VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Ruqaiyya Mahreen (1BM23EE044)

in partial fulfilment 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 **Ruqaiyya Mahreen (1BM23EE044),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object-Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

Lab Faculty In Charge: Dr Prasad GR Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	23/10/2024	Implement Quadratic Equation	4
2	23/10/2024	Student SGPA Calculation	11
3	30/10/2024	Book Details	18
4	30/10/2024	Shape Area Calculation	23
5	13/11/2024	Bank Account Management	29
6	13/11/2024	CIE and SEE Marks	41
7	20/11/2024	Exception Handling	51
8	27/11/2024	Thread based Message display	57
9	27/11/2024	Integer Division User Interface	61
10	27/11/2024	Interprocess Communication and Deadlock	67

GitHub Link:
https://github.com/RuqaiyyaMahreen/JavaLab

Program 1
Implement Quadratic Equation

Algorithm:

INDEX

NAME: RUQAIYYA M STD.: ____ SEC.: ____ ROLL NO.: ____ SUB.: JAVA OBS

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	23/10/24	Week 1 - Quadratic	5	
2	23/10/24	Week 2 - SGPA	8	
3	30/10/24	Week 3 - Book	13	
4	30/10/24	Week 4 - Area	16	
5	13/11/24	Week 5 - Bank	20	
6	13/11/24	Week 6 - CIE and SEE	27	
7	20/11/24	Week 7 - Exceptions	33	- 27/11/24
8	27/11/24	Week 8 - Threads	37	
9	27/11/24	Week 9 - Interface	39	
10	27/11/24	Week 10 - IPC and deadlock	43	

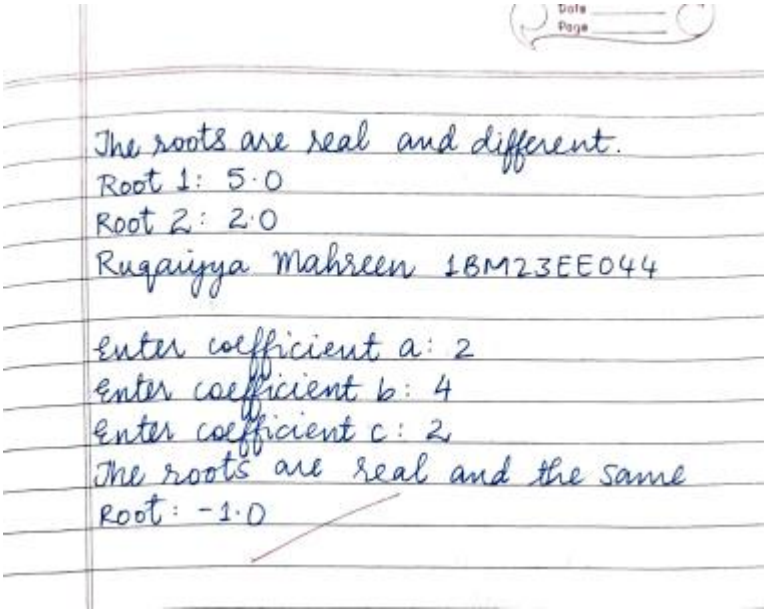
23/10/24 WEEK 1

```
import java.util.Scanner;
import java.lang.Math;
class q
{
 public static void main (String args[]){
  Scanner in = new Scanner (System.in);
  System.out.println ("Enter coefficient a:");
  double a = in.nextDouble();
  System.out.println ("Enter coefficient b:")
  double b = in.nextDouble();
  System.out.println ("Enter coefficient c:");
  double c = in.nextDouble();
  double d = b*b - 4*a*c;
  if (d>0){
    double r1 = (-b+Math.sqrt(d))/(2*a);
    double r2 = (-b-Math.sqrt(d))/(2*a);
    System.out.println ("The roots are
real and different.");
    System.out.println ("Root 1: "+r1);
    System.out.println ("Root 2: "+r2);
  }
  else if (d==0){
    double r = -b/(2*a);
    System.out.println ("The roots are
real and the same");
    System.out.println ("Root: "+r);
  }
```

```
else {
  System.out.println ("The roots are
  complex");
double realpart = -b/(2*a);
double imaginarypart = Math.sqrt(-d)/
                       (2*a);
System.out.println ("Root 1: " +realpart+"
                    imaginarypart + "i");
System.out.println ("Root 2: " +realpart +"
                    + imaginarypart + "i");
}
  Scanner.close();
  System.out.println("Rugaiyya Mahreen
  1BM23EE044");
}
}
```

Output:

```
Enter coefficient a : 2
Enter coefficient b : 4
Enter coefficient c : 8
The roots are complex
Root 1: -1.0+1.73205080i
Root 2: -1.0+1.73205080i
Rugaiyya Mahreen 1BM23EE044

Enter coefficient a: 1
Enter coefficient b: -7
Enter coefficient c: 10
```

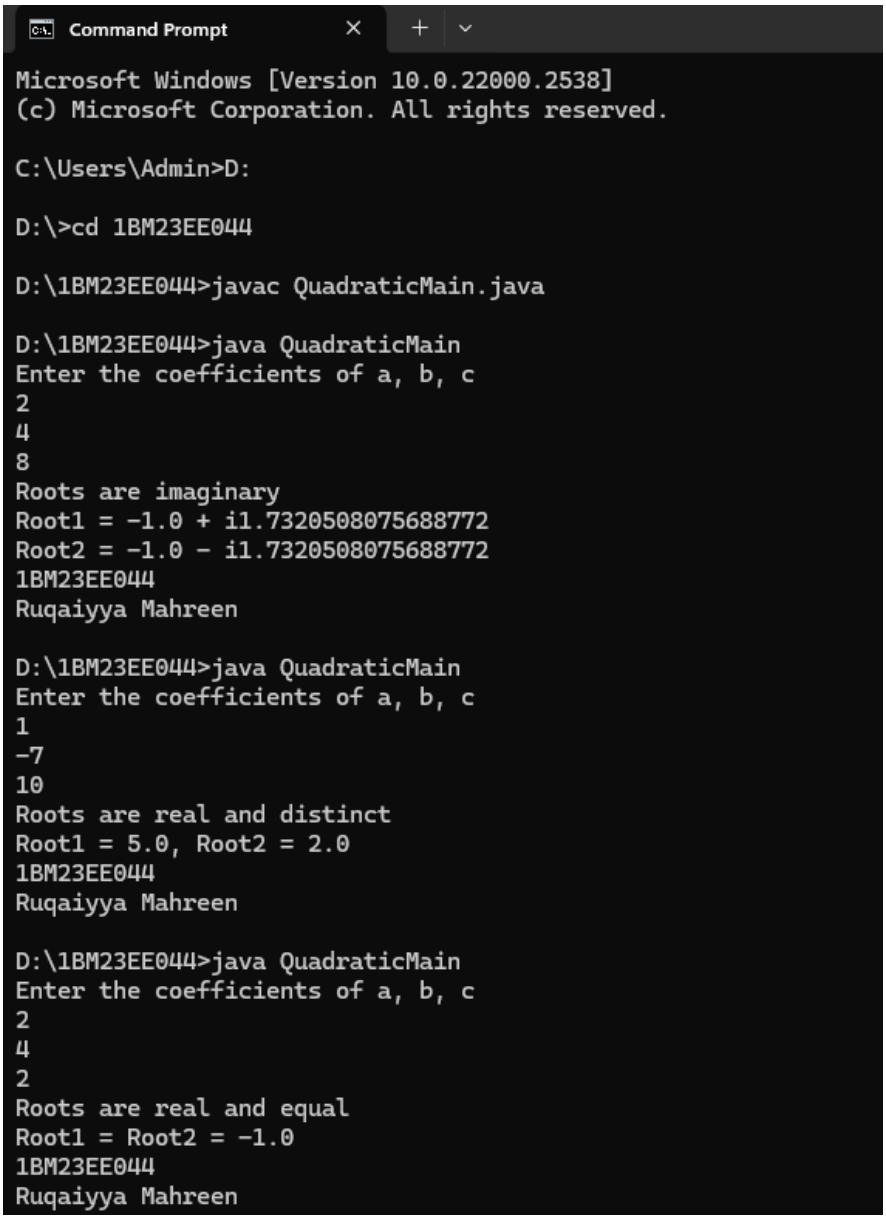


Code:

```
import java.util.*;

class Quadratic {
    int a, b, c;
    double r1, r2, d;

    void getd(Scanner in) {
        System.out.println("Enter the coefficients of a, b, c");
        a = in.nextInt();
        b = in.nextInt();
        c = in.nextInt();
    }

    void compute() {
        while (a == 0) {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non-zero value for a:");
            Scanner in = new Scanner(System.in);
            a = in.nextInt();
        }
        d = b * b - 4 * a * c;
        if (d == 0) {
            r1 = (-b) / (2.0 * a);
            System.out.println("Roots are real and equal");
            System.out.println("Root1 = Root2 = " + r1);
        } else if (d > 0) {
            r1 = (-b + Math.sqrt(d)) / (2.0 * a);
            r2 = (-b - Math.sqrt(d)) / (2.0 * a);
```

```
            System.out.println("Roots are real and distinct");
            System.out.println("Root1 = " + r1 + ", Root2 = " + r2);
        } else {
            System.out.println("Roots are imaginary");
            r1 = (-b) / (2.0 * a);
            r2 = Math.sqrt(-d) / (2.0 * a);
            System.out.println("Root1 = " + r1 + " + i" + r2);
            System.out.println("Root2 = " + r1 + " - i" + r2);
        }
    }
}

class QuadraticMain {
    public static void main(String args[]) {
        Scanner in = new Scanner(System.in);
        Quadratic q = new Quadratic();
        q.getd(in);
        q.compute();
        in.close();
            System.out.println("1BM23EE044");
            System.out.println("Ruqaiyya Mahreen");
    }
}
```

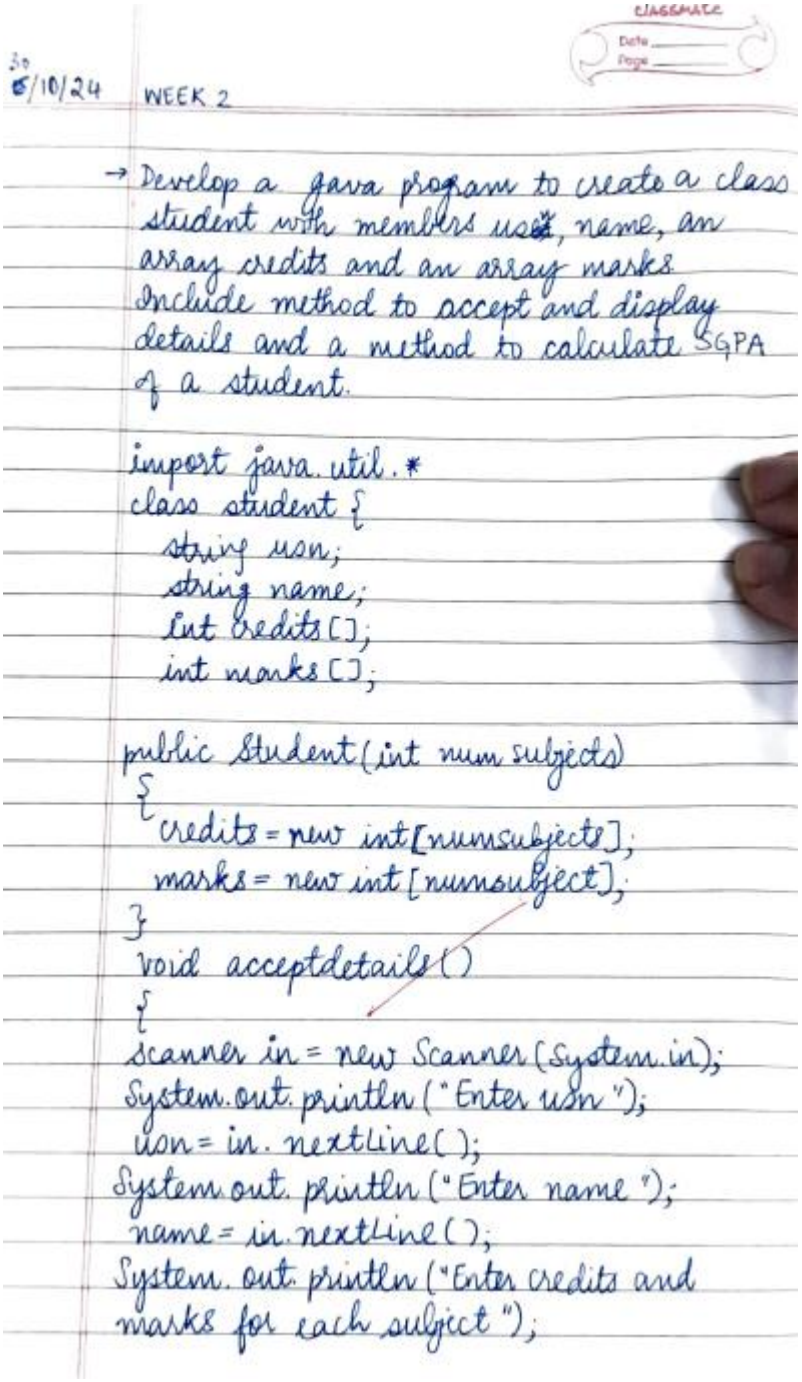
Output:


Command Prompt
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.
C:\Users\Admin>D:
D:\>cd 1BM23EE044
D:\1BM23EE044>javac QuadraticMain.java
D:\1BM23EE044>java QuadraticMain
Enter the coefficients of a, b, c
2
4
8
Roots are imaginary
Root1 = -1.0 + i1.7320508075688772
Root2 = -1.0 - i1.7320508075688772
1BM23EE044
Ruqaiyya Mahreen
D:\1BM23EE044>java QuadraticMain
Enter the coefficients of a, b, c
1
-7
10
Roots are real and distinct
Root1 = 5.0, Root2 = 2.0
1BM23EE044
Ruqaiyya Mahreen
D:\1BM23EE044>java QuadraticMain
Enter the coefficients of a, b, c
2
4
2
Roots are real and equal
Root1 = Root2 = -1.0
1BM23EE044
Ruqaiyya Mahreen

Program 2

Student SGPA Calculation

Algorithm:

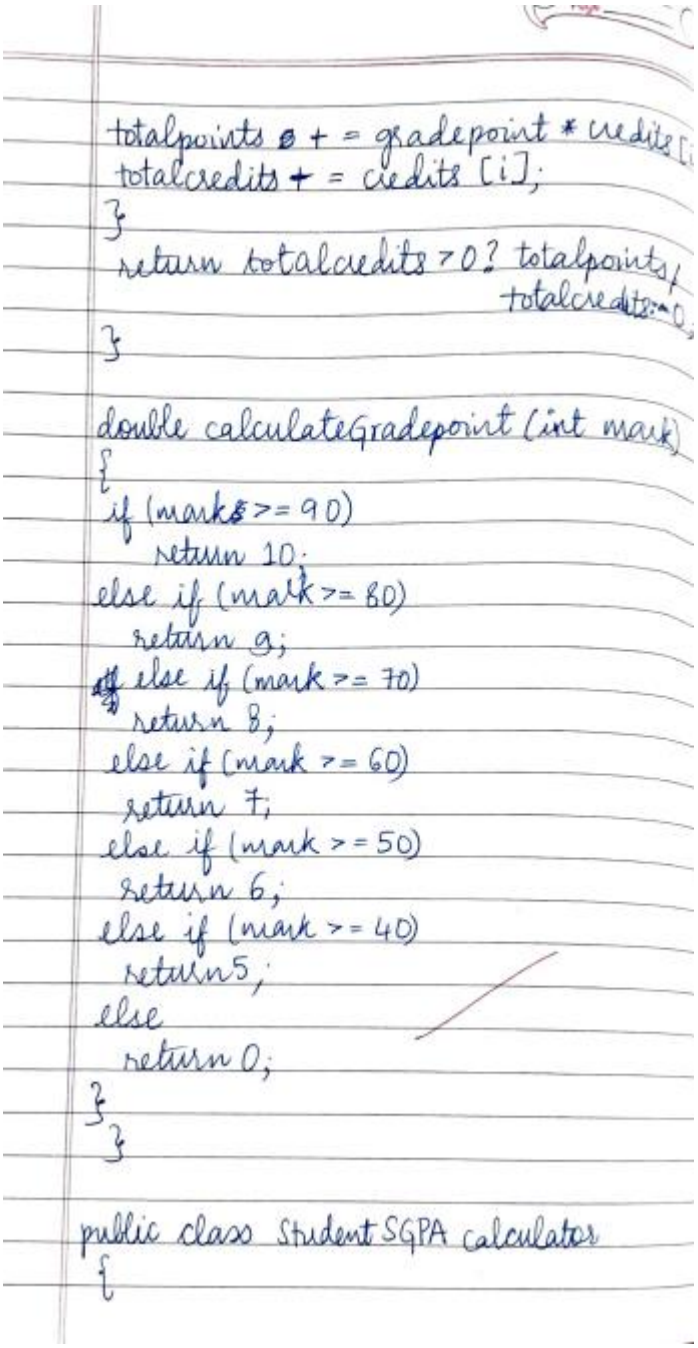


30
6/10/24 WEEK 2

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

```
import java.util.*
class student {
  string usn;
  string name;
  int credits[];
  int marks[];

public Student(int num subjects)
 {
  credits = new int[numsubjects];
  marks = new int[numsubject];
 }
 void acceptdetails()
 {
scanner in = new Scanner(System.in);
System.out.println("Enter usn");
 usn = in.nextline();
System.out.println("Enter name");
 name = in.nextLine();
System.out.println("Enter credits and
marks for each subject");
```

```
for(int i=0; i<credits.length; i++){
  System.out.println("Credits for
                subject "+(i+1));
  credits[i]=in.nextInt();
  System.out.println("Marks for subject
                +(i+1));
  marks[i]=sc.nextInt();
}
}
void displaydetails()
{
 System.out.println("Usn:"+usn);
 System.out.println("Name:"+name);
 for(int i=0; i<credits.length; i++)
 {
  System.out.println("Subject "+(i+1)+
   "credits:"+credits[i]+"Marks:"+
   marks[i]);}
 System.out.println("SGPA :%.2f \n",
                calculateSGPA());

double calculateSGPA(){
  double totalcredits=0;
  double total points=0;
for(int i=0; i<credits.length; i++){
  double gradepoint=calculateGradepoint(
                marks[i]);
```

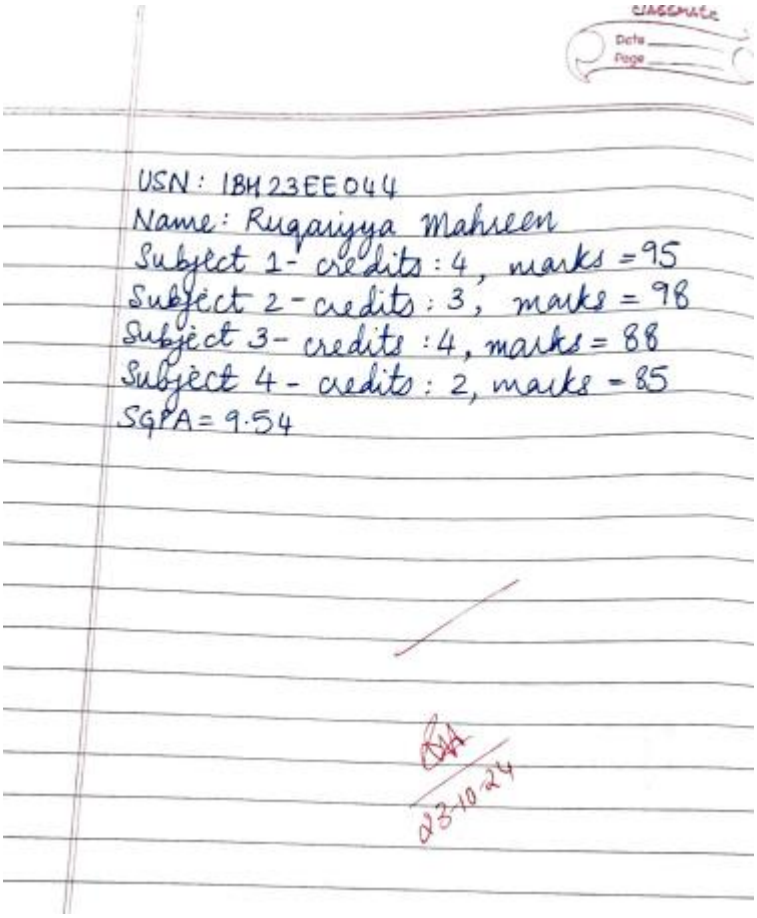


```
totalpoints += gradepoint * credits[i];
totalcredits += credits[i];
}
return totalcredits > 0 ? totalpoints / totalcredits : 0;
}

double calculateGradepoint(int mark)
{
if (marks >= 90)
   return 10;
else if (mark >= 80)
  return 9;
else if (mark >= 70)
  return 8;
else if (mark >= 60)
  return 7;
else if (mark >= 50)
  return 6;
else if (mark >= 40)
  return5;
else
  return 0;
}
}

public class StudentSGPAcalculator
{
```

```
public static void main (String args[])
{
 Scanner in = new Scanner (System.in)
 System.out.println("Enter the no. of
    subjects");
 int numsubjects = in.nextInt();
 in.nextLine();
 student student = new student (numsubjects);
 student.acceptdetails();
 student.displaydetails();
 sc.close();
}
}
```

Output:

```
Enter the number of subjects : 4
Enter usn : 1BM23EE044
Enter name : Ruqaiyya
Enter credits and marks for each
subject :
Credits for subject 1 : 4
Marks : 95
Credits for subject 2 : 3
Marks = 98
Credits for subject 3 : 4
Marks = 88
Credits for subject 4 : 2
Marks = 85
```



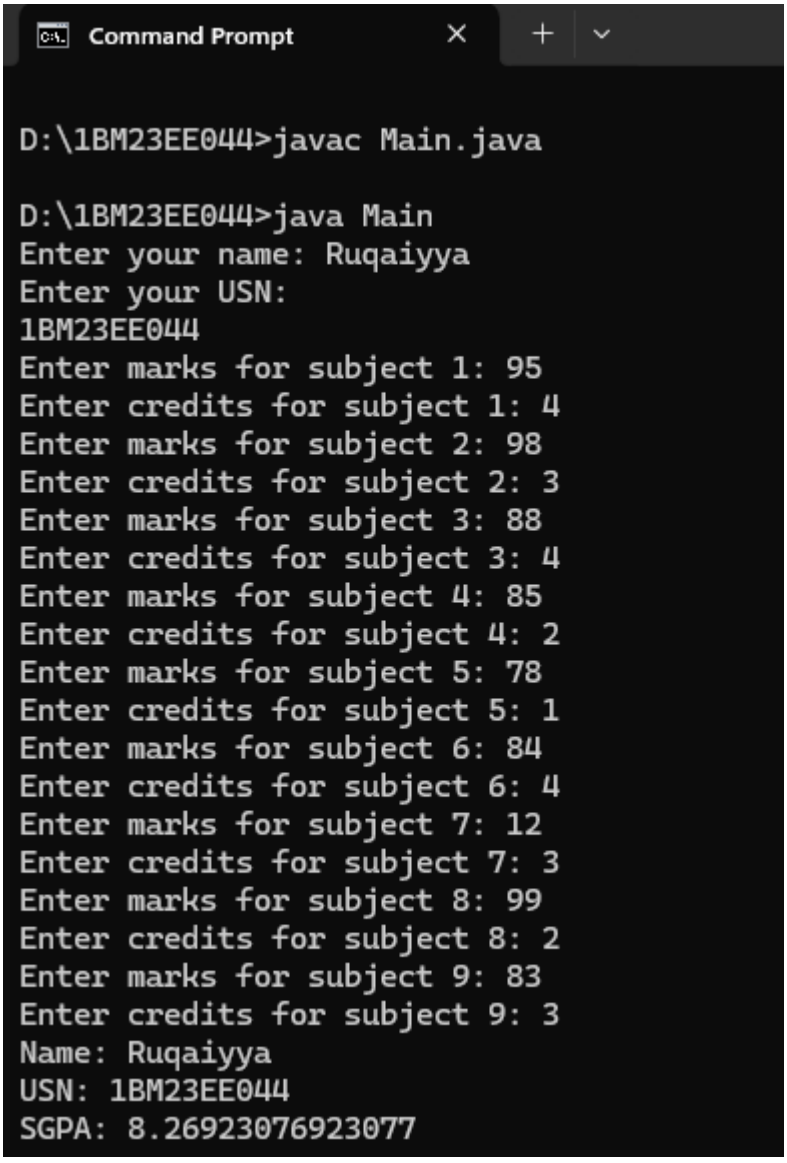
Code:

```
import java.util.*;
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.println("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 Main
{
 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);
        }
}
```

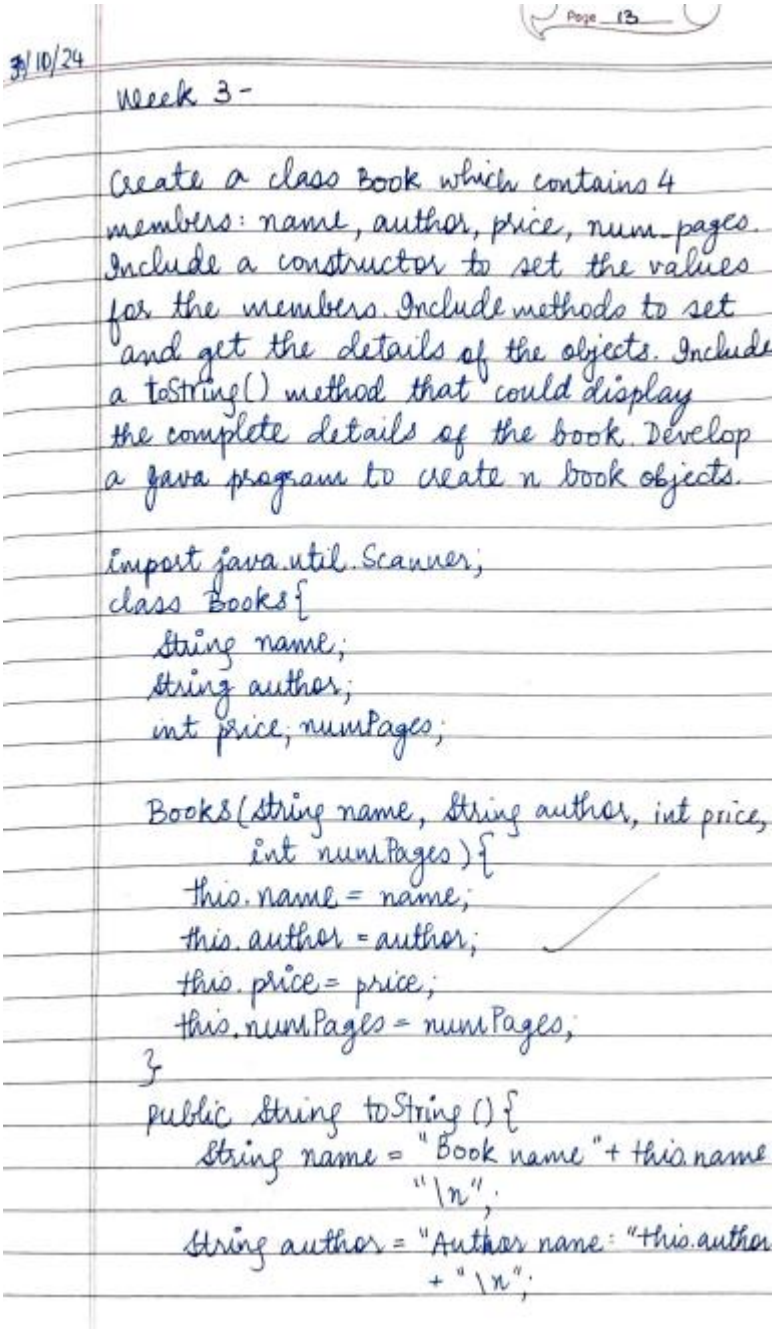
Output:



Program 3

Book Details

Algorithm:



Page 13

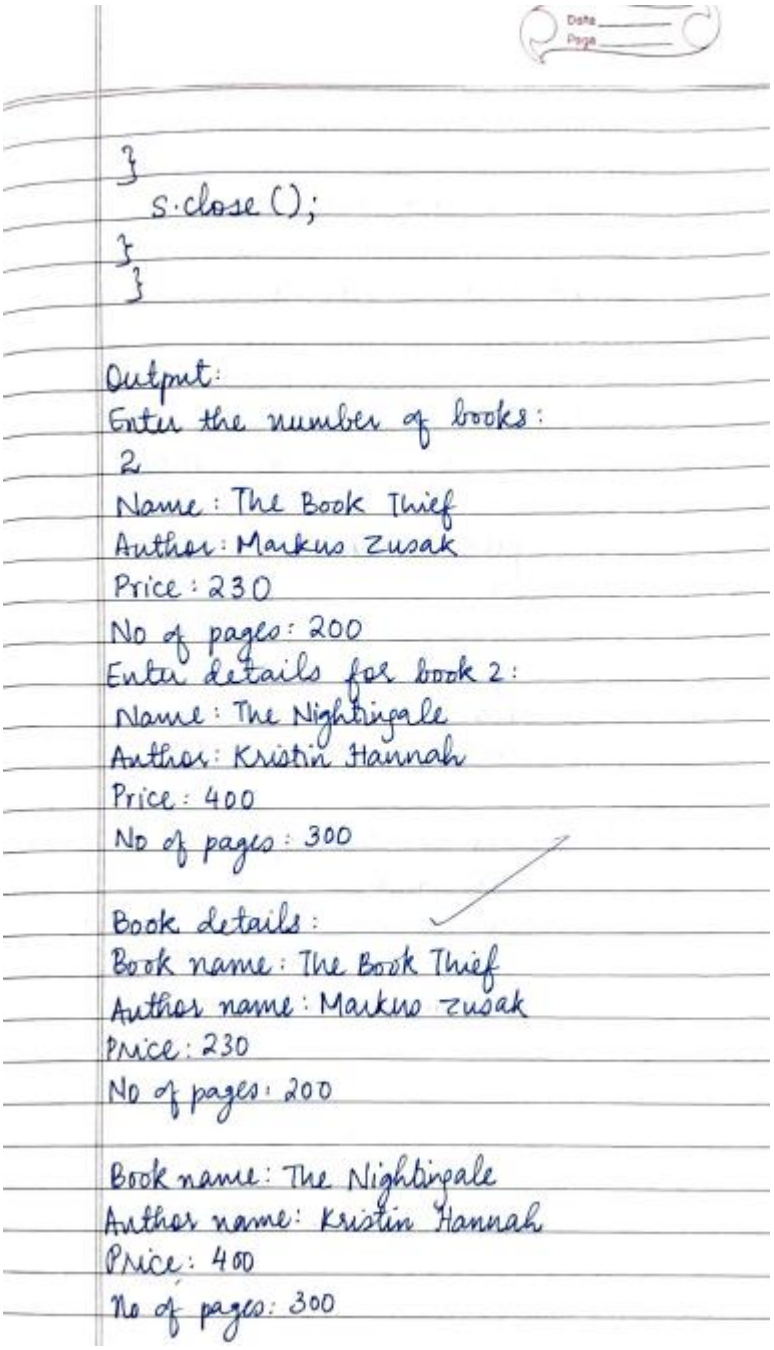
3/10/24

Week 3 -

Create a class Book which contains 4 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.

```
import java.util.Scanner;
class Books{
    String name;
    String author;
    int price, numPages;

    Books(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 = "Book name" + this.name
                        "\n";
        String author = "Author name: "this.author
                        + "\n";
```

```
String price = "Price : " + this.price + "/n";
String numPages = "Number of Pages: " +
                  this.numPages + "\n";
return name + author + price + numPages;
}
}
public class Main{
    public static void main(String args[]){
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the number
            of books : ");
        int n = s.nextInt();
        Books b[] = new Books[n];
        for(int i=0; i<n; i++){
            System.out.println("Enter the
            details for book " + (i+1) + " : ");
            System.out.print("Name : ");
            String name = s.next();
            System.out.print("Author : ");
            String author = s.next();
            System.out.print("Price : ");
            int price = s.nextInt();
            b[i] = new Books(name, author,
                    price, numPages);
        }
        System.out.println("\n Book Details");
        for(int i=0; i<n; i++){
            System.out.println(b[i].toString());
```



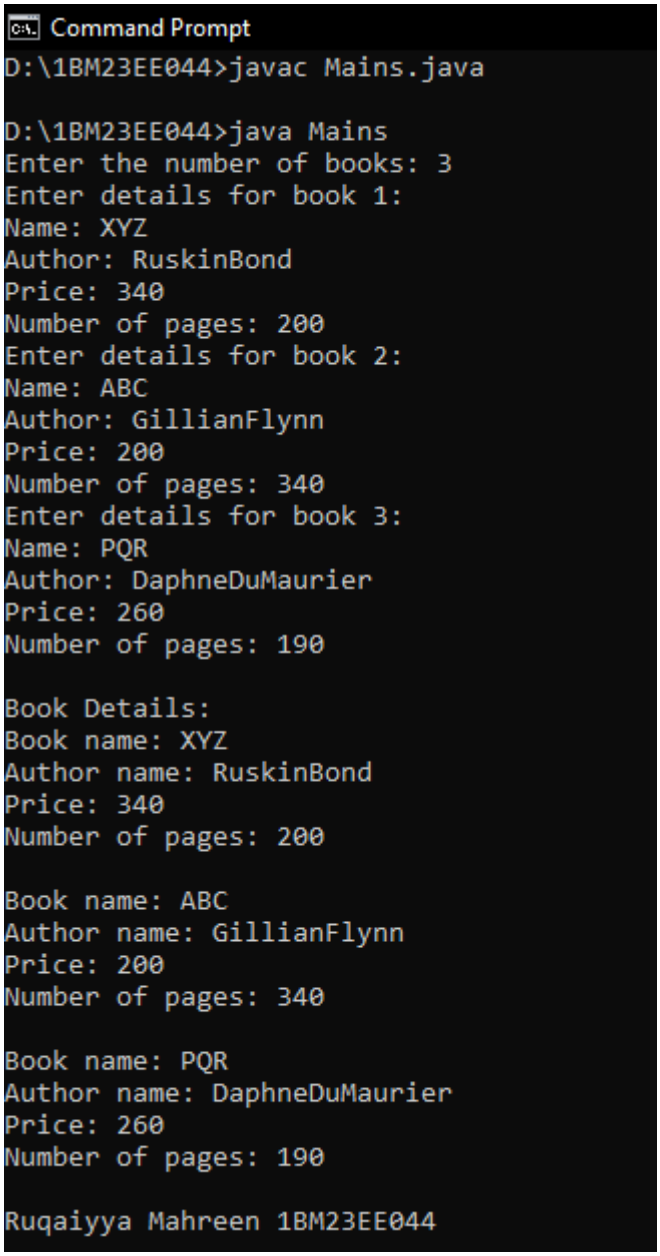
Code:

```
import java.util.Scanner;

class Books {
    String name;
    String author;
    int price;
```

```
    int numPages;

    Books(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 = "Book name: " + this.name + "\n";
        String author = "Author name: " + this.author + "\n";
        String price = "Price: " + this.price + "\n";
        String numPages = "Number of pages: " + this.numPages + "\n";
        return name + author + price + numPages;
    }
}

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

        System.out.print("Enter the number of books: ");
        int n = s.nextInt();

        Books b[] = new Books[n];

        for (int i = 0; i < n; i++) {
            System.out.println("Enter details for book " + (i + 1) + ":");
            System.out.print("Name: ");
            String name = s.next();
            System.out.print("Author: ");
            String author = s.next();
            System.out.print("Price: ");
            int price = s.nextInt();
            System.out.print("Number of pages: ");
            int numPages = s.nextInt();
            b[i] = new Books(name, author, price, numPages);
        }

        System.out.println("\nBook Details:");
```

```
        for (int i = 0; i < n; i++) {
            System.out.println(b[i].toString());
        }
        System.out.println("Ruqaiyya Mahreen 1BM23EE044");
        s.close();
    }
}
```

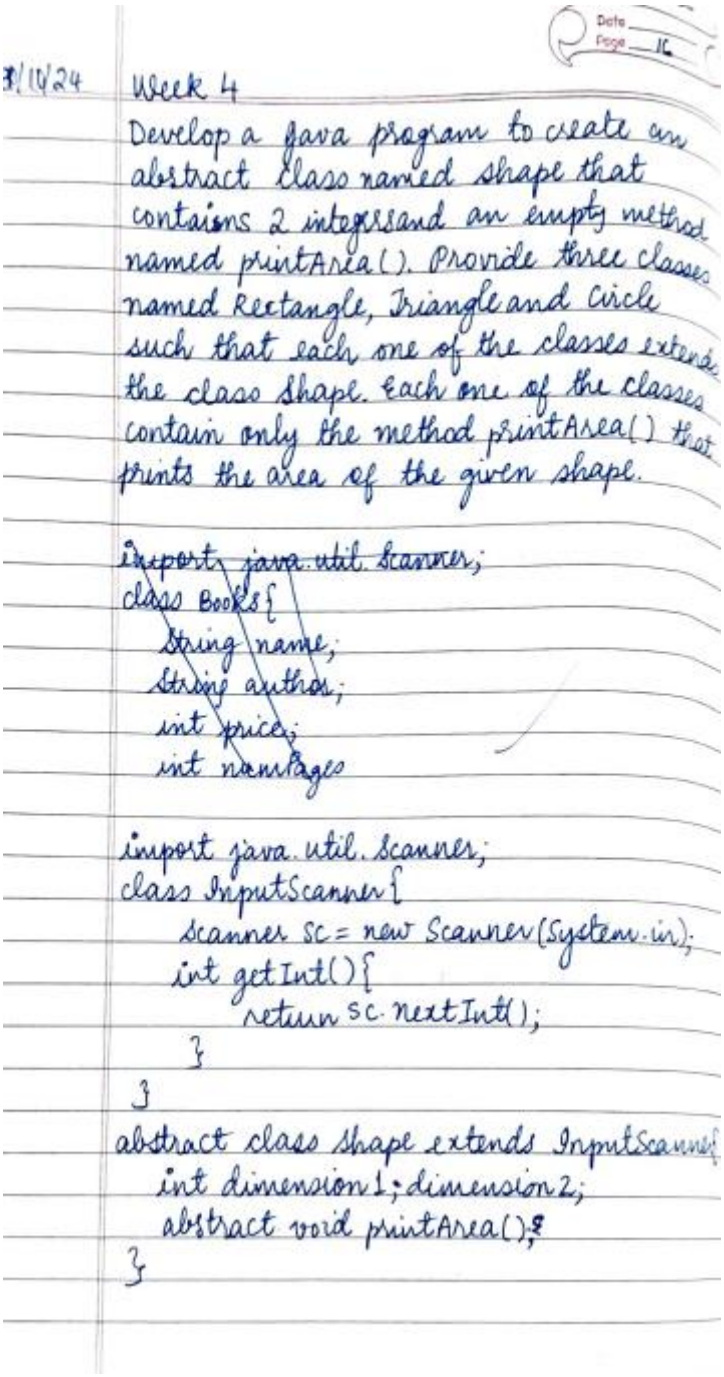
Output:



Program 4

Shape Area Calculation

Algorithm:



Week 4

Develop a java program to create an abstract class named shape that contains 2 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.

```
import java.util.Scanner;
class Books{
   String name;
   String author;
   int price;
   int numPages
```

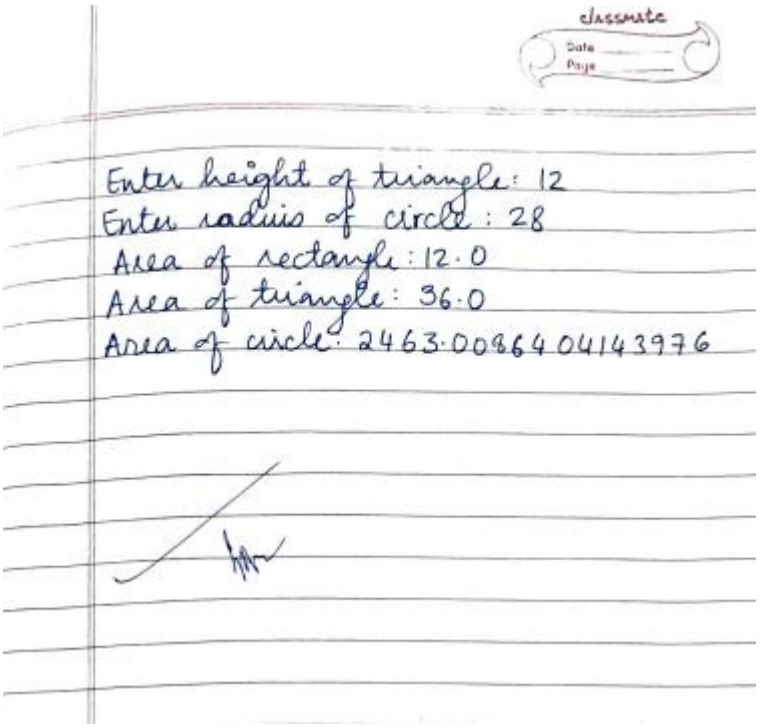
```
import java.util.Scanner;
class InputScanner{
    Scanner sc = new Scanner(System.in);
    int getInt(){
        return sc.nextInt();
    }
}
abstract class Shape extends InputScanner{
   int dimension1; dimension2;
   abstract void printArea();
}
```

```
class Rectangle extends Shape{
  Rectangle(){
    System.out.println("Enter length of
                        rectangle:");
    dimension1 = getInt();
    System.out.println("Enter breadth of
                        rectangle:");
    dimension2 = getInt();
  }
  void printArea(){
    double area = dimension1 * dimension2;
    System.out.println("Area of
                        rectangle:"+area);
  }
}
class Triangle extends Shape{
  Triangle(){
    System.out.println("Enter base of
                        Triangle:");
    dimension1 = getInt();
    System.out.println("Enter height of
                        Triangle:");
    dimension2 = getInt();
  }
  void printArea(){
    double area = 0.5 * dimension1 * dimension
    System.out.println("Area of triangle:"
                        +area);
  }}
```

```
class Circle extends Shape {
    Circle(){
        System.out.println("Enter radius of
                                circle: ");
        dimension1 = getInt();
        dimension2 = 0;
    }
    void printArea(){
        double area = Math.PI * dimension1 *
                                dimension1;
        System.out.println("Area of circle: "+
                                area);
    }
}
public class Main {
    public static void main(String args[]){
        Shape rect = new Rectangle();
        Shape tri = new Triangle();
        Shape cir = new Circle();
        rect.printArea();
        tri.printArea();
        cir.printArea();
    }
}
```

Output:

```
Enter length of rectangle: 4
Enter breadth of rectangle: 3
Enter base of triangle: 6
```



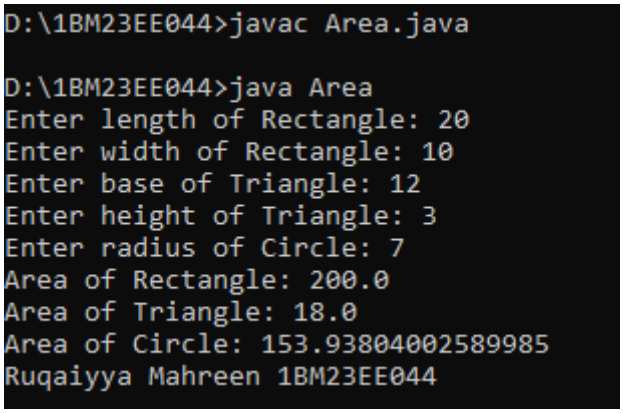
Code:

```
import java.util.Scanner;

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

    int getInt() {
        return scanner.nextInt();
    }
}

abstract class Shape extends InputScanner {
    int dimension1;
    int dimension2;

    abstract void printArea();
}

class Rectangle extends Shape {
    Rectangle() {
        System.out.print("Enter length of Rectangle: ");
        dimension1 = getInt();
        System.out.print("Enter width of Rectangle: ");
```

```
        dimension2 = getInt();
    }

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

class Triangle extends Shape {
    Triangle() {
        System.out.print("Enter base of Triangle: ");
        dimension1 = getInt();
        System.out.print("Enter height of Triangle: ");
        dimension2 = getInt();
    }

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

class Circle extends Shape {
    Circle() {
        System.out.print("Enter radius of Circle: ");
        dimension1 = getInt();
        dimension2 = 0;
    }

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

public class Area {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle();
        Shape triangle = new Triangle();
        Shape circle = new Circle();
```

```
        rectangle.printArea();
        triangle.printArea();
        circle.printArea();
        System.out.println("Ruqaiyya Mahreen 1BM23EE044");
    }
}
```

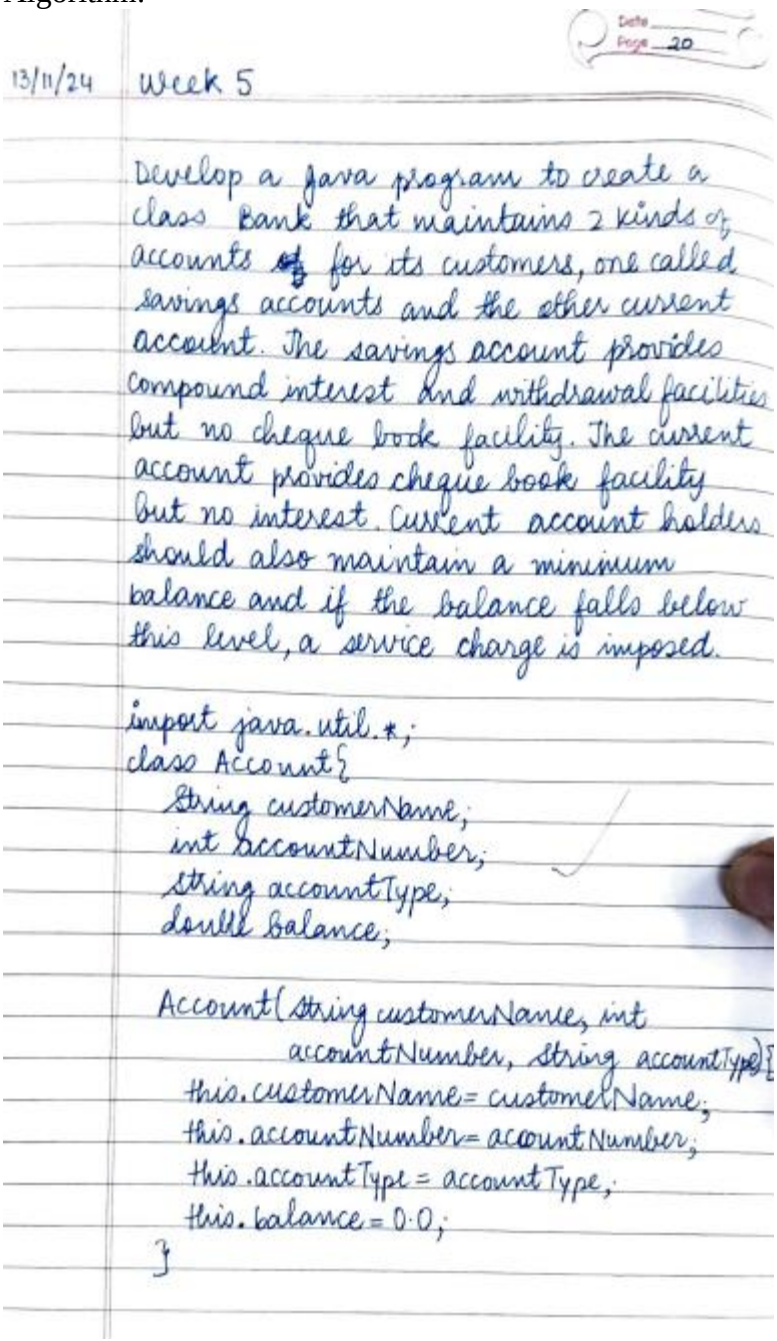
Output:



Program 5

Bank Account Management

Algorithm:



13/11/24 Week 5

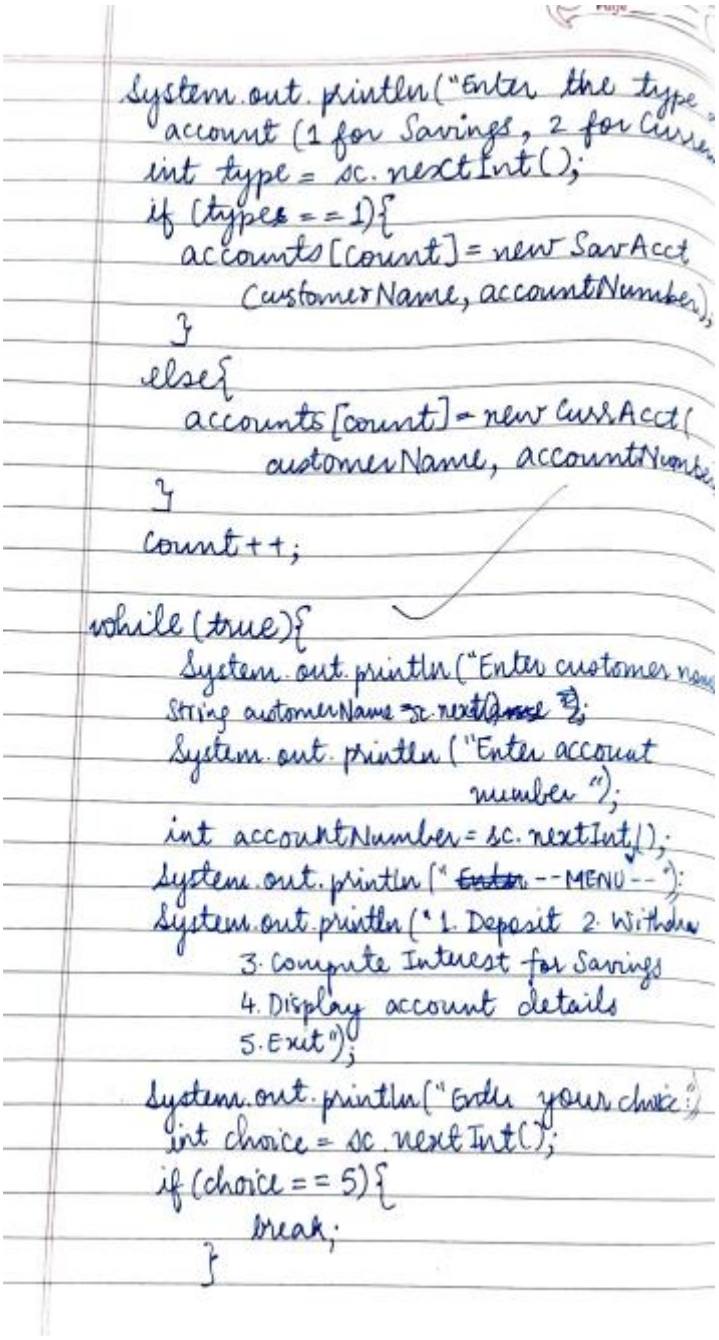
Develop a java program to create a class Bank that maintains 2 kinds of accounts for its customers, one called savings accounts and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book 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.

```
import java.util.*;
class Account{
   String customerName;
   int accountNumber;
   String accountType;
   double balance;

   Account(String customerName, int
         accountNumber, String accountType){
    this.customerName= customerName;
    this.accountNumber= accountNumber;
    this.accountType = accountType;
    this.balance = 0.0;
   }
```

```
void deposit(double amount){
    balance += amount;
    System.out.println("Amount
            deposited: "+amount);
  }

void displayBalance (){
    System.out.println("Customer name :"+
                        customerName);
    System.out.println("Account number: "+
                        accountNumber);
    System.out.println("Types of account: "
      +accountType+ "account");
    System.out.println("Balance = "+balance);
    }
}

class SavAcct extends Account{
    SavAcct(String customerName, int
                accountNumber){
      super(customerName, accountNumber,
                "savings");
    }
  void computeInterest(double rate, int
time){
    balance += balance * Math.pow(1+rate/100,
                time) - balance;
    }
```

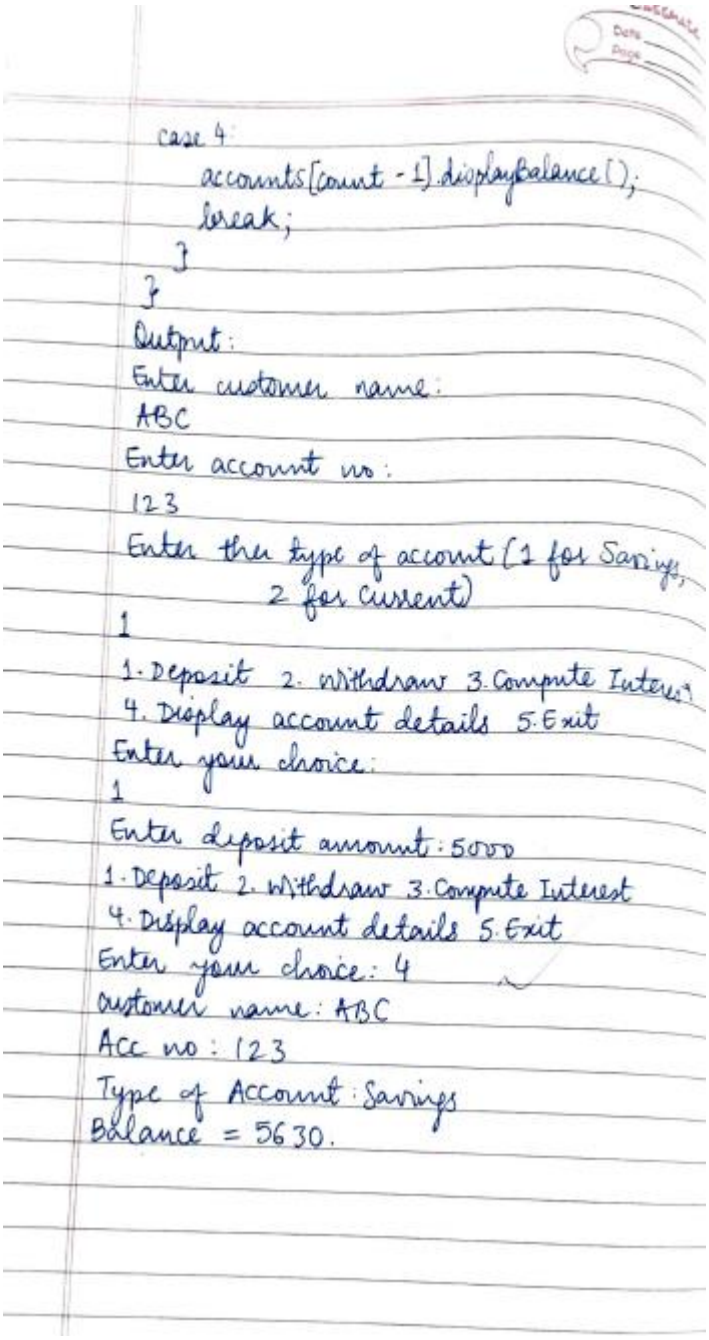
```
void withdraw(double amount){
    if (balance >= amount){
        balance -= amount;
        System.out.println("Amount withdrawn
                + amount);
    } else{
        System.out.println("Insufficient
                balance ");
    }
  }
 }
}

class CurAcct extends Account{
  static final double MIN_BALANCE= 500.0
  static final double PENALTY= 50.0;

  CurAcct(String customerName, int accountNu
  {
    super(customerName, accountNumber,
          "Current");
  }
  void withdraw(double amount){
    if (balance - amount >= MIN_BALANCE)
        balance -= amount;
        System.out.println("Amount withdr
          " + amount);
    }
```

```
    else{
        System.out.println("Min balance
        requirement not met. Cannot withdraw");
    }
}
void checkMinimumBalance(){
    if(balance < MIN_BALANCE){
        balance -= PENALTY;
        System.out.println("Penalty of " +
        PENALTY + " imposed due to insufficient
        balance");
    }
}
}
public class Bank{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        Account[] accounts = new Account[10];
        int count = 0;

        while(true){
        System.out.println("Enter customer
        name");
        String customerName = sc.next();
        System.out.println("Enter account
        number");
        int accountNumber = sc.nextInt();
```



```
System.out.println("Enter the type
  account (1 for Savings, 2 for Curre
int type = sc.nextInt();
if (type == 1){
  accounts[count] = new SavAcct(
      CustomerName, accountNumber);
  }
else{
  accounts[count] = new CurrAcct(
      customerName, accountNumber
  }
count++;

while (true){
    System.out.println("Enter customer name
    String customerName = sc.nextLine();
    System.out.println("Enter account
                        number");
    int accountNumber = sc.nextInt();
    System.out.println("Enter --MENU--");
    System.out.println("1. Deposit 2. Withdra
        3. Compute Interest for Savings
        4. Display account details
        5. Exit");
    System.out.println("Enter your choice:");
    int choice = sc.nextInt();
    if (choice == 5){
        break;
      }
```

```
switch(choice){
    case 1:
        System.out.println("Enter the deposit
            amount");
        double depositAmount = sc.nextDouble();
        accounts[count-1].deposit(depositAmount);
        break;
    case 2:
        System.out.println("Enter the withdrawal
                            amount : ");
        double withdrawAmount = sc.nextDouble();
        if (type == 1){                    withdrawAmount
            ((SavAcct)accounts[count-1].withdraw(
        }else{
        ((CurrAcct)accounts[count-1].withdraw(withdrawAmount);
        ((CurrAcct)accounts[count-1].checkMinimumBalance();
        }break;
    case 3:
        if (type == 1){                    time");
            System.out.println("Enter the interest rate and
            double rate = sc.nextDouble();
            int time = sc.nextInt();
            ((SavAcct)accounts[count-1]).computeInterest(rate, time);
            System.out.println("Interest computed and
                                added to balance");
        }else{
            System.out.println("No interest
                computation for Current Account");
        }break;
```



Code:

```
import java.util.Scanner;

class Account {
    String customerName;
    int accountNumber;
    String accountType;
    double balance;
```

```
    Account(String customerName, int accountNumber, String accountType) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = 0.0;
    }

    void deposit(double amount) {
        balance += amount;
        System.out.println("Amount deposited: " + amount);
    }

    void displayBalance() {
        System.out.println("Customer name: " + customerName);
        System.out.println("Account number: " + accountNumber);
        System.out.println("Type of Account: " + accountType + " account");
        System.out.println("Balance = " + balance);
    }
}

class SavAcct extends Account {
    SavAcct(String customerName, int accountNumber) {
        super(customerName, accountNumber, "Savings");
    }

    void computeInterest(double rate, int time) {
        balance += balance * Math.pow(1 + rate / 100, time) - balance;
    }

    void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Amount withdrawn: " + amount);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

class CurAcct extends Account {
    static final double MIN_BALANCE = 500.0;
    static final double PENALTY = 50.0;

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

```
    void withdraw(double amount) {
        if (balance - amount >= MIN_BALANCE) {
            balance -= amount;
            System.out.println("Amount withdrawn: " + amount);
        } else {
            System.out.println("Minimum balance requirement not met. Cannot withdraw.");
        }
    }

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

public class Bank {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
            System.out.println("Ruqaiyya Mahreen 1BM23EE044");
        Account[] accounts = new Account[10];
        int count = 0;

        while (true) {
            System.out.println("Enter customer name: ");
            String customerName = sc.next();
            System.out.println("Enter account Number: ");
            int accountNumber = sc.nextInt();
            System.out.println("Enter the type of account (1 for Savings, 2 for Current): ");
            int type = sc.nextInt();

            if (type == 1) {
                accounts[count] = new SavAcct(customerName, accountNumber);
            } else {
                accounts[count] = new CurAcct(customerName, accountNumber);
            }
            count++;

            while (true) {
                System.out.println("-----MENU-----");
                System.out.println("1. Deposit\n2. Withdraw\n3. Compute interest for SavingsAccount\n4.
Display account details\n5. Exit");
                System.out.println("Enter your choice: ");
                int choice = sc.nextInt();

                if (choice == 5) break;
```

```
            switch (choice) {
                case 1:
                    System.out.println("Enter the deposit amount: ");
                    double depositAmount = sc.nextDouble();
                    accounts[count - 1].deposit(depositAmount);
                    break;

                case 2:
                    System.out.println("Enter the withdrawal amount: ");
                    double withdrawAmount = sc.nextDouble();
                    if (type == 1) {
                        ((SavAcct) accounts[count - 1]).withdraw(withdrawAmount);
                    } else {
                        ((CurAcct) accounts[count - 1]).withdraw(withdrawAmount);
                        ((CurAcct) accounts[count - 1]).checkMinimumBalance();
                    }
                    break;

                case 3:
                    if (type == 1) {
                        System.out.println("Enter the interest rate and time in years: ");
                        double rate = sc.nextDouble();
                        int time = sc.nextInt();
                        ((SavAcct) accounts[count - 1]).computeInterest(rate, time);
                        System.out.println("Interest computed and added to balance.");
                    } else {
                        System.out.println("No interest computation for Current Account.");
                    }
                    break;

                case 4:
                    accounts[count - 1].displayBalance();
                    break;
            }
        }

        System.out.println("Do you want to add another account? (yes/no): ");
        String response = sc.next();
        if (response.equalsIgnoreCase("no")) break;
    }

    sc.close();
  }
}
```

Output:

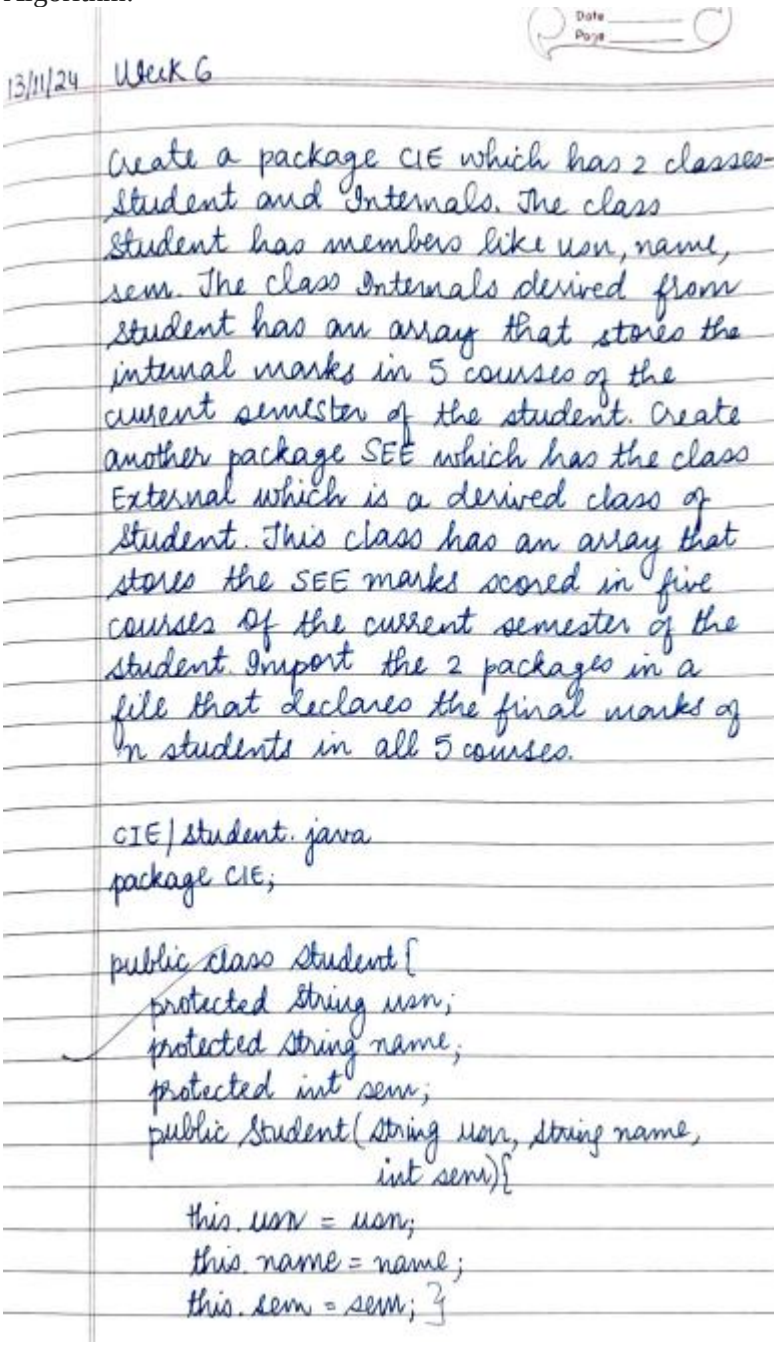
```
D:\1BM23EE044>javac Bank.java

D:\1BM23EE044>java Bank
Enter customer name:
ABC
Enter account Number:
123
Enter the type of account (1 for Savings, 2 for Current):
1
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
1
Enter the deposit amount:
5000
Amount deposited: 5000.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
3
Enter the interest rate and time in years:
2
6
Interest computed and added to balance.
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
4
Customer name: ABC
Account number: 123
Type of Account: Savings account
Balance = 5630.81209632
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
5
Do you want to add another account? (yes/no):
yes
Enter customer name:
Ruqaiyya
Enter account Number:
456
Enter the type of account (1 for Savings, 2 for Current):
2
-----MENU-----
1. Deposit
```

```
2
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
1
Enter the deposit amount:
10000
Amount deposited: 10000.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
2
Enter the withdrawal amount:
200
Amount withdrawn: 200.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
3
No interest computation for Current Account.
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
4
Customer name: Ruqaiyya
Account number: 456
Type of Account: Current account
Balance = 9800.0
-----MENU-----
1. Deposit
2. Withdraw
3. Compute interest for SavingsAccount
4. Display account details
5. Exit
Enter your choice:
5
Do you want to add another account? (yes/no):
no
```

Program 6

CIE and SEE Marks

Algorithm:



13/11/24 Week 6

Create a package CIE which has 2 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 in 5 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 2 packages in a file that declares the final marks of n students in all 5 courses.

CIE/student.java

```
package CIE;

public class Student {
    protected String usn;
    protected String name;
    protected int sem;
    public Student(String usn, String name,
                   int sem){
        this.usn = usn;
        this.name = name;
        this.sem = sem; }
```

```
public void displayStudentInfo(){
    System.out.println("USN: "+usn+
"Name: "+name+"Semester"+sem);
 }
}
```

CIE/Internals.java

```
package CIE;
public class Internals{
  private int[] internalMarks = new int[5];
  ~~in~~
  public Internals(int[]marks){
    ~~if(marks.length==5){~~
      System.~~out~~.arraycopy(marks,0,
internalMarks,0,5);
  }
  public int[] getInternalMarks(){
      return internalMarks;
  }
}
```

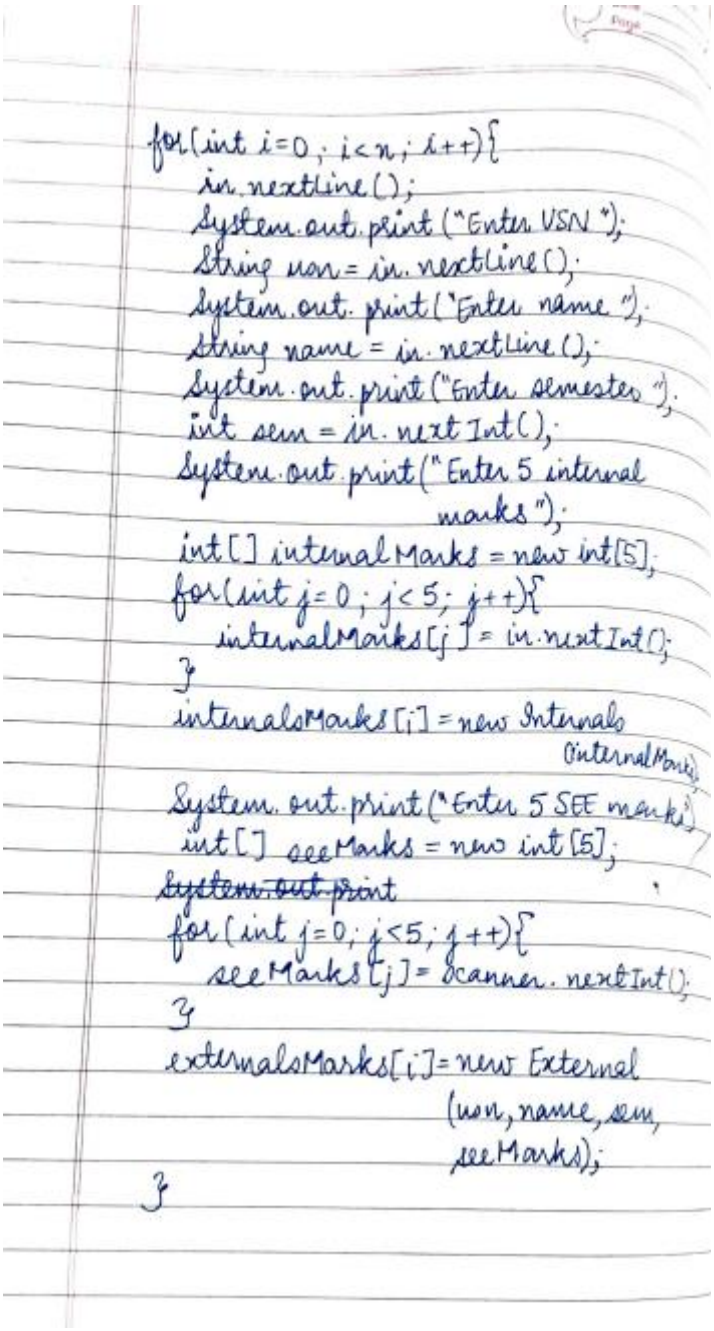
SEE/External.java

```
package SEE;
import CIE.Student;
public class External extends Student{
  private int[] seeMarks = new int[5];
```

```
public External(String usn, String name,
 int sem, int[] seeMarks){
 super(usn, name, sem);
 if(seeMarks)
    System.arraycopy(seeMarks, 0,
    this.seeMarks, 0, 5);
 }
public int[] getSeeMarks(){
    return seeMarks;
 }
}
```

Main.java in same root

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

public class Main{
   public static void main(String args[]){
      Scanner in = new Scanner(System.in);
      System.out.println("Enter the no. of
       students");
      int n = in.nextInt();
      Student[] students = new Student[n];
      Internals[] internalsMarks = new Internals[
      Externals[] externalMarks = new External[5]
```

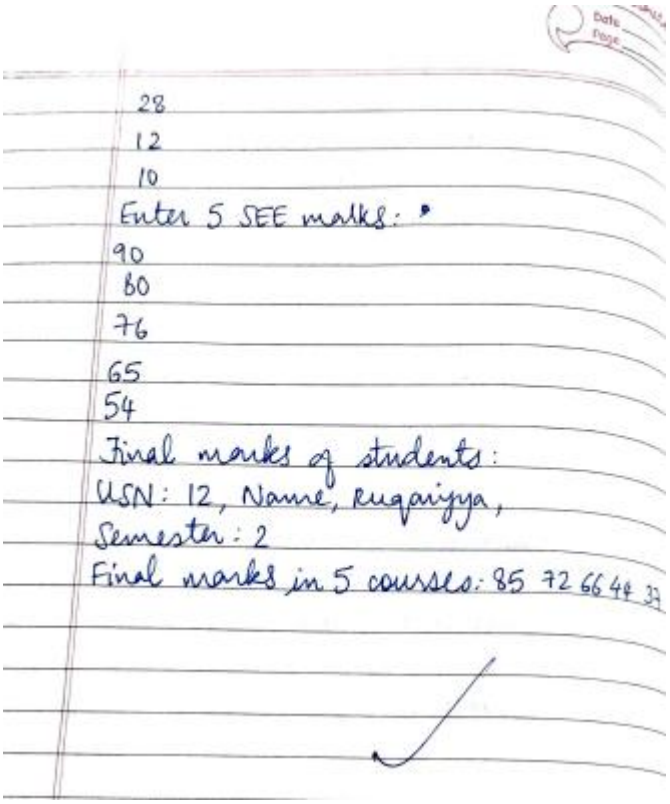


```
for(int i=0; i<n; i++){
    in.nextLine();
    System.out.print("Enter USN ");
    String usn = in.nextLine();
    System.out.print("Enter name ");
    String name = in.nextLine();
    System.out.print("Enter semester ");
    int sem = in.nextInt();
    System.out.print("Enter 5 internal
                      marks");
    int[] internalMarks = new int[5];
    for(int j=0; j<5; j++){
        internalMarks[j] = in.nextInt();
    }
    internalsMarks[i] = new Internals
                        (internalMarks);
    System.out.print("Enter 5 SEE marks");
    int[] seeMarks = new int[5];
    System.out.print
    for(int j=0; j<5; j++){
        seeMarks[j] = scanner.nextInt();
    }
    externalsMarks[i] = new External
                        (usn, name, sem,
                         seeMarks);
}
```

```
System.out.println("\n Final Marks of
                    Students: ");
for(int i=0; i<n; i++){
  Students[i] = externalsMarks[i];
  Students[i].displayStudentInfo();
  int[] internal = internalsMarks[i].get
                   InternalMarks();
  int[] external = externalsMarks[i].getSee
                                    Marks();

  System.out.print("Final Marks in 5courses");
  for(int j=0; j<5; j++){
    int finalMark = internal[j] + external[j]/2;
    System.out.print(finalMark + " ");
  }
  System.out.println();
}
in.close();
}
}
```

Output:

```
Enter no. of students: 1
Enter USN: 56
Enter name: Rugaiyya
Enter semester: 2
Enter 5 internal marks:
40
32
```



Code:

CIE/Student.java

```
package CIE;

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

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

    public void displayStudentInfo() {
        System.out.println("USN: " + usn + ", Name: " + name + ", Semester: " + sem);
    }
}
```

CIE/Internals.java

```
package CIE;

public class Internals {
```

```
    private int[] internalMarks = new int[5];

    public Internals(int[] marks) {
        if (marks.length == 5) {
            System.arraycopy(marks, 0, internalMarks, 0, 5);
        } else {
            throw new IllegalArgumentException("Exactly 5 internal marks are required.");
        }
    }

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

SEE/External.java

```
package SEE;

import CIE.Student;

public class External extends Student {
    private int[] seeMarks = new int[5];

    public External(String usn, String name, int sem, int[] seeMarks) {
        super(usn, name, sem);
        if (seeMarks.length == 5) {
            System.arraycopy(seeMarks, 0, this.seeMarks, 0, 5);
        } else {
            throw new IllegalArgumentException("Exactly 5 SEE marks are required.");
        }
    }

    public int[] getSeeMarks() {
        return seeMarks;
    }
}

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

Main.java

```
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
            System.out.println("Ruqaiyya Mahreen 1BM23EE044");
        System.out.print("Enter the number of students: ");
```

```
int n = scanner.nextInt();

Student[] students = new Student[n];
Internals[] internalsMarks = new Internals[n];
External[] externalsMarks = new External[n];

for (int i = 0; i < n; i++) {
    scanner.nextLine();  // Consume the newline character

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

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

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

    System.out.print("Enter 5 internal marks: ");
    int[] internalMarks = new int[5];
    for (int j = 0; j < 5; j++) {
        internalMarks[j] = scanner.nextInt();
    }
    internalsMarks[i] = new Internals(internalMarks);

    System.out.print("Enter 5 SEE marks: ");
    int[] seeMarks = new int[5];
    for (int j = 0; j < 5; j++) {
        seeMarks[j] = scanner.nextInt();
    }
    externalsMarks[i] = new External(usn, name, sem, seeMarks);
}

System.out.println("\nFinal Marks of Students:");
for (int i = 0; i < n; i++) {
    students[i] = externalsMarks[i];  // Since External is a subclass of Student
    students[i].displayStudentInfo();

    int[] internal = internalsMarks[i].getInternalMarks();
    int[] external = externalsMarks[i].getSeeMarks();
    System.out.print("Final Marks in 5 Courses: ");
    for (int j = 0; j < 5; j++) {
        int finalMark = internal[j] + (external[j] / 2);  // SEE marks are divided by 2
        System.out.print(finalMark + " ");
    }
    System.out.println();
}
```

```
        scanner.close();
    }
}
```

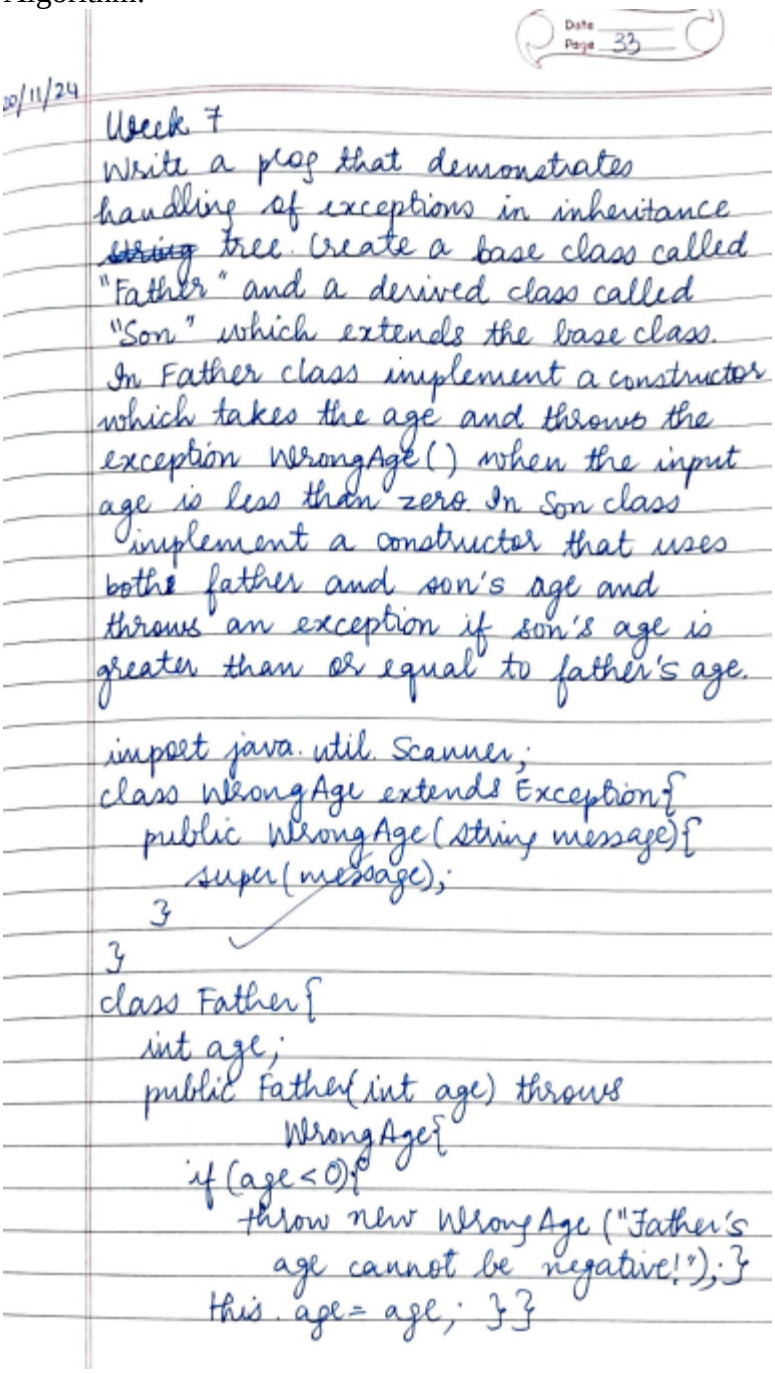
Output:

```
D:\1BM23EE044>javac Main.java

D:\1BM23EE044>java Main
Ruqaiyya Mahreen 1BM23EE044
Enter the number of students: 1
Enter USN: 56
Enter Name: Ruqaiyya
Enter Semester: 3
Enter 5 internal marks: 34
23
21
40
36
Enter 5 SEE marks: 80
78
67
75
54

Final Marks of Students:
USN: 56, Name: Ruqaiyya, Semester: 3
Final Marks in 5 Courses: 74 62 54 77 63

D:\1BM23EE044>javac Main.java

D:\1BM23EE044>java Main
Ruqaiyya Mahreen 1BM23EE044
Enter the number of students: 2
Enter USN: 12
Enter Name: Ruqaiyya
Enter Semester: 2
Enter 5 internal marks: 40
32
28
12
10
Enter 5 SEE marks: 90
80
76
65
54
Enter USN: 87
Enter Name: Shraddha
Enter Semester: 3
Enter 5 internal marks: 4
40
39
10
12
Enter 5 SEE marks: 76
72
74
78
65

Final Marks of Students:
USN: 12, Name: Ruqaiyya, Semester: 2
Final Marks in 5 Courses: 85 72 66 44 37
USN: 87, Name: Shraddha, Semester: 3
Final Marks in 5 Courses: 42 76 76 49 44
```

Program 7

Exception Handling

Algorithm:



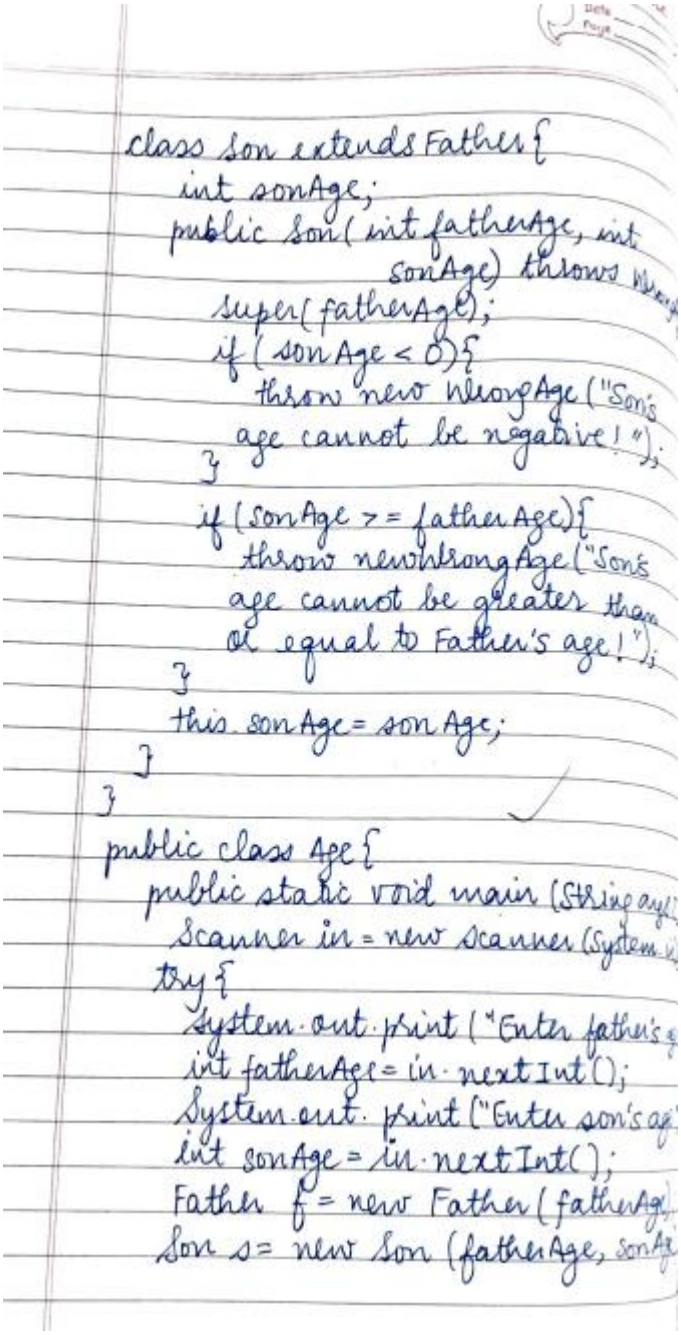
Date
Page 33

20/11/24

Week 7

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

```
import java.util.Scanner;
class WrongAge extends Exception{
  public WrongAge(String message){
    super(message);
  }
}
class Father{
  int age;
  public Father(int age) throws
          WrongAge{
    if(age<0){
      throw new WrongAge("Father's
        age cannot be negative!");}
    this.age=age;}}
```



```
class Son extends Father{
    int sonAge;
    public Son(int fatherAge, int
                    sonAge) throws WrongAge{
        super(fatherAge);
        if(sonAge < 0){
            throw new WrongAge("Son's
            age cannot be negative!");
        }
        if(sonAge >= fatherAge){
            throw new WrongAge("Son's
            age cannot be greater than
            or equal to Father's age!");
        }
        this.sonAge = sonAge;
    }
}
public class Age{
    public static void main(String args[]
        Scanner in = new Scanner(System.
        try{
            System.out.print("Enter father's
            int fatherAge = in.nextInt();
            System.out.print("Enter son's ag
            int sonAge = in.nextInt();
            Father f = new Father(fatherAge,
            Son s = new Son(fatherAge, sonAg
```

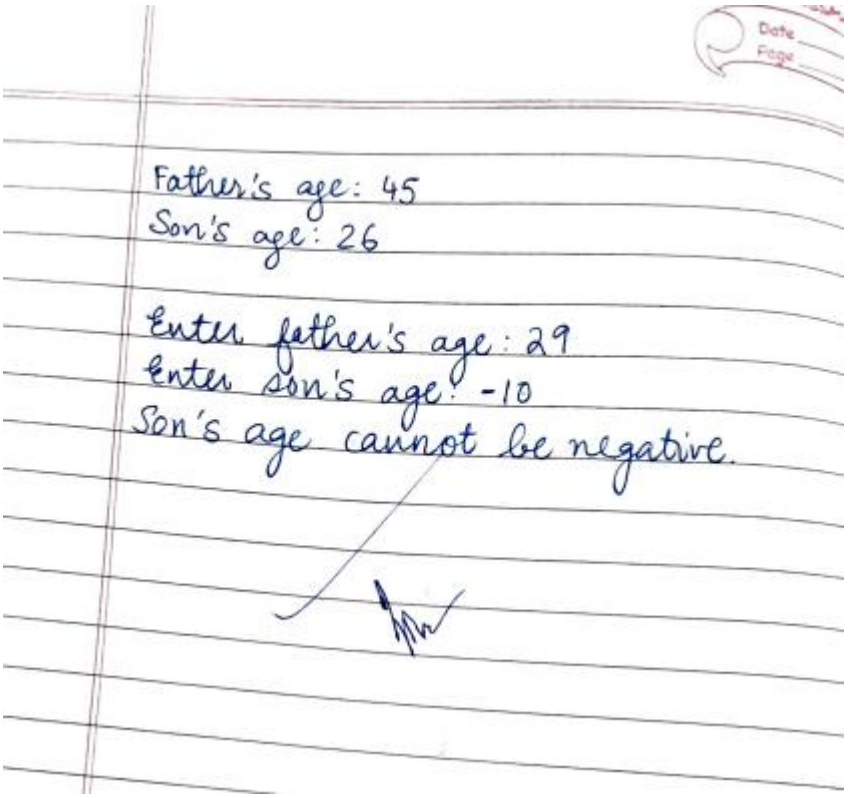
```
            System.out.println("Father's age: "+
                                        fatherAge);
            System.out.println("Son's age: "+
                                        sonAge);
        }catch(WrongAge e){
            System.out.println("Error: "+e.getMessage());
        }catch(Exception e){
            System.out.println("Enter integers
                                only");
        }
        finally{
            in.close();
        }
    }
}
```

Output:

Enter father's age: 23
Enter son's age: 46
Son's age cannot be greater than or equal to father's age.

Enter father's age: -19
Enter son's age: -9
Father's age cannot be negative.

Enter father's age: 45
Enter son's age: 26



Code:

```
import java.util.Scanner;

class WrongAgeException extends Exception {
    public WrongAgeException(String message) {
        super(message);
    }
}
class Father {
    int age;

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

    public Son(int fatherAge, int sonAge) throws WrongAgeException {
        super(fatherAge);
        if (sonAge < 0) {
```

```
            throw new WrongAgeException("Son's age cannot be negative.");
        }
        if (sonAge >= fatherAge) {
            throw new WrongAgeException("Son's age cannot be greater than or equal to Father's age.");
        }
        this.sonAge = sonAge;
    }
}
public class Age{
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
            System.out.println("Ruqaiyya Mahreen 1BM23EE044");
        try {
            System.out.print("Enter Father's age: ");
            int fatherAge = in.nextInt();
            System.out.print("Enter Son's age: ");
            int sonAge = in.nextInt();
            Father father = new Father(fatherAge);
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Father's age: " + father.age);
            System.out.println("Son's age: " + son.sonAge);
        } catch (WrongAgeException e) {
            System.out.println("Error: " + e.getMessage());
        } finally {
            in.close();
        }
    }
}
```

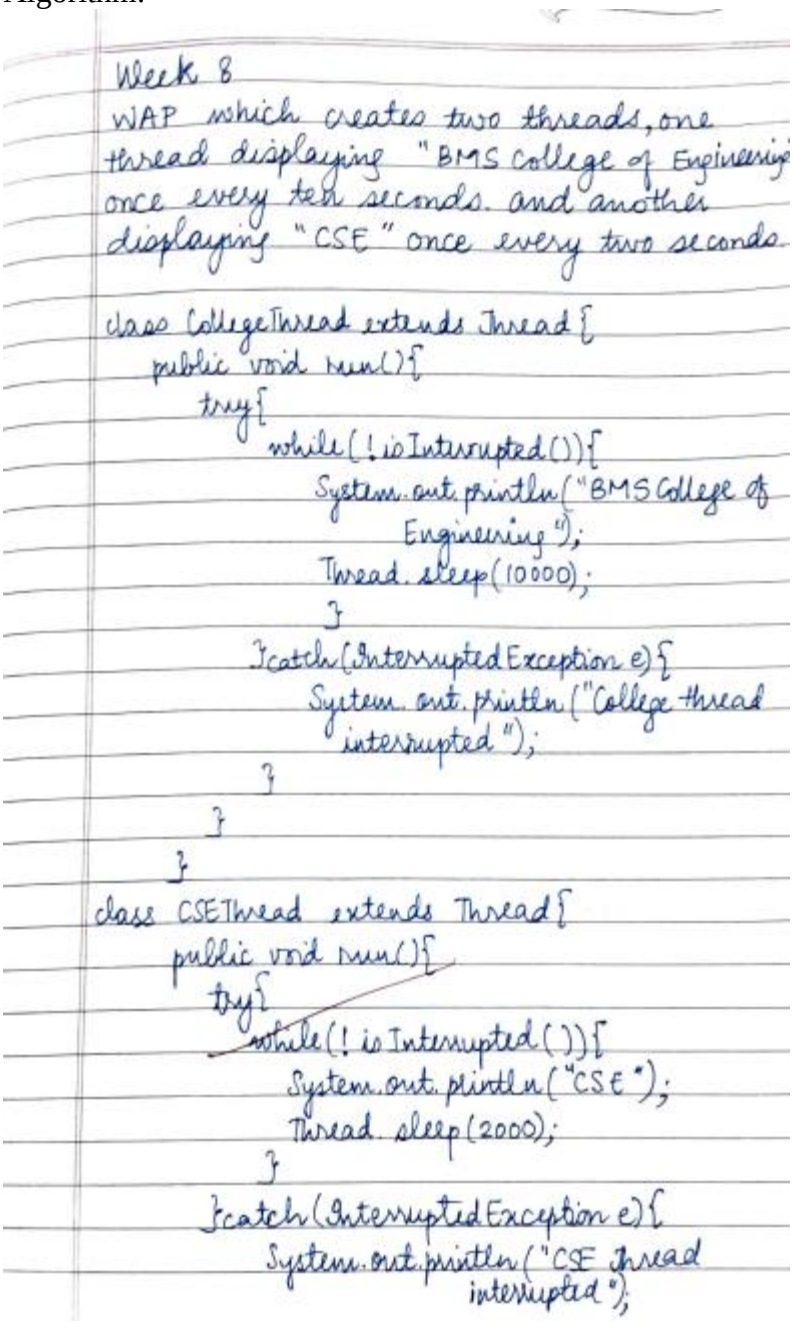
Output:

```
PS D:\1BM23EE044> java Age
Ruqaiyya Mahreen 1BM23EE044
Enter Father's age: 12
Enter Son's age: 23
Error: Son's age cannot be greater than or equal to Father's age.
PS D:\1BM23EE044> java Age
Ruqaiyya Mahreen 1BM23EE044
Enter Father's age: -9
Enter Son's age: 12
Error: Father's age cannot be negative.
PS D:\1BM23EE044> java Age
Ruqaiyya Mahreen 1BM23EE044
Enter Father's age: 32
Enter Son's age: -9
Error: Son's age cannot be negative.
PS D:\1BM23EE044> java Age
Ruqaiyya Mahreen 1BM23EE044
Enter Father's age: 31
Enter Son's age: 31
Error: Son's age cannot be greater than or equal to Father's age.
PS D:\1BM23EE044> java Age
Ruqaiyya Mahreen 1BM23EE044
Enter Father's age: 43
Enter Son's age: 18
Father's age: 43
Son's age: 18
```

Program 8

Thread Based Message Display

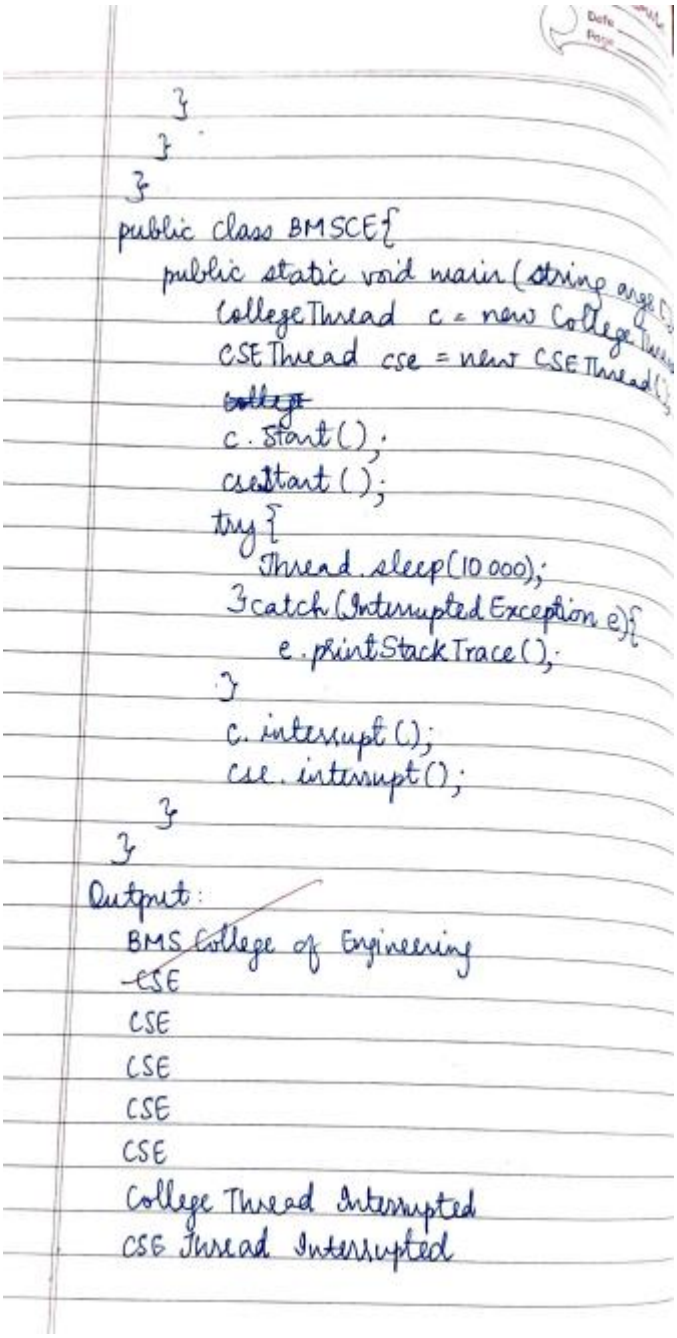
Algorithm:



Week 8

WAP which creates two threads, one thread displaying "BMS College of Engineering" once every ten seconds and another displaying "CSE" once every two seconds.

```
class CollegeThread extends Thread{
    public void run(){
        try{
            while(!isInterrupted()){
                System.out.println("BMS College of
                        Engineering");
                Thread.sleep(10000);
                }
            }catch(InterruptedException e){
                System.out.println("College thread
                    interrupted");
            }
        }
    }
class CSEThread extends Thread{
    public void run(){
        try{
            while(!isInterrupted()){
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        }catch(InterruptedException e){
            System.out.println("CSE thread
                        interrupted");
```



Code:

```
class CollegeThread extends Thread {
   public void run() {
      try {
         while (!isInterrupted()) {
            System.out.println("BMS College of Engineering");
            Thread.sleep(10000); // 10 seconds
         }
      } catch (InterruptedException e) {
```

```
            System.out.println("College Thread interrupted");
        }
    }
}

class CSEThread extends Thread {
    public void run() {
        try {
            while (!isInterrupted()) {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            System.out.println("CSE Thread interrupted");
        }
    }
}

public class BMSCE {
    public static void main(String[] args) {
        CollegeThread collegeThread = new CollegeThread();
        CSEThread cseThread = new CSEThread();
            System.out.println("Ruqaiyya Mahreen 1BM23EE044");
        collegeThread.start();
        cseThread.start();

        try {
            Thread.sleep(10000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        collegeThread.interrupt();
        cseThread.interrupt();
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>D:

D:\>cd 1BM23EE044

D:\1BM23EE044>javac BMSCE.java

D:\1BM23EE044>java BMSCE
Ruqaiyya Mahreen 1BM23EE044
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
College Thread interrupted
CSE Thread interrupted
```

Program 9

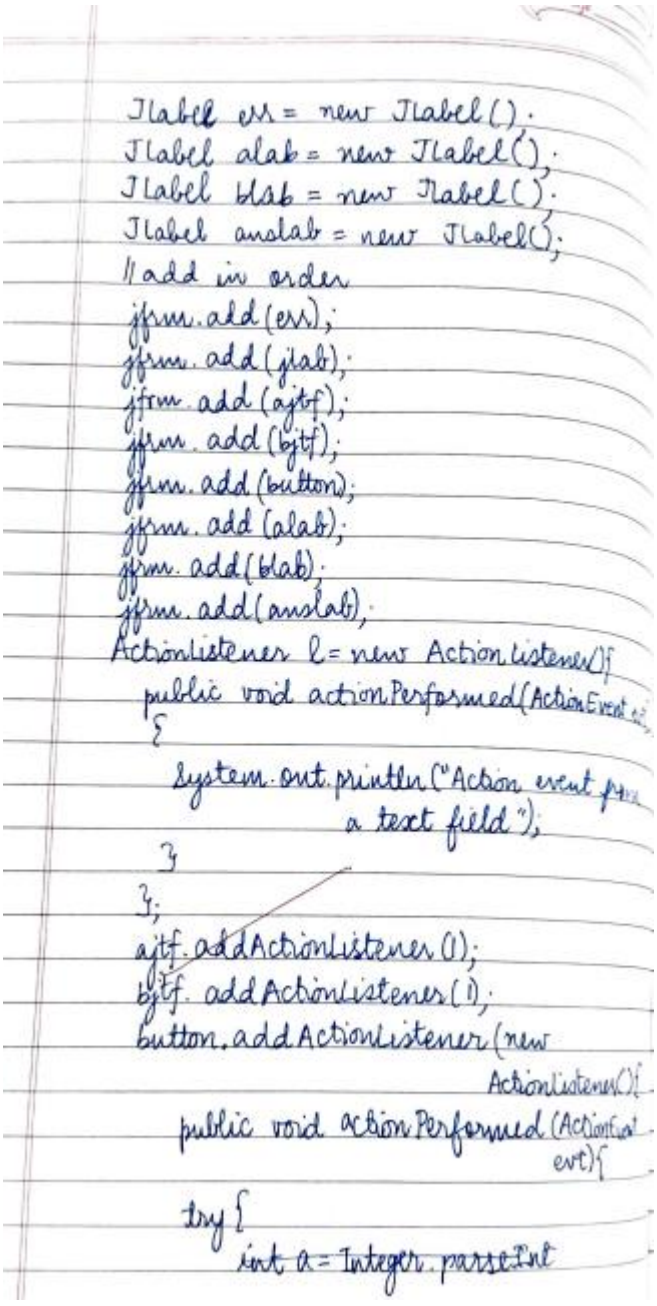
Integer Division User Interface

Algorithm:

Week 9

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

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo{
    SwingDemo(){
     JFrame jfrm = new JFrame("Divider
                              App");
     jfrm.setSize(275,150);
     jfrm.setLayout(new FlowLayout());
     jfrm.setDefaultCloseOperation(JFrame.EXIT
                                   ON_CLOSE);
     JLabel jlab = new JLabel("Enter the
                    divider and dividend:");
     JTextField ajtf = new JTextField(8);
     JTextField bjtf = new JTextField(8);
     JButton button = new JButton("Calculate
```



```
Jlabel err = new Jlabel().
Jlabel alab = new Jlabel();
Jlabel blab = new Jlabel().
Jlabel anslab = new Jlabel();
//add in order
jfrm.add(err);
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);
ActionListener l = new ActionListener(){
  public void actionPerformed(ActionEvent e)
  {
    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
```

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

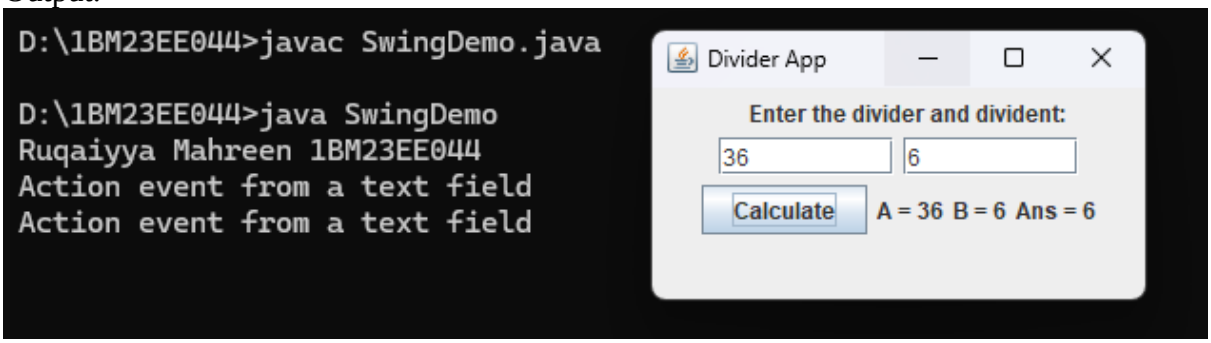
Code:

```
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[]){
System.out.println("Ruqaiyya Mahreen 1BM23EE044");
// create frame on event dispatching thread
SwingUtilities.invokeLater(new Runnable(){
public void run(){
new SwingDemo();
}
});
}
```

}

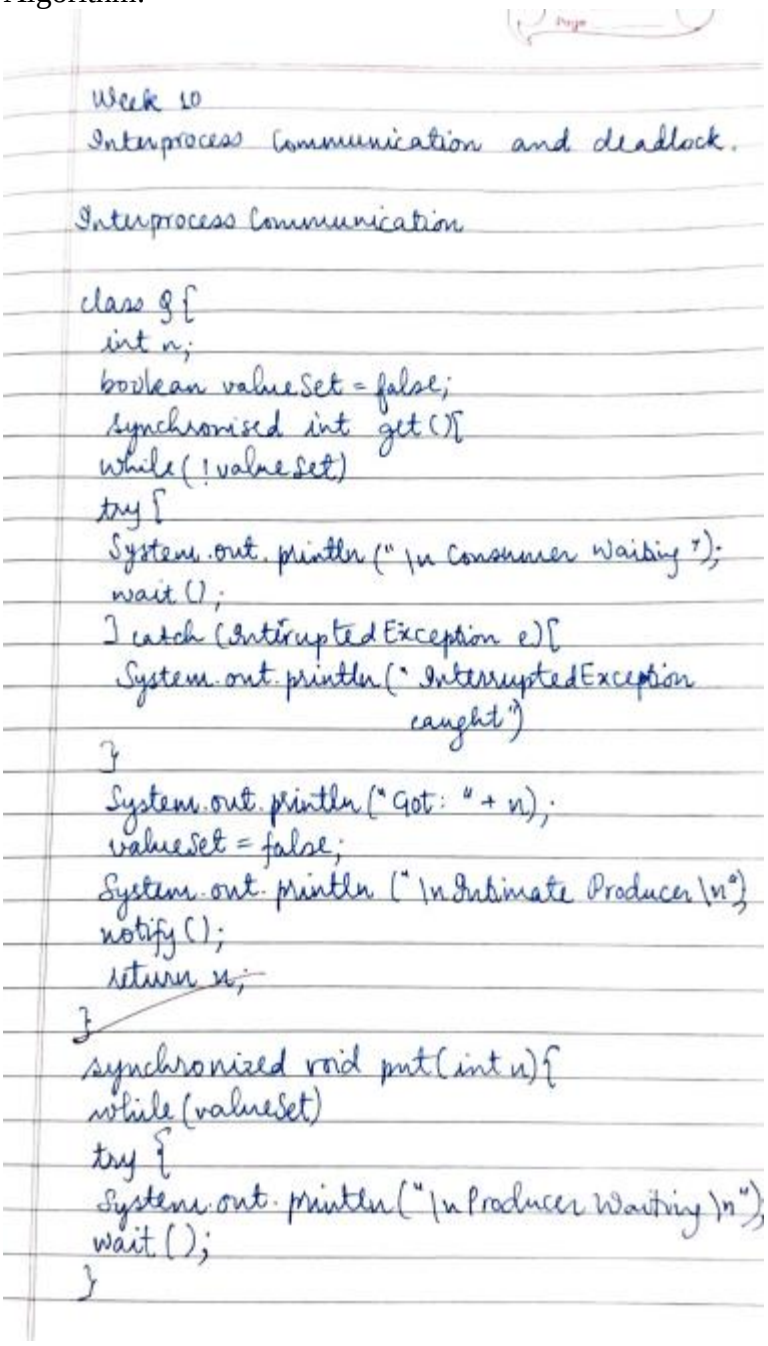
Output:


D:\1BM23EE044>javac SwingDemo.java
D:\1BM23EE044>java SwingDemo
Ruqaiyya Mahreen 1BM23EE044
Action event from a text field
Action event from a text field
Divider App
Enter the divider and divident:
36
6
Calculate
A = 36 B = 6 Ans = 6

Program 10

Interprocess Communication and Deadlock

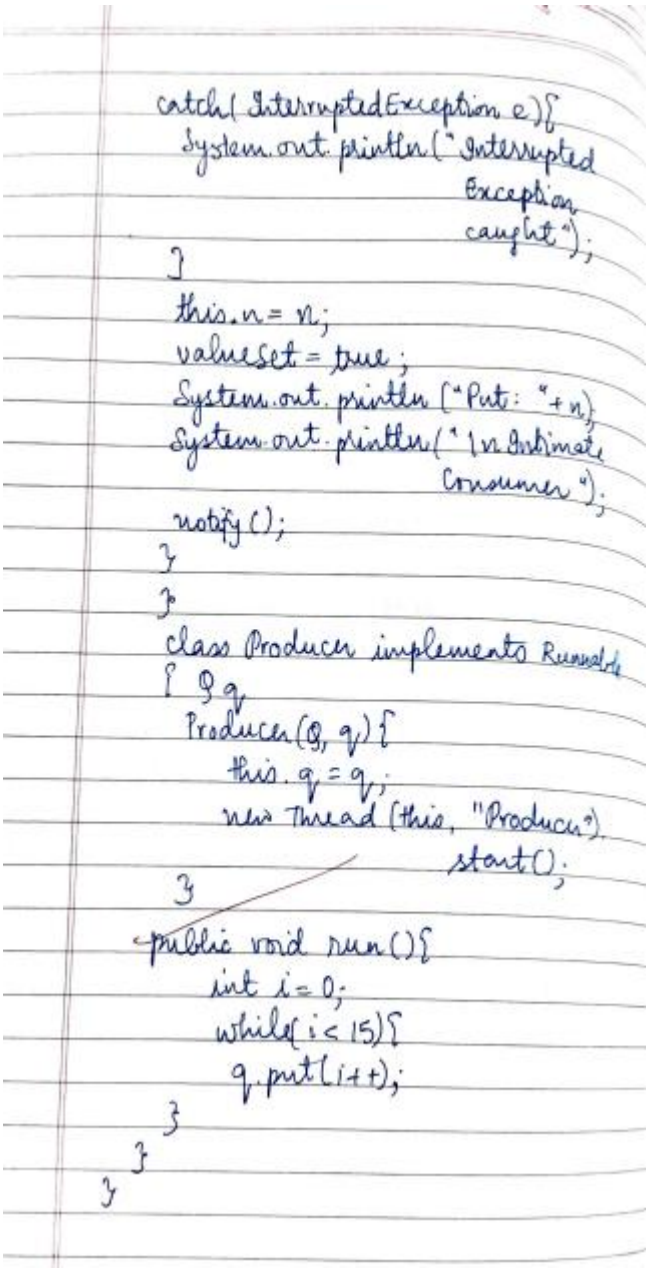
Algorithm:



Week 10
Interprocess Communication and deadlock.

Interprocess Communication

```
class Q {
 int n;
 boolean valueSet = false;
 synchronized int get(){
 while(!valueSet)
 try {
  System.out.println("\n Consumer Waiting");
  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("\n Producer Waiting\n");
  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
                    Consumer");
notify();
}
}
class Producer implements Runnable
{ Q q;
  Producer(Q, q){
    this.q = q;
    new Thread(this, "Producer").
                    start();
  }
public void run(){
    int i = 0;
    while(i < 15){
      q.put(i++);
  }
 }
}
```

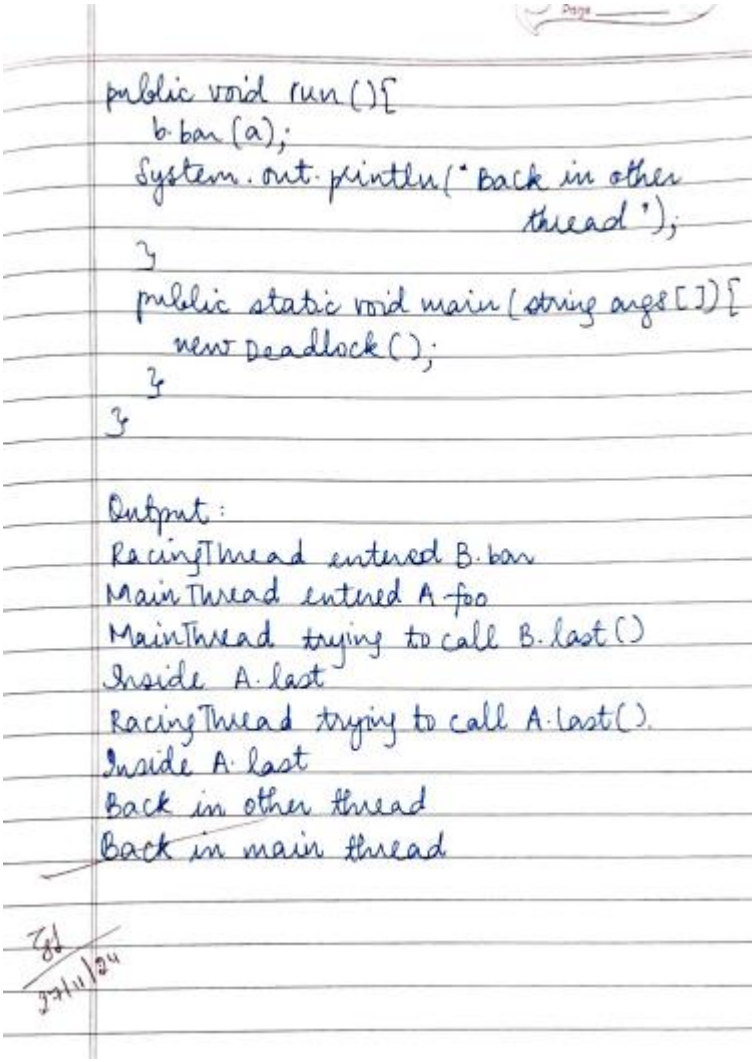
```
class Consumer implements Runnable{
  Q q;
  Consumer(Q, q){
  this.q = q;
  new Thread(this, "Consumer").start();
  }
  public void run(){
   int i = 0;
   while(i<15){
    int r = q.get();
    System.out.println("Consumed:" + r);
    i++;
   }
  }
}
class PCFixed{
  public static void main(String args[]){
   Q q = new Q();
   new Producer(q);
   new Consumer(q);
   System.out.println("Press control C to
                       stop");
  }
}
```

Output:
Press Control-C to stop
Put: 0
Intimate Consumer
Producer waiting
Got: 0
Intimate Producer
Put: 1
Intimate Consumer
Processor waiting
consumed: 0
Got: 1
Intimate producer
Consumed: 1
Put: 2
Intimate consumer
Producer waiting
Got: 2
Intimate Producer
consumed: 2
Put: 3
Intimate Consumer
Producer waiting
Got: 3

Deadlock -

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



Code:

//Inter process communication

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

```
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
System.out.println("\nProducer waiting\n");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("\nIntimate Consumer\n");
notify();
}
}
class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(i<15) {
q.put(i++);
}
}
}
class Consumer implements Runnable {
Q q;
Consumer(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
int i=0;
while(i<15) {
int r=q.get();
System.out.println("consumed:"+r);
i++;
}
}
}
class PCFixed {
```

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

Output:

```
D:\1BM23EE044>javac PCFixed.java

D:\1BM23EE044>java PCFixed
Press Control-C to stop.
Put: 0

Intimate Consumer


Producer waiting

Got: 0

Intimate Producer

Put: 1

Intimate Consumer


Producer waiting

consumed:0
Got: 1

Intimate Producer

consumed:1
Put: 2

Intimate Consumer


Producer waiting

Got: 2

Intimate Producer

consumed:2
Put: 3

Intimate Consumer


Producer waiting

Got: 3

Intimate Producer
```

```
//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 {
synchronized void bar(A a) {
String name = Thread.currentThread().getName();
System.out.println(name + " entered B.bar");
try {
Thread.sleep(1000);
} catch(Exception e) {
System.out.println("B Interrupted");
}
System.out.println(name + " trying to call A.last()");
a.last();
}
void last() {
System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable
{
A a = new A();
B b = new B();
Deadlock() {
Thread.currentThread().setName("MainThread");
Thread t = new Thread(this,"RacingThread");
t.start();
a.foo(b); // get lock on a in thisthread.
System.out.println("Back in main thread");
}
public void run() {
b.bar(a); // get lock on b in other thread.
System.out.println("Back in other thread");
```

```
}
public static void main(String args[]) {
new Deadlock();
}
}
```

Output: