VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

MADHAN K (1BM23CS173)

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 **MADHAN K (1BM23CS173),** 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.

Srushti C S Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Github Link:
https://github.com/MadhanK173/Java-LAB/tree/main

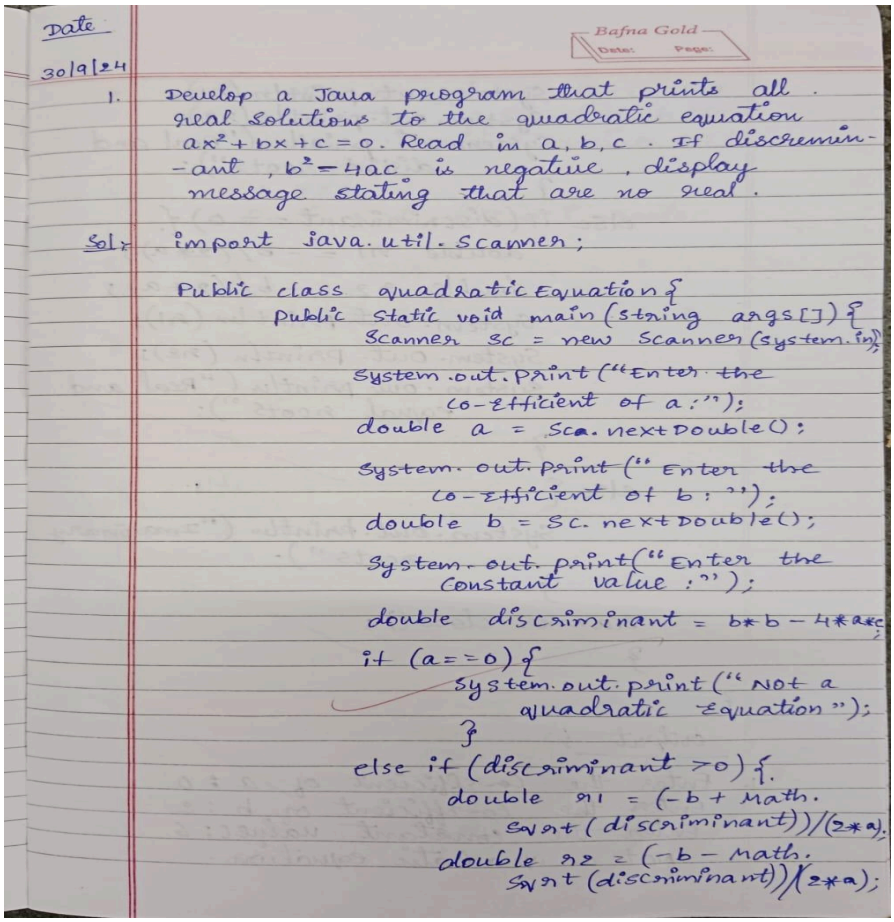
Program 1 : Quadratic Equation

```
import java.util.Scanner;

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

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

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

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

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

		if(a==0){
			System.out.print("Not a quadratic equation");
			}

		else if(discriminant>0){
			double r1 = (-b + Math.sqrt(discriminant))/(2*a);
			double r2 = (-b - Math.sqrt(discriminant))/(2*a);
			System.out.println(r1);
			System.out.println(r2);
			System.out.println("Real and distinct roots");
			}

		else if(discriminant==0){
			double r1 = -b/(2*a);
			double r2 = -b/(2*a);
			System.out.println(r1);
			System.out.println(r2);
			System.out.println("Real and equal roots");
			}
		else{
			System.out.println("Imaginary roots");
			}
	scanner.close();
	}
}
```

Output :

```
PS D:\1BM23CS001-Java\1BM23CS173> javac quadraticEquation.java
PS D:\1BM23CS001-Java\1BM23CS173> java quadraticEquation
Enter the coefficient of a : 0
Enter the coefficient of b : 2
Enter the constant value c : 6
Not a quadratic equation
PS D:\1BM23CS001-Java\1BM23CS173> javac quadraticEquation.java
PS D:\1BM23CS001-Java\1BM23CS173> java quadraticEquation
Enter the coefficient of a : 2
Enter the coefficient of b : 6
Enter the constant value c : 1
-0.17712434446770464
-2.8228756555322954
Real and distinct roots
PS D:\1BM23CS001-Java\1BM23CS173> java quadraticEquation.java
Enter the coefficient of a : 2
Enter the coefficient of b : 4
Enter the constant value c : 2
-1.0
-1.0
Real and equal roots
PS D:\1BM23CS001-Java\1BM23CS173>
```

Source Code :



Date
30/9/24

1. Develop a Java program that prints all real solutions to the quadratic equation $ax^2+bx+c=0$. Read in a, b, c. If discriminant, b^2-4ac, is negative, display message stating that are no real.

Sol:

```
import java.util.Scanner;

Public class quadraticEquation{
    Public static void main(string args[]){
        Scanner sc = new Scanner(system.in);
        System.out.print("Enter the
            co-efficient of a:");
        double a = sca.nextDouble();
        System.out.print("Enter the
            co-efficient of b:");
        double b = sc.nextDouble();
        System.out.print("Enter the
            constant value:");
        double discriminant = b*b-4*a*c;
        if (a==0){
            system.out.print("Not a
                quadratic Equation");
        }
        else if (discriminant >0){
            double r1 = (-b + Math.
                sqrt(discriminant))/(2*a);
            double r2 = (-b - Math.
                sqrt(discriminant))/(2*a);
```

```
        System.out.println(r1);
        System.out.println(r2);
        System.out.println("Real and
                distint-roots");
        }
    else if(discriminant == 0){
        double r1 = -b/(2*a);
        double r2 = -b/(2*a);
        System.out.println(r1);
        System.out.println(r2);
        System.out.println("Real and
                equal roots");
        }
    else{
        System.out.println("Imaginary
                roots");
        }
    scanner.close();
    }
}
```

output :

1. Enter the co-efficient of a : 0
   Enter the co-efficient of b : 2
   Enter the constant value c : 6.
   Not a quadratic equation.

2. Enter the co-efficient of a : 2.
   Enter the co-efficient of b : 6
   Enter the constant value c : 1.
   -0.177
   -2.822.
   Real and distinct roots.

3. Enter the coefficient of a : 2
   Enter the coefficient of b : 4
   Enter the constant value c : 2
   -1.0
   -1.0

Program 2 : Student SGPA

```
import java.util.Scanner;

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


    Student(int numSubjects) {
        this.numSubjects = numSubjects;
        credits = new int[numSubjects];
        marks = new int[numSubjects];
    }


    void acceptDetails() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter USN: ");
        usn = scanner.nextLine();
        System.out.print("Enter Name: ");
        name = scanner.nextLine();
        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            credits[i] = scanner.nextInt();
            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            marks[i] = scanner.nextInt();
        }
    }


    void displayDetails() {
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        for (int i = 0; i < numSubjects; i++) {
            System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i] + ", Marks: " + marks[i]);
        }
    }


    double calculateSGPA() {
        int totalCredits = 0;
        double totalPoints = 0.0;
```

```
    for (int i = 0; i < numSubjects; i++) {
        int gradePoint = getGradePoint(marks[i]);
        totalPoints += gradePoint * credits[i];
        totalCredits += credits[i];
    }

    return totalPoints / totalCredits;
}


int getGradePoint(int marks) {
    if (marks >= 90) return 10;
    else if (marks >= 80) return 9;
    else if (marks >= 70) return 8;
    else if (marks >= 60) return 7;
    else if (marks >= 50) return 6;
    else if (marks >= 40) return 5;
    else return 0;
}

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


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


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


    Student[] students = new Student[numStudents];


    for (int i = 0; i < numStudents; i++) {
        System.out.println("\nEntering details for Student " + (i + 1));
        students[i] = new Student(numSubjects);
        students[i].acceptDetails();
    }


    for (int i = 0; i < numStudents; i++) {
        System.out.println("\nDetails of Student " + (i + 1));
        students[i].displayDetails();
        double sgpa = students[i].calculateSGPA();
        System.out.printf("SGPA: %.2f\n", sgpa);
```

```
        }
            System.out.println("Name : MADHAN K");
            System.out.println("USN : 1BM23CS173");
    }
}
```

Output :

```
Enter number of students: 2
Enter number of subjects per student: 8

Entering details for Student 1
Enter USN: 1BM23CS173
Enter Name: MADHAN K
Enter credits for subject 1: 4
Enter marks for subject 1: 97
Enter credits for subject 2: 4
Enter marks for subject 2: 88
Enter credits for subject 3: 3
Enter marks for subject 3: 93
Enter credits for subject 4: 3
Enter marks for subject 4: 87
Enter credits for subject 5: 3
Enter marks for subject 5: 82
Enter credits for subject 6: 1
Enter marks for subject 6: 90
Enter credits for subject 7: 1
Enter marks for subject 7: 94
Enter credits for subject 8: 1
Enter marks for subject 8: 85

Entering details for Student 2
Enter USN: 1BM23CS155
Enter Name: Saketh Ram
Enter credits for subject 1: 4
Enter marks for subject 1: 76
Enter credits for subject 2: 4
Enter marks for subject 2: 91
Enter credits for subject 3: 3
Enter marks for subject 3: 94
Enter credits for subject 4: 3
Enter marks for subject 4: 82
Enter credits for subject 5: 3
Enter marks for subject 5: 78
Enter credits for subject 6: 1
Enter marks for subject 6: 71
Enter credits for subject 7: 1
Enter marks for subject 7: 83
Enter credits for subject 8: 1
Enter marks for subject 8: 90
```

```
Details of Student 1
USN: 1BM23CS173
Name: MADHAN K
Subject 1 - Credits: 4, Marks: 97
Subject 2 - Credits: 4, Marks: 88
Subject 3 - Credits: 3, Marks: 93
Subject 4 - Credits: 3, Marks: 87
Subject 5 - Credits: 3, Marks: 82
Subject 6 - Credits: 1, Marks: 90
Subject 7 - Credits: 1, Marks: 94
Subject 8 - Credits: 1, Marks: 85
SGPA: 9.45

Details of Student 2
USN: 1BM23CS155
Name: Saketh Ram
Subject 1 - Credits: 4, Marks: 76
Subject 2 - Credits: 4, Marks: 91
Subject 3 - Credits: 3, Marks: 94
Subject 4 - Credits: 3, Marks: 82
Subject 5 - Credits: 3, Marks: 78
Subject 6 - Credits: 1, Marks: 71
Subject 7 - Credits: 1, Marks: 83
Subject 8 - Credits: 1, Marks: 90
SGPA: 9.00
Name : MADHAN K
USN : 1BM23CS173
```

Source Code :

6/10/24

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.

Sol:-

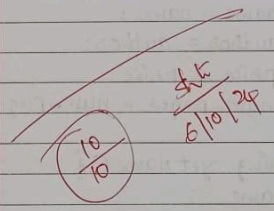
```
import java.util.Scanner;

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

    Student(int numSubjects) {
        this.numSubjects = numSubjects;
        credits = new int[numSubjects];
        marks = new int[numSubjects];
    }

    void acceptDetails() {
        Scanner scanner = new Scanner
                          (System.in);
        System.out.print("Enter USN: ");
        usn = scanner.nextLine();
        System.out.print("Enter Name:");
        name = scanner.nextLine();
        for(int i=0; i<numSubjects; i++){
            System.out.print("Enter credits
for Subject "+(i+1)+":");
            credits[i] = scanner.nextInt();
            System.out.print("Enter marks for
Subject "+(i+1)+":");
            marks[i] = scanner.nextInt();
        }
    }

    void displayDetails() {
        System.out.println("USN:"+usn);
        System.out.println("Name:"+name);
        for(int i=0; i<numSubjects; i++) {
            System.out.println("Subject"+
            (i+1)+"- Credits:"+credits[i]
            +", Marks:"+marks[i]);
        }
    }

    double calculateSGPA() {
        int totalCredits = 0;
        double totalPoints = 0.0;

        for(int i=0; i<numSubjects; i++){
            int gradePoint ~~(marks[i])~~;
            gradePoint = getGradePoint
                         (marks[i]);
            totalPoints += gradePoint *
                           credits[i];
            totalCredits += credits[i];
        }
        return totalPoints/totalCredits;
    }
```

```
    int getGradePoint(int marks) {
        if(marks >= 90) return 10;
        else if(marks >= 80) return 9;
        else if(marks >= 70) return 8;
        else if(marks >= 60) return 7;
        else if(marks >= 50) return 6;
        else if(marks >= 40) return 5;
        else return 0;
    }

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

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

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

        Student[] students = new Student[num
                             Students];

        for(int i=0; i<numStudents; i++) {
            System.out.println("\n Entering details for
            Student "+(i+1));
            students[i] = new Student(numSubjects);
            students[i].acceptDetails();
        }

        for(int i=0; i<numStudents; i++) {
            System.out.println("\n Details of Student"
                               +(i+1));
            students[i].displayDetails();
            double sgpa = students[i].calculateSGPA();
            System.out.printf("SGPA: %.2f\n", sgpa);
        }
        System.out.println("Name: MADHAN K");
        System.out.println("USN: 1BM23CS173");
    }
}
```

Output :-

```
Enter number of students: 2
Enter number of subjects per Student: 8

Entering details for Student 1
Enter USN: 1BM23CS173
Enter Name: MADHAN K
Enter Credits for Subject 1: 4
Enter marks for Subject 1: 97
Enter ~~ma~~credits for Subject 2: 4
Enter marks for Subject 2: 88
Enter Credits for Subject 3: 3
Enter marks for Subject 3: 93
Enter Credits for Subject 4: 3
Enter marks for Subject 4: 87
Enter Credits for Subject 5: 3
Enter marks for Subject 5: 82
Enter Credits for Subject 6: 1
Enter marks for Subject 6: 90
```

```
Enter credits for Subject 7 : 1
Enter marks for Subject 7 : 94
Enter credits for Subject 8 : 1
Enter marks for Subject 8 : 85

Entering details for student 2
Enter USN : 1BM23CS143
Enter Name : GANESH
Enter Credits for Subject 1 : 4
Enter marks for subject 1: 76
Enter Credits for Subject 2 : 4
Enter marks for Subject 2 : 91
Enter credits for Subject 3 : 3
Enter marks for Subject 3 : 94
Enter Credits for subject 4 : 3
Enter marks for Subject 4 : 82
Enter Credits for Subject 5 : 3
Enter marks for Subject 5 : 78
Enter Credits for Subject 6 : 1
Enter marks for Subject 6 : 71
Enter Credits for Subject 7 : 1
Enter marks for Subject 7 : 83
Enter Credits for Subject 8 : 1
Enter marks for Subject 8 : 90

Details of Student 1
USN : 1BM23CS173
Name : MADHAN.K
Subject 1 - Credits : 4 , Marks : 97
Subject 2 - Credits : 4 , Marks : 88
Subject 3 - Credits : 3 , Marks : 93
Subject 4 - Credits : 3 , Marks : 87
Subject 5 - Credits : 3 , Marks : 82
Subject 6 - Credits : 1 , Marks : 90
Subject 7 - Credits : 1 , Marks : 94
Subject 8 - Credits : 1 , Marks : 85
SGPA : 9.45

Details for Student 2
USN : 1BM23CS143
Name : GANESH
Subject 1 - Credits : 4 , Marks : 76
Subject 2 - Credits : 4 , Marks : 91
Subject 3 - Credits : 3 , Marks : 94
Subject 4 - Credits : 3 , Marks : 82
Subject 5 - Credits : 3 , Marks : 78
Subject 6 - Credits : 1 , Marks : 71
Subject 7 - Credits : 1 , Marks : 83
Subject 8 - Credits : 1 , Marks : 90.
```



Program 3 : Book Details

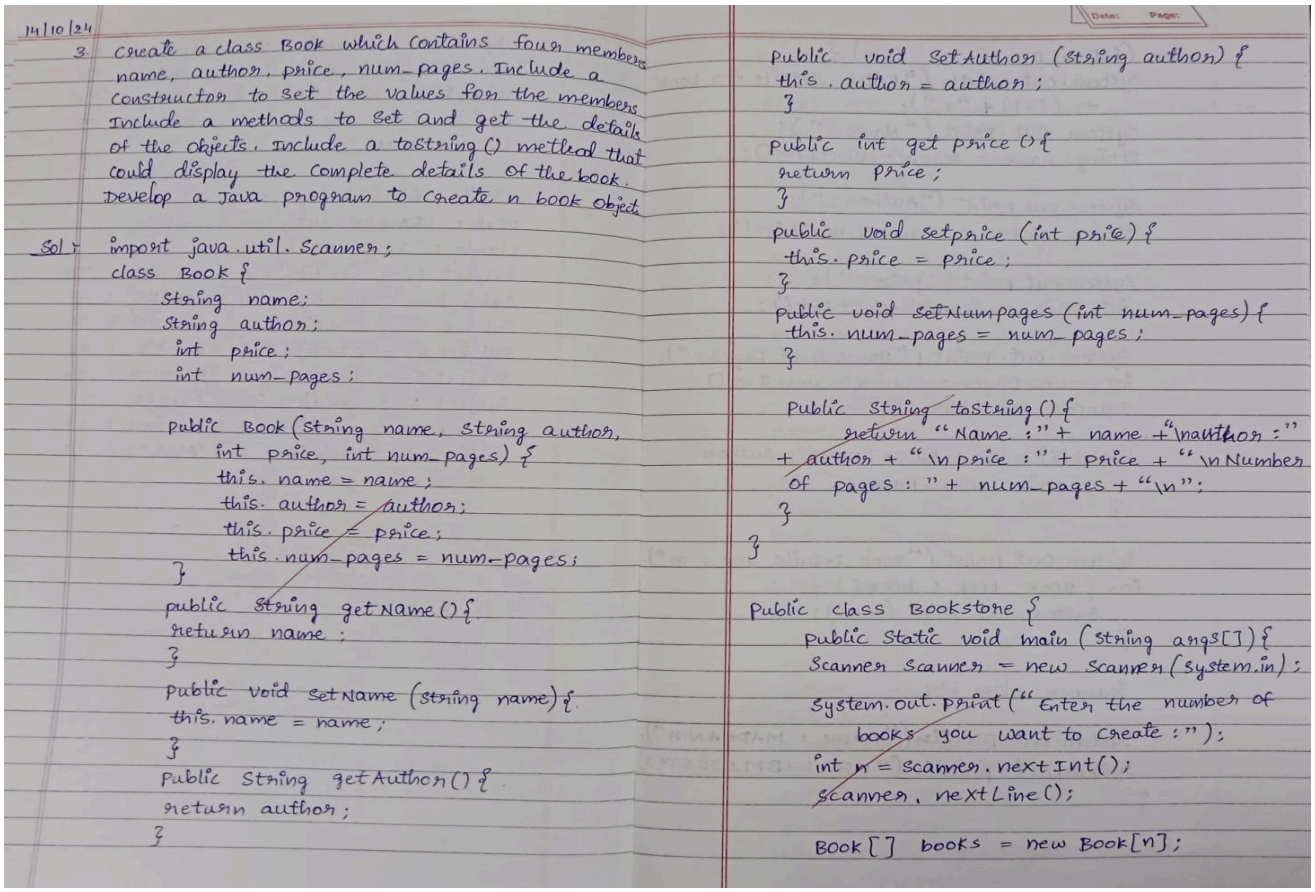
```
import java.util.Scanner;

class Book {
String name;
String author;
int price;
int num_pages;

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

public String getName() {
return name;
}

public void setName(String name) {
this.name = name;
}

public String getAuthor() {
return author;
}

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

public int getPrice() {
return price;
}

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

public int getNumPages() {
return num_pages;
}

public void setNumPages(int num_pages) {
this.num_pages = num_pages;
```

```
}

public String toString() {
       return "Name: " + name + "\nAuthor: " + author + "\nPrice: " + price + "\nNumber of Pages: " +
num_pages + "\n";
   }
}


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

System.out.print("Enter the number of books you want to create : ");

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

Book[] books = new Book[n];

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

System.out.println("Enter details for book " + (i + 1) + ":");

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

System.out.print("Author: ");
String author = scanner.nextLine();

System.out.print("Price: ");
int price = scanner.nextInt();

System.out.print("Number of Pages: ");
int num_pages = scanner.nextInt();
scanner.nextLine();
books[i] = new Book(name, author, price, num_pages);
}

System.out.print("Book Details are : \n");
for(Book book : books) {
System.out.print(book);
}
scanner.close();
System.out.println("Name : MADHAN K");
System.out.println("USN : 1BM23CS173");
}}
```

Output :

```
D:\1BM23CS173>javac Bookstore.java

D:\1BM23CS173>java Bookstore
Enter the number of books you want to create : 3
Enter details for book 1:
Name: java
Author: rajesh
Price: 1200
Number of Pages: 854
Enter details for book 2:
Name: c++
Author: praveen
Price: 1000
Number of Pages: 400
Enter details for book 3:
Name: java script
Author: Madhan K
Price: 2000
Number of Pages: 980
Book Details are :
Name: java
Author: rajesh
Price: 1200
Number of Pages: 854
Name: c++
Author: praveen
Price: 1000
Number of Pages: 400
Name: java script
Author: Madhan K
Price: 2000
Number of Pages: 980
Name : MADHAN K
USN : 1BM23CS173
```

Source Code :



14/10/24

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 a 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.

Sol:

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

    public Book(String name, String author,
        int price, int num_pages) {
        this.name = name;
        this.author = author;
        this.price = price;
        this.num_pages = num_pages;
    }
    public String getName() {
    return name;
    }
    public void setName (String name) {
    this.name = name;
    }
    Public String getAuthor() {
    return author;
    }
    public void setAuthor (String author) {
    this.author = author;
    }
    public int getprice() {
    return price;
    }
    public void setprice (int price) {
    this.price = price;
    }
    public void setNumpages (int num_pages) {
    this.num_pages = num_pages;
    }

    Public String toString() {
        return "Name :" + name + "\nauthor :"
    + author + "\n price :" + price + "\n Number
    of pages : " + num_pages + "\n";
    }
}

Public class Bookstore {
    Public Static void main (String args[]) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the number of
        books you want to create :");
    int n = scanner.nextInt();
    scanner.nextLine();

    Book[] books = new Book[n];

    for (int i=0; i<n; i++) {
    System.out.println("Enter details for book"
        + (i+1) + ":");
    System.out.print("Name :");
    String name = scanner.nextLine();

    System.out.print("Author :");
    String author = scanner.nextLine();

    System.out.print("price :");
    int price = scanner.nextInt();

    System.out.print("Number of pages :");
    int num_pages = scanner.nextInt();
    scanner.nextLine();

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

    System.out.print("Book Details are :\n");
    for (Book book : books) {
        System.out.print(book);
    }

    scanner.close();

    System.out.println("Name : MADHAN K");
    System.out.println("USN : 1BM23CS173");
    }
}
```

output :-

```
Enter the number of books you want to
 Create : 3
Enter details for book 1 :
Name : java
Author : rajesh
price : 1200
Number of pages : 854
Enter details for book 2 :
Name : c++
Author : praveen
price : 1000
Number of pages : 400
Enter details for book 3 :
Name : java script
Auther : Madhan K
price : 2000
Number of Pages : 980
Book details are :
Name : java
Author : rajesh
price : 1200
Number of Pages : 854
Name : c++
Author : praveen
price : 1000
Number of pages : 400
Name : java script
Author : Madhan.K
price : 2000
Number of pages : 980
Name : MADHAN.K:
USN : 1BM23CS173.
```

10/10

Program 4 : Area of Shapes

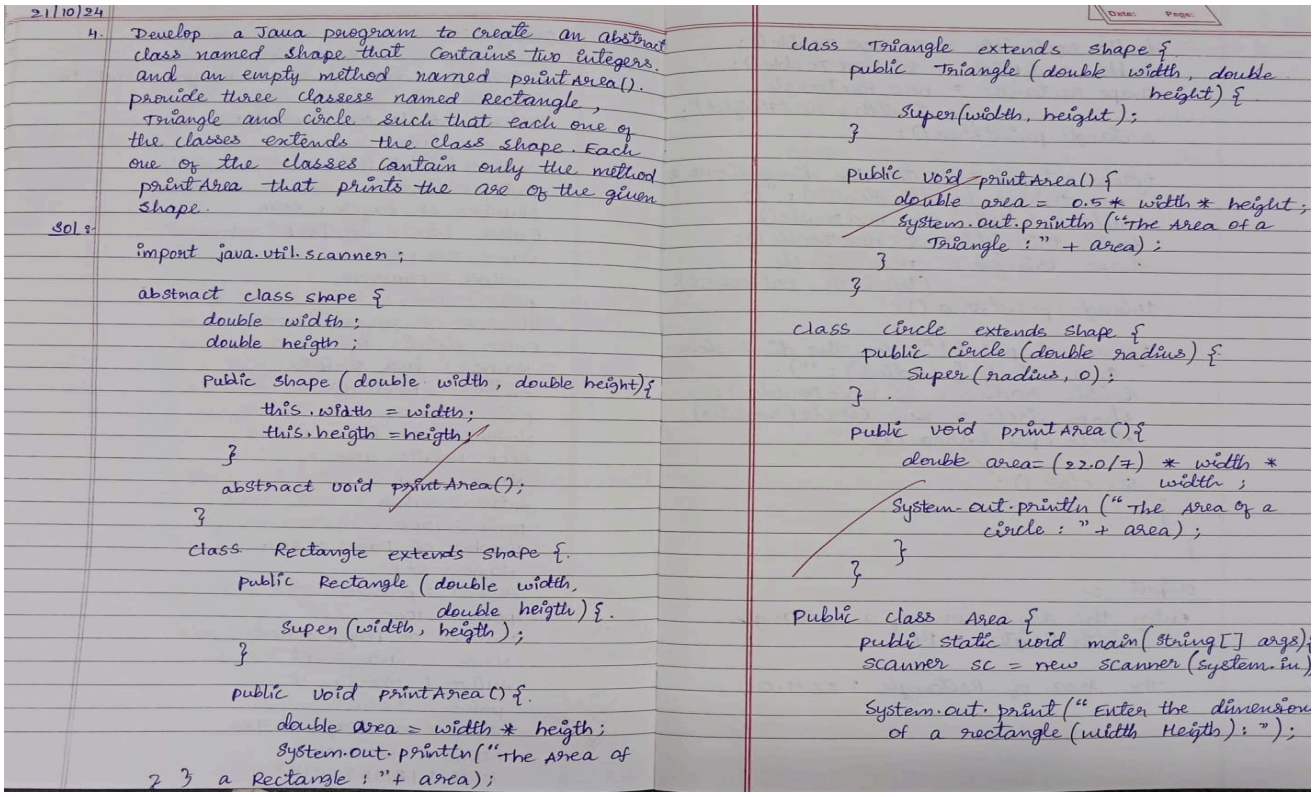
```
import java.util.Scanner;

abstract class Shape {
   double width;
   double height;

   abstract void printArea();
}

class Rectangle extends Shape {
   public Rectangle(double width, double height) {
      this.width = width;
      this.height = height;

   }


   public void printArea() {
      double area = width * height;
      System.out.println("The Area of a Rectangle: " + area);
   }
}

class Triangle extends Shape {
   public Triangle(double width, double height) {
      this.width = width;
      this.height = height;

   }


   public void printArea() {
      double area = 0.5 * width * height;
      System.out.println("The Area of a Triangle: " + area);
   }
}

class Circle extends Shape {
   public Circle(double radius) {
      this.width = radius;

   }


   public void printArea() {
```

```
        double area = (22.0 / 7) * width * width;
        System.out.println("The Area of a Circle: " + area);
    }
}

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


        System.out.print("Enter the dimensions of a rectangle (Width Height): ");
        double rectWidth = sc.nextDouble();
        double rectHeight = sc.nextDouble();
        Shape rectangle = new Rectangle(rectWidth, rectHeight);
        rectangle.printArea();


        System.out.print("Enter the dimensions of a triangle (Width Height): ");
        double triWidth = sc.nextDouble();
        double triHeight = sc.nextDouble();
        Shape triangle = new Triangle(triWidth, triHeight);
        triangle.printArea();


        System.out.print("Enter the dimensions of a circle (Radius): ");
        double radius = sc.nextDouble();
        Shape circle = new Circle(radius);
        circle.printArea();

        sc.close();
    }
}
```

Output :

```
Enter the dimensions of a rectangle (Width Height): 14
16
The Area of a Rectangle: 224.0
Enter the dimensions of a triangle (Width Height): 12
14
The Area of a Triangle: 84.0
Enter the dimensions of a circle (Radius): 16
The Area of a Circle: 804.5714285714286
```

Source Code :



21/10/24

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.

Sol :-

```
import java.util.scanner;

abstract class shape {
    double width;
    double height;

    Public shape (double width, double height);{
        this.width = width;
        this.height = height;
    }
    abstract void printArea();
}

class Rectangle extends shape {
    public Rectangle (double width,
                      double height) {
        super (width, height);
    }

    public void printArea () {
        double area = width * height;
        System.out.println("The area of
a Rectangle : "+ area);
    }
}

class Triangle extends shape {
    public Triangle (double width, double
                     height) {
        super(width, height);
    }

    Public void printArea() {
        double area = 0.5 * width * height;
        System.out.println("The Area of a
Triangle : " + area);
    }
}

class circle extends shape {
    public circle (double radius) {
        super (radius, 0);
    }
    public void printArea () {
        double area= (22.0/7) * width *
                     width;
        System.out.println("The Area of a
circle : "+ area);
    }
}

Public class Area {
    publi static void main(string[] args);
    scanner sc = new scanner (system.in)

    System.out.print("Enter the dimensions
of a rectangle (width Height): ");
    double rectwidth = sc.nextDouble();
    double rectHeigth = sc.nextDouble();
    Shape rectangle = new Rectangle
                      (rectwidth, rectHeight);
    rectangle.printArea();

    System.out.print("Enter the dimensions of
a triangle (width Height): ");
    double triwidth = sc.nextDouble();
    double triHeight = sc.nextDouble();
    Shape triangle = new Triangle
                     (triwidth, triHeight);
    triangle.printArea();

    System.Out.print("Enter the dimensions
of a circle (radius): ");
    double radius = sc.nextDouble();
    Shape circle = new circle(radius);
    circle.printArea();

    sc.close();
  }
}
```

10/10

output :-

```
Enter the dimensions of a rectangle
(width Height) : 14
                 16
the Area of Rectangle : 224.0.

Enter the dimensions of a triangle
(width Height) : 12
                 14
The Area of a Triangle : 84.0

Enter the dimensions of a circle
(radius) : 16
The Area of a circle : 804.5714285
```

21/10/24

Program 5 : Bank Savings

```
import java.util.Scanner;

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

   public Account(String customerName, String accountNumber, double initialBalance) {
      this.customerName = customerName;
      this.accountNumber = accountNumber;
      this.balance = initialBalance;
   }

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

   public double getBalance() {
      return balance;
   }

   public String getAccountDetails() {
      return "Account Number: " + accountNumber + ", Name: " + customerName + ", Balance: " +
balance;
   }
}

class SavingsAccount extends Account {
   double interestRate;

   public SavingsAccount(String customerName, String accountNumber, double initialBalance,
double interestRate) {
      super(customerName, accountNumber, initialBalance);
      this.interestRate = interestRate;
   }

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

   public void withdraw(double amount) {
      if (amount > balance) {
```

```
            System.out.println("Insufficient balance for withdrawal.");
        } else {
            balance -= amount;
            System.out.println("Withdrawn: " + amount);
        }
    }
}

class CurrentAccount extends Account {
    private static final double MINIMUM_BALANCE = 500.0;
    private static final double SERVICE_CHARGE = 50.0;

    public CurrentAccount(String customerName, String accountNumber, double initialBalance) {
        super(customerName, accountNumber, initialBalance);
    }

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

    void checkMinimumBalance() {
        if (balance < MINIMUM_BALANCE) {
            balance -= SERVICE_CHARGE;
            System.out.println("Minimum balance not maintained. Service charge of " +
SERVICE_CHARGE + " applied.");
        }
    }
}

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

        System.out.println("Bank type:");
        System.out.println("1. Savings");
        System.out.println("2. Current");
        System.out.print("Enter your choice: ");
        int type = scanner.nextInt();
        scanner.nextLine(); // Consume newline
```

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

System.out.print("Enter account number: ");
String accountNumber = scanner.nextLine();

System.out.print("Enter initial balance: ");
double initialBalance = scanner.nextDouble();

if (type == 1) {
    System.out.print("Enter interest rate: ");
    double interestRate = scanner.nextDouble();
    account = new SavingsAccount(name, accountNumber, initialBalance, interestRate);
} else if (type == 2) {
    account = new CurrentAccount(name, accountNumber, initialBalance);
} else {
    System.out.println("Invalid account type.");
    return;
}

while (true) {
    System.out.println("\nChoose an operation:");
    System.out.println("1. Deposit");
    System.out.println("2. Display Balance");
    System.out.println("3. Withdraw");
    if (account instanceof SavingsAccount) {
        System.out.println("4. Compute and Deposit Interest");
    }

    int choice = scanner.nextInt();

    switch (choice) {
        case 1:
            System.out.print("Enter deposit amount: ");
            double depositAmount = scanner.nextDouble();
            account.deposit(depositAmount);
            break;
        case 2:
            System.out.println(account.getAccountDetails());
            break;
        case 3:
            System.out.print("Enter withdrawal amount: ");
            double withdrawAmount = scanner.nextDouble();
            if (account instanceof SavingsAccount) {
                ((SavingsAccount) account).withdraw(withdrawAmount);
            } else {
                ((CurrentAccount) account).withdraw(withdrawAmount);
```

```
                }
                break;
            case 4:
                if (account instanceof SavingsAccount) {
                    ((SavingsAccount) account).computeAndDepositInterest();
                } else {
                    System.out.println("This option is not available for Current Accounts.");
                }
                break;

                    default:
                System.out.println("Invalid choice. Please try again.");
        }
    }
  }
}
```

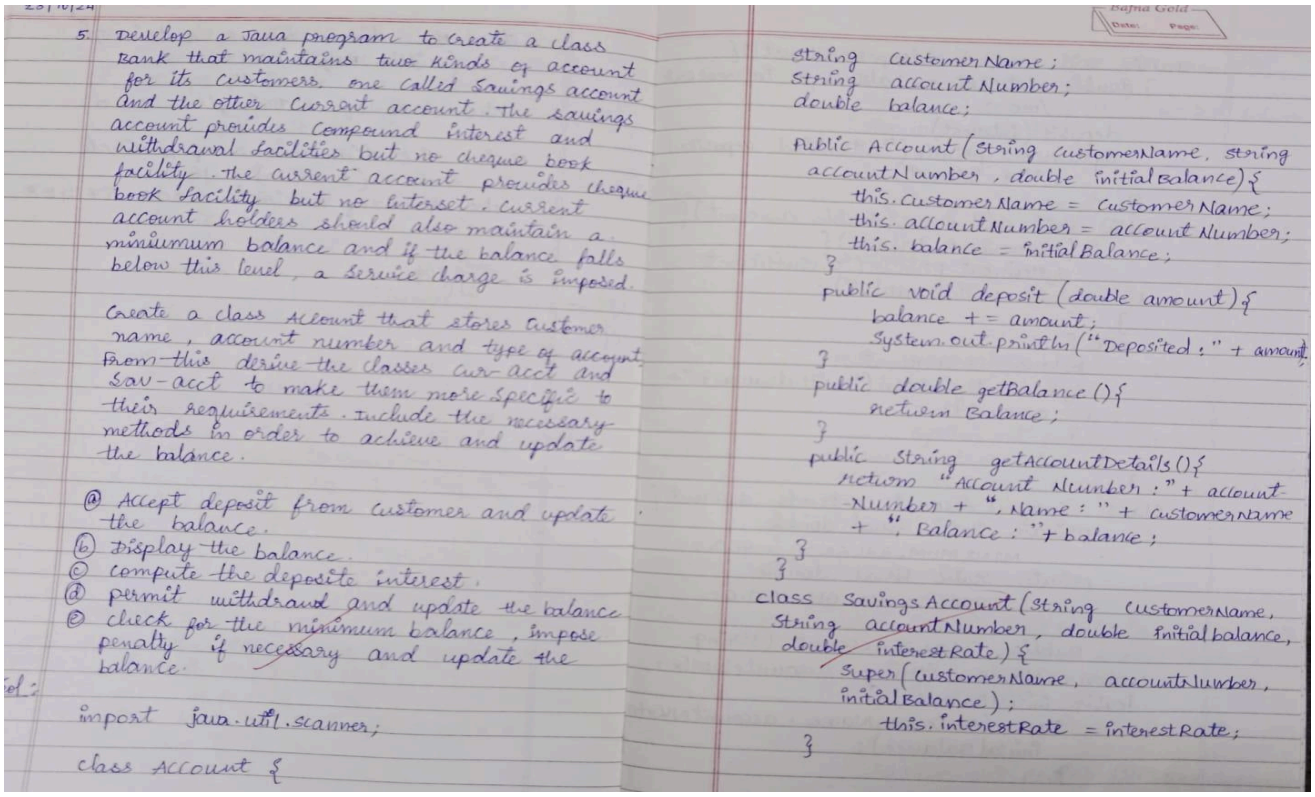
Output :

```
Bank type:
1. Savings
2. Current
Enter your choice: 1
Enter customer name: Madhan k
Enter account number: 1234
Enter initial balance: 50000
Enter interest rate: 2

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
1
Enter deposit amount: 40000
Deposited: 40000.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
2
Account Number: 1234, Name: Madhan k, Balance: 90000.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
3
Enter withdrawal amount: 20000
Withdrawn: 20000.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
4
Deposited: 1400.0
Interest deposited: 1400.0
```

```
Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
2
Account Number: 1234, Name: Madhan k, Balance: 71400.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
3
Enter withdrawal amount: 80000
Insufficient balance for withdrawal.
```

```
Bank type:
1. Savings
2. Current
Enter your choice: 2
Enter customer name: Madhan k
Enter account number: 2345
Enter initial balance: 50000

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
1
Enter deposit amount: 20000
Deposited: 20000.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
2
Account Number: 2345, Name: Madhan k, Balance: 70000.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
3
Enter withdrawal amount: 30000
Withdrawn: 30000.0

Choose an operation:
1. Deposit
2. Display Balance
3. Withdraw
2
Account Number: 2345, Name: Madhan k, Balance: 40000.0
```

Source Code :

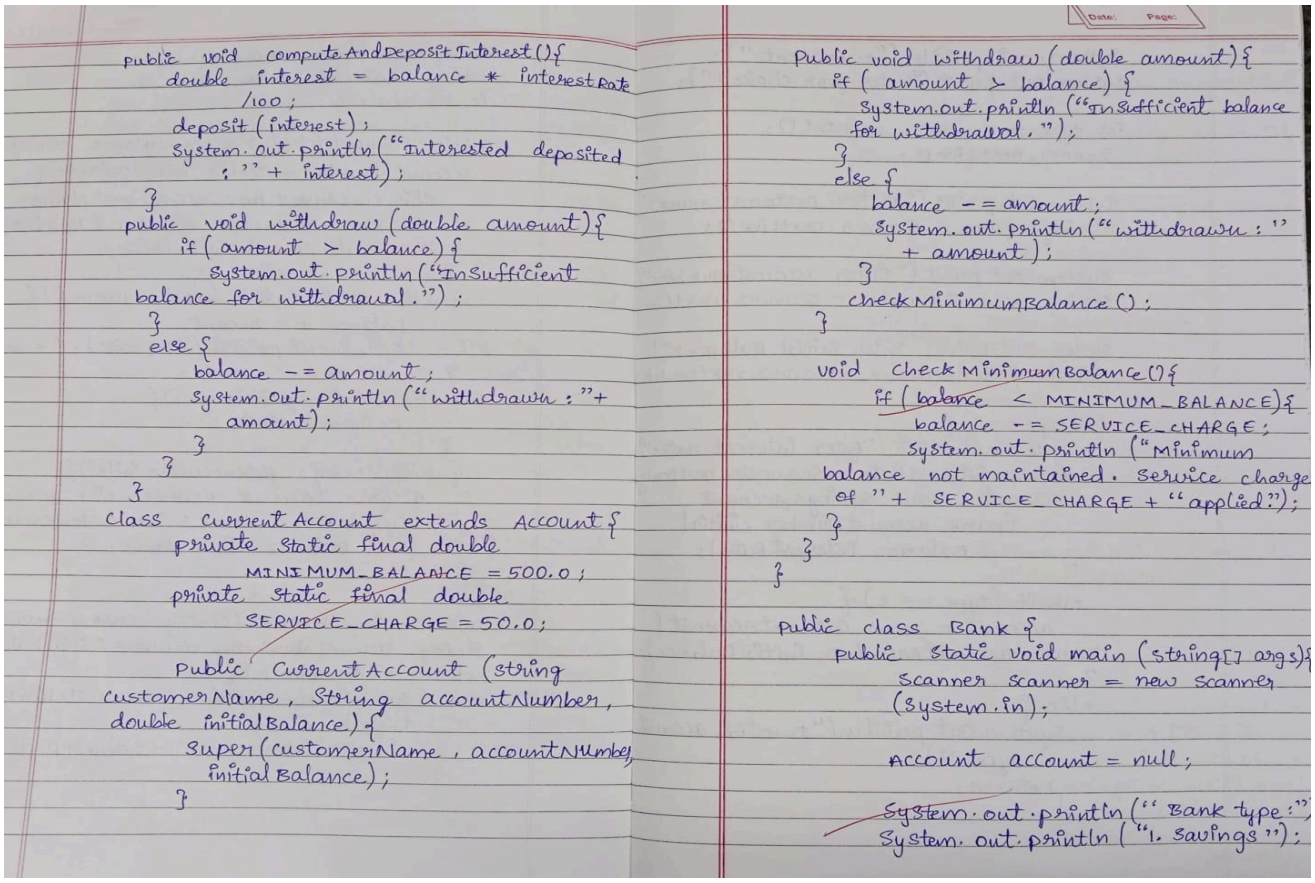


5. Develop a Java program to create a class Bank that maintains two kinds of account for its customers one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book 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 and update the balance.

ⓐ Accept deposit from customer and update the balance.
ⓑ Display the balance.
ⓒ compute the deposite interest.
ⓓ permit withdrawal and update the balance
ⓔ check for the minimum balance, impose penalty if necessary and update the balance.

```
import java.util.Scanner;

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

    Public Account(String customerName, String
     accountNumber, double initialBalance){
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.balance = initialBalance;
    }
    public void deposit(double amount){
        balance += amount;
        System.out.println("Deposited: " + amount);
    }
    public double getBalance(){
        return balance;
    }
    public String getAccountDetails(){
        return "Account Number: " + accountNumber + ", Name: " + customerName
        + ", Balance: " + balance;
    }
}
class SavingsAccount(String customerName,
 String accountNumber, double initialbalance,
 double interestRate){
    super(customerName, accountNumber,
    initialBalance);
        this.interestRate = interestRate;
    }
```



```
public void computeAndDepositInterest(){
    double interest = balance * interestRate
        /100;
    deposit(interest);
    System.out.println("interested deposited
        :" + interest);
}
public void withdraw(double amount){
    if(amount > balance){
        System.out.println("InSufficient
 balance for withdrawal.");
    }
    else{
        balance -= amount;
        System.out.println("withdrawn:"+
            amount);
    }
}
}
class CurrentAccount extends Account{
    private static final double
        MINIMUM_BALANCE = 500.0;
    private static final double
        SERVICE_CHARGE = 50.0;

    Public CurrentAccount(string
 customerName, String accountNumber,
 double initialBalance){
        super(customerName, accountNumber,
         initialBalance);
    }
Public void withdraw(double amount){
    if( amount > balance){
        System.out.println("InSufficient balance
        for withdrawal.");
    }
    else{
        balance -= amount;
        System.out.println("withdrawn:"
            + amount);
    }
    checkMinimumBalance();
}

void checkMinimumBalance(){
    if(balance < MINIMUM_BALANCE){
        balance += SERVICE_CHARGE;
        System.out.println("Minimum
 balance not maintained. Service charge
 of" + SERVICE_CHARGE + "applied.");
    }
}
}

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

        Account account = null;

        System.out.println("Bank type:");
        System.out.println("1. savings");
```

```
System.out.println("2. Current");
System.out.print("Enter your choice:");

int type = scanner.nextInt();
scanner.nextLine();

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

System.out.print("Enter account number:");
String accountNumber = scanner.nextLine();

System.out.print("Enter initial Balance:");
double initialBalance = scanner.nextDouble();

if (type == 1) {
    System.out.print("Enter interest rate:");
    double interestRate = scanner.nextDouble();
    account = new SavingsAccount
        (name, accountNumber, initial
        Balance, interestRate);
}
else if (type == 2) {
    account = new CurrentAccount(
name, accountNumber, initialBalance);
}
else {
    System.out.println("Invalid account
        type.");
    return;
}
```

```
while (true) {
    System.out.println("\n choose an operation:");
    System.out.println("1. Deposit");
    System.out.println("2. Display Balance");
    System.out.println("3. withdraw");

    if (account instance of SavingsAccount){
        System.out.println("4. compute and
            Deposit Interest");
    }

int choice = scanner.nextInt();
switch (choice) {
    case 1: System.out.print("Enter
        deposit amount:");
        double depositAmount =
            scanner.nextDouble();
        account.deposit(depositAmount);
        break;

    case 2: System.out.println(account.
        getAccountDetails());
        break;

    case 3: System.out.print("Enter
        withdrawal amount:");
        double withdrawAmount =
            scanner.nextDouble();
        if (account instance of
            SavingsAccount) {
        ((SavingsAccount) account).
            withdraw(withdraw
                Amount);
    }
```

```
else {
((CurrentAccount)account).withdraw
    (withdrawAmount);
}
break;

Case 4: if (account instance of SavingsAccount) {
    ((SavingsAccount) account).
    compute And Deposit Interest();
    }
    else {
    system.out.println("this option
is not available for Current
Accounts.");
    }
    break;

default: system.out.println("Invalid
    choice. please try again.");
    }
  }
 }
}
```

Output :-

①

```
Bank type :
1. Savings
2. Current
Enter your choice : 1
Enter customer name : Madhan K
Enter account number : 1234
Enter initial balance : 50000
Enter interest rate : 2

Choose an operation :
1. Deposit
2. Display Balance
3. withdraw
4. Compute and Deposit Interest
1
Enter deposit amount : 40000
Deposited : 40000.0

choose an operation :
1. Deposit
2. Display Balance
3. withdraw
4. Compute and Deposit Interest
2
Account Number : 1234, Name : Madhan K,
Balance : 90000.0

Choose an operation :
1. Deposit
2. Display Balance
3. withdraw
4. Compute and Deposite Interest
3
Enter withdrawal amount : 20000.0
withdrawn : 20000.0
```

Choose an operation :
1. ~~Display~~ Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposite Interest
4
Deposited : 1400.0
Interest deposited : 1400.0.

Choose an operation :
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
2
Account Number : 1234, Name : Madhan K,
Balance : 71400.0.

Choose an operation :
1. Deposit
2. Display Balance
3. Withdraw
4. Compute and Deposit Interest
3
Enter withdrawal Amount : 80000
Insufficient balance for withdrawal.

(2)
Bank type :
1. Savings
2. Current
Enter your choice : 2
Enter Customer name : Madhan K
Enter account number : 2345
Enter initial balance : 50000

Choose an operation :
1. Deposit
2. Display Balance
3. Withdraw
1
Enter deposit amount : 20000
Deposited : 20000.0

Choose an operation :
1. Deposit
2. Display Balance
3. Withdraw
2
Account Number : 2345, Name : Madhan K,
Balance : 70000.0

Choose an operation :
1. Deposit
2. Display Balance
3. Withdraw
3
Enter withdrawal amount : 30000
Withdrawn : 30000.0

Choose an operation :
1. Deposit
2. Display Balance
3. Withdraw
2
Account Number : 2345, Name : Madhan K,
Balance : 40000.0.

Program 6 : Packages

```
package cie;

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

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

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

```
package cie;

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

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

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

```
package see;

import cie.Student;

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

   public External(String usn, String name, int sem, int[] seeMarks) {
      super(usn, name, sem);
      this.seeMarks = seeMarks;
   }
}


import cie.*;
import see.*;
import java.util.Scanner;

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

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

      Internals[] internalStudents = new Internals[n];
      External[] externalStudents = new External[n];


      for (int i = 0; i < n; i++) {
         System.out.println("Enter details for Student " + (i + 1) + ":");

         System.out.print("USN: ");
         String usn = scanner.next();
         System.out.print("Name: ");
         String name = scanner.next();
         System.out.print("Semester: ");
         int sem = scanner.nextInt();

         int[] internalMarks = new int[5];
         System.out.println("Enter 5 internal marks: ");
         for (int j = 0; j < 5; j++) {
            internalMarks[j] = scanner.nextInt();
         }
         internalStudents[i] = new Internals(usn, name, sem, internalMarks);

         int[] seeMarks = new int[5];
```

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


        System.out.println("\nFinal Marks of Students:");
        for (int i = 0; i < n; i++) {
            System.out.println("\nDetails of Student " + (i + 1) + ":");
            internalStudents[i].display();

            int[] finalMarks = new int[5];
            for (int j = 0; j < 5; j++) {
                finalMarks[j] = internalStudents[i].internalMarks[j] + (externalStudents[i].seeMarks[j] / 2);
            }

            System.out.println("Final Marks in 5 Courses:");
            for (int mark : finalMarks) {
                System.out.print(mark + " ");
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

Output :

```
D:\package>java FinalMarksCalculator
Enter number of students: 3
Enter details for Student 1:
USN: 1BM23CS148
Name: Rohan
Semester: 3
Enter 5 internal marks:
40
45
44
47
50
Enter 5 SEE marks:
78
88
90
94
96
Enter details for Student 2:
USN: 1BM23CS203
Name: Soham
Semester: 3
Enter 5 internal marks:
47
48
43
44
47
Enter 5 SEE marks:
98
97
91
88
89
Enter details for Student 3:
USN: 1BM23CS258
Name: Praveen
Semester: 3
Enter 5 internal marks:
45
43
40
47
50
Enter 5 SEE marks:
89
83
90
79
92

Final Marks of Students:

Details of Student 1:
USN: 1BM23CS148
Name: Rohan
Semester: 3
Final Marks in 5 Courses:
79 89 89 94 98

Details of Student 2:
USN: 1BM23CS203
Name: Soham
Semester: 3
Final Marks in 5 Courses:
96 96 88 88 91

Details of Student 3:
USN: 1BM23CS258
Name: Praveen
Semester: 3
Final Marks in 5 Courses:
89 84 85 86 96
```

Source Code :

11/11/24

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.

Sol :

```
Package cie;
public class Student {
    public String usn;
    public String name;
    public int sem;

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

    Public void display (){
        System.out.println("USN : "+usn);
        System.out.println("Name :"+name);
        System.out.println("semester : "+sem);
    }
}
```

Bafna Gold

```
Package cie;
public class Internals extends Student {
    public int[] internalMarks = new int[5];
    public Internals (String usn, String name,
int sem, int [] internalMarks){
        super(usn, name, sem);
        this.internalMarks = internalMarks;
    }
}
```

```
package see;
import cie.student;
public class External extends Student {
    public int[] seeMarks = new int[5];
    public External (String usn, String name,
int sem, int [] seeMarks) {
        super(usn, name, sem);
        this.seeMarks = seeMarks;
    }
}
```

```
import cie.*;
import see.*;
import java.util.Scanner;

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

        System.out.print("Enter number of
Students : ");
        int n = scanner.nextInt();
```

```
Internals [] internalStudents = new
        Internals [n];
External [] externalStudents = new
        External [n];

for (int i=0; i<n ; i++){
    System.out.println("Enter details for
        Student " + (i+1) + ": ");

    System.out.print("USN: ");
    String usn = scanner.next();
    System.out.print("Name:");
    String name = scanner.nextLine();
    System.out.print("Semester :");
    int sem = scanner.nextInt();

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

    internalStudents[i] = new Internals
(usn, name, sem, internalMarks);

    int [] seeMarks = new int[5]
    System.out.println("Enter 5 SEE
        marks :");
    for (int j=0; j<5; j++){
        seeMarks[j] = scanner.nextInt();
    }
```

```
    externalStudents[i] = new External (usn,
        name, sem, seeMarks);
}

System.out.println("\n Final Marks of
    Student:");
for( int i=0; i<n; i++){
    System.out.println("\n Details of
        Student "+(i+1)+":");

    internalStudents[i].display();

    int [] finalMarks = new int[5];
    for (int j=0; j<5; j++){
        finalMarks[j] = internalStudents
[i].internalMarks[j] + (external
        Students[i].seeMarks[j] /2);
    }

    System.out.println("Final Marks in
        5 Courses :");
    for (int mark : finalMarks){
        System.out.print(mark+" ");
    }
    System.out.println();
}
Scanner.close();
}
}
```

output :-

```
Enter number of Students : 3
Enter details for Student 1:
USN : 1BM23CS148
Name : Rohan
Semester : 3
Enter 5 internal marks:
40
45
44
47
50
Enter 5 SEE marks:
78
88
90
94
96
Enter details for Student 2:
USN : 1BM23CS203
Name : Soham
Semester : 3
Enter 5 internal marks:
47
48
43
44
47
Enter 5 SEE marks:
98
97
91
88
89
Enter details for Student 3:
USN : 1BM23CS258
Name : Praveen
Semester : 3
Enter 5 internal marks:
45
43
40
47
50
Enter 5 SEE marks:
89
83
90
79
92

Final Marks of Students:

Details of Student 1:
USN : 1BM23CS148
Name : Rohan
Semester : 3
Final Marks in 5 Courses : 79 89 89 94 98
Details of Student 2:
USN : 1BM23CS203
Name : Soham
Semester : 3
Final Marks in 5 Courses : 96 96 88 88 91
Details Of Student 3:
USN : 1BM23CS258
Name : Praveen
Semester : 3
Final Marks in 5 Courses : 89 84 85 86 96.
```

Program 7 : Exceptional Handling

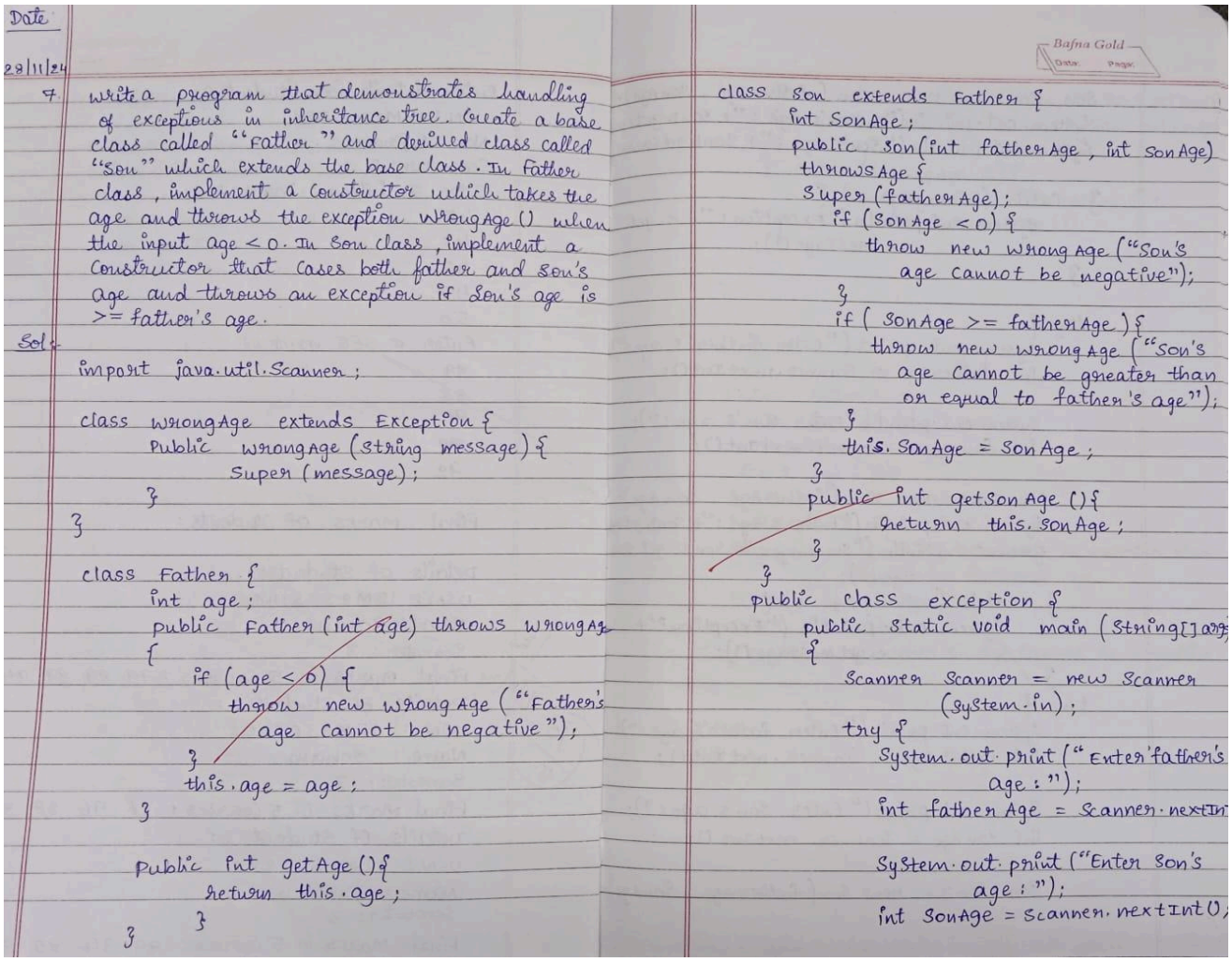
```
import java.util.Scanner;

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;
   }

   public int getAge() {
      return this.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 int getSonAge() {
      return this.sonAge;
   }
}

public class exception {
```

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

        try {
            System.out.print("Enter father's age: ");
            int fatherAge = scanner.nextInt();

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

            Son son1 = new Son(fatherAge, sonAge);
            System.out.println("Father's Age: " + son1.getAge());
            System.out.println("Son's Age: " + son1.getSonAge());

        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }

        try {
            System.out.print("Enter father's age: ");
            int fatherAge = scanner.nextInt();

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

            Son son2 = new Son(fatherAge, sonAge);

        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }

        try {
            System.out.print("Enter father's age: ");
            int fatherAge = scanner.nextInt();

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

            Son son3 = new Son(fatherAge, sonAge);

        } catch (WrongAge e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
```

Output :

```
D:\1BM23CS173>javac exception.java

D:\1BM23CS173>java exception
Enter father's age: 42
Enter son's age: 20
Father's Age: 42
Son's Age: 20
Enter father's age: 20
Enter son's age: 30
Exception: Son's age cannot be greater than or equal to father's age
Enter father's age: -20
Enter son's age: 10
Exception: Father's age cannot be negative
```

Source Code :



Date
29/11/24

7. Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge() when the input age < 0. In Son class, implement a constructor that cases both father and son's age and throws an exception if Son's age is >= father's age.

Sol:

```
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;
    }

    Public int getAge (){
        return this.age;
    }
}
```

Bafna Gold

```
class Son extends Father {
    int SonAge;
    public Son(int fatherAge, int SonAge)
    throws Age {
    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 int getSonAge (){
        return this.SonAge;
    }
}
public class exception {
    public static void main (String[] args
    {
        Scanner scanner = new Scanner
            (System.in);
        try {
            System.out.print ("Enter father's
                Age:");
            int fatherAge = scanner.nextIn

            System.out.print ("Enter Son's
                age:");
            int SonAge = scanner.nextInt();
```

```
Son son1 = new Son(fatherAge, SonAge);
System.out.println("Father's Age :" + son1.getAge());
System.out.println("Son's Age :" + son1.getSonAge());

} Catch (WrongAge e) {
    System.out.println("Exception :" + e.get
                Message());
}

try {
    System.out.print("Enter father's age:");
    int fatherAge = Scanner.nextInt();

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

    Son son2 = new Son(fatherAge, SonAge);
    System.out.println("Father's Age :" + son2.getAge());
    System.out.println("Son's Age :" + son2.getSon
                Age());
} catch (WrongAge e) {
    System.out.println("Exception:" +
                e.getMessage());
}
try {
    System.out.print("Enter father's age:");
    int fatherAge = Scanner.nextInt();

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

    Son son3 = new Son(fatherAge, SonAge);
    System.out.println("Father's Age :" + son3.getAge());
    System.out.println("Son's Age :" + son3.getSonAge());

} catch (WrongAge e) {
    System.out.println("Exception :" +
                e.getMessage());
}
}
}
```

Output :-

```
Enter father's Age : 42
Enter Son's Age : 20
Father's Age : 42
Son's Age : 20
Enter father's Age : 20
Enter Son's Age : 30
Exception : Son's age cannot be greater than
        or equal to father's age.
Enter father's Age : -20
Enter Son's Age : 10
Exception : Father's age cannot be negative.
```

10/10

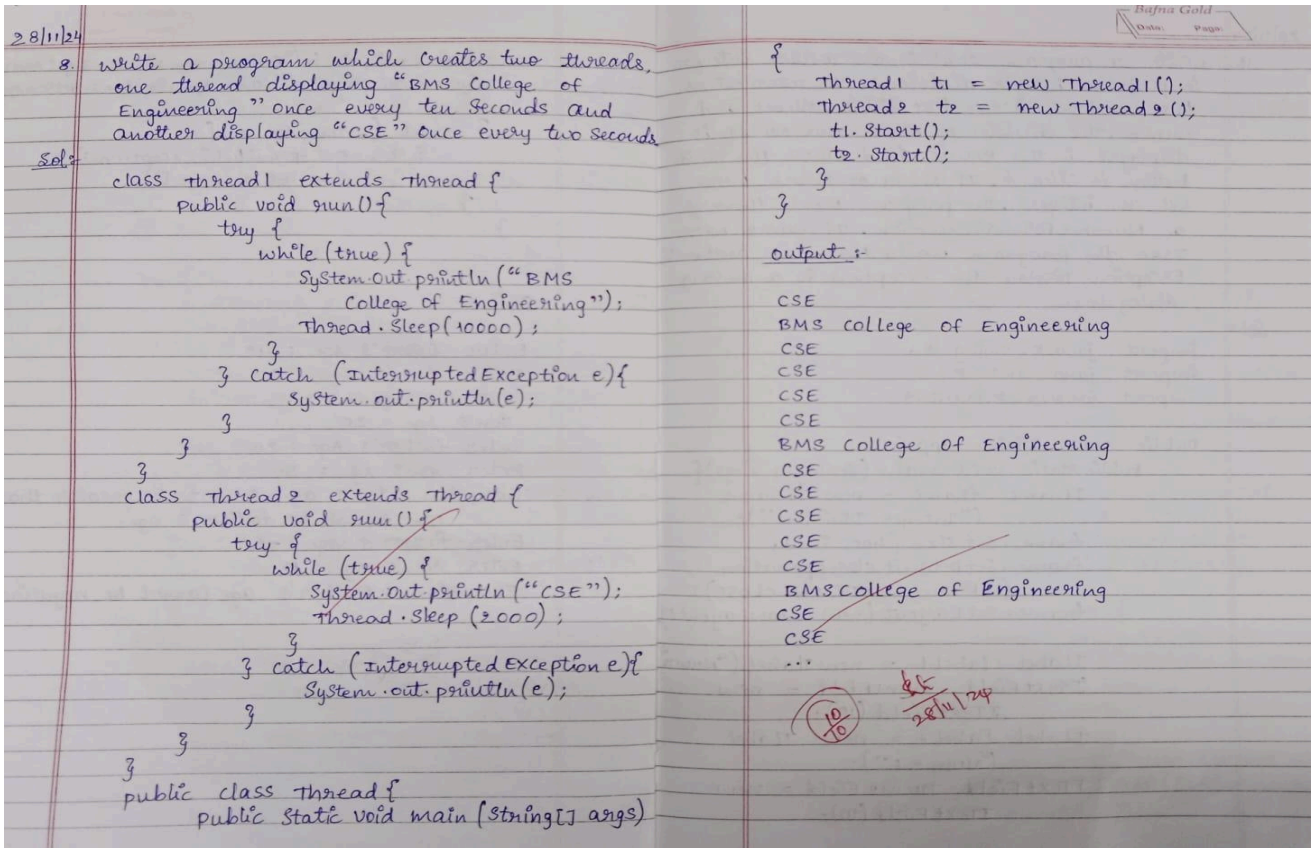
Program 8 : Threads

```
class Thread1 extends Thread {
   public void run() {
      try {
         while (true) {
            System.out.println("BMS College of Engineering");
            Thread.sleep(10000);
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}

class Thread2 extends Thread {
   public void run() {
      try {
         while (true) {
            System.out.println("CSE");
            Thread.sleep(2000);
         }
      } catch (InterruptedException e) {
         System.out.println(e);
      }
   }
}

public class thread {
   public static void main(String[] args) {
      Thread1 t1 = new Thread1();
      Thread2 t2 = new Thread2();
      t1.start();
      t2.start();
   }
}
```

Output :

```
D:\1BM23CS173>javac thread.java

D:\1BM23CS173>java thread
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
```

Source Code :



28/11/24

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

Sol:

```
class Thread1 extends Thread {
    public void run(){
        try {
            while (true) {
                System.out.println("BMS
                    College of Engineering");
                Thread.Sleep(10000);
            }
        } catch (InterruptedException e){
            System.out.println(e);
        }
    }
}
class Thread2 extends Thread {
    public void run(){
        try {
            while (true) {
                System.out.println("CSE");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e){
            System.out.println(e);
        }
    }
}
public class Thread {
    public static void main(String[] args)
{
    Thread1 t1 = new Thread1();
    Thread2 t2 = new Thread2();
    t1.start();
    t2.start();
    }
}
```

Output :-

```
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMSCollege of Engineering
CSE
CSE
...
```

10/10

28/11/24

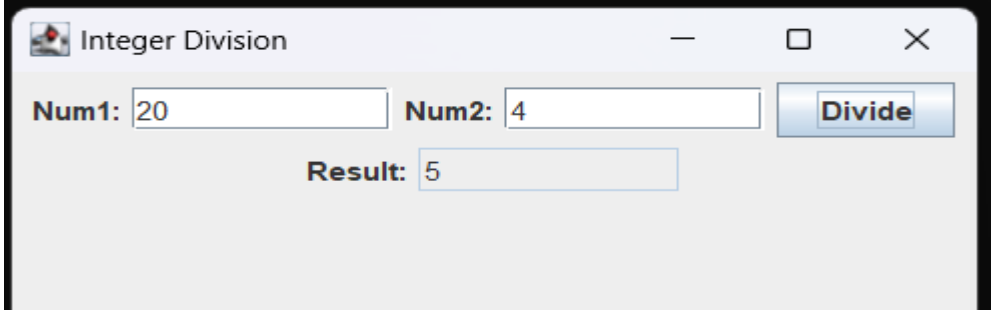
<u>Program 9</u> : Open Ended Question(Division App)

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

public class DivisionApp {
   public static void main(String[] args) {
      JFrame frame = new JFrame("Integer Division");
      frame.setSize(400, 300);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setLayout(new FlowLayout());

      JLabel label1 = new JLabel("Num1:");
      JTextField num1Field = new JTextField(10);
      JLabel label2 = new JLabel("Num2:");
      JTextField num2Field = new JTextField(10);
      JButton divideButton = new JButton("Divide");
      JTextField resultField = new JTextField(10);
      resultField.setEditable(false);

      frame.add(label1);
      frame.add(num1Field);
      frame.add(label2);
      frame.add(num2Field);
      frame.add(divideButton);
      frame.add(new JLabel("Result:"));
      frame.add(resultField);

      divideButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            try {
               int num1 = Integer.parseInt(num1Field.getText());
               int num2 = Integer.parseInt(num2Field.getText());

               if (num2 == 0) {
                  throw new ArithmeticException("Division by zero is not allowed.");
               }

               int result = num1 / num2;
               resultField.setText(String.valueOf(result));
            } catch (NumberFormatException ex) {
               JOptionPane.showMessageDialog(frame, "Invalid input. Please enter integers.", "Error",
JOptionPane.ERROR_MESSAGE);
            } catch (ArithmeticException ex) {
               JOptionPane.showMessageDialog(frame, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
```

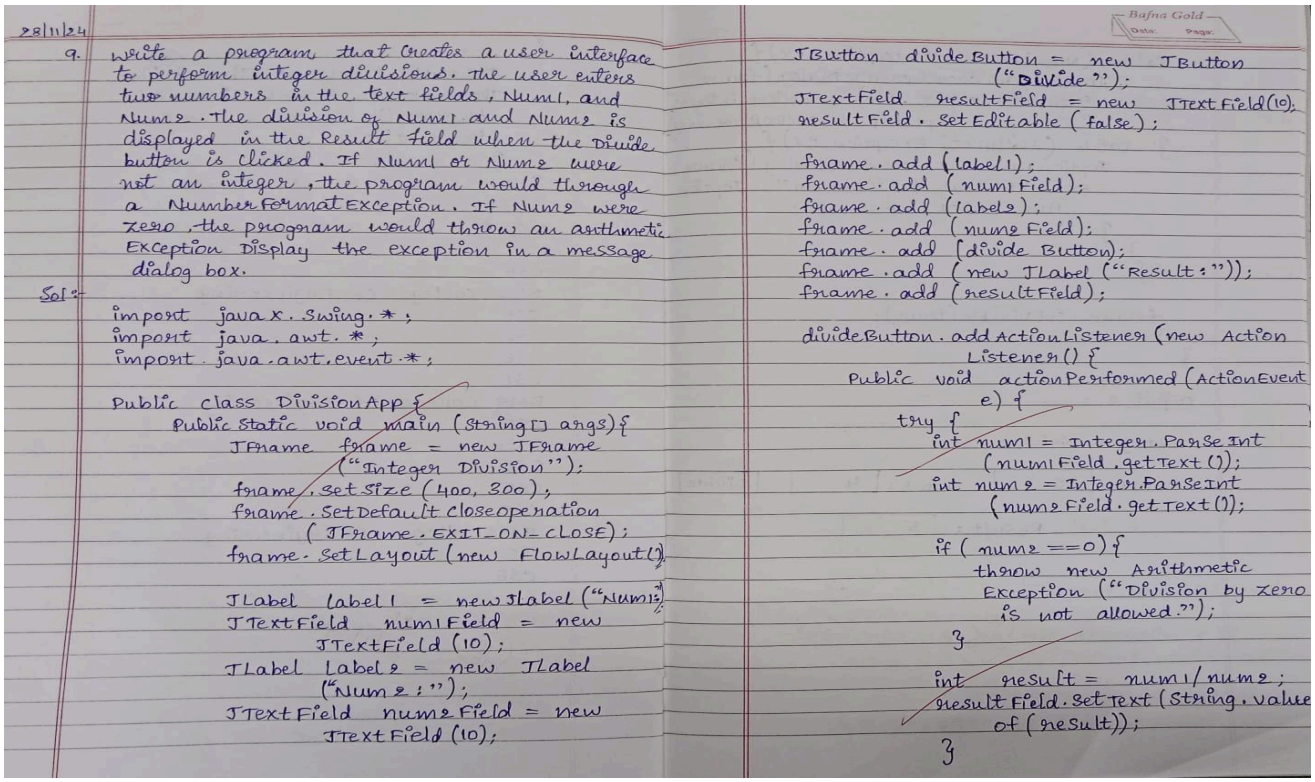
```
            }
        }
    });

    frame.setVisible(true);
  }
}
```

Output :

```
PS C:\Users\madha> d:
PS D:\> cd 1BM23CS173
PS D:\1BM23CS173> javac DivisionApp.java
PS D:\1BM23CS173> java DivisionApp
```



Source Code :



28/11/24

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 through a NumberFormatException. If Num2 were zero, the program would throw an arithmetic Exception Display the exception in a message dialog box.

Sol:-

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

Public class DivisionApp {
    Public static void main (String[] args){
        JFrame frame = new JFrame
            ("Integer Division");
        frame.setSize (400, 300);
        frame.SetDefaultCloseOperation
            (JFrame.EXIT_ON_CLOSE);
        frame.setLayout (new FlowLayout());

        JLabel label1 = new JLabel ("Num1:");
        JTextField num1Field = new
            JTextField (10);
        JLabel label2 = new JLabel
            ("Num2:");
        JTextField num2Field = new
            JTextField (10);
        JButton divideButton = new JButton
            ("Divide");
        JTextField resultField = new JTextField(10);
        resultField.setEditable (false);

        frame.add (label1);
        frame.add (num1Field);
        frame.add (label2);
        frame.add (num2Field);
        frame.add (divideButton);
        frame.add (new JLabel ("Result:"));
        frame.add (resultField);

        divideButton.addActionListener (new ActionListener() {
            Public void actionPerformed (ActionEvent e) {
                try {
                    int num1 = Integer.ParseInt
                        (num1Field.getText());
                    int num2 = Integer.ParseInt
                        (num2Field.getText());

                    if (num2 ==0) {
                        throw new Arithmetic
                            Exception ("Division by zero
                            is not allowed.");
                    }

                    int result = num1/num2;
                    resultField.setText (String.valueOf (result));
                }
```

```
        catch (NumberFormatException ex){
            JOptionPan.ShowMessageDialog(frame,
            "Invalid Input. Please enter integers",
            "Error", JOptionPane.ERROR_MESSAGE);
        } catch (ArithmeticException ex) {
            JOptionPane.showMessageDialog(frame,
            ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
        }
    });
    frame.setVisible(true);
}
}
```

output :-

Integer Division

Num1 : [20] Num2 : [4] [Divide]

Result : [5]

10/10

26/11/24

<u>Program 10</u> : Open Ended Exercise(Inter Process Communication and Deadlock)

i. 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<05) {
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<05) {
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 :

```
PS D:\1BM23CS173> javac PCFixed.java
PS D:\1BM23CS173> 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

consumed:3
Put: 4

Intimate Consumer


Producer waiting

Got: 4

Intimate Producer
```

Source Code :

Date:

Bafna Gold
Date: Page:

0/11/24

10. Demonstrate inter process communication and deadlock.

Sol:- (i) Inter process communication (IPC).

```
class Q {
int n;
boolean valueSet = false;
Synchronized int get() {
while (!valueset)
try{
System.out.println("in consumer waiting\n");
wait();
} Catch ( InterruptedException e) {
System.out.println("InterruptedExcepting
                    caught");
}
System.out.println("Got : "+ n);
Value Set = 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("InterruptedException
                    caught");
}
this.n = n;
```

```
valueSet = true;
System.out.println("Put:" + n);
System.out.println("\n Intimate 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<05) {
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<05) {
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.
producer waiting
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.
Cproducer waiting
Got : 3
Intimate producer
Consumed : 3
put : 4
Intimate Consumer
Got : 4
Intimate producer.
Consumed : 4
```

ii. 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) {
```

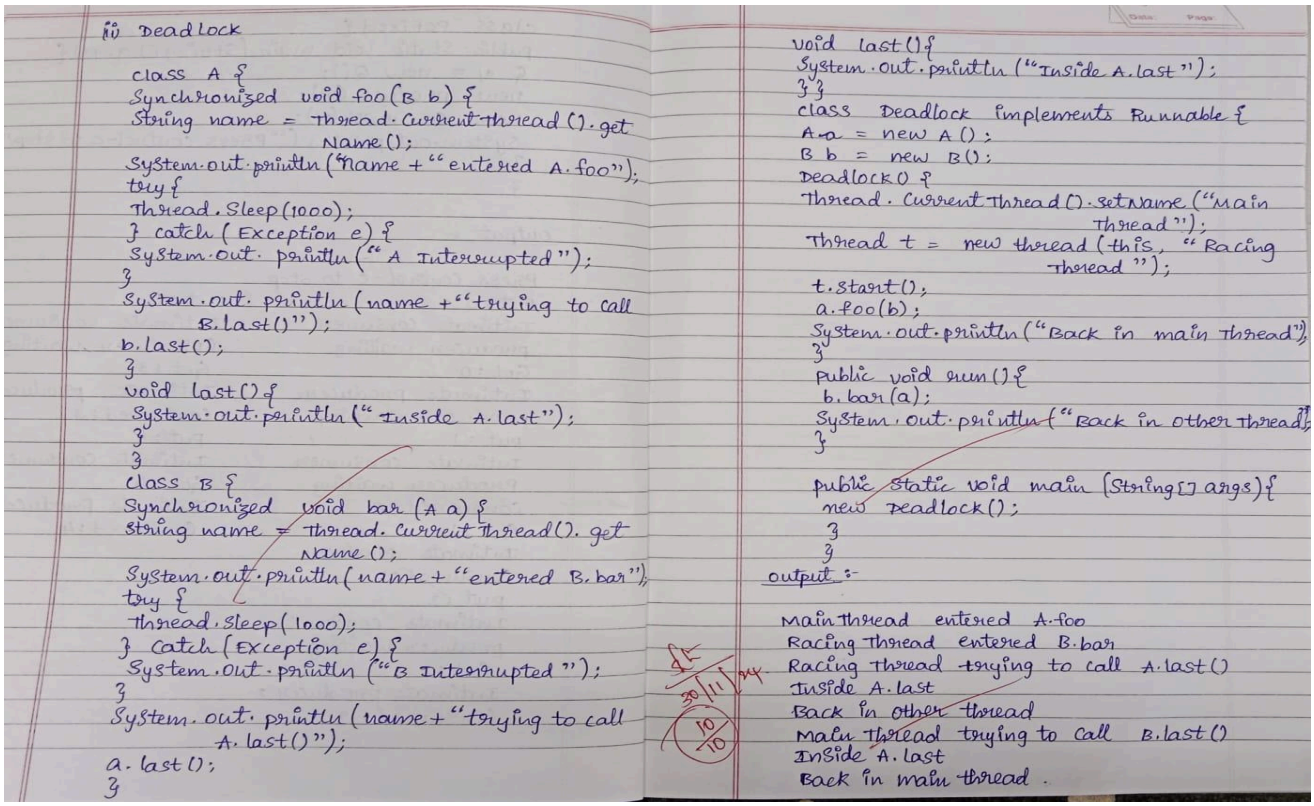
```
System.out.println("B Interrupted");
}
System.out.println(name + " trying to call A.last()");
a.last();
}
void last() {
System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable
{
A a = new A();
B b = new B();
Deadlock() {
Thread.currentThread().setName("MainThread");
Thread t = new Thread(this, "RacingThread");
t.start();
a.foo(b); // get lock on a in this thread.
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 :

```
PS D:\1BM23CS173> javac Deadlock.java
PS D:\1BM23CS173> java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
MainThread trying to call B.last()
Inside A.last
RacingThread trying to call A.last()
Inside A.last
Back in other thread
Back in main thread
```

Source Code :



```
// 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("Main Thread");
Thread t = new thread(this, "Racing Thread");
t.start();
a.foo(b);
System.out.println("Back in main Thread");
}
public void run() {
b.bar(a);
System.out.println("Back in other Thread");
}

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

output :-

```
Main Thread entered A.foo
Racing Thread entered B.bar
Racing Thread trying to call A.last()
Inside A.last
Back in other thread
Main Thread trying to call B.last()
Inside A.last
Back in main thread.
```