VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

DIPTANSHU SHEKHAR(1BM23IC022)

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 **Diptanshu Shekhar(1BM23IC022),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

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

Index

Github Link:
https://github.com/Diptanshu-Shekhar12/Java-lab-programs

Program 1

Develop a Java program that prints all real solutions to the quadratic equation ax2 +bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2 -4ac is negative, display a message stating that there are no real solutions.

Algorithm:

Lab Program 1

Q) Develop a Java Program that prints all real sol to quadratic eqn $ax^2 + bx + c = 0$. Read in a, b, c & use quadratic formula.

```
→ import java.util.Scanner;

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

double discriminant = b*b - 4*a*c;
if (a == 0) {
        if (b == 0) {
            if (c == 0)
System.out.println("Infinite sol");
        }
        else
        { System.out.println("No solution");
        }
else {
        double solution = -c/b;
System.out.println("The eqn is linear,
        solution = x" + solution);
        }
```

```
else if (discriminant > 0)
        double root 1 = (-b + Math.sqrt
                (discriminant))/(2* a);
double root 2 = (-b - Math.sqrt(discriminant))/(2*a);
System.out.println("Two real solutions: x1 = " + root1,
                root2);

else if (discriminant == 0) {
    double root = -b/(2*a);
System.out.println("One real solution: x = " + root);
else
{
    System.out.println("No real solutions");
}
}
Scanner.close();
}
```

Output:-

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

Two real solutions x1 = 2.0, x2 = 1.0

Code:

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

}
else if(d<0)
{
System.out.println("Roots are imaginary");
r1 = (-b)/(2*a);
r2 = Math.sqrt(-d)/(2*a);
System.out.println("Root1 = " + r1 + " + i"+r2);
System.out.println("Root1 = " + r1 + " - i"+r2);
}
}
}
class QuadraticMain
{
public static void main(String args[])
```

```
{
Quadratic q = new Quadratic();
q.getd();
q.compute();
System.out.println("Diptanshu Shekhar");
System.out.println("IBM23IC022");
}
}
```

```
D:\IBM23IC022>javac  QuadraticMain.java

D:\IBM23IC022>java  QuadraticMain
Enter the coefficients of a,b,c
4 1 2
Roots are imaginary
Root1 = 0.0 + i0.6959705453537527
Root1 = 0.0 - i0.6959705453537527
Diptanshu Shekhar
IBM23IC022

D:\IBM23IC022>javac  QuadraticMain.java

D:\IBM23IC022>java  QuadraticMain
Enter the coefficients of a,b,c
5 6 1
Roots are real and distinct
Roo1 = -0.2 Root2 = -1.0
Diptanshu Shekhar
IBM23IC022
```

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

Algorithm:

Lab Program 2

a) Develop a Java Program to create a class student with member USN, name, an array credit & an array marks.

→ import java.util.scanner;

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

public void accept Details () {
    Scanner scanner = new Scanner(system.in);
System.out.print ("Enter USN");
    usn = scanner.next line();

System.out.print("Enter name:");
    name = Scanner.nextline();

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

Credits = new int [numSubjects];
marks = new int [ num Subjects];

System.out.println ("Enter credit & marks
for each subjects");
```

```
for (int i = 0; i < numSubjects; i++) {
        System.out.println();
        marks[i] = scanner.nextInt();

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

public double calculateSGPA() {
        int totalCredits = 0;
        double weightedGradePoints = 0.0;

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

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

    return 0;
```

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

Output :-

```
=> Enter USN : 1BM23IS022
Enter name : Dipanshu
Enter number of subjects : 3
Enter credit & marks for each subject :
Credits for subject 1 : 4
Marks for subject 1 : 85
Credit for subject 2 : 3
Marks for subject 2 : 75
Credit for subject 3 : 2
Marks for subject 3 : 92

CGPA = 8.78
```

Code:

```
import java.util.Scanner;

class Subject {
   int subjectMarks;
   int credits;
   int grade;
}

class Student {
   Subject subject[];
   String name;
   String usn;
   double SGPA;
   Scanner s;

   Student() {
```

```
        subject = new Subject[9];
        for (int i = 0; i < 9; i++) {
            subject[i] = new Subject();
        }
        s = new Scanner(System.in);
    }

    void getStudentDetails() {
        System.out.print("Enter your Name: ");
        name = s.nextLine();
        System.out.print("Enter your USN: ");
        usn = s.nextLine();
    }

    void getMarks() {
        for (int i = 0; i < 9; i++) {
            System.out.print("Enter marks for subject " + (i + 1) + ": ");
            subject[i].subjectMarks = s.nextInt();
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            subject[i].credits = s.nextInt();

            int marks = subject[i].subjectMarks;
            if (marks >= 90) subject[i].grade = 10;
            else if (marks >= 80) subject[i].grade = 9;
            else if (marks >= 70) subject[i].grade = 8;
            else if (marks >= 60) subject[i].grade = 7;
            else if (marks >= 50) subject[i].grade = 6;
            else if (marks >= 40) subject[i].grade = 5;
            else subject[i].grade = 0;
        }
    }

    void computeSGPA() {
        int effectiveScore = 0;
        int totalCredits = 0;
        for (int i = 0; i < 9; i++) {
            effectiveScore += (subject[i].grade * subject[i].credits);
            totalCredits += subject[i].credits;
        }
        SGPA = (double) effectiveScore / totalCredits;
    }
}

public class Main {
    public static void main(String args[]) {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
```

```
        s1.computeSGPA();

        System.out.println("Name: " + s1.name);
        System.out.println("USN: " + s1.usn);
        System.out.println("SGPA: " + s1.SGPA);

        System.out.println("Diptanshu Shekhar");
        System.out.println("1BM23IC022");
    }
}
```

```
D:\LAB2>java Main
Enter your Name: Diptanshu Shekhar
Enter your USN: 1BM23IC022
Enter marks for subject 1: 89
Enter credits for subject 1: 4
Enter marks for subject 2: 86
Enter credits for subject 2: 3
Enter marks for subject 3: 95
Enter credits for subject 3: 4
Enter marks for subject 4: 92
Enter credits for subject 4: 3
Enter marks for subject 5: 87
Enter credits for subject 5: 3
Enter marks for subject 6: 90
Enter credits for subject 6: 1
Enter marks for subject 7: 87
Enter credits for subject 7: 1
Enter marks for subject 8: 98
Enter credits for subject 8: 1
Enter marks for subject 9: 97
Enter credits for subject 9: 1
Name: Diptanshu Shekhar
USN: 1BM23IC022
SGPA: 9.476190476190476
Diptanshu Shekhar
1BM23IC022
```

Program 3

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

Algorithm:

Lab Program 3

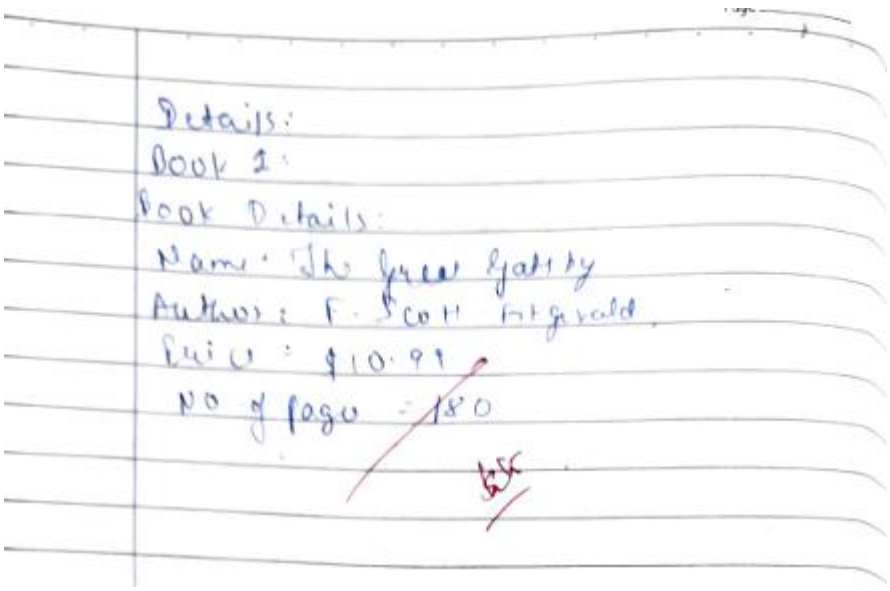
a) Create class Book which contain 4 members, name, author, price, num_pages. Include a constructor to get value for members.

```
import java.util.Scanner;

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

public Book(String name, String author,
    double price, int numPages) {

        this.name = name;
        this.author = author;
        this.price = price;
        this.numPage = numPage; }

public String getName() {
            return Name;
        }
public class BookManager {
    public static void main(String[] args) {
        Scanner Scanner = new Scanner(System.in);
```

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

BOOK[]books = new Book[n];

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

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

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

System.out.print("Enter no. of pages");

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

Output :-

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



Code:

```
import java.util.Scanner;

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

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

    @Override
    public String toString() {
        return "Book Details:\nName: " + name + "\nAuthor: " + author + "\nPrice: $" + price +
"\nPages: " + numPages;
    }
}

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

        System.out.print("Enter the number of books: ");
        int n = scanner.nextInt();
        scanner.nextLine(); // Consume newline

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

```
        for (int i = 0; i < n; i++) {
            System.out.println("\nEnter 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: ");
            double price = scanner.nextDouble();
            System.out.print("Pages: ");
            int pages = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            books[i] = new Book(name, author, price, pages);
        }

        System.out.println("\nBooks Entered:");
        for (Book book : books) {
            System.out.println(book);
        }

        scanner.close();
    }
}
```

```
D:\lab3>java BookDemo
Enter the number of books: 2

Enter details for Book 1:
Name: harry potter
Author: jk rowling
Price: 850
Pages: 400

Enter details for Book 2:
Name: brahma
Author: vinesh
Price: 550
Pages: 420

Books Entered:
Book Details:
Name: harry potter
Author: jk rowling
Price: $850.0
Pages: 400
Book Details:
Name: brahma
Author: vinesh
Price: $550.0
Pages: 420

D:\lab3>
```

Program 4

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

Algorithm:

Lab Program 4

Q) Develop a Java program to create an abstract class named shape that contain two integers and an empty method named printArea().
Provide three classes named Rectangle, Triangle & Circle

```
-> import java.util.Scanner;

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

    public abstract void printArea();
}
class Rectangle extends Shape {

    public Rectangle(int length, int breadth) {
        this.dimension1 = length;
        this.dimension2 = breadth;
}
class Circle extends Shape {
        private int radius;
    public Circle(int radius) {
        this.radius = radius;
    }

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

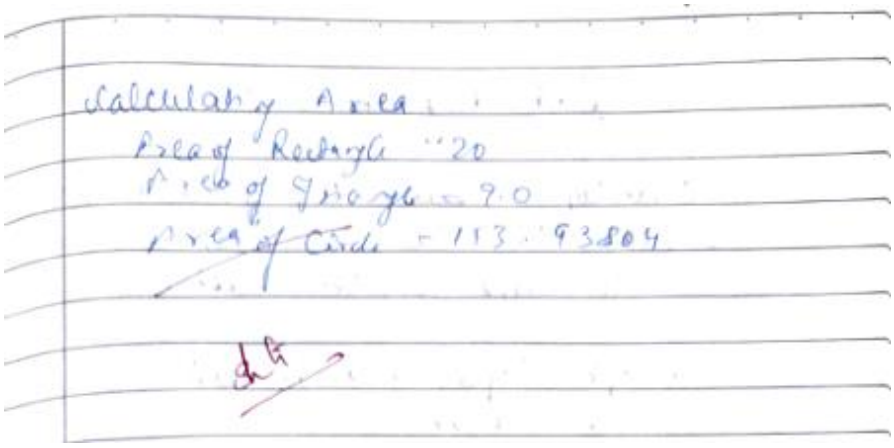
```
System.out.println("Enter length");
System.out.println("Enter Breadth");

Shape Rectangle = new Rectangle (length, breadth);

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

Shape circle = new Circle (radius);

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

Scanner.close();
}
```

Output:-

→ Enter dimension of Rectangle
length = 5
Breadth = 4

Enter dimension of Triangle
Base = 6
Height = 3

Enter radius of Circle
Radius = 7



Code:

```
import java.util.Scanner;

abstract class Shape {
    int dim1, dim2;

    Shape(int dim1, int dim2) {
        this.dim1 = dim1;
        this.dim2 = dim2;
    }

    abstract void printArea();
}

class Rectangle extends Shape {
    Rectangle(int length, int breadth) {
        super(length, breadth);
    }

    void printArea() {
        System.out.println("Area of Rectangle: " + (dim1 * dim2));
    }
}
```

```
class Triangle extends Shape {
    Triangle(int base, int height) {
        super(base, height);
    }

    void printArea() {
        System.out.println("Area of Triangle: " + (0.5 * dim1 * dim2));
    }
}

class Circle extends Shape {
    Circle(int radius) {
        super(radius, 0);
    }

    void printArea() {
        System.out.println("Area of Circle: " + (Math.PI * dim1 * dim1));
    }
}

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

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

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

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

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

        scanner.close();
    }
}
```

```
D:\LAB4>java ShapeDemo
Enter length and breadth of the rectangle: 5 6
Enter base and height of the triangle: 6 7
Enter radius of the circle: 5
Area of Rectangle: 30
Area of Triangle: 21.0
Area of Circle: 78.53981633974483
Diptanshu Shekhar
1BM23IC022
```

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.

Algorithm:

Lab Program 5

a) Develop a Java Program to create a class bank that maintain two kinds of account for its customer.

a) Accept deposits from customer & update balance.
b) Display the balance
c) compute & deposit interest
d) Permit withdrawal & update balance

```
import java.util.Scanner;

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

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

```
class SavAct extends Account {
  private static final double int_rate = 0.05;
}

public void computeDepositInterest(int years){
double interest = balance * Math.pow(1+int_rate,
                                    balance);

balance += interest;
System.out.println("Interest of" + interest);
  displayBalance();
}

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

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

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

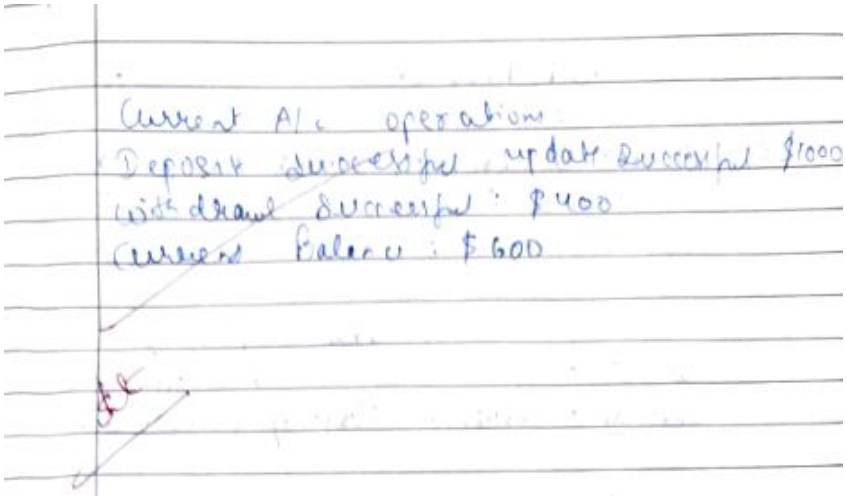
```
public class Bank {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

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

        System.out.println("Saving Account ope");
        Saving Ac.deposit(1000);
        Saving Ac.withdrawal(500);

        System.out.println("Current A/C operations");
        Current Ac deposit(1000);
        Current Ac withdrawal(600);

        Scanner.close();
    }
```

Output:

Create Saving A/c:
Enter name: Alice
Enter A/c number: SAV123

create current A/c:
Enter name: Bob
Enter A/c number: CUR541

Saving operation:
Deposit successful Balance $1000
Interest of $102.5 for 2 years
Current Balance = $1102.5
Withdrawal successful Updated Balance $602.5



Code:

```
import java.util.Scanner;

abstract class Account {
    String name, accNo;
    double balance;

    Account(String name, String accNo, double balance) {
        this.name = name;
        this.accNo = accNo;
        this.balance = balance;
    }

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

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

    abstract void withdraw(double amount);
}

class SavAcct extends Account {
    SavAcct(String name, String accNo, double balance) {
        super(name, accNo, balance);
    }

    void computeInterest() {
        balance += balance * 0.05;
        System.out.println("Interest added.");
    }
```

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

class CurAcct extends Account {
    CurAcct(String name, String accNo, double balance) {
        super(name, accNo, balance);
    }

    void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
            if (balance < 1000) balance -= 50; // Penalty
        } else System.out.println("Insufficient balance.");  }}

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

        System.out.print("Enter name and balance for Savings Account: ");
        SavAcct sa = new SavAcct(sc.next(), "SA001", sc.nextDouble());
        System.out.print("Enter name and balance for Current Account: ");
        CurAcct ca = new CurAcct(sc.next(), "CA001", sc.nextDouble());

        System.out.println("\nSavings Account Operations:");
        sa.deposit(500);
        sa.computeInterest();
        sa.withdraw(300);
        sa.displayBalance();
        System.out.println("\nCurrent Account Operations:");
        ca.deposit(200);
        ca.withdraw(400);
        ca.withdraw(1200);
        ca.displayBalance();

        sc.close();
    }
}
```

```
Enter name and balance for Savings Account: Diptanshu 10000
Enter name and balance for Current Account: Dhirendra 15000

Savings Account Operations:
Deposited: $500.0
Interest added.
Balance: $10725.0

Current Account Operations:
Deposited: $200.0
Balance: $13600.0
Diptanshu Shekhar
1BM23IC022
```

Program 6

Create a package CIE which has two classes - Personal 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 Personal. This class has an array that stores the SEE marks scored in five courses of the current semester of the student. Import the two packages in a file that declares the final marks of n students in all five courses.

Algorithm:

Lab Program 6

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

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

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

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

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

```
public void SetMarks(int[] marks){
        if (marks.length == 5){
        System.arraycopy(marks, 0, 5);
    System.out.println("Please provide marks");
    }
}
import CIE*;
import SEE*;
import java.util.Scanner;
public class FinalMarks {
        public static void(String[] args)

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

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

for (int j=0; j<5; j++){
    int finalMark = internalMarks[i].internalMark[j]
            + students[i].toMarks[j];
        System.out.print(finalMark);
}
```

Output

Enter number of students : 1

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

Final Marks of Students :

Students 1 : John Doe
USN : 123
Semester : 5
Final marks course wise
55 63 55 61 54

Code:

```
package CIE;

import java.util.Scanner;

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

    // Method to input student details
    public void inputStudentDetails() {
        Scanner s = new Scanner(System.in);
        System.out.print("Enter USN: ");
```

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

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

```
package CIE;

import java.util.Scanner;

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

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

```
package SEE;

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

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

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

```
    public void inputSEEmarks() {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter SEE marks for 5 subjects:");
        for (int i = 0; i < 5; i++) {
            System.out.print("Enter SEE marks for subject " + (i + 1) + ": ");
            marks[i] = s.nextInt();
        }
    }

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

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



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

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

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

        Externals[] students = new Externals[n];

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

```
        }

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

        s.close();
    }
}
```

```
> javac CIE/Personal.java CIE/Internals.java SEE/External.java FinalMarks.java
> java FinalMarks

Enter the number of students: 2

Enter details for Student 1:
USN: 1RV23CS001
Name: Alice
Semester: 5
Enter CIE marks for 5 courses:
18 20 15 19 17
Enter SEE marks for 5 courses:
40 38 45 42 39

Enter details for Student 2:
USN: 1RV23CS002
Name: Bob
Semester: 5
Enter CIE marks for 5 courses:
20 22 18 19 21
Enter SEE marks for 5 courses:
36 40 44 35 38

Final Marks of Students:
Student 1 - USN: 1RV23CS001
Name: Alice, Semester: 5
Final Marks: 38 39 37 40 36

Student 2 - USN: 1RV23CS002
Name: Bob, Semester: 5
Final Marks: 38 42 40 36 40
```

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=father’s age.

Algorithm:

Lab Program 7

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

→ class WrongAge Exception extends Exception {
public WrongAge Exception (String message) {
super (mes, age);
}
}

class Father {
protected int age;
public Father(int age) throws WrongAgeException
{ if (age < 0) {
throw new WrongAge Exception("Father's Age cannot be negative");
}
this . age = age;
System.out.println ("Fathers Age is" + this.age);
}
}

class Son extends Father {
private int SonAge
if (age < 0) {
throw new WrongAge Exception ("Son's age can't be
if (SonAge >= father.Age) {
throw new WrongAge Exception ("Son age

```
cannot be equal to or greater than father");
    }

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

try {
System.out.println("Creating Father & Son objects");
Father father = new Father(40);
Son son = new Son(40,15);
    }
catch (WrongAgeException e) {
System.out.println("Exception caught" + e.getMessage());
    }
Scanner.close()
    }
}
```

Output:-

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

Testing invalid inputs
Attempting to set father's age to -5
Exception caught: Father's age cannot be -ve

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

Code:

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

```
    }
}

// Father class
class Father {
    int age;

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

// Son class that extends Father
class Son extends Father {
    int sonAge;

    public Son(int fatherAge, int sonAge) throws WrongAgeException {
        super(fatherAge); // Call the parent (Father) constructor
        if (sonAge >= fatherAge)
            throw new WrongAgeException("Son's age cannot be greater than or equal to
father's age.");
        this.sonAge = sonAge;
    }
}

// Main class
public class ExceptionDemo {
    public static void main(String[] args) {
        System.out.println("DIPTANSHU-1BM23IC022");

        try {
```

```
        Father father = new Father(40); // Create a Father object
        Son son = new Son(40, 20);      // Create a Son object
        System.out.println("Father's Age: " + father.age + ", Son's Age: " + son.sonAge);
    } catch (WrongAgeException e) {
        System.out.println("Exception: " + e.getMessage());
    }

    try {
        Son invalidSon = new Son(30, 35); // This will throw an exception
    } catch (WrongAgeException e) {
        System.out.println("Exception: " + e.getMessage());
    }
  }
}
```

```
DIPTANSHU - 1BM23IC022
Father's Age: 40, Son's Age: 20
Exception: Son's age cannot be greater than or equal to father's age.
Diptanshu Shekhar
1BM23IC022
```

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.

Algorithm:

```
Lab Program 8

8) Write a program to which creates
threads, one thread displaying
"BMS college of Engineering" once every ten
seconds & other displaying "CSE"
every two seconds.

class College Thread extends Thread {
public void run() {
    try {
      while (true) {
System.out.println("BMS College of engineering");
    Thread.sleep(10000);
    }
}
catch ("Interruption Exception") {
System.out.println("College thread interrupted:"+
                    e.getmessage());
}
}
}
class Department thread extends Thread {
  public void run() {
    try {
      while (true) {
      System.out.println("CSE");
      Thread.sleep(2000);
    }
  }
catch (InterruptedException e) {
System.out.println("Department thread interrupted:"+
                    e.getmessage());
```

```
    }
    }
}

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

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

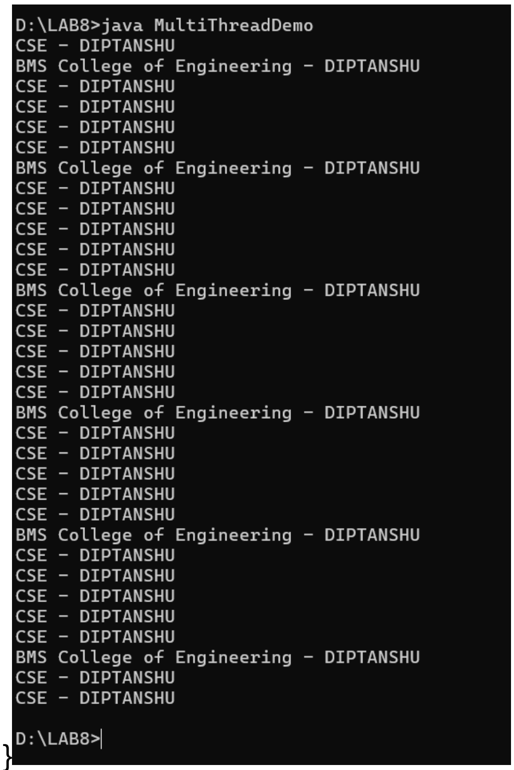
Output:

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

Code:

```
class MessageThread extends Thread {
    private String message;
    private int interval;

    public MessageThread(String message, int interval) {
        this.message = message;
        this.interval = interval;
    }

    @Override
```

```
    public void run() {
        try {
            while (true) {
                System.out.println(message + " - DIPTANSHU");
                Thread.sleep(interval * 1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted.");
        }
    }
}

public class MultiThreadDemo {
    public static void main(String[] args) {
        new MessageThread("BMS College of Engineering", 10).start();
        new MessageThread("CSE", 2).start();
    }
```



```
D:\LAB8>java MultiThreadDemo
CSE - DIPTANSHU
BMS College of Engineering - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
BMS College of Engineering - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
BMS College of Engineering - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
BMS College of Engineering - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
BMS College of Engineering - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU
BMS College of Engineering - DIPTANSHU
CSE - DIPTANSHU
CSE - DIPTANSHU

D:\LAB8>
```

```
}
```

Program 9

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

Algorithm:

Lab Program 9

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

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

public class DivisionUI {
    public static void main(String[] args) {
        JFrame frame = new frame("Integer division");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(new GridLayout(4,2,10,10));

        JLabel label1 = new JLabel("Num1:");
        JTextField num1field = new JTextfield();

        frame.add(label1);
        frame.add(num1field);
        frame.add(label2);
        frame.add(num2field);
        frame.add(resultlabel);
        frame.add(resultfield);
        frame.add(dividbutton);
```

divideButton.addActionListener(new ActionListener(){
try {
int num1 = Integer.parseInt(num1Field.getText());
int num2 = Integer.parseInt(num2Field.getText());
}
catch (ArithmeticException ex) {
JOptionPane.showMessageDialog(frame,
ex.getMessage());
"Arithmetic Error", JOptionPane.ERROR_MESSAGE);
}
frame.setVisible(true);
}
});

Output:-

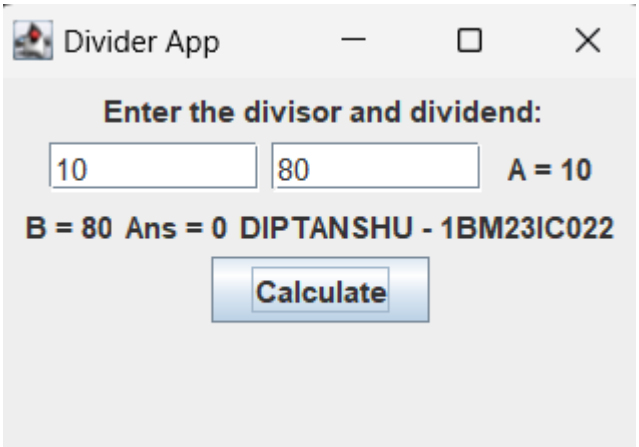
Num1 = 10
Num2 = 2
5.
Num1 = 10
Num2 = abc
Please enter valid integer for Num1 & Num2
Num1 = 10
Num2 = 0
Cannot divide by zero.

Code:

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

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

```
// To terminate on close
jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

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

// Add text fields for both numbers
JTextField ajtf = new JTextField(8);
JTextField bjtf = new JTextField(8);

// Labels to display results
JLabel err = new JLabel();
JLabel alab = new JLabel();
JLabel blab = new JLabel();
JLabel anslab = new JLabel();
JLabel nameLabel = new JLabel("DIPTANSHU-1BM23IC022");

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

// Calculate button
JButton button = new JButton("Calculate");
jfrm.add(button);

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

            // Display the results
            alab.setText("A = " + a);
            blab.setText("B = " + b);
            anslab.setText("Ans = " + ans);
            err.setText(""); // Clear any previous error message

        } catch (NumberFormatException e) {
            alab.setText("");
            blab.setText("");
            anslab.setText("");
```

```
                err.setText("Enter only integers!");
            } catch (ArithmeticException e) {
                alab.setText("");
                blab.setText("");
                anslab.setText("");
                err.setText("B should be NON-zero!");
            }
        }
    });

    // Display frame
    jfrm.setVisible(true);
  }

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



Program 10

Demonstrate Inter process Communication and deadlock

IPC

Algorithm:

Lab Program 10:

Q) Demonstrate Inter Process Communication & Deadlock

```
class SharedResource {
synchronized void methodA(SharedResource
    other Resource) {
System.out.println(Thread.currentThread().getN
    try {
        Thread.sleep(1000);
    }
catch (InterruptedException e) {
System.out.println("Interrupted");
    }

synchronized void methodB() {
System.out.println(Thread.currentThread()
        getName() + " is executing methodB");
}
public class DeadLockDemo {
    public static void main(String[] args) {
        SharedResource Resource1 = new SharedResource();
        SharedResource Resource2 = new SharedResource();

        Thread thread1 = new Thread() -> Resource1.methodA();
        Thread thread2 = new Thread() -> Resource2.methodB();

        thread1.start();
        thread2.start(); }
    }
```

Output: (Dead Block)

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

Output: (Dead Lock Resolved)

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

Code:

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

    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
```

```
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\nIntimate Producer\n");
        notify();
        return n;
    }

    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);
        System.out.println("\nIntimate Consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;

    Producer(Q q) {
        this.q = q;
        new Thread(this, "Producer").start();
    }
```

```
    public void run() {
        int i = 0;
        while (i < 15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
    Q q;

    Consumer(Q q) {
        this.q = q;
        new Thread(this, "Consumer").start();
    }

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

public class PCFixed {
    public static void main(String args[]) {
        System.out.println("DIPTANSHU SHEKHAR 1BM23IC022") ;
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
```

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

```
Diptanshu Shekhar 1BM23IC022
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

Put: 6

Intimate Consumer


Producer waiting
```

```
Producer waiting

Consumed: 5
Got: 6

Intimate Producer

Put: 7

Intimate Consumer


Producer waiting

Consumed: 6
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
```

```
Got: 9

Intimate Producer

Consumed: 9
Put: 10

Intimate Consumer


Producer waiting

Got: 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
```

```
Producer waiting

Got: 12

Intimate Producer

Consumed: 12
Put: 13

Intimate Consumer


Producer waiting

Got: 13

Intimate Producer

Consumed: 13
Put: 14

Intimate Consumer

Got: 14

Intimate Producer

Consumed: 14

D:\LAB10\IPC>
```

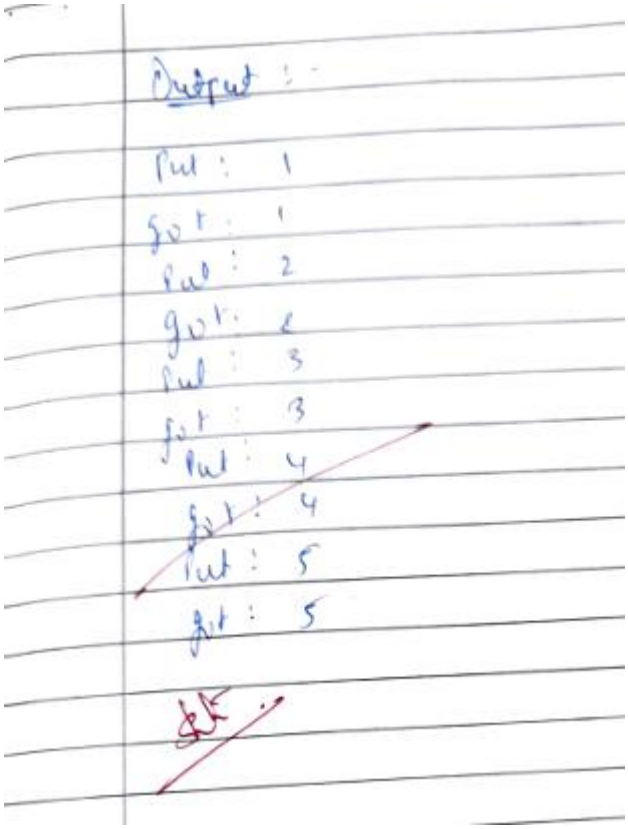
DeadLock
Algorithm:

Lab Program 10

a) Demonstrate IPC

```
class Q {
    int x;
    boolean value set = false;
    synchronized int get () {

    while (!value set) {
        try {
    System.out.println ("Consumer Waiting");
        wait();
        }
    catch { InterruptionException e) {
    System.out.println ("Exception");
        }
    System.out.println ("Got:" + x);
    value set = false;
    System.out.println ("Intimate Producer");
    notify ();
    return x;
    }
    synchronized void put (int x) {
        while (value set) {
        try {
    System.out.println ("Producer Waiting");
    wait();
    }
    catch (InterruptedException e) {
        System.out.println ("Put" + x);
    System.out.println ("Put" + x);
```

```
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 i = q.get();
        i++;
    }
}
}
```



Code:

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

    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); // Simulating some work
        } catch (Exception e) {
            System.out.println("B Interrupted");
        }
        System.out.println(name + " trying to call A.last()");
        a.last(); // Deadlock occurs here
    }

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

class Deadlock implements Runnable {
    A a = new A();
    B b = new B();

    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b); // 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[]) {
        System.out.println("DIPTANSHU SHEKHAR 1BM23IC022") ;
        new Deadlock();
    }
}
```

```
D:\LAB10\Deadlock>java Deadlock
DIPTANSHU SHEKHAR 1BM23IC022
RacingThread entered B.bar
MainThread entered A.foo
RacingThread trying to call A.last()
MainThread trying to call B.last()
```