VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

Object Oriented Java Programming (23CS3PCOOJ)

Submitted by

SIRIPURAPU MANASWI(1BM23CS331)

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 **SIRIPURAPU MANASWI (1BM23CS331),** 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. Nandhini Vineeth Associate Professor, Department of CSE, BMSCE, Bengaluru	**Dr. Kavitha Sooda** Professor and Head, Department of CSE BMSCE, Bengaluru

Index

Github Link:

https://github.com/s-manaswi/MANASWI-1BM23CS331-JAVA

Program 1

Implement Quadratic Equation

LAB PROGRAM 1

Quadratic equation

```
import java.util.scan;
public class QuadraticEqn {
    psvm (string[] args) {
        scanner sc = new scanner (system.in);
        System.out.println (" Enter coefficient a: ");
        double a = sc.nextDouble();
        System.out.println ("Enter coefficient b: ");
        double b = sc.nextDouble();
        System.out.println ( "Enter coefficient c: ");
        double c = sc.nextDouble();
        double discriminant = b*b - 4*a*c;
        if (discriminat >0) {
            double root1 = (-b + Math.sqrt (discrimi-
                                nant))/(2*a);
            double root2 = (-b+ Math.sqrt (discriminant
                                /(2*a);
        }
            S.O.P. ("Roots are real & distinct ");
            S.O.P. ( " Root 1 : " + root1);
            S.O.P. ("Root 2 : " + root2);
        }
        else if (discriminant == 0) {
            double root= -b /(2*a);
            S.O.P. ("Roots are real & equal");
            S.O.P. ( "Root= " + root );
        }
```

```
        else {
            S.O.P. ("There are no real solutions");
        }
        Sc.close();
    }
}
```

Output:

```
Enter a coefficient a:
1
Enter a coefficient b:
-3
Enter a coefficient c;
2
The equation has 2 real & distinct roots
root 1 = 2.0
root 2 = 1.0
```

CODE :

```
import java.util.Scanner;
import java.lang.Math;
class Quadratic
 {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter coefficient a: ");
      double a = scanner.nextDouble();

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

      System.out.print("Enter coefficient c: ");
      double c = scanner.nextDouble();

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

      if (d > 0) {
         double r1 = (-b + Math.sqrt(d)) / (2 * a);
         double r2 = (-b - Math.sqrt(d)) / (2 * a);
         System.out.println("The roots are real and different:");
         System.out.println("Root 1: " + r1);
         System.out.println("Root 2: " + r2);
      } else if (d == 0) {
         double r = -b / (2 * a);
         System.out.println("The roots are real and the same:");
         System.out.println("Root: " + r);
      } else {
```

```
            System.out.println("The roots are complex:");
            double realPart = -b / (2 * a);
            double imaginaryPart = Math.sqrt(-d) / (2 * a);
            System.out.println("Root 1: " + realPart + " + " + imaginaryPart + "i");
            System.out.println("Root 2: " + realPart + " - " + imaginaryPart + "i");
        }

        scanner.close();

    }
}
```

Program 2

Implement SGPA Calculator

LAB PROGRAM - 2

SGPA Calculation

```
import java.util.Scanner;
class Student;
    String name;
    String usn;
    int numsub;
    int [] credits, marks[];
    Student (int num) {
        numsub = num;
        credits = new int [numsub];
        marks = new int [numsub];
    }
    Void details () {
        Scanner sc = new Scanner (system.in);
        System.out.println ("Enter usn: ");
        usn = sc.next();
        System.out.println ("Enter name: ");
        name = sc.next();
        System.out.println ("Enter credits
            for each subject: ");
        for (int i=0 ; i<numsub; i++) {
            S.O.P. ("Credits for subject "+
                (i+1) + ":");
            Credits [i] = sc.nextInt();
        }
        S.O.P. ("Marks for subjects: ");
        for (int i=0 ; i<numsub ; i++) {
            S.O.P. ("Marks for subject "+(i+1)+ ":");
            marks [i] = sc.nextInt();
        }
    }
```

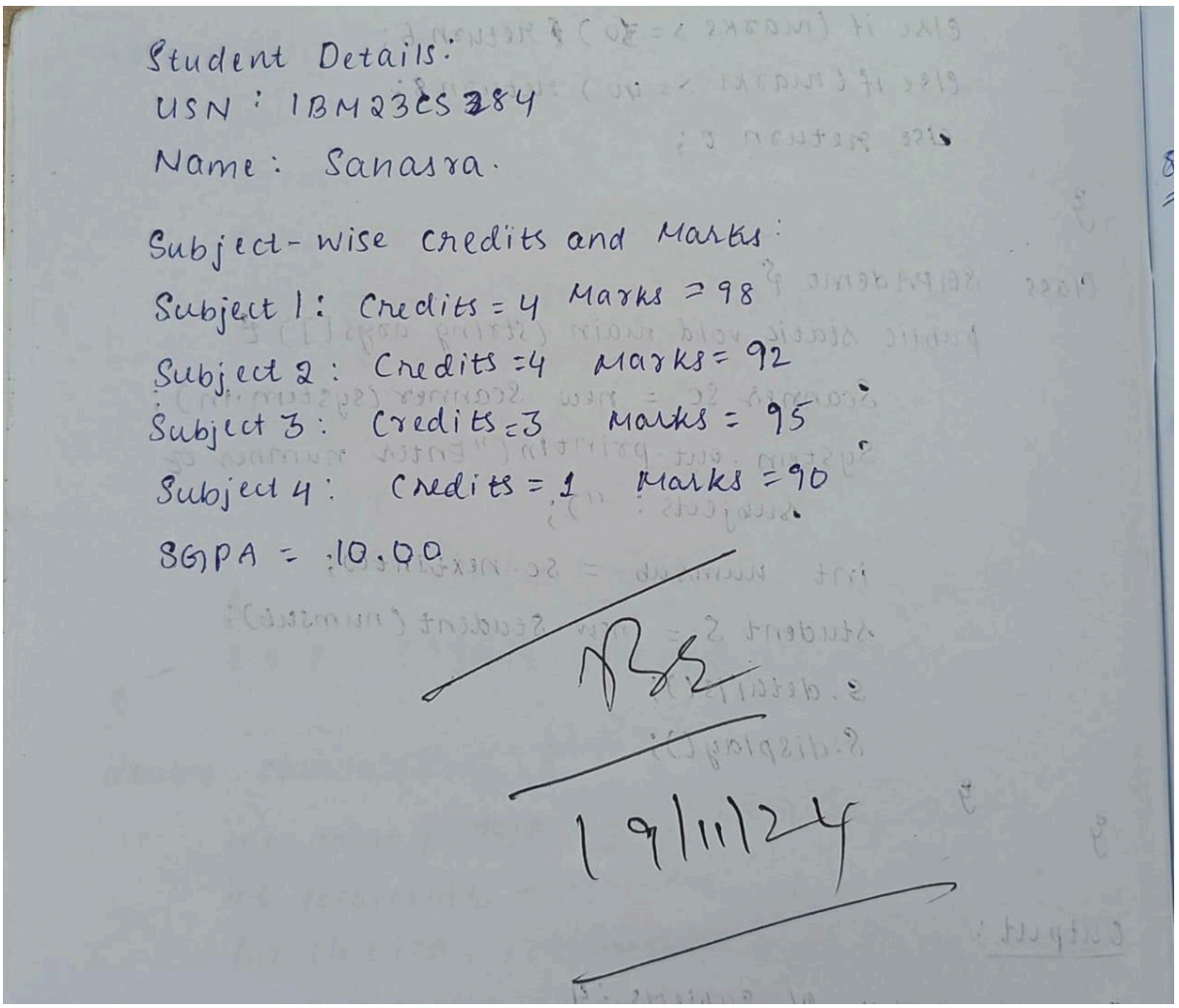
```
void display () {
    S.O.P. ("\nStudent details : ");
    System.out.println ("USN : "+usn);
    system.out.println ("Name : "+name);
    system.out.println (" \nsubject-wise
        credits & marks ");
    for (int i=0 ; i<numsub ; i++) {
        S.O.P. ("subject "+(i+1) + " credits = "
        +credits[i] + " Marks = " + marks[i]);
    }
    S.O.P. ("SGPA : " + calculate SGPA());
}
double calculateSGPA() {
    int totalgradepts = 0;
    int totalcredits = 0;
    for (int i=0 ; i<numsub ; i++) {
        totalcredits += credits[i];
        int gradepoint = calculategradepoint (marks[i]);
        totalgradepoints += gradepoint * credits[i];
    }
    return (double) totalgradepoints / totalcredits;
}
int calculategradepoint (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;

}
class SGPAdemo {
    public static void main (String args[]) {
        Scanner sc = new Scanner (system.in);
        System.out.println("Enter number of
            subjects : ");
        int numsub = sc.nextInt();
        Student S = new Student (numsub);
        S.details();
        S.display();
    }
}
```

Output:

```
Enter number of subjects : 4
Enter USN : 1BM23CS384.
Enter Name : Sanasra
Enter credits for each subject:
Credits for subject 1 : 4
Credits for subject 2 : 4
Credits for subject 3 : 3
Credits for subject 4 : 2
Enter credits for each subject:
Marks for subject 1 : 98
Marks for subject 2 : 92
Marks for subject 3 : 95
Marks for subject 4 : 90.
```



Student Details:
USN : 1BM23CS284
Name : Sanasra.

Subject-wise credits and Marks:
Subject 1: Credits = 4 Marks = 98
Subject 2: Credits = 4 Marks = 92
Subject 3: Credits = 3 Marks = 95
Subject 4: Credits = 1 Marks = 90

SGPA = 10.00

19/11/24

CODE :

```
import java.util.Scanner;


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

    Sgpacalc(int x) {
        numSubjects = x;
```

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

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

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

void displayDetails() {
    System.out.println("\nStudent Details:");
    System.out.println("USN: " + usn);
```

```
        System.out.println("Name: " + name);

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

        System.out.println("SGPA: " + calculateSGPA());
    }

    double calculateSGPA() {
        int totalCredits = 0;
        int totalGradePoints = 0;

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

        return (double) totalGradePoints / totalCredits;
    }

    int calculateGradePoint(int marks) {
        if (marks >= 90) return 10;
        else if (marks >= 80) return 9;
        else if (marks >= 70) return 8;
```

```
        else if (marks >= 60) return 7;
        else if (marks >= 50) return 6;
        else if (marks >= 40) return 5;
        else return 0;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of subjects: ");
        int numSubjects = sc.nextInt();

        Sgpacalc s = new Sgpacalc(numSubjects);
        s.acceptDetails();
        s.displayDetails();
    }
}
```

Program 3:

Create Objects for Books

LAB PROGRAM-3

Q) Write a Java program to create a class Book which contains 4 members: name, author, price, num_pages. Include a constructor to set values for the members. Include methods to set & 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 objects.

```
import java.util.Scanner;

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

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

    void details() {
        System.out.print("Enter name of book " +
                (i+1) + ": ");

        String name = s.nextLine();
```

```
System.out.print ("Enter author of book: "
                    + (i+1) + ": ");

String author = s.nextLine;
S.O.P. (" Enter price of the book "+(i+1)+ ": ");
int price = s.nextInt();
S.O.P. ("Enter no. of pages in book " + (i+1)
                    + ": ");
int numpages = s.nextInt();
}

public String toString () {
    return "Book name: " + name + "\n Author"
    + author + "\n Price " + price + "\n Number
    of pages: " + numpages;
}

Class Mybook {
    public Static void main (String args[]) {
        Scanner sc = new Scanner (system.in);
        S.O.P. ("Enter number of books: ");
        int n = sc.nextInt();
        Book [] books = new book[n];
        for (int i=0 ; i<n; i++) {
            books [i] = new Book();
            books [i].details ();
        }
        S.O.P. ("\n Book details : ");
        for ( Book book : books )
        {   S.O.P. (book); }
    }
}
```

Output:

Enter thenumber of books :

2

Enter name of the book:

Silent Patient

Enter name of the Author:

Alex Michaelides

Enter price :

499

Enter no. of pages :

325

Enter name of the Author :

Holly Jackson

Enter price :

450

Enter number of pages :

400

Book details:

Book Name : Silent Patient

Author : Alex Michaelides

Price : 499

Number of pages : 325

Book Name : AGGGTM

Author : Holly Jackson

Price : 450

Number of pages : 400.

12/11/24

CODE :

```
import java.util.Scanner;

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

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

    @Override
    public String toString() {
        String bookDetails = "Book Name: " + this.name + "\n" +
                             "Author Name: " + this.author + "\n" +
                             "Price: " + this.price + "\n" +
                             "Number of Pages: " + this.numPages + "\n";
        return bookDetails;
    }
}

public class BooksData {
```

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

        System.out.print("Enter the Number of Books: ");
        int n = s.nextInt();
        Book[] books = new Book[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter name of book " + (i + 1) + ": ");
            String name = s.next();
            System.out.print("Enter author of book " + (i + 1) + ": ");
            String author = s.next();
            System.out.print("Enter price of book " + (i + 1) + ": ");
            int price = s.nextInt();
            System.out.print("Enter number of pages in book " + (i + 1) + ": ");
            int numPages = s.nextInt();

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

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

        s.close();
    }
}
```

Program 4:

Implement Abstract Class

LAB PROGRAM - 4

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

```
import java.util.Scanner;
abstract class Shape {
    float dim1, dim2;
    Shape() {}
    abstract void printArea();
}
Class Rectangle extends Shape {
    Rectangle() {}
    void getd() {
        Scanner sc = new Scanner(system.in);
        System.out.println("Enter length: ");
        dim1 = sc.nextFloat;
        S.O.P. ("Enter breadth: ");
        dim2 = sc.nextFloat;
    }
    void printArea() {
        double area = dim1 * dim2;
        System.out.println("Area of rectangle: "
                                    + area);
    }
}
```

```
class Triangle extends Shape {
    Triangle(){}
    void getd() {
        Scanner sc = new Scanner(System.in);
        S.O.P. ("Enter height : ");
        dim1 = sc.nextFloat();
        S.O.P. ("Enter base : ");
        dim2 = sc.nextFloat();
    }
    void printArea() {
        double Area = 0.5 * dim1 * dim2;
        S.O.P. ("Area of Triangle : " + Area);
    }
}

class cincle extends shape {
    circle(){}
    void getd() {
        Scanner sc = next Scanner (System.in);
        System.out.println ("Enter radius : ");
        dim1 = sc.nextFloat();
        dim2 = 0.0f;
    }
    void printArea() {
        double area = math PI * dim1 * dim1;
        S.O.P. ("Area of cincle : " + area);
    }
}
```

```
class Main {
    psvm (String args[]) {
        Rectangle rect = new Rectangle();
        Triangle tri = new Triangle();
        Circle circ = new Circle();
        rect.getd();
        tri.getd();
        circ.getd();
        rect.printArea();
        tri.printArea();
        circ.printArea();
    }
}
```

Output:

```
Enter length of rectangle:
2.5
Enter breadth of rectangle;
2.5
Enter height of triangle: 4.2
Enter base of triangle : 2
Enter radius of circle: 5.5

Area of Rectangle : 6.25
Area of Triangle : 4.1999
Area of circle : 95.03317
```

19/11/24

CODE :

```
import java.util.Scanner;
abstract class Shape
{
        float dim1, dim2;
        Shape() {}
        abstract void printArea();
}

class Rectangle extends Shape
{
        Rectangle() {}
        void getd()
        {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter length of rectangle:");
            dim1=sc.nextFloat();
            System.out.println("Enter breadth of rectangle:");
            dim2=sc.nextFloat();
        }

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

```
class Triangle extends Shape
{
        Triangle() {}
        void getd()
        {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter height of triangle:");
            dim1=sc.nextFloat();
            System.out.println("Enter base of triangle:");
            dim2=sc.nextFloat();
        }
        void printArea()
        {
        double area = 0.5 * dim1 * dim2;
        System.out.println("Area of Triangle: " + area);
        }
}

class Circle extends Shape
{
   Circle() {}
        void getd()
        {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter radius of circle:");
            dim1=sc.nextFloat();
```

```
            dim2=0.0f;
        }
        void printArea()
        {
        double area = Math.PI * dim1 * dim1;
        System.out.println("Area of Circle: " + area);
        }
}

class Main
{
   public static void main(String[] args)
        {
        Rectangle rect = new Rectangle();
        Triangle tri = new Triangle();
        Circle circ = new Circle();
            rect.getd();
            tri.getd();
            circ.getd();
        rect.printArea();
        tri.printArea();
        circ.printArea();
        }
}
```

Program 5:

Bank Account Management

LAB PROGRAM-5

Q) Create a banking system to acheive following tasks:
a) accept deposit from customer & update balance.
b) Display balance.
c) Compute & deposit interest.
d) Permit withdrawal & update balance.
check for minimal balance, impose penalty if necessary & update the balance.

```
import java.util.Scanner;
class Account {
    String custname, accnum;
    double deposit, balance, withdrawalamt;
    void getd() {
        Scanner sc = new Scanner(system.in);
        S.O.P. ("Enter Customer name: ");
        custname = sc.nextline();
        S.O.P. ("Enter acc no: ");
        accnum = sc.nextline();
        S.O.P. ("Enter deposit amount:");
        deposit = sc.nextDouble();
        balance = deposit;
        S.O.P. ();
    }
    void putd() {
        S.O.P. ("Customer name: " + custname);
        S.O.P. ("Account number: " + accnum);
        S.O.P. ();
    }
}
```

```
class curracc extends account {
    void balancecheck() {
        if (balance <= 1000) {
            S.O.P. ("You have less than min. balance!
                    ₹500 deducted! ");

            balance -= 500;
        }
    }
    void calcDisplayBalance() {
        S.O.P. ("Current acc details: ");
        putd();
        S.O.P. ("Amount to be withdrawn");
        Scanner sc = new Scanner (System.in);
        withdrawalamt = sc.nextDouble();
        balance -= withdrawalamt;
        balancecheck();
        S.O.P. ("Balance : " + Balance);
        S.O.P. ();
    }
}
class savacc extends account {
    void interestcalc () {
        balance = balance + (0.07 * balance);
    }
    void calcdisplayBalance () {
        Scanner sc = new Scanner (System.in);
        S.O.P. ("Enter customer name ");
        S.O.P. ("Savings acc details");
        putd();
        S.O.P. ("Amount to be withdrawn");
        withdrawal amt -= sc.nextDouble();
        S.O.P. ("Balance before addition of interest:
                + balance);
```

```
            interestcalc();
            S.o.p. ("Balance after addn of interest: "+balance)
            S.o.p.();
        }
}
class Bank {
    public static void main (String args[]) {
        Scanner sc = new Scanner (System.in);
        String acctype;
        S.o.p. ("Enter acc type: ");
        acctype = sc.nextline();
        if (acctype.equals ("Savings account")) {
            Savacc sacc = new Savacc();
            Sacc.getd();
            Sacc.calcDisplayBalance();
        }
        else if (acctype.equals ("current account")) {
            curracc cacc = new curracc();
            cacc.getd();
            cacc.calcdisplay balance();
        }
        else {
            S.o.p. ("Invalid acc type");
        }
    }
}
```

Output:

Enter type of account (savings account / current account):
Savings account
Enter customer name:
Manaswi
Enter customer account number:
123ABC789
Enter deposit amount:
25000

Savings account details:
Customer name: Manaswi
Account number: 123ABC789

Enter amount to be withdrawn
10000
Balance before addn of interest: 15000.00
Balance after addn of interest: 16050.00

19/11/24

CODE :

```
import java.util.Scanner;


class Account {
    protected String customerName;
    protected String accountNumber;
    protected double balance;
    protected String accountType;
    public Account(String customerName, String accountNumber, String
accountType, double initialBalance) {
        this.customerName = customerName;
        this.accountNumber = accountNumber;
        this.accountType = accountType;
        this.balance = initialBalance;
    }
    public void deposit(double amount) {
        balance += amount;
        System.out.println("Deposit successful! Current balance: " + balance);
    }
    public void displayBalance() {
        System.out.println("Account balance: " + balance);
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawal successful! Current balance: " +
balance);
        } else {
```

```
            System.out.println("Insufficient balance! Withdrawal failed.");
        }
    }
    public String getAccountType() {
        return accountType;
    }
}
class SavAcct extends Account {
    private static final double interestRate = 0.04;
    public SavAcct(String customerName, String accountNumber, double
initialBalance) {
        super(customerName, accountNumber, "Savings", initialBalance);
    }
    public void computeInterest() {
        double interest = balance * interestRate;
        balance += interest;
        System.out.println("Interest of " + interest + " has been added to your
account. New balance: " + balance);
    }
}
class CurAcct extends Account {
    private static final double MIN_BALANCE = 500;
    private static final double PENALTY = 50;
    public CurAcct(String customerName, String accountNumber, double
initialBalance) {
        super(customerName, accountNumber, "Current", initialBalance);
    }
```

```
    public void checkMinimumBalance() {
        if (balance < MIN_BALANCE) {
            balance -= PENALTY;
            System.out.println("Balance is below minimum. A penalty of " +
PENALTY + " has been charged. New balance: " + balance);
        }
    }
}

public class Bank {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter customer name: ");
        String customerName = scanner.nextLine();
        System.out.print("Enter account type (1 for Savings, 2 for Current): ");
        int accountChoice = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter account number: ");
        String accountNumber = scanner.nextLine();
        Account account = null;
        if (accountChoice == 1) {
            System.out.print("Enter initial deposit for Savings account: ");
            double initialDeposit = scanner.nextDouble();
account = new SavAcct(customerName, accountNumber, initialDeposit);
        } else if (accountChoice == 2) {
            System.out.print("Enter initial deposit for Current account: ");
            double initialDeposit = scanner.nextDouble();
```

```
            account = new CurAcct(customerName, accountNumber,
initialDeposit);
        } else {
            System.out.println("Invalid choice! Exiting program.");
            return;
        }
        boolean running = true;
        while (running) {
            System.out.println("\nBank Operations Menu:");
            System.out.println("1. Deposit \n 2.withdraw \n 3.Display Balance 4.
Compute Interest (Savings account only) 5.      Check and apply minimum
balance penalty (Current account only) 6.Exit ");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter deposit amount: ");
                    double depositAmount = scanner.nextDouble();
                    account.deposit(depositAmount);
                    break;
                case 2:
                    System.out.print("Enter withdrawal amount: ");
                    double withdrawAmount = scanner.nextDouble();
                    account.withdraw(withdrawAmount);
                    break;
                case 3:
                    account.displayBalance();
                    break;
```

```
            case 4:
                if (account instanceof SavAcct) {
                    ((SavAcct) account).computeInterest();
                } else {
                    System.out.println("Interest can only be computed for Savings
account.");
                }
                break;
            case 5:
                if (account instanceof CurAcct) {
                    ((CurAcct) account).checkMinimumBalance();
                } else {
                    System.out.println("Minimum balance check can only be applied
to Current account.");
                }
                break;
            case 6:
                System.out.println("Exiting program.");
                running = false;
                break;
            default:
                System.out.println("Invalid choice! Please select a valid option.");
        }
    }
    scanner.close();
  }
}
```

Program 6:

Implement Packages

LAB PROGRAM - 6

Q) Create package CIE with 2 classes Student & Internals create another package SEE which has class externals which in turn is a child class of Student.

```
Package CIE;
import java.util.Scanner;
public class Student {
    String name;
    String usn;
    int sem;
    public void getd() {
        Scanner sc = new Scanner(System.in);
        S.O.P.("Enter USN: ");
        usn = sc.nextline ();
        S.O.P.("Enter name:");
        name = sc.nextline ();
        S.O.P.("Enter semester: ");
        sem = sc.nextInt();
    }
    public void display() {
        S.O.P.();
        S.O.P.("usn: " + usn);
        S.O.P.("name: " + name);
        S.O.P.("semester: " + sem);
        S.O.P.();
    }
}
```

```
package CIE;
import java.util.Scanner;

public class Internals {
    public int markscie[] = new int[5];
    public void getmarks() {
        for (int i=0; i<5; i++) {
            Scanner sc = new scanner (system.in);
            S.O.P. (" marks in subject " + (i+1));
            markscie[i] = sc.nextInt();
        }
    }
    public int returnmarks cie (int i) {
        return markscie [i];
    }
}


package SEE;
import java.util.scanner;
import CIE.Student;
import CIE.Internals;

public class externals extends student {
    int marksee[] = new int[5];
    public void getmarks() {
        for (int i=0; i<5; i++) {
            Scanner sc = new Scanner(system.in);
            System.out.println (" marks in subject " + (i+1));
            marksee[i] = sc.nextInt();
        }
    }
}
```

```
    public void calctotalmarks (internals il) {
        for (int i=0 ; i<5 ; i++) {
            S.O.P. (" Subject " + (i+1) + ": " +
                (il.returnmarksciie [i])+ (marksee[i]/2));
        }
        S.O.P.();
    }
}

import CIE.Student ;
import CIE.internals;
import SEE.Externals;
import java.util.Scanner;

public class Main {
    psvm(String args[]) {
        Scanner sc = new Scanner(system.in);
        S.O.P. (" Number of Students: ");
        int n = sc.nextInt();
        internals [] il = new Internals [n];
        externals [] el = new Externals [n];
        for (int i=0; i<n; i++) {
            S.O.P. ("Student " +(i+1)+ " details: ");
            el[i] = new externals ();
            il [i] = new internals ();
            el [i] . getd ();
            il [i] . getmarks ();
            el [i] . getmarks ();
        }
        for (int i=0 ; i<n ; i++) {
            el [i] . display ();
            el [i] . calctotalmarks (il [i]);
        }
    }
}
```

(javac CIE /*.java SEE/*.java Main.java)

Output:

```
Number of students:
1
Studen 1 details:
Enter usn: 1BM23CS320
Enter name: Ram
Enter semester: 03
Enter CIE marks
marks in subject 1: 45
marks in subject 2: 49
marks in subject 3: 48
marks in subject 4: 40
marks in subject 5: 50
Enter SEE marks
marks in subject 1: 100
marks in subject 2: 95
marks in subject 3: 98
marks in subject 4: 99
marks in subject 5: 100

Student usn: 1BM23CS320
Student name: Ram
Semester: 03
Subject 1: 95
Subject 2: 97
Subject 3: 98
Subject 4: 90
Subject 5: 100
```

10/11/24

CODE :

```
package CIE;
import java.util.Scanner;
public class Internal {
	public int marksCie[] = new int[5];
	public void getMarks() {
		for(int i=0;i<5;i++) {
			Scanner sc = new Scanner(System.in);
			System.out.println("Enter CIE marks in subject "+(i+1));
			marksCie[i]=sc.nextInt();
		}
	}
	public int returnCieMarks(int i) {
		return marksCie[i];
	}
}


package SEE;
import CIE.Student;
import CIE.Internal;
import java.util.Scanner;
public class External extends Student {
	int marksSee[] = new int[5];
	public void getMarks() {
		for(int i=0;i<5;i++) {
			Scanner sc = new Scanner(System.in);
```

```
                System.out.println("Enter SEE marks in subject "+(i+1));
                marksSee[i]=sc.nextInt();
            }
        }
        public void calcTotalMarks(Internal i1) {
            for(int i=0;i<5;i++) {
                System.out.println("Subject "+(i+1)+":
"+(i1.returnCieMarks(i)+(marksSee[i]/2)));
            }
            System.out.println();

        }
}


import CIE.Student;
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;
public class Main_package {
        public static void main(String args[]) {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the number of students whose details you
want to enter");
            int n = sc.nextInt();
            Internals[] i1 = new Internals[n];
            Externals[] e1 = new Externals[n];
            for(int i=0;i<n;i++) {
```

```
                System.out.println("Student "+(i+1)+" details:");
                e1[i] = new Externals();
                i1[i] = new Internals();
                e1[i].getd();
                i1[i].getMarks();
                e1[i].getMarks();
            }
            for(int i=0;i<n;i++) {
                e1[i].display();
                e1[i].calcTotalMarks(i1[i]);
            }
        }
}
```

Program 7:

Implement Exception Handling

LAB PROGRAM 7

Q) Demonstrate exception handling by creating a base class "Father" and derived class "Son" which extends base class. Create a WrongAgeException when age is negative and also when son's age exceeds father's age.

Sol)

```
import java.util.Scanner;

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

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

Class Son extends Father {
    private int SonAge;
    public Son (int fatherAge, int SonAge)
      throws WrongAgeException {
        super (fatherAge);
```

```
        if (sonAge <0) {
            throw new WrongAgeException ("Son's
              age cannot be negative. ");
        }
        if (sonAge >= fatherAge) {
            throw new WrongAgeException ("Son's
              age cannot be greater than or equal
              to father's age. ");
        }
        this.sonAge = sonAge;
    }
    public String toString() {
        return "Father's Age: " + age +
          ", Son's Age: " + sonAge;
    }
}

class ExceptionInheritanceDemo {
    public static void main (String[] args) {
        try {
            Father father = new Father(45);
            System.Out.println (" Father created
              with age: " + father.age);
            Son son = new Son(45, 20);
            System.out.println (son);
        }
        catch (WrongAgeException e) {
            System.err.println ("Exception occured: "+a);
        }
```

```
try {
    Son invalidSon = new Son (40,40);
}
catch ( WrongAgeException e) {
    S.O.P. ("Exception occured : "+ e.getMessage());
}
try {
    Father invalidFather = new Father(-50);
}
catch ( WrongAgeException e) {
    S.O.P. ("Exception occured : " + e.getMessage());
}
}
}
```

Output :

Father's age :
45
Father's age : 45 , Son's Age : 20

Exception occured : Son's age cannot be greater than or equal to father's age.

Exception occured : Father's age cannot be negative.

26/11/24

CODE :

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

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

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

```
        this.sonAge = sonAge;
    }
    public String toString() {
        return "Father's Age: " + age + ", Son's Age: " + sonAge;
    }
}

public class ExceptionInheritanceDemo {
    public static void main(String[] args) {
        try {
            Father father = new Father(45);
            System.out.println("Father created with age: " + father.age);
            Son son = new Son(45, 20);
            System.out.println(son);
        } catch (WrongAgeException e) {
            System.err.println("Exception occurred: " + e.getMessage());
        }
        try {
            Son invalidSon = new Son(40, 40);
        } catch (WrongAgeException e) {
            System.err.println("Exception occurred: " + e.getMessage());
        }
        try {
            Father invalidFather = new Father(-5);
        } catch (WrongAgeException e) {
            System.err.println("Exception occurred: " + e.getMessage());
        }
    }
}
```

Program 8:

Multithreading, Creating Threads in Java

Lab program 8

Q) WAP which creates two threads, one thread displaying "BMS college of Engineering" once every 10 sec. and another display "CSE" once every 2 sec.

```
Class BMSCollegethread extends thread {
    public void run(){
        while (true) {
            try {
                S.O.P. ("BMS College of Engineering");
                Thread. Sleep (10000);
            }
            catch (InterruptedException e) {
                S.O.P. (e);
            }
        }
    }
}
class CSEThread extends Thread {
    public void run(){
        while (true){
            try {
                S.O.P. ("CSE");
                Thread.sleep(2000);
            }
            catch (InterruptedException e) {
                S.O.P. (e);
            }
        }
    }
}
```

```
public class Main {
    psvm (String [] args) {
        BMSCollegeThread bmsThread = new
                         BMSCollegeThread();
        CSEThread Csethread = new CSEThread;
        bmsThread.start();
        CSEThread.start();
    }
}
```

O/P:

```
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
BMS College of Engineering
CSE
CSE
^C
```

29/11/24

CODE :

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

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

```
public class Main {
    public static void main(String[] args) {
        BMSCollegeThread bmsThread = new BMSCollegeThread();
        CSEThread cseThread = new CSEThread();
        bmsThread.start();
        cseThread.start();
    }
}
```

Program 9:

Interface to Perform Integer Division

Lab Program - 9

Q) WAP that creates a user interface to perform integer divisions. The user enters 2 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 an 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 DivisionMain1() {
        setLayout (new Flowlayout());
        dResult = new Button ("RESULT");
```

```
Label Number 1 = new Label ("Number 1:",
                                Label.RIGHT);
Label Number2 = new Label("Number 2: ",Label.LEFT);
num1 = new TextField (5);
num2 = new TextField (5);
outResult = new Label ("Result: ", Label.RIGHT);
add (number1);
add (num1);
add (number 2);
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);
        }
});
}
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 (ArithmeticException 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;
    }
}
```

CODE :

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

JLabel anslab = new JLabel();
// add in order :)
jfrm.add(err); // to display error bois
```

```
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);
ActionListener l = new ActionListener() {
public void actionPerformed(ActionEvent evt) {
System.out.println("Action event from a text field");
}
};
ajtf.addActionListener(l);
bjtf.addActionListener(l);
button.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent evt) {
try{
int a = Integer.parseInt(ajtf.getText());
int b = Integer.parseInt(bjtf.getText());
int ans = a/b;
alab.setText("\nA = " + a);
blab.setText("\nB = " + b);
anslab.setText("\nAns = "+ ans);
}
catch(NumberFormatException e){
alab.setText("");
blab.setText("");
```

```
anslab.setText("");

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

Program 10:

Implement Deadlock and Inter-process Communication

Lab program 10

Q) Demonstrate Interprocess communication and deadlock.

```
class Q {
  int n;
  boolean valueSet = false;
  synchronized int get() {
      while (!valueSet) {
          try {
              S.O.P. ("\n Consumer waiting \n");
              wait();
          }
          catch (InterruptedException e) {
              S.O.P. ("Got: " + n);
          }
          S.O.P. ("Got " + n );
          valueset = false;
          S.O.P. ("\n Intimate producer \n");
          notify();
          return n;
      }
      synchronized void put (int n) {
          while
synchronized void put (int n) {
    while(valueset)
        try {
            S.O.P.( &quot ; \n Producer waiting
                \n &qout;);
        }   wait();
```

```
        catch (InterruptedException e){
            S.O.P. (&quot; InterruptedException
                caught &quot;);
        }
        this.n = n
        valueSet = true;
        S.O.P. (&quot; Put: &quot; +n);
        S.O.P. (&quot; \n Intimate consumer \n &quot;);
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q = q;
        new Thread (this, &quot; Producer &quot;);
    }
    public void run() {
        int i = 0;
        while (i &lt; 15) {
            q.put (i++);
        }
    }
}
```

```
class Consumer implements Runnable {
    Q q;
    Consumer (Q q) {
        this q = q;
        new Thread ( this, &quot; Consumer &quot).
                                    start ();
    }
    public void run() {
        int i =0;
        while ( i&lt; 15) {
            int r = q.get();
            S.O.P. (&quot; consumed : &quot;+r);
            i++;
        }
    }
}
class PCfixed {
    public static void main (String args[]) {
        Qq = new Q();
        new Producer( q);
        new Consumer (q);
        S.O.P. ( &quot; Press Control -C to stop. &quot;);
    }
}
```

Deadlock →

```
class A {
    synchronized void foo(B b) {
        String name = Thread.currentThread().getName();
        S.O.P. (name + " entered A.foo");
        try {
            Thread.sleep(1000);
        }
        catch (Exception e) {
            S.O.P. ("A Interrupted);
        }
        S.O.P. (name + " tryinto 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();
    }
}
```

CODE :

```
//PCFixed.java
class Q {
int n;
boolean valueSet = false;
synchronized int get() {
while(!valueSet)
try {
System.out.println("Consumer waiting");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedExceptioncaught");
}
System.out.println("Got: " + n);
valueSet = false;
System.out.println("Intimate Producer");
notify();
return n;
}
synchronized void put(int n) {
while(valueSet)
try {
System.out.println("Producer waiting");
wait();
} catch(InterruptedException e) {
System.out.println("InterruptedException caught");
}
```

```
this.n = n;
valueSet = true;
System.out.println("Put: " + n);
System.out.println("Intimate Consumer");
notify();
}
}
class Producer implements Runnable {
Q q;
Producer(Q q) {
this.q = q;
new Thread(this, "Producer").start();
}
public void run() {
int i = 0;
while(i<15) {
q.put(i++);
}
}
}
class Consumer implements Runnable {
Q q;
Consumer(Q q) {
this.q = q;
new Thread(this, "Consumer").start();
}
public void run() {
```

```
int i=0;
while(i<15) {
int r=q.get();
System.out.println("consumed:"+r);
i++;
}
}
}
class PCFixed {
public static void main(String args[]) {
Q q = new Q();
new Producer(q);
new Consumer(q);
System.out.println("Press Control-C to stop.");
}
}

//Deadlock.java
class A {
synchronized void foo(B b) {
String name =Thread.currentThread().getName();
System.out.println(name + " enteredA.foo");
try {
Thread.sleep(1000);
} catch(Exception e) {
System.out.println("A Interrupted");
}
```

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

```
B b = new B();
Deadlock() {
Thread.currentThread().setName("MainThread");
Thread t = new Thread(this,"RacingThread");
t.start();
a.foo(b); // get lock on a in this thread.
System.out.println("Back in mainthread");
}
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();
}
}
```