel();
// add in order :)
jfrm.add(err); // to display error bois
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);
ActionListener l = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
System.out.println("Action event from a text field");
}
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
try{
int a = Integer.parseInt(ajtf.getText());
int b = Integer.parseInt(bjtf.getText());
int ans = a/b;
```

```
alab.setText("\nA = " + a);
blab.setText("\nB = " + b);
anslab.setText("\nAns = "+ ans);
}
catch(NumberFormatException e){
alab.setText("");
blab.setText("");
anslab.setText("");

err.setText("Enter Only Integers!");
}
catch(ArithmeticException e){
alab.setText("");
blab.setText("");
anslab.setText("");
err.setText("B should be NON zero!");
}
}
});
// display frame
jfrm.setVisible(true);
}
public static void main(String args[]){
// create frame on event dispatching thread
SwingUtilities.invokeLater(new Runnable(){
public void run(){
new SwingDemo();
}
});
}
}
```

Program 10

Demonstrate Inter process Communication and deadlock

Algorithm:



Lab - 10

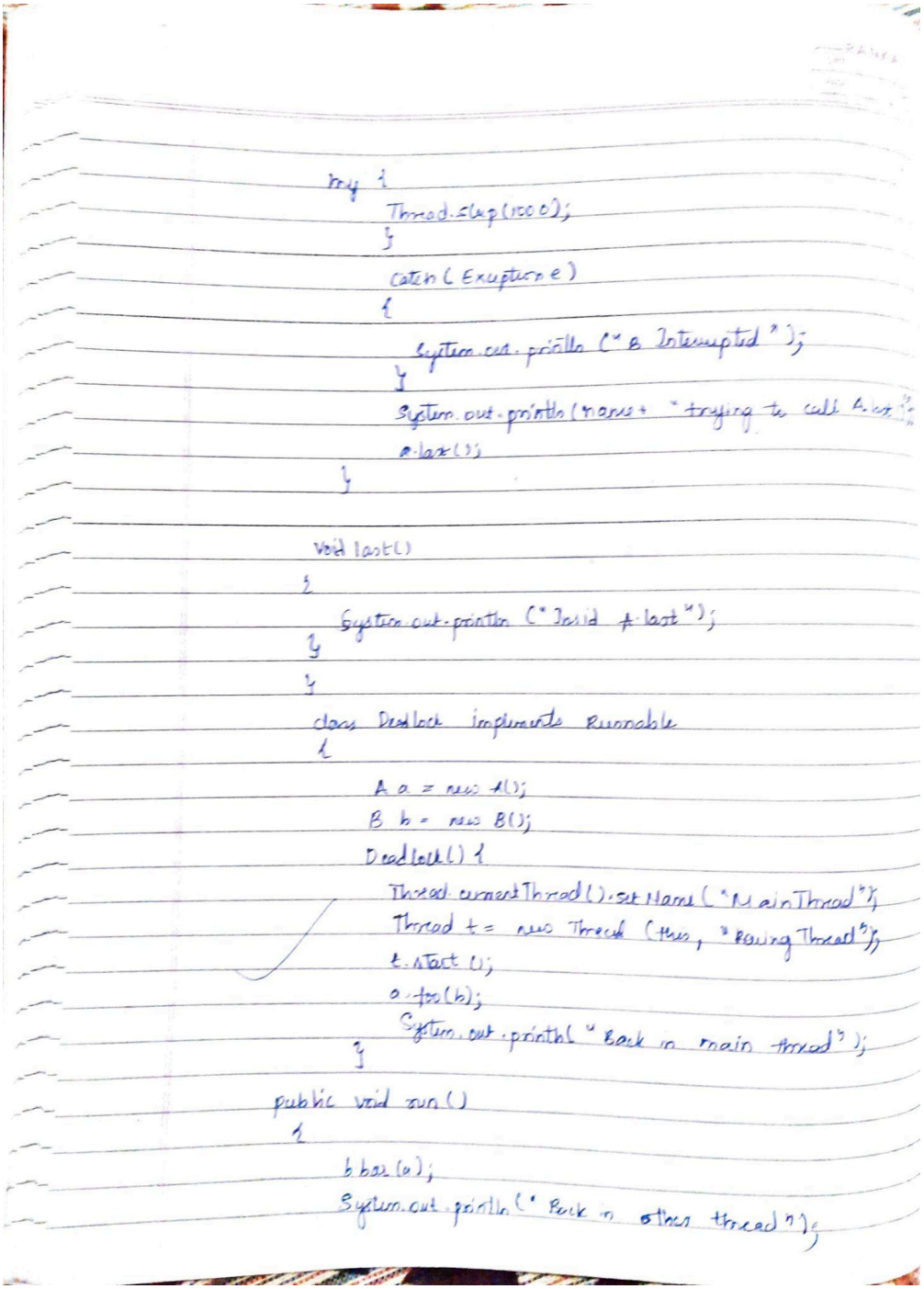
1) class Q
{
int n;
boolean ValueSet = false;
synchronized int get()
{
while (!valueSet)
try {
System.out.println("\n Consumer waiting \n");
wait();
} catch(InterruptedException e)
{
System.out.println("InterruptedException caught");
}
System.out.println("Got : "+n);
ValueSet = false
System.out.println("\n intimate producer \n");
notify();
return n;
}
Synchronized void put(int n)
{
while(valueSet)
try {
System.out.println("\nProducer waiting");
wait();
}
catch (InterruptedException e)
{

```
        System.out.println(" Interrupted Exception caught ");
    }
    this.n=n;
    ValueSet =true;
    System.out.println (" Put : "+n);
    System.out.println (" \n Intimate Customer \n");
    notify();
  }
}
class prod Producer implements Runnable
{
    Q q;
    Producer (Q q) {
      this.q =q;
      newThread (this, "Producer ").start();
    }
    public void run()
    {
      int i=0;
      while (i<15)
      {
        q.put(i++);
      }
    }
}
class Consumer implements Runnable
{
    Q q;
    Consumer (Q q)
    {
      this.q = q;
```

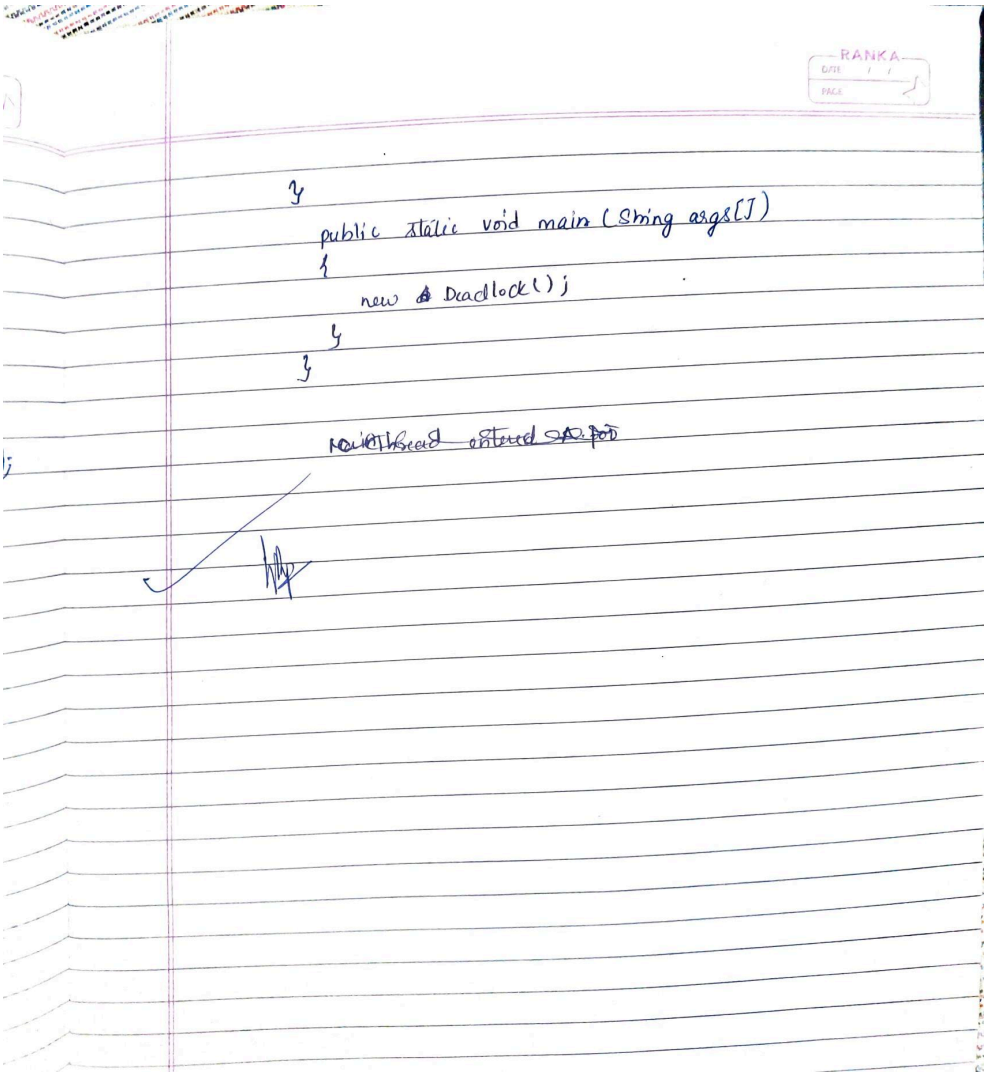
```
        new Thread(this, "Consumer").start();
    }

    public void run()
    {
        int i=0;
        while(i<15)
        {
            int r=q.get();
            System.out.println("Consumed:"+r);
            i++;
        }
    }
}

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

=# 2) Deadlock :

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

class B
{
    Synch
    synchronized void bar(A a)
    {
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");
```



```
try {
    Thread.sleep(1000);
    }
    catch (Exception e)
    {
      System.out.println("B Interrupted");
    }
    System.out.println(name + "trying to call A.last()");
    a.last();
  }

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

class Deadlock implements Runnable
{
    A a = new A();
    B b = new B();
    Deadlock() {
      Thread.currentThread().setName("MainThread");
      Thread t = new Thread(this, "RacingThread");
      t.start();
      a.foo(b);
      System.out.println("Back in main thread");
    }

public void run()
{
    b.bar(a);
    System.out.println("Back in other thread");
```


RANKA
DATE / /
PAGE
}
public static void main (String args[])
{
new Deadlock();
}
}

Code:
1.Interface:

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

    // Synchronized method to get the value
    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("\nConsumer waiting...");
                wait();  // Consumer waits until a value is produced
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }

        // Consume the product
        System.out.println("Got: " + n);
        valueSet = false;

        // Notify the producer that consumer has consumed the product
        System.out.println("\nIntimate Producer...");
        notify();
        return n;
    }

    // Synchronized method to put the value
    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting...");
                wait();  // Producer waits until the consumer consumes the product
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }

        // Produce a product
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);

        // Notify the consumer that the producer has put a new product
        System.out.println("\nIntimate Consumer...");
        notify();
```

```
    }
}

class Producer implements Runnable {
    Q q;

    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();  // Create and start the producer thread
    }

    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++);  // Producer puts a value into the queue
        }
    }
}

class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();  // Create and start the consumer thread
    }

    public void run() {
        int i = 0;
        while (i < 15) {
            int r = q.get();  // Consumer gets a value from the queue
            System.out.println("Consumed: " + r);
            i++;
        }
    }
}

class PCFixed {
    public static void main(String args[]) {
        Q q = new Q();  // Shared queue between producer and consumer

        new Producer(q);  // Start the producer thread
        new Consumer(q);  // Start the consumer thread

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

2.Deadlock:

```
class A {
   synchronized void foo(B b) {
      String name = Thread.currentThread().getName();
      System.out.println(name + " entered A.foo");

      try {
         Thread.sleep(1000); // Simulate some work
      } catch (Exception e) {
         System.out.println("A Interrupted");
      }

      System.out.println(name + " trying to call B.last()");
      b.last(); // Call method in B
   }

   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); // Simulate some work
      } catch (Exception e) {
         System.out.println("B Interrupted");
      }

      System.out.println(name + " trying to call A.last()");
      a.last(); // Call method in A
   }

   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(); // Start the second thread

        a.foo(b); // Main thread locks A and tries to call B's last() method
        System.out.println("Back in main thread");
    }

    public void run() {
        b.bar(a); // Other thread locks B and tries to call A's last() method
        System.out.println("Back in other thread");
    }

    public static void main(String[] args) {
        new Deadlock(); // Start the deadlock simulation
    }
}
```