

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT

on

OBJECT ORIENTED JAVA PROGRAMMING

Submitted by

SATISH GIRISH KUDARE (1BM23CS306)

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)

B. M. S. College of Engineering,
Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled "**OBJECT ORIENTED JAVA PROGRAMMING**" carried out by **SATISH GIRISH KUDARE (1BM23CS306)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of **Object-Oriented Java Programming Lab - (23CS3PCOOJ)** work prescribed for the said degree.

Dr. Nandhini Vineeth

Associate Professor,
Department of CSE,
BMSCE, Bengaluru

Dr. Kavitha Sooda

Professor and Head,
Department of CSE
BMSCE, Bengaluru

INDEX

Sl. No.	Date	Experiment Title	Page No.
1	26/09/2024	Quadratic Equation	4-7
2	03/10/2024	Student SGPA Calculator	8-14
3	19/10/2024	Book Details	15-19
4	24/10/2024	Abstract Shape	20-24
5	07/11/2024	Bank Simulation	25-35
6	14/11/2024	CIE/SEE Packages	36-45
7	21/11/2024	Father-Son Verification (Exceptions)	46-50
8	05/12/2024	Multi-Threading	51-54
9	12/12/2024	Custom Division Using Awt	55-60
10	19/12/2024	Inter-process Communication and Deadlock	61-69

PROGRAM 1:

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

Observation Writeup:

WEEK - 1

CLASSMATE
Date _____
Page _____

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

```
import java.util.Scanner;
class Quadratic {
    float d;
    Scanner sc = new Scanner(System.in);

    void solver() {
        System.out.println("enter the values for a,b and c");
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();

        if (a==0) {
            System.out.println("invalid equation");
        } else {
            d = b*b - 4*a*c;
            System.out.println(d);
            System.out.println("solutions are:");
        }
    }

    if (d>0) {
        System.out.println("roots are unique");
        double r1 = (-b + Math.sqrt(d))/(2*a);
        double r2 = (-b - Math.sqrt(d))/(2*a);
        System.out.println(r1 + " " + r2);
    }
}
```

if ($d == 0$) {

 System.out.println ("roots are equal");

 double r = -b / (2 * a);

 System.out.println (r);

}

if ($d < 0$) {

 System.out.println ("There are no real roots");

}

}

}

public static void main (String [] args) {

 Quadratic q1 = new Quadratic (),

 q1.solver ();

}

}

Output:

1) enter the values of a, b, and c

2 3 6

-39.0

the solutions are

There are no real roots.

2) Enter the values of a, b, c

✓ 1

✗ 2

28/11/24 1

0.0

the solutions are
roots^{are} equal

-1.0

③ Enter the values of

a, b and c

1 5 3

13.0

the solutions are

roots are unique

-0.6972243622680054

-4.30277563773195

Soft copy of the program:

```
import java.util.Scanner;

class Quadratic {
    float d;
    Scanner sc = new Scanner(System.in);

    void solver()

    {
        System.out.println("enter the values of a,b, and c");
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();

        if (a == 0) {
            System.out.println("invalid equation");
        }
        else{
            d= b*b - 4*a*c;
            System.out.println(d);
            System.out.println("the solutions are");
            if(d>0){
                System.out.println("roots are unique ");
                double r1 = (-b+Math.sqrt(d))/(2*a);
                double r2 = (-b-Math.sqrt(d))/(2*a);
                System.out.println(r1 +" " + r2);
            }
            if(d==0){
                System.out.println("roots are equal ");
                double r = -b/(2*a);
                System.out.println(r);
            }
            if(d<0){
                System.out.println("There are no real roots" );
            }
        }
    }

}
```

```
public class Main {  
    public static void main(String[] args) {  
        Quadratic q1 = new Quadratic();  
        q1.solver();  
    }  
}
```

Output:

```
PS C:\Users\satis\OneDrive\Desktop\project> javac Main.java  
PS C:\Users\satis\OneDrive\Desktop\project> java Main  
enter the values of a,b, and c  
2 3 6  
-39.0  
the solutions are  
There are no real roots  
PS C:\Users\satis\OneDrive\Desktop\project> javac Main.java  
PS C:\Users\satis\OneDrive\Desktop\project> java Main  
enter the values of a,b, and c  
1 2 1  
0.0  
the solutions are  
roots are equal  
-1.0  
PS C:\Users\satis\OneDrive\Desktop\project> javac Main.java  
PS C:\Users\satis\OneDrive\Desktop\project> java Main  
enter the values of a,b, and c  
1 5 3  
13.0  
the solutions are  
roots are unique  
-0.6972243622680054 -4.302775637731995
```

PROGRAM 2:

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

Observation Writeup :

WEEK - 2

classmate
Date _____
Page _____

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

```
import java.util.Scanner;
class Student {
    String usn;
    String name;
    int numSubjects;
    int [] credits;
    int [] marks;
    double sgpa;

    public void acceptDetails() {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter name : ");
        name = sc.nextLine();
        System.out.println("Enter usn : ");
        usn = sc.nextLine();
        System.out.println("Enter no. of subjects : ");
        numSubjects = sc.nextInt();

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

        for (int i=0; i<numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i+1) + ": ");
            credits[i] = sc.nextInt();
            System.out.print("Enter marks for subject " + (i+1) + ": ");
            marks[i] = sc.nextInt();
        }
    }

    public void displayDetails() {
        System.out.println("Name : " + name);
        System.out.println("USN : " + usn);
        System.out.println("Number of Subjects : " + numSubjects);
        System.out.println("Credits : " + Arrays.toString(credits));
        System.out.println("Marks : " + Arrays.toString(marks));
    }

    public double calculateSGPA() {
        double totalCredits = 0;
        double totalMarks = 0;
        for (int i=0; i<numSubjects; i++) {
            totalCredits += credits[i];
            totalMarks += marks[i];
        }
        double sgpa = totalMarks / totalCredits;
        return sgpa;
    }
}
```

Page _____

```

public void displayDetails() {
    System.out.println("Student details:");
    System.out.println("USN: " + usn);
    System.out.println("Name: " + name);
    System.out.println("Subjects and Marks:");
}

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

public void calculateSGPA() {
    int totalCredits = 0;
    int totalGradePoints = 0;

    for (int i = 0; i < numSubjects; i++) {
        int grade = calculateGrade(marks[i]);
        totalGradePoints += grade * credits[i];
        totalCredits += credits[i];
    }

    sgpa = (double) totalGradePoints / totalCredits;
}

private int calculateGrade(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 void displaySGPA() {

System.out.println ("SGPA of the student is :")

+ sgpa);

}

}

public stat

@This Main

public static void main (String [] args) {

Student student = new Student ();

student.acceptDetails ();

student.displayDetails ();

student.calculateSGPA ();

student.displaySGPA ();

}

}

WEEK - 2

Output

Enter USN: 16m23cs306

Enter Name: sagar

Enter the number of subjects 6

Enter credits for subject 1: 4

Enter marks for subject 1: 95

Enter credits for subject 2: 3

Enter marks for subject 2: 85

Enter credits for subject 3: 3

Enter marks for subject 3: 75

Enter credits for subject 4: 4

Enter marks for subject 4: 88

~~Enter marks for 8~~

Enter credits for subject 5: 2

Enter marks for subject 5: 60

Enter credits for subject 6: 1

Enter marks for subject 6: 97

Student Details:

USN: 16m23cs306

Name: sagar

Subjects and Marks:

✓ Subject 1: Marks = 95, Credits = 4

✓ Subject 2: Marks = 85, Credits = 3

✓ Subject 3: Marks = 75, Credits = 3

✓ Subject 4: Marks = 88, Credits = 4

✓ Subject 5: Marks = 60, Credits = 2

✓ Subject 6: Marks = 97, Credits = 1

SGPA: 8.882352941176471

Soft copy of the program:

```
import java.util.Scanner;
class Student {
    String usn;
    String name;
    int numSubjects;
    int[] credits;
    int[] marks;
    double sgpa;

    public void acceptDetails() {
        Scanner sc = new Scanner(System.in);

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

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

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

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

        for (int i = 0; i < numSubjects; i++) {
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            credits[i] = sc.nextInt();

            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            marks[i] = sc.nextInt();
        }
    }

    public void displayDetails() {
        System.out.println("\nStudent Details:");
        System.out.println("USN: " + usn);
        System.out.println("Name: " + name);
        System.out.println("Subjects and Marks:");

        for (int i = 0; i < numSubjects; i++) {
            System.out.println("Subject " + (i + 1) + ": Marks = " + marks[i]
+ ", Credits = " + credits[i]);
        }
    }
}
```

```

public void calculateSGPA() {
    int totalCredits = 0;
    int totalGradePoints = 0;

    for (int i = 0; i < numSubjects; i++) {
        int grade = calculateGrade(marks[i]);
        totalGradePoints += grade * credits[i];
        totalCredits += credits[i];
    }

    sgpa = (double) totalGradePoints / totalCredits;
}

private int calculateGrade(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 void displaySGPA() {
    System.out.printf("SGPA:" + sgpa);
}

public static void main(String[] args) {
    Student student = new Student();
    student.acceptDetails();
    student.displayDetails();
    student.calculateSGPA();
    student.displaySGPA();
}
}

```

Output:

```
C:\1BM23CS306>javac Student.java
```

```
C:\1BM23CS306>java Student
Enter USN: 1bm23cs306
Enter Name: sagar
Enter the number of subjects: 6
Enter credits for subject 1: 4
Enter marks for subject 1: 95
Enter credits for subject 2: 3
Enter marks for subject 2: 85
Enter credits for subject 3: 3
Enter marks for subject 3: 75
Enter credits for subject 4: 4
Enter marks for subject 4: 88
Enter credits for subject 5: 2
Enter marks for subject 5: 60
Enter credits for subject 6: 1
Enter marks for subject 6: 97
```

Student Details:

USN: 1bm23cs306

Name: sagar

Subjects and Marks:

```
Subject 1: Marks = 95, Credits = 4
Subject 2: Marks = 85, Credits = 3
Subject 3: Marks = 75, Credits = 3
Subject 4: Marks = 88, Credits = 4
Subject 5: Marks = 60, Credits = 2
Subject 6: Marks = 97, Credits = 1
```

SGPA:8.882352941176471

PROGRAM 3:

Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a `toString()` method that could display the complete details of the book. Develop a Java program to create n book objects.

Observation Writeup :

WEEK - 3

CLASSTIME _____
Date _____
Page _____

• Create a class Book which contains four members: name, author, price, num_pages. Include a constructor to set the values for the members. Include methods to set and get the details of the objects. Include a `toString()` method that could display the complete details of the book. Develop a Java program to create n books.

→ Class Book {

```
String name;
String author;
int price;
int num_pages;
```

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

Scanner sc = new Scanner (System.in);

void set() {
 name = sc.next();
 author = sc.next();
 price = sc.nextInt();
 num_pages = sc.nextInt();

```
void display()
{
    cout ("The book details are");
    system.out.println(name);
    system.out.println(author);
    system.out.println(price);
    system.out.println(numPages);
}
```

```
public String toString()
{
```

```
    system.out.println("The book details are displayed by toString method");
}
```

```
    system.out.println(name);
    system.out.println(author);
    system.out.println(price);
    system.out.println(numPages);
}
```

```
} Class main
{
    public static void main (String [] args)
{
```

```
    Scanner s1 = new Scanner (System.in);
}
```

```
    System.out.println ("enter the number of objects");
}
```

```
    int n = s1.nextInt();
}
```

```
    Book [] b1 = new Book [n];
}
```

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

```
        b1[i] = new Book (200, "sachin",
    
```

```
    "The pride", 120);
}
```

```
    b1[i].display();
}
```

```
b1[i].display();  
b1[i].set();  
b1[i].display();
```

```
System.out.println(b1[i]);
```

3
3

Output:

enter the number of books

1

these are Book Details:

Price: 200

Author: Sachin

Name: The Pride

Pages: 111

enter the price, author, name and pages of the book

150

virat

TheCentury

120

✓ Book Details:

✓ Price: 150

✓ Author: virat

✓ Name: TheCentury

Pages: 120

these are book details.

Soft copy of the program:

```
import java.util.Scanner;

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

    public Book(int price, String author, String name, int pages) {
        this.price = price;
        this.author = author;
        this.name = name;
        this.pages = pages;
    }
    public void setter() {
        System.out.println("enter the price,author,name and pages of the
book");
        Scanner sc = new Scanner(System.in);
        this.price=sc.nextInt();
        this.author= sc.nextLine();
        this.name=sc.nextLine();
        this.pages=sc.nextInt();
    }

    public void getter() {
        System.out.println("Book Details:");
        System.out.println("Price:"+price);
        System.out.println("Author:"+author);
        System.out.println("Name:"+name);
        System.out.println("Pages:"+pages);
    }

    public String toString() {
        return "these are book details";
    }
}

public class Pro {
    public static void main(String[] args) {
        Scanner s1 = new Scanner(System.in);
        System.out.println("enter the number of books");
        int n = s1.nextInt();

        Book []b1 = new Book[n];
```

```
for(int i=0;i<n;i++){
    b1[i] = new Book(200,"sachin","The Pride",111);
    b1[i].getter();
    b1[i].setter();
    b1[i].getter();
    System.out.println(b1[i]);
}

}
}
```

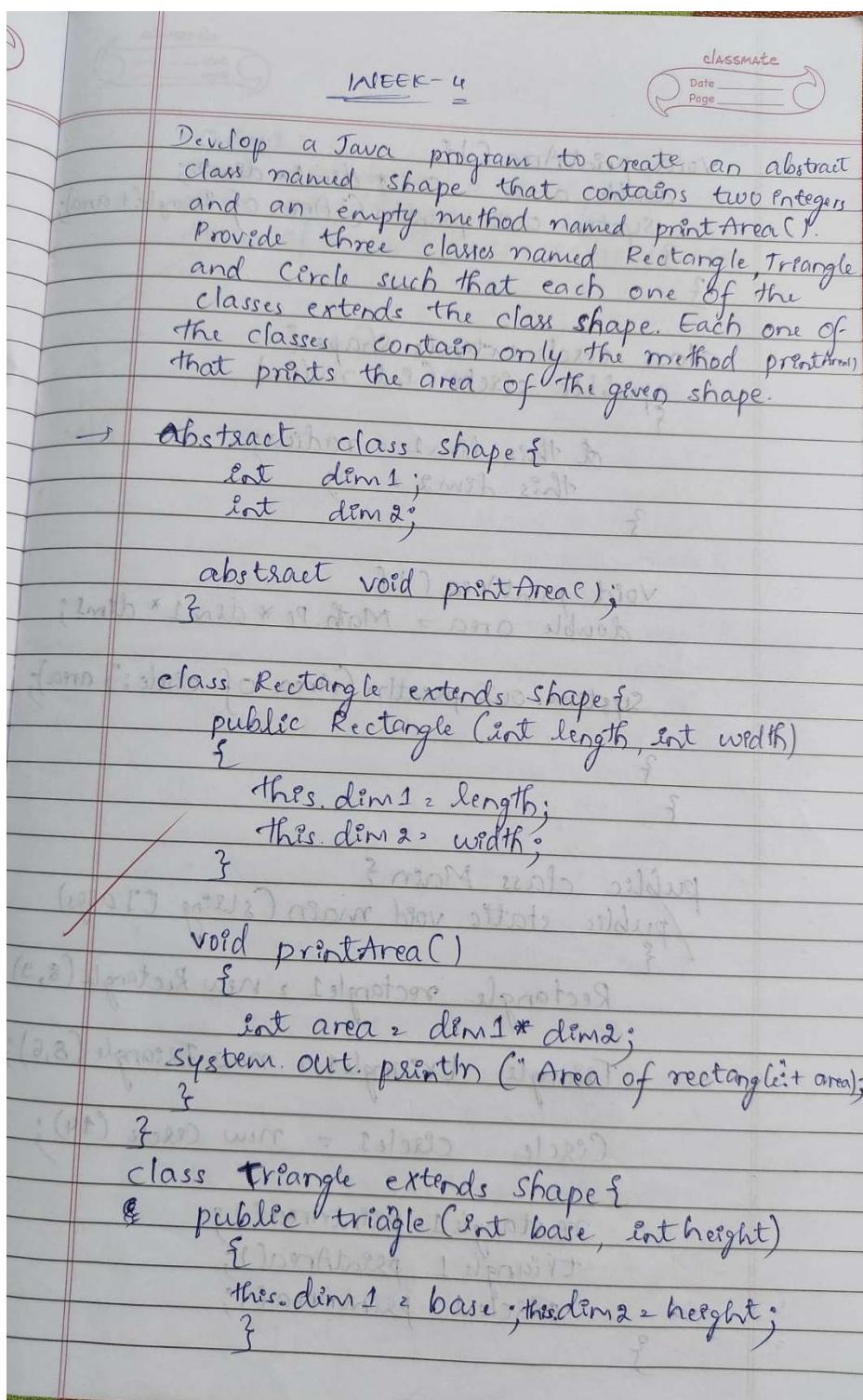
Output:

```
PS C:\Users\satis\OneDrive\Documents\oopj_lab> javac Pro.java
PS C:\Users\satis\OneDrive\Documents\oopj_lab> java Pro
enter the number of books
1
Book Details:
Price:200
Author:sachin
Name:The Pride
Pages:111
enter the price,author,name and pages of the book
150
virat
TheCentury
120
Book Details:
Price:150
Author:virat
Name:TheCentury
Pages:120
these are book details
```

PROGRAM 4 :

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

Observation Writeup :



```
void printArea () {  
    double area = 0.5 * dim1 * dim2;  
    System.out.println ("Area of Triangle:" + area);  
}  
  
class Circle extends Shape {  
    public Circle (int radius) {  
        this.dim1 = radius;  
        this.dim2 = 0;  
    }
```

& this.dim1 = radius;
this.dim2 = 0;

}

```
void printArea () {  
    double area = Math.PI * dim1 * dim1;
```

System.out.println ("Area of circle:" + area);

}

}

public class Main {

public static void main (String [] args) {

Rectangle rectangle1 = new Rectangle (8, 9);

Triangle triangle1 = new Triangle (8, 6);

Circle circle1 = new Circle (14);

rectangle1.printArea ();

triangle1.printArea ();

circle1.printArea ();

}

Output:

Area of rectangle: 72

Area of triangle: 24

Area of circle: 615.7521604035994.

Isosceles has three diagonals which are equal length and equal are the midpoints of each other which divides them into four triangles.

✓ Isosceles triangle three medians are equal.

Isosceles triangle median is a line segment which connects vertex to the base and divides it into two equal halves.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Isosceles triangle has three medians which are equal in length and divide the triangle into six equal parts.

Soft copy of the program:

```
abstract class Shape {  
    int dim1;  
    int dim2;  
  
    abstract void printArea();  
}  
  
class Rectangle extends Shape {  
    public Rectangle(int length, int width) {  
        this.dim1 = length;  
        this.dim2 = width;  
    }  
  
    void printArea() {  
        int area = dim1 * dim2;  
        System.out.println("Area of Rectangle: " + area);  
    }  
}  
  
class Triangle extends Shape {  
    public Triangle(int base, int height) {  
        this.dim1 = base;  
        this.dim2 = height;  
    }  
  
    void printArea() {  
        double area = 0.5 * dim1 * dim2;  
        System.out.println("Area of Triangle: " + area);  
    }  
}  
  
class Circle extends Shape {  
    public Circle(int radius) {  
        this.dim1 = radius;  
        this.dim2 = 0;  
    }  
  
    void printArea() {  
        double area = Math.PI * dim1 * dim1;  
        System.out.println("Area of Circle: " + area);  
    }  
}  
  
public class Main {
```

```
public static void main(String[] args) {  
    Shape rectangle = new Rectangle(8,9);  
    Shape triangle = new Triangle(8, 6);  
    Shape circle = new Circle(14);  
  
    rectangle.printArea();  
    triangle.printArea();  
    circle.printArea();  
}  
}
```

Output:

```
PS C:\Users\satis\OneDrive\Documents\oop_lab> javac Main.java  
PS C:\Users\satis\OneDrive\Documents\oop_lab> java Main  
Area of Rectangle: 72  
Area of Triangle: 24.0  
Area of Circle: 615.7521601035994  
PS C:\Users\satis\OneDrive\Documents\oop_lab> |
```

PROGRAM 5 :

Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called savings account and the other current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class Account that stores customer name, account number and type of account. From this derive the classes Cur-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

- a) Accept deposit from customer and update the balance.
- b) Display the balance.
- c) Compute and deposit interest
- d) Permit withdrawal and update the balance

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

Observation Writeup :

WEEK-5
7

CLASSMATE
Date _____
Page _____

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 withdrawl 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 8s imposed.

Create a class Account that stores customer name, account number and type of account. From this derive the classes curr-acct and Sav-acct to make them more specific to their requirements. Include the necessary methods in order to achieve the following tasks:

- a) Accept deposit from customer and update the balance
- b) Display the balance.
- c) Compute and deposit interest
- d) Permit withdrawl 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;  
    int accountNumber;  
    String accountType;  
    double balance;
```

```

public Account (String customerName, int accountNum
- ber, String accountType) {
    this.customerName = customerName;
    this.accountNumber = accountNumber;
    this.accountType = accountType;
    this.balance = 0.0;
}

```

```

public void deposit (double amount) {
    if (amount > 0) {
        balance += amount;
        System.out.println ("Amount deposited:" + amount);
        System.out.println ("Updated balance:" + balance);
    } else {
        System.out.println ("Invalid deposit amount!");
    }
}

```

```

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

```

```

class SavingsAcct extends Account {
    private double interestRate;
    public SavingsAcct (String customerName, int
    accountNumber, double interestRate) {
        super (customerName, accountNumber);
        this.interestRate = interestRate;
    }
}

```

```

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

```

```

public void withdraw(double amount) {
    if (amount <= balance) {
        balance -= amount;
        S.O.P ("Amount withdrawn:" + amount);
        S.O.P ("Updated balance:" + balance);
    } else {
        S.O.P ("Insufficient balance");
    }
}

```

\exists If (amount > balance) triggers below message
 \exists Insufficient balance

```

class CurrentAcct extends Account {
    double minimumBalance;
    double serviceCharge;
}

public CurrentAcct (String customerName, int
accountNumber, double minimumBalance, double
serviceCharge) {
    super (customerName, accountNumber, "Current");
    this.minimumBalance = minimumBalance;
    this.serviceCharge = serviceCharge;
}

```

```

public void withdraw(double amount) {
    if (amount <= balance) {
        balance -= amount;
        S.O.P ("Amount withdrawn:" + amount);
    } else if (balance < minimumBalance) {
        imposePenalty();
        S.O.P ("Updated balance:" + balance);
    }
}

```

```
else {  
    "Insufficient balance");  
}  
  
private void imposePenalty(){  
    balance -= serviceCharge;  
    S.O.P ("Balance fell below minimum. Service  
charge imposed: " + serviceCharge);  
  
public class Bank{  
    public static void main (String [ ] args){  
        Scanner sc = new Scanner (System.in);  
        S.O.P ("Choose account type: 1. Savings  
Account 2. Current Account");  
        int choice = sc.nextInt();  
        if (choice == 1){  
            S.O.P ("Enter customer name: ");  
            String name = sc.nextLine();  
            S.O.P ("Enter account number: ");  
            int accNum = sc.nextInt();  
  
            if (choice == 1){  
                S.O.P ("Enter interest rate for savings account: ");  
                double interestRate = sc.nextDouble();  
                SavingsAcct savAccount = new SavingsAcct (name,  
accNum, interestRate);  
                S.O.P ("Enter amount to deposit: ");  
                double deposit = sc.nextDouble();  
            }  
        }  
    }  
}
```

```
scvAccount. computeAndDepositInterest();  
s.o.p ("Enter amount to withdraw:");  
double withdrawAmount = sc.nextDouble();
```

```
scvAccount.withdraw(withdrawAmount);
```

```
{ else if (choice == 2) {  
    s.o.p ("Enter minimum balance for current  
    account:"); minBalance = sc.nextDouble();  
    s.o.p ("Enter service charge for falling below  
    minimum balance:");  
    double serviceCharge = sc.nextDouble();
```

```
CurrentAcct curAccount = new CurrentAcct (name,  
    accNum, minBalance, serviceCharge);
```

```
s.o.p ("Enter amount to deposit:");  
double deposit = sc.nextDouble();  
curAccount.deposit(deposit);
```

```
s.o.p ("Enter amount to withdraw:");  
double withdrawAmount = sc.nextDouble();
```

```
curAccount.withdraw(withdrawAmount);
```

```
{ else if (choice == 3) {  
    s.o.p ("Invalid account type selected.");  
}
```

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

```
{ } // loop at line 16
```

```
{ } // end of line 32 is loop end
```

Output:

* Choose account type:

1. Savings Account
2. Current Account

1

Enter customer name:

sagar

Enter account number: 1234

Enter interest rate for savings account: 3.1.

Enter amount to deposit: 5000.0

Updated balance: 5000.0

Interest added: 150.0

Updated balance: 5150.0

Enter amount to withdraw: 4800.0

Amount withdrawn: 4800.0

Updated balance: 350.0

* ~~Choose account type:~~

~~1. Savings Account~~

~~2. Current Account~~

2

Enter customer name: chetan

Enter account number: 9876

Enter minimum balance for current account: 1000

Enter service charge for falling below minimum balance:

150

Enter amount to deposit: 6000.0

Amount deposited: 6000.0

Updated balance: 6000.0

Enter amount to withdraw: 5200

Amount withdrawn: 5200.0

Balance fell below minimum. Service charge

imposed: 150.0

Updated balance: 650.0

Soft copy of the program :

```
import java.util.Scanner;

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

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

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            System.out.println("Amount deposited: " + amount);
            System.out.println("Updated balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount!");
        }
    }

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

class SavAcct extends Account {
    private double interestRate;

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

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

```

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

class CurAcct extends Account {
    double minimumBalance;
    double serviceCharge;

    public CurAcct(String customerName, int accountNumber, double
minimumBalance, double serviceCharge) {
        super(customerName, accountNumber, "Current");
        this.minimumBalance = minimumBalance;
        this.serviceCharge = serviceCharge;
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            System.out.println("Amount withdrawn: " + amount);
            if (balance < minimumBalance) {
                imposePenalty();
            }
            System.out.println("Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance!");
        }
    }

    private void imposePenalty() {
        balance -= serviceCharge;
        System.out.println("Balance fell below minimum. Service charge
imposed: " + serviceCharge);
    }
}

public class Bank {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Choose account type:\n1. Savings Account\n2.
Current Account");
        int choice = scanner.nextInt();
    }
}

```

```

scanner.nextLine();

System.out.println("Enter customer name: ");
String name = scanner.nextLine();
System.out.println("Enter account number: ");
int accNum = scanner.nextInt();

if (choice == 1) {
    System.out.println("Enter interest rate for savings account: ");
    double interestRate = scanner.nextDouble();
    SavAcct savAccount = new SavAcct(name, accNum, interestRate);

    System.out.println("Enter amount to deposit: ");
    double deposit = scanner.nextDouble();
    savAccount.deposit(deposit);

    savAccount.computeAndDepositInterest();
    System.out.println("Enter amount to withdraw: ");
    double withdrawAmount = scanner.nextDouble();
    savAccount.withdraw(withdrawAmount);

} else if (choice == 2) {
    System.out.println("Enter minimum balance for current account: ");
    double minBalance = scanner.nextDouble();
    System.out.println("Enter service charge for falling below minimum
balance: ");
    double serviceCharge = scanner.nextDouble();
    CurAcct curAccount = new CurAcct(name, accNum, minBalance,
serviceCharge);

    System.out.println("Enter amount to deposit: ");
    double deposit = scanner.nextDouble();
    curAccount.deposit(deposit);

    System.out.println("Enter amount to withdraw: ");
    double withdrawAmount = scanner.nextDouble();
    curAccount.withdraw(withdrawAmount);

} else {
    System.out.println("Invalid account type selected.");
}

scanner.close();
}
}

```

Output :

```
PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac Bank.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java Bank
Choose account type:
1. Savings Account
2. Current Account
1
Enter customer name:
sagar
Enter account number:
1234
Enter interest rate for savings account:
3
Enter amount to deposit:
5000
Amount deposited: 5000.0
Updated balance: 5000.0
Interest added: 150.0
Updated balance: 5150.0
Enter amount to withdraw:
4800
Amount withdrawn: 4800.0
Updated balance: 350.0
```

```
PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac Bank.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java Bank
Choose account type:
1. Savings Account
2. Current Account
2
Enter customer name:
chetan
Enter account number:
9876
Enter minimum balance for current account:
1000
Enter service charge for falling below minimum balance:
150
Enter amount to deposit:
6000
Amount deposited: 6000.0
Updated balance: 6000.0
Enter amount to withdraw:
5200
Amount withdrawn: 5200.0
Balance fell below minimum. Service charge imposed: 150.0
Updated balance: 650.0
```

PROGRAM 6 :

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

Observation Writeup :

WEEK - 6	Date _____ Page _____
<p>6) Create a package CIE which has two classes Student and Internals. The class Student has members like usn, name, sem. The class Internals has an array that stores the internal marks scored in five courses of the current semester of the student. Create another package SEE which has the class External which is a derived class of Student. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import two packages in a file that declares the final marks of n students in all five courses.</p> <p>→ package CIE; " + name" q. o. 2</p> <pre> 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 displayStudentDetails() { System.out.println("USN: " + usn + ", Name: " + name + ", Semester: " + sem); } </pre>	

```

package CIE;
public class Internals extends Student {
    private int[] internalMarks = new int[5];
    public Internals (String usn, String name, int sem, int[] internalMarks) {
        super (usn, name, sem);
        internalMarks = internalMarks;
    }
    public void displayInternalMarks() {
        S.O.P ("Internal Marks: ");
        for (int mark : internalMarks) {
            S.O.P (mark + " ");
        }
    }
    public int[] getInternalMarks() {
        return internalMarks;
    }
}

package SEE;
import CIE.Student;
public class External extends Student {
    private int[] externalMarks = new int[5];
    public External (String usn, String name, int sem, int[] externalMarks) {
        super (usn, name, sem);
    }
}

```

thes. externalMarks = externalMarks;

{

public void displayExternalMarks() {

s. o. p ("External Marks : ");

for (int mark : externalMarks) {

 s. o. p (mark + " ");

}

 s. o. p ();

public int[] getExternalMarks() {

 return externalMarks;

}

import CIE. Internals;

import SEE. External;

import java.util. Scanner;

public class Studentmarks {

 public static void main (String [] args) {

 Scanner sc = new Scanner (System. in);

 s. o. p ("Enter number of students : ");

 int n = sc. nextInt();

 sc. nextLine();

 Internals [] cieStudents = new Internals [n];

 Externals [] seeStudents = new External [n];

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

 s. o. p ("Enter detail of the student for CIE " +

 (i + 1) + ":");

 s. o. p ("USN : ");

```

string usn = sc.nextLine();
s.o.p ("Name:");
string name = sc.nextLine();
s.o.p ("Semester:");
int sem = sc.nextInt();
int [] internalMarks = new int[5];
s.o.p ("Enter internal marks for 5 courses:");
for (int j=0; j<5; j++) {
    internalMarks[j] = sc.nextInt();
}
ciestudents[i] = new Internals (usn, name, sem,
internalMarks);
sc.nextLine();

```

```

s.o.p ("Enter details of the student for see");
(i+1) + ":" );
s.o.p ("USN:");
usn = sc.nextLine();
s.o.p ("Name:");
name = sc.nextLine();
s.o.p ("Semester:");
sem = sc.nextInt();
int [] externalMarks = new int[5];
s.o.p ("Enter external marks for five courses:");

```

```

for (int j=0; j<5; j++) {
    externalMarks[j] = sc.nextInt();
}
ceiStudents[i] = new External (usn, name, sem,
externalMarks);
sc.nextLine();
}

```

```

s.o.p ("In Final Marks for all students:");
for (int i=0; i<n; i++) {
    cestudents[i].displayStudentDetails();
    seestudents[i].display();
    cestudents[i].displayInternalMarks();
    seestudents[i].display();
    cestudents[i].displayExternalMarks();
    seestudents[i].display();
}
int [ ] internalMarks = cestudents[i].getInternalMarks();
int [ ] externalMarks = seestudents[i].getExternalMarks();
int [ ] finalMarks = new int[5];

```

```

for (int j=0; j<5; j++) {
    finalMarks[j] = internalMarks[j] + externalMarks[j];
}

```

s.o.p ("Final Marks");

```

for (int mark : finalMarks) {
    s.o.p(mark + " ");
}

```

s.o.p ("\n");

sc.close();

}

Experiments notes made 6/1/20
08:00 AM AP 88/08 : 2021 Long

Output:

Enter number of students : 2

(1) Enter details for CIE Student 1 :

USN: 1

Name: sagar

Semester: 2

Enter internal marks for 5 courses:

38 40 41 45 46

Enter details for SEE Student 1 :

USN: 123

Name: sagar

Semester: 2

Enter external marks for 5 courses:

39 42 45 50 48

Enter details for CIE student 2 :

USN: 2

Name: chetan

Semester: 3

Enter internal marks for 5 courses:

40 42 43 44 45

Enter details for SEE student 2 :

USN: 2

Name: chetan

Semester: 3

Enter external marks for 5 courses:

40 44 46 47 50

Final marks for all students :

USN: 1, Name: sagar, Semester: 2

Final Marks : 77 82 86 95 94

USN: 2 Name: chetan Semester: 3

Final Marks : 80 88 92 94 100

Soft copy of the program :

```
import CIE.Internals;
import SEE.External;
import java.util.Scanner;

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

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

        Internals[] cieStudents = new Internals[n];
        External[] seeStudents = new External[n];

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

            System.out.println("Enter details for CIE Student " + (i + 1) + ":");
            System.out.print("USN: ");
            String usn = scanner.nextLine();
            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 internal marks for 5 courses: ");
            for (int j = 0; j < 5; j++) {
                internalMarks[j] = scanner.nextInt();
            }
            cieStudents[i] = new Internals(usn, name, sem, internalMarks);
            scanner.nextLine();

            System.out.println("Enter details for SEE Student " + (i + 1) + ":");
            System.out.print("USN: ");
            usn = scanner.nextLine();
            System.out.print("Name: ");
            name = scanner.nextLine();
            System.out.print("Semester: ");
            sem = scanner.nextInt();
            int[] externalMarks = new int[5];
            System.out.println("Enter external marks for 5 courses: ");
```

```

        for (int j = 0; j < 5; j++) {
            externalMarks[j] = scanner.nextInt();
        }
        seeStudents[i] = new External(usn, name, sem, externalMarks);
        scanner.nextLine();
    }

System.out.println("\nFinal Marks for all students:");

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

    cieStudents[i].displayStudentDetails();
    cieStudents[i].displayInternalMarks();

    seeStudents[i].displayStudentDetails();
    seeStudents[i].displayExternalMarks();

    int[] internalMarks = cieStudents[i].getInternalMarks();
    int[] externalMarks = seeStudents[i].getExternalMarks();
    int[] finalMarks = new int[5];

    for (int j = 0; j < 5; j++) {
        finalMarks[j] = internalMarks[j] + externalMarks[j];
    }

    System.out.print("Final Marks: ");
    for (int mark : finalMarks) {
        System.out.print(mark + " ");
    }
    System.out.println("\n");
}

scanner.close();
}
}

```

```

package CIE;

public class Internals extends Student {

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

```

```

public Internals(String usn, String name, int sem, int[] internalMarks) {
    super(usn, name, sem); // Call parent constructor
    this.internalMarks = internalMarks;
}

public void displayInternalMarks() {
    System.out.print("Internal Marks: ");
    for (int mark : internalMarks) {
        System.out.print(mark + " ");
    }
    System.out.println();
}

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

```

```

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 displayStudentDetails() {
        System.out.println("USN: " + usn + ", Name: " + name + ", Semester: "
+ sem);
    }
}

```

```

package SEE;

import CIE.Student;

public class External extends Student {

```

```

private int[] externalMarks = new int[5];

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

public void displayExternalMarks() {
    System.out.print("External Marks: ");
    for (int mark : externalMarks) {
        System.out.print(mark + " ");
    }
    System.out.println();
}

public int[] getExternalMarks() {
    return externalMarks;
}
}

```

Output :

```

PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac Studentmarks.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java Studentmarks
Enter number of students: 2
Enter details for CIE Student 1:
USN: 1
Name: sagar
Semester: 2
Enter internal marks for 5 courses:
38 40 41 45 46
Enter details for SEE Student 1:
USN: 1
Name: sagar
Semester: 2
Enter external marks for 5 courses:
39 42 45 50 48
Enter details for CIE Student 2:
USN: 2
Name: chetan
Semester: 3
Enter internal marks for 5 courses:
40 44 46 47 50
Enter details for SEE Student 2:
USN: 2
Name: chetan
Semester: 3
Enter external marks for 5 courses:
40 44 46 47 50

Final Marks for all students:
USN: 1, Name: sagar, Semester: 2
Internal Marks: 38 40 41 45 46
USN: 1, Name: sagar, Semester: 2
External Marks: 39 42 45 50 48
Final Marks: 77 82 86 95 94

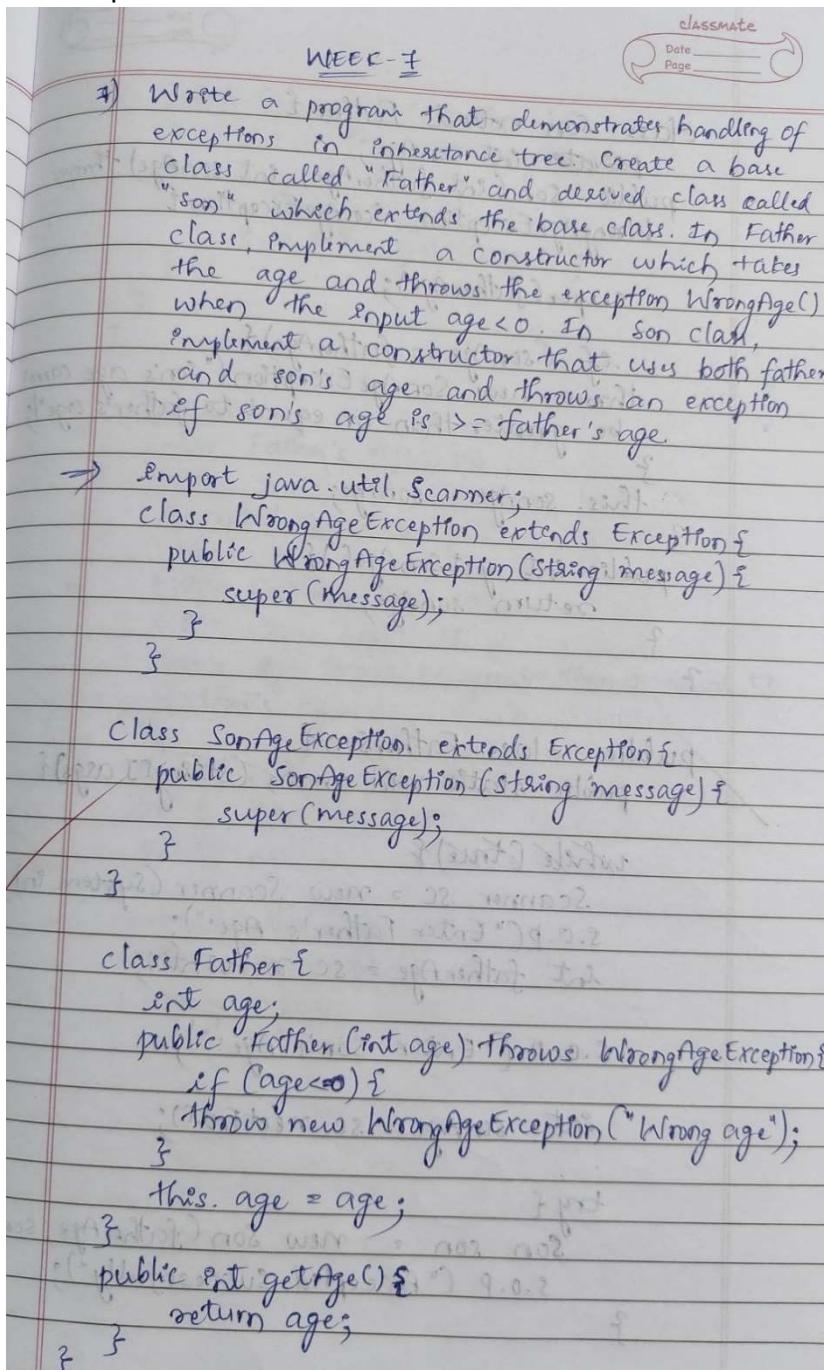
USN: 2, Name: chetan, Semester: 3
Internal Marks: 40 44 46 47 50
USN: 2, Name: chetan, Semester: 3
External Marks: 40 44 46 47 50
Final Marks: 80 88 92 94 100

```

PROGRAM 7 :

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

Observation Writeup :



```
class son extends Father {  
    private int sonAge;  
    public son (int fatherAge, int sonAge) throws  
        WrongAgeException, SonAgeException {  
        super(fatherAge);  
        if (sonAge >= fatherAge) {  
            throw new SonAgeException ("Son's age cannot  
            be greater than or equal to father's age");  
        }  
        this.sonAge = sonAge;  
    }  
    public int getSonAge() {  
        return sonAge;  
    }  
}
```

```
public class Fatherson {  
    public static void main (String [] args) {  
        Scanner sc = new Scanner (System.in);  
        S.O.P ("Enter Father's Age:");  
        int fatherAge = sc.nextInt();  
        S.O.P ("Enter Son's Age:");  
        ("Enter son's age") int sonAge = sc.nextInt();  
        try {  
            Son son = new Son (fatherAge, sonAge);  
            S.O.P ("Accepted successfully");  
        }  
    }  
}
```

```
catch (WrongAgeException e) {  
    s.o.p (e.getMessage());  
}
```

```
catch (SonAgeException e) {  
    s.o.p (e.getMessage());  
}
```

```
3  
3  
3
```

Output:

* Enter Father's Age: 46

Enter Son's Age: 26

Accepted successfully

* Enter Father's Age: 30

Enter Son's Age: 32

Son's age cannot be greater than or equal to
father's age

* Enter Father's Age: 0

Enter Son's Age: 15

Wrong age.

N
28/11/21

* Enter Father's Age: 30

Enter Son's Age: 0

Wrong age

E = 21.0M 2F = 21.0M 3S = 21.0M

F = 21.0M 8S = 21.0M 11T = 21.0M

S = 21.0M 0D = 21.0M 3. T = 21.0M

1 = 21.0M FC = 21.0M !@Twijpuz

AF:5251E8P8Z8888.8 : M92

Soft copy of the program :

```
import java.util.Scanner;

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

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

class Father {
    int age;
    public Father(int age) throws WrongAgeException {
        if (age <= 0) {
            throw new WrongAgeException("Wrong age");
        }
        this.age = age;
    }
    public int getAge() {
        return age;
    }
}

class Son extends Father {
    int sonAge;
    public Son(int fatherAge, int sonAge) throws WrongAgeException,
SonAgeException {
        super(fatherAge);
        if (sonAge >= fatherAge) {
            throw new SonAgeException("Son's age cannot be greater than or
equal to father's age");
        }
        if(sonAge <= 0){
            throw new WrongAgeException("Wrong age");
        }
        this.sonAge = sonAge;
    }
    public int getSonAge() {
        return sonAge;
    }
}
```

```

public class FatherSon{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Father's Age: ");
        int fatherAge = sc.nextInt();
        System.out.print("Enter Son's Age: ");
        int sonAge = sc.nextInt();
        try {
            Son son = new Son(fatherAge, sonAge);
            System.out.println("Accepted Succesfully");
        }
        catch (WrongAgeException e) {
            System.out.println(e.getMessage());
        }
        catch (SonAgeException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

Output :

```

PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac FatherSon.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java FatherSon
Enter Father's Age: 44
Enter Son's Age: 26
Accepted Succesfully
PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac FatherSon.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java FatherSon
Enter Father's Age: 30
Enter Son's Age: 32
Son's age cannot be greater than or equal to father's age

```

```

PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac FatherSon.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java FatherSon
Enter Father's Age: 30
Enter Son's Age: 0
Wrong age

```

```

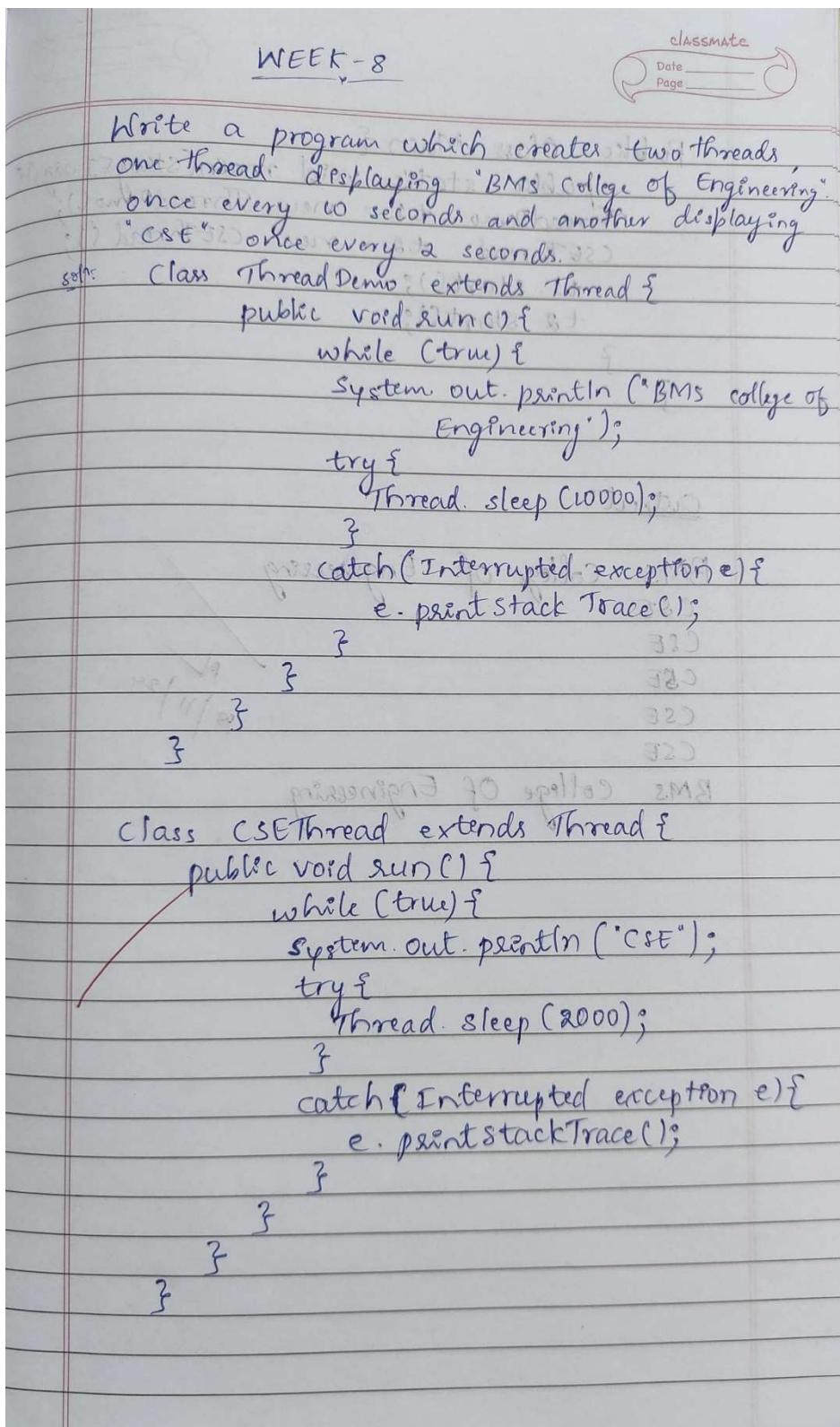
PS C:\Users\satis\OneDrive\Documents\ooj_lab> javac FatherSon.java
PS C:\Users\satis\OneDrive\Documents\ooj_lab> java FatherSon
Enter Father's Age: 0
Enter Son's Age: 15
Wrong age

```

PROGRAM 8 :

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

Observation Writeup :



```
public class Main {  
    public static void main(String[] args) {  
        ThreadDemo t1 = new ThreadDemo();  
        CSEThread t2 = new CSEThread();  
        t1.start();  
        t2.start();  
    }  
}
```

Output(q32) board

BMS College of Engineering

CSE

C-REF

CSE

CSE

205

BMS College Of Engineering

✓ 11/29

;(320) ~~other~~ two ways
;(008) ~~first~~ board

Soft copy of the program :

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

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

public class Demo{
    public static void main(String[] args){
        ThreadDemo t1 = new ThreadDemo();
        CSEThread t2 = new CSEThread();
        t1.start();
        t2.start();
    }
}
```

Output :

```
PS C:\Users\satis\OneDrive\Documents\oop_lab> javac Demo.java
PS C:\Users\satis\OneDrive\Documents\oop_lab> java Demo
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
CSE
BMS College Of Engineering
CSE
CSE
CSE
CSE
CSE
```

PROGRAM 9 :

Write a program that creates a user interface to perform integer divisions. The user enters two numbers in the text fields, Num1 and Num2. The division of Num1 and Num2 is displayed in the Result field when the Divide button is clicked. If Num1 or Num2 were not an integer, the program would throw a NumberFormatException. If Num2 were Zero, the program would throw an ArithmeticException. Display the exception in a message dialogbox.

Observation Writeup:

LAB Program - 9

classmate
Date _____
Page _____

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

→ import java.awt.*;
→ import java.awt.event.*;

```
public class DivisionMain1 extends Frame
    implements ActionListener
{
    TextField num1, num2;
    Button dResult;
    Label outResult;
    String out = "";
    double resultNum;
    int flag = 0;
}

public void actionPerformed(ActionEvent e)
{
    setLayout(new FlowLayout());
    dResult = new Button("RESULT");
    Label number1 = new Label("Number1:", Label.RIGHT);
    Label number2 = new Label("Number2:", Label.RIGHT);
    num1 = new TextField(5);
    num2 = new TextField(5);
    outResult = new Label("Result:", Label.RIGHT);
}
```

```

        add (number1);
        add (num1);
        add (number2);
        add (name);
        add (dResult);
        add (outResult);
        num1.addActionListener(this);
        num2.addActionListener(this);
        dResult.addActionListener(this);
        addWindowListener(new WindowsAdapter())
    }

    public void windowClosing(WindowEvent we)
    {
        System.exit(0);
    }

    public void actionPerformed(ActionEvent ae)
    {
        int n1, n2;
        try {
            if (ae.getSource() == dResult)
                n1 = Integer.parseInt (num1.getText());
                n2 = Integer.parseInt (num2.getText());
                if (n2 == 0) { throw new ArithmeticException();
                out = n1 + " * " + n2 + "=";
                resultNum = n1/n2;
                out += String.valueOf(resultNum);
                repaint();
            }
        }
    }

```

```

        catch (NumberFormatException e1)
    {
        flag = 1;
        out = "Number Format Exception" + e1;
        repaint();
    }

    catch (ArithmaticException e2)
    {
        flag = 2;
        out = "Divide by 0 Exception" + e2;
        repaint();
    }

    public void paint (Graphics g)
    {
        if (flag == 0)
            g.drawString (out, outResult.getX() +
                outResult.getWidth(), outResult.getY() +
                outResult.getHeight() - 8);
        else
            g.drawString (out, 100, 200);
        flag = 0;
    }

    public static void main (String args[])
    {
        DivisionMain dm = new DivisionMain();
        dm.setSize (new Dimension (800, 400));
        dm.setTitle ("Division Of Integers");
        dm.setVisible (true);
    }
}

```

Soft copy of the program :

```
import java.awt.*;
import java.awt.event.*;
public class DivisionMain1 extends Frame implements ActionListener
{
    TextField num1, num2;
    Button dResult;
    Label outResult;
    String out = "";
    double resultNum;
    int flag = 0;

    public DivisionMain1()
    {
        setLayout(new FlowLayout());

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

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

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

        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent we)
            {
                System.exit(0);
            }
        });
    }

    setTitle("Division Calculator");
    setSize(300, 200);
    setVisible(true);
}

public void actionPerformed(ActionEvent ae)
```

```

{
    int n1, n2;
    try
    {
        if (ae.getSource() == dResult)
        {
            n1 = Integer.parseInt(num1.getText());
            n2 = Integer.parseInt(num2.getText());

            if (n2 == 0)
            {
                throw new ArithmeticException("Cannot divide by zero!");
            }

            out = n1 + " / " + n2 + " = ";
            resultNum = (double) n1 / n2;
            out += String.valueOf(resultNum);
            repaint();
        }
    }
    catch(NumberFormatException e1)
    {
        flag = 1;
        out = "Number Format Exception! " + e1;
        repaint();
    }
    catch(ArithmeticeException e2)
    {
        flag = 1;
        out = "Divide by 0 Exception! " + e2;
        repaint();
    }
}

public void paint(Graphics g)
{
    if(flag == 0)
    {
        g.drawString(out, outResult.getX() + outResult.getWidth(),
outResult.getY() + outResult.getHeight() - 8);
    }
    else
    {
        g.drawString(out, 100, 200);
        flag = 0;
    }
}

```

```

public static void main(String[] args)
{
    DivisionMain1 dm=new DivisionMain1();
    dm.setSize(new Dimension(800,400));
    dm.setTitle("DivisionOfIntegers");
    dm.setVisible(true);
}
}

```

Output:



PROGRAM 10 :

Demonstrate Inter process Communication and deadlock

1. Demonstration of Inter process Communication

Observation Writeup:

Lab Program - 10

1) Demonstrate Interprocess communication and deadlock.

2) Interprocess communication.

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

    synchronized int get() {
        while (!valueset)
            try {
                S.O.P ("In Consumer waiting");
                wait();
            } catch (InterruptedException e) {
                S.O.P ("InterruptedException caught");
            }
        if (valueset)
            S.O.P ("Got: " + n);
        valueset = false;
        S.O.P ("Intimate Producer");
        notify();
        return n;
    }

    synchronized void put (int n) {
        while (valueset)
            try {
                S.O.P ("Producer waiting");
                wait();
            } catch (InterruptedException e) {
                S.O.P ("InterruptedException caught");
            }
        this.n = n;
        valueset = true;
        S.O.P ("Put: " + n);
        S.O.P ("Intimate Consumer");
        notify();
    }
}
```

```

        catch (NumberFormatException e1)
    {
        flag = 1;
        out = "Number Format Exception!" + e1;
        repaint();
    }
    catch (ArithmaticException e2)
    {
        flag = 1;
        out = "Divide by 0 Exception!" + e2;
        repaint();
    }
}

public void paint (Graphics g)
{
    if (flag == 0)
        g.drawString (out, outResult.getX() +
                      outResult.getWidth(), outResult.getY() +
                      outResult.getHeight() - 8);
    else
        g.drawString (out, 100, 200);
    flag = 0;
}

public static void main (String[] args)
{
    Dimension Main dmi = new DimensionMain();
    dmi.setSize (new Dimension (800, 400));
    dmi.setTitle ("Division Of Integers");
    dmi.setVisible (true);
}

new Consumer (2);
s.o.p ("Press Control-C to stop.");
}
:()fwtz ("method", "method")

```

Soft copy of the program :

```
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:

```
C:\Users\satis\OneDrive\Documents\ooj_lab>javac PCFixed.java
C:\Users\satis\OneDrive\Documents\ooj_lab>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
consumed:4
Put: 5
Intimate Consumer
Producer waiting
Got: 5
Intimate Producer
consumed:5
Put: 6
Intimate Consumer
Producer waiting
Got: 6
```

```
Intimate Producer
consumed:6
Put: 7
Intimate Consumer
Producer waiting
Got: 7
Intimate Producer
consumed:7
Put: 8
Intimate Consumer
Producer waiting
Got: 8
Intimate Producer
consumed:8
Put: 9
Intimate Consumer
Producer waiting
Got: 9
Intimate Producer
consumed:9
Put: 10
```

```
Intimate Producer
consumed:10
Put: 11
Intimate Consumer
Producer waiting
Got: 11
Intimate Producer
consumed:11
Put: 12
Intimate Consumer
Producer waiting
Got: 12
Intimate Producer
consumed:12
Put: 13
Intimate Consumer
Producer waiting
Got: 13
Intimate Producer
consumed:13
Put: 14
Intimate Consumer
```

2. Demonstration of Deadlock

Observation Writeup:

ii)

Demonstration of deadlock

class A

{

synchronized void foo (B b)

{ String name = Thread.currentThread().get
Name(); }

s.o.p (name + " entered A - foo");

try { Thread.sleep (1000); }

catch (Exception e) { s.o.p ("A Interrupted"); }

s.o.p (name + " trying to call B.last()");

b.last();

}

synchronized void last () { s.o.p ("Inside A.last") }

{

class B

{ synchronized void bar (A a) {

String name = Thread.currentThread().getName();

s.o.p (name + " entered B.bar");

try { Thread.sleep (1000); }

catch (Exception e) { s.o.p ("B Interrupted"); }

s.o.p (name + " trying to call A.last()"); a.last();

synchronized void last () { s.o.p ("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);

s. o. p ("Back in main thread");

}

public void run() { b. bar(a);

s. o. p ("Back in other thread");

}

public static void main(String[] args) {

new Deadlock();

}

}

Soft copy of the program :

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

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

```
C:\Users\satis\OneDrive\Documents\oopj_lab>javac Deadlock.java
C:\Users\satis\OneDrive\Documents\oopj_lab>java Deadlock
MainThread entered A.foo
RacingThread entered B.bar
RacingThread trying to call A.last()
MainThread trying to call B.last()
^C
C:\Users\satis\OneDrive\Documents\oopj_lab>
```