VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

Object Oriented Java Programming

(22CS3PCOOJ)

Submitted by

MONISH P (1BM22CS163)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019 December-2023 to April-2024

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 Programming in Java (22CS3PCDOOJ)" carried out by **MONISH P (1BM22CS163),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2023-24. The Lab report has been approved as itsatisfies the academic(22CS3PCDBM) work prescribed requirements in respect of a Database Management Systems for the said degree.

Prof. NANDHINI VINEETH.

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 discriminant b2-4ac is negative, display a message stating that there are no real solutions.

```
import java.util.*;
import java.math.*;
class Quadratic
{
int a,b,c;
double r1,r2,d;
void coeff()
{
Scanner s=new Scanner(System.in);
System.out.println("Enter coeeficients a,b,c");
a = s.nextInt();
b = s.nextInt();
c = s.nextInt();
d = (b*b) - (4*a*c);
}
void evalu()
{
while(a==0)
{
System.out.println("Not a QE.");
System.out.println("Enter non zero coefficient");
Scanner s = new Scanner(System.in);
a = s.nextInt();
}
if(d==0)
{
System.out.println("Roots are real and equal.");
r1 = (-b)/(2*a);
System.out.println("Root1=Root2="+r1);
}

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("Root2="+r1+"-i"+r2);
}
else
{
System.out.println("Roots are real and distinct");
r1 = (-b+(Math.sqrt(d)))/(2*a);
```

```
r2 = (-b-(Math.sqrt(d)))/(2*a);
System.out.println("root1= "+r1+"root2= "+r2);
}
}
}
class QuadraticEq
{
public static void main(String sx[])
{
Quadratic q = new Quadratic();
q.coeff();
q.evalu();
}
}
```

PROGRAM (1) - QUADRATIC EQN.

```
import java.util.*;
class main1;
{
    public static void main(String args[]);
    {
        Scanner S1 = new Scanner(System.in);
        double a,b,c;
        System.out.println("Enter the coefficients of the equation");
        a = s1.nextdouble();
        b = S1.nextdouble();
        c = s1.nextdouble();
        if (a==0)
        {
          System.out.println("Enter the valid coefficient");
        }
        else
        {
            double d = b*b - 4.0*a*c;
            if (d==0)
            {
                double n1 = -b/(2*a);
                System.out.println("The roots are real and equal");
                System.out.println("The roots are "+n1+"and"+n2);
            }
            else if (d>0)
            {
             double n1 = (-b+Math.pow(d,0.5))/(2*a);
             double n2 = (-b-Math.pow(d,0.5))/(2*a);
             System.out.println("The roots are real & unequal");
             System.out.println("The roots are "+n1+" and "+n2);
            }
```

```
else if (d<0)
{
    System.out.println("The roots are imaginary");
    System.out.println("The roots are "+(-b/(2*a))+"+i"+
(d/(2*a))+" and "+(-b/(2*a))+"-i"+(d/(2*a)));
}
}
}
}
```

Output:

enter the coefficients
1
2
3
The roots are imaginary
The roots are -1.0 + i -4.0 and -1.0 - i -4.0.

enter the coefficients
1
6
9
The roots are real and equal
The roots are -3.0 and -3.0

enter the coefficients
1
11
15
The roots are real and unequal
The roots are -2.5 and 3.0

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

```
import java.util.Scanner;
class Student {
String usn,name;
int[] credits,marks;
// Method to accept details of a student
void acceptDetails() {
Scanner hello = new Scanner(System.in);
System.out.print("Enter USN: ");
this.usn = hello.next();
System.out.print("Enter Name: ");
this.name = hello.next();
credits = new int[4];
marks = new int[4];
System.out.println("Enter details of credits and marks in order for 4 subjects:");
for (int i = 0; i < 4; i++) {
System.out.print("Enter credits for Subject " + (i + 1) + ": ");
credits[i] = hello.nextInt();
System.out.print("Enter marks for Subject " + (i + 1) + ": ");
marks[i] = hello.nextInt();
}
}
// Method to display details of a student
void display() {
System.out.println("Student Details.");
System.out.println("USN: " + usn);
System.out.println("Name: " + name);

for (int i = 0; i < 4; i++) {
System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i] + ", Marks: "

+ marks[i]);
}
}
// Method to calculate SGPA of a student
double calc() {
int totalCredits = 0;
double totalGradePoints = 0;
for (int i = 0; i < 4; i++) {
totalCredits += credits[i];
totalGradePoints += gradePoints(marks[i]) * credits[i];
}
return totalGradePoints / totalCredits;
}
```

```
// Method to calculate grade points based on marks
int gradePoints(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 class SGPA {
public static void main(String[] args) {
// Example usage of the Student class
Student student = new Student();
student.acceptDetails();
System.out.println("\nStudent Details:");
student.display();
System.out.println("\nSGPA: " + student.calc());
}
}
```

LAB-02

1)
```
import java.util.Scanner;
class Student {
    String USN;
    Scanner sl = new Scanner(System.in);
    String name;
    int credits[] = new int[8];
    double marks[] = new double[8];
    double Sgpa;
    void details() {
        System.out.println("Enter USN:");
        USN = sl.next();
        System.out.println("Enter Student name");
        name = sl.next();
        System.out.println("Enter the credits of subjects");
        for (int i=0; i<8; i++)
        {
            System.out.println("credits of subject:");
            credits[i] = sl.nextInt();
            System.out.println("marks of subject:");
            marks[i] = sl.nextDouble();
        }
    void displaydetails() {
        System.out.println("name:" + name);
        System.out.println("USN:" + USN);
        for (int i=0; i<8; i++)
        {
            System.out.println("Subject" + (i+1));
            System.out.println("Credits" + credits[i]);
            System.out.println("Marks" + marks[i]);
        }
    }
void calcSgpa() {
```

```
for(int i=0; i<8; i++)
{   int grade;
    if (marks[i] >= 90 && marks[i] <= 100)
    { grade = 10; }
    else if (marks[i] >= 80 && < 90)
    { grade = 9; }
    else if (marks[i] >= 70 && < 80)
    { grade = 8; }
    else if (marks[i] >= 60 && < 70)
    { grade = 7; }
    else if (marks[i] >= 50 && marks[i] < 60)
    { grade = 6; }
    else if (marks[i] >= 40 && marks[i] < 50)
    { grade = 5; }
    else if (marks[i] >= 30 && marks[i] < 40)
    { grade = 4; }
    else {
        System.out.println("Fail no sgpa");
        exit(0);
    }
    double gpa = grade * credits[i];
    gpa += (grade * credits[i]);
}
double sgpa;
sgpa = gpa / 20;
System.out.println("SGPA = " + sgpa);
}

void student{
{ psvm(String args[])
    Student a = new Student();
    A.setdetails();
    A.displaydetails();
}
}
```

```
Enter the USN: 163CS
Enter name: MONISH
Enter credits: 4
Enter marks: 90
             : 4
             : 80
             : 3
             : 94
             : 3
             : 97
             : 3
             : 86
             : 1
             : 90
             : 1
             : 92
             : 1
             : 96
The details:
USN: 163CS
Name: MONISH
~~Credits~~ Subject 1
Credits : 4
Marks: 90
Subject 2
Credits: 4
Marks: 80
Subject 3
Credits : 3
Marks, 94
Subject 4
Credits, 3
Marks : 96
```

Date: / /

Subject 5
Credits: 3
Marks: 86

Subject 6
Credits: 1
Marks: 90

Subject 7
Credits: 1
Marks: 92

SGPA: 9.6

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.

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

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

public void setDetails() {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter book name: ");
this.name = scanner.nextLine();
System.out.print("Enter author name: ");
this.author = scanner.nextLine();
System.out.print("Enter price: ");
this.price = scanner.nextDouble();
System.out.print("Enter number of pages: ");
this.numPages = scanner.nextInt();

}

public void getDetails() {
System.out.println("Book Name: " + name);
System.out.println("Author: " + author);
System.out.println("Price: $" + price);
System.out.println("Number of Pages: " + numPages);
}

public String toString() {
return "Book Details:\n" +
"Name: " + name + "\n" +
"Author: " + author + "\n" +
"Price: $" + price + "\n" +
"Number of Pages: " + numPages;

}
```

```
}
public class Books {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter the number of books: ");
int n = scanner.nextInt();
// Creating an array to store n book objects
Book[] books = new Book[n];
// Creating n book objects and setting their details
for (int i = 0; i < n; i++) {
System.out.println("\nEnter details for Book " + (i + 1) + ":");
books[i] = new Book("", "", 0.0, 0);
books[i].setDetails();
}
System.out.println("\nDetails of all books:");
for (int i = 0; i < n; i++) {
System.out.println("\nBook " + (i + 1) + ":");
books[i].getDetails();
}
System.out.println("\nComplete details of all books:");
for (int i = 0; i < n; i++) {
System.out.println("\nBook " + (i + 1) + ":\n" + books[i].toString());
}
}
}
```

Book Problem

```
import java.util.Scanner;
class Book {
    string name;
    string author;
    double price;
    int num_pages;
    Scanner sl = new Scanner(System.in);
    Book(){}
    Book(string n, string authorn, double pricn, int num_pagn)
    {
      this.name = n;
      this.author = author;
      this.price = price;
      this.num_pages = pages;
    }
    void setName(){
      System.out.println("Enter name of book");
      name = sl.next().();
    }
    void setAuthor(){
      System.out.println("Enter the name of author");
      author = sl.next().();
    }
    void setPrice(){
    System.out.println("Enter price of book");
    price = sl.nextDouble();
    }
    void setNumpages(){
    System.out.println("Enter no of pages");
    num_pages = sl.nextInt();
    }
```

```
string getname() { return name; }
string getauthor() { return author; }
double getprice() { return price; }
int getpages() { return num_pages; }
public string tostring()
{
    return ("Name: " + getName() + "\n" + "Author: " + getAuthor()
    + "\n" + "Price: " + getPrice() + "\n" + "Num pages: " +
        getPages());
}
}
class main {
    public static void main (String args[]) {
        Book B[] = new Book[2];
        Book B1 = new Book ("XYZ", "ABC", 1200, 200);
        S.O.P(B1);
        for (int i=0; i<2; i++)
            B[i] = new Book();
        { B[i].setName();
          B[i].setAuthor();
          B[i].setPrice();
          B[i].setNumPages();
          System.out.println(B[i]);
        }
    }
}
```

Output:

Enter name of book:
XYZ
Enter name of author:
ABC
Enter price of Book:
200.20

Enter no of pages:
100
X47 ABC 200.20 , 100.
Enter name of book:
DEF
Enter name of Author:
ABC
Enter price of Book:
500
Enter number of pages:
300
DEF ABC 500, 300

Error:
Enter name of Book
ABC
Enter name of Author:
DEF
Enter price of Book:
ABC
↳ Error.

311

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.

```
abstract class Shape {
   int a;
   int b;

   }

   abstract void printArea();
}

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

      void printArea() {
      int area = a * b;
      System.out.println("Area of Rectangle: " + area);
   }
}

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


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

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

      void printArea() {
      double area = 3.14* a * a;
      System.out.println("Area of Circle: " + area);
   }
```

```
}

public class Main {
    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 10);
        Triangle triangle = new Triangle(4, 6);
        Circle circle = new Circle(7);

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

4) SHAPE PROGRAM

```
abstract class Shape {
    int a, b;
    abstract void printArea();
    Scanner sc = new Scanner(); S.O.P("Enter Dimensions")
    a = sc.nextInt();
    b = sc.nextInt();
    }
class rectangle extends shape, {
    rectangle () {}
    rectangle (double x, double h){
    this.x = x;
    this.h = h;
    }
    void printArea(){
    System.out.println("Area of rectangle"+(x*h));
    }
}
class triangle extends shape{
    triangle(){}
    triangle(double x, double h)
    { this.x = x;
      this.h = h;
    }
    void printArea(){
    System.out.println("Area of triangle"+((1/2)*x*h));
    }
}
class circle extends shape{
    circle(){}
    circle(double x){
     System.out.println("Area of circle"+(3.14*x*x));
}}
```

```
class Area {
    PSVM(String args[])
{
    rectangle r1 = new rectangle (20,3);
    triangle t1 = new triangle (5,10);
    circle c1 = new circle (4);
    r1.printArea();
    t1.printArea();
    c1.printArea();
}
}
```

Output:

Area of rectangle: 60
Area of triangle: 25.0
Area of circle: 50.24

22/1

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.

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

   Account(String name, int accNo, String type, double bal) {
      customerName = name;
      accountNumber = accNo;
      accountType = type;
      balance = bal;
   }

   void deposit(double amount) {
      balance += amount;
      System.out.println("Deposit of Rs." + amount + " successful");
   }

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

   void withdraw(double amount) {
      if (balance - amount >= 0) {
         balance -= amount;
         System.out.println("Withdrawal of Rs." + amount + " successful");
      } else {
         System.out.println("Insufficient balance for withdrawal");
      }
   }
}
```

```
class CurAcct extends Account {
   double minimumBalance;
   double serviceCharge;

   CurAcct(String name, int accNo, String type, double bal, double minBal, double charge) {
      super(name, accNo, type, bal);
      minimumBalance = minBal;
      serviceCharge = charge;
   }

   void withdraw(double amount) {
      if (balance - amount >= minimumBalance) {
         balance -= amount;
         System.out.println("Withdrawal of Rs." + amount + " successful");
      } else {
         System.out.println("Insufficient balance for withdrawal. Service charge of Rs." +
serviceCharge + " applied.");
         balance -= serviceCharge;
      }
   }

   void checkbook() {
      System.out.println("Checkbook facilities are available and will be sent soon.");
   }
}

class SavAcct extends Account {
   double interestRate;

   SavAcct(String name, int accNo, String type, double bal, double rate) {
      super(name, accNo, type, bal);
      interestRate = rate;
   }

   void computeInterest() {
      double interest = balance * (interestRate / 100);
      balance += interest;
      System.out.println("Interest of Rs." + interest + " added to account");
   }

   void checkbook() {
      System.out.println("Checkbook facilities not available.");
   }
}

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

```
        Scanner scanner = new Scanner(System.in);

        CurAcct currentAccount = new CurAcct("Monish", 123456, "Current", 5000, 1000, 50);
        SavAcct savingsAccount = new SavAcct("Navaneeth", 654321, "Savings", 10000, 5);

        System.out.println("Current Account Details:");
        currentAccount.displayBalance();
        currentAccount.deposit(2000);
        currentAccount.displayBalance();
        currentAccount.withdraw(7000);
        currentAccount.displayBalance();
        currentAccount.withdraw(3000);
        currentAccount.displayBalance();
        currentAccount.checkbook();

        System.out.println("\nSavings Account Details:");
        savingsAccount.displayBalance();
        savingsAccount.deposit(5000);
        savingsAccount.displayBalance();
        savingsAccount.computeInterest();
        savingsAccount.displayBalance();
        savingsAccount.withdraw(15000);
        savingsAccount.displayBalance();
        savingsAccount.checkbook();

    }
}
```

BANKING PROBLEM:

```
import java.util.Scanner;
class Account {
    String customerName;
    int accountNumber;
    String accountType;
    double balance;
    Account (String name, int accNo, String type, double bal) {
        customerName = name;
        accountNumber = accNo;
        accountType = type;
        balance = bal;
    }
    void deposit (double amount) {
        balance += amount;
        S.O.P("Deposit of Rs." + amount + "successfull");
    }
    void displayBalance () {
        S.O.P("Account Balance : Rs." + balance);
    }
    void withdraw (double Amount) {
        if (balance - amount >= 0) {
            balance -= amount;
            S.O.P ("withdrawal of Rs." + amount + "successful");
        }
        else {
            S.O.P("Insufficient balance for withdraw");
        }
    }
}
```

```
Class CurrAcct extends Account{
    double minimumBalance;
    double SurviceCharge;
    CurrAcct (String name, int accNo, String type,
double Bbal, double minBal, double charge){
    super (name, accNo, type, bal);
    minimumBalance = minBal;
    SurvicCharge = charge;
    }
void withdraw (double amount){
  if (balance - amount >= minimumBalance){
     balance -= amount;
     S.o.P ("Withdrawal of Rs." + amount + "succesful");
  }
  else
     S.o.P ("Insufficient balance for withdrawal.
Survice charge of Rs." + surviceCharge + "applied");
     balance -= suviceCharge;
  }
}
void checkbook(){
     S.o.P ("Checkbook facilities are available
and will be sent soon");
  }
}

Class SaveAcct extends Acct{
    double interestrate;
    Save Acct (String name, int accNo, String type,
      double bal, double rate){
  super (name, accNo, type, bal);
  interest rate = rate;
}
```

```
void computeInterest(){
    double interest = balance * (interestRate/100);
    balance += interest;
    S.o.p("Interest of Rs." + interest + "added to account");
  }
void checkbook(){
    S.o.p("Checkbook facilities not available");
  }
}

public class Bank {
  psvm(String args[]){
  CurrAcct currentAccount = new CurrAcct("Manish", 123456, "Current", 5000, 100050);
  SavAcct savingsAccount = new SavAcct("Navaneeth", 645321, "Savings", 10000, 5);

S.o.p("Current Account Details:");
currentAccount.displayBalance();
currentAccount.deposit(2000);
currentAccount.displayBalance();
currentAccount.withdraw(7000);
currentAccount.displayBalance();
currentAccount.withdraw(3000);
currentAccount.displayBalance();
currentAccount.checkbook();

S.o.p("\n Savings Account details:");
savingsAccount.displayBalance();
savingsAccount.deposit(5000);
savingsAccount.displayBalance();
savingsAccount.computeInterest();
savingsAccount.displayBalance();
savingsAccount.withdraw(15000);
```

```
savingsAccount.displayBalance();
savingsAccount.checkbook();

}
}
```

O/p:

Current Account Details:
Account Balance: Rs 5000.0
Deposit of Rs. 2000 successful.
Account Balance: 7000.0
Insufficient balance for withdrawal. Service charge of Rs.500 applied.
Account Balance: Rs. 6950.0
Withdrawal of Rs. 3000 successful.
Account Balance: Rs. 3950.0
Checkbook facilities are available and will be sent soon.

Savings Account Details:
Account Balance: Rs. 10000.0
Deposit of Rs. 5000 successful.
Account balance: 15000.0
Interest of Rs. 750.0 added to account.
Account Balance: 15750.0
Withdrawal of Rs 5000 successful.
Account balance: Rs 750.0
Checkbook facilities not available.

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

```
//file Student.java in Monish/CIE
package CIE;
import java.util.Scanner;
public class Student{
public String usn;
public String name;
public int sem;

public void accept(){
Scanner s = new Scanner(System.in);
System.out.println("Enter Name:");
this.name = s.nextLine();
System.out.println("Enter usn:");
this.usn = s.nextLine();
System.out.println("Enter sem");
this.sem = s.nextInt();
}
public void display(){
System.out.println("Name: " + this.name + "\nUSN: " + this.usn + "\nSem: " + this.sem);
}
}

//file Internal.java in Monish/CIE
package CIE;
import java.util.Scanner;
public class Internal extends CIE.Student{
public int m[] = new int[5];
CIE.Student student = new CIE.Student();
public void accept(){
student.accept();
Scanner s1 = new Scanner(System.in);
System.out.println("Enter Internal Marks:");
for(int i=0;i<5;i++){
m[i] = s1.nextInt();
}
}
public void display(){
student.display();
for(int i=0;i<5;i++){
```

```
System.out.println("Marks of sub" + (i+1) + " = " + m[i]);
}
}
}

//file External.java in Monish/SEE
package SEE;
import java.util.Scanner;
import CIE.Internal;
import CIE.Student;
public class External extends CIE.Student{
public int x[] = new int[5];
public void accept(){
Scanner s2 = new Scanner(System.in);
System.out.println("Enter External Marks:");
for(int i=0;i<5;i++){
x[i] = s2.nextInt();
}
}
public void display(){
super.display();
for(int i=0;i<5;i++){
System.out.println("Marks of sub" + (i+1) + " = " + x[i]);
}}}
//file Final.java in Monish
import java.util.Scanner;
import CIE.Student;
import CIE.Internal;
import SEE.External;
public class Final{
public static void main(String[] args) {
Scanner n = new Scanner(System.in);
System.out.println("Enter n:");
int y = n.nextInt();
CIE.Internal[] c1 = new CIE.Internal[y];
SEE.External[] c2 = new SEE.External[y];
for(int i=0;i<y;i++){
c1[i] = new CIE.Internal();
c2[i] = new SEE.External();
c1[i].accept();
c2[i].accept();
// c1[i].accept();c2[i].accept();
c1[i].display();c2[i].display();
for(int j=0;j<5;j++){
double calc = c1[i].m[j]+((c2[i].x[j])/2);
System.out.println("Final marks of sub["+(i+1)+"]= "+calc);
}}}
```

5) CIE & SEE (Package Program)

```
package CIE;
public class student{
    public string USN;
    public string name;
    public int sem;
    public student (string u, string n, int s){
        this.usn = u;
        this.name = n;
        this.sem = s;
    }
}
public class internals extends CIE.student{
    public double imarks[];
    public internals (string u, string n, int s, double m[])
    {
        super(u,n,s);
        this.imarks = m;
    }
}

package SEE;
import CIE.student;
public class externals extends CIE.student{
    public double marks[];
    public externals (string u, String n, int s, double m[]){
        super(u,n,s);
        this.marks = m;
    }
}
```

```
package Result;
import CIE.Student;
import CIE.Internals;
import SEE.Externals;
public class Test {
    public static void main (String args[]) {
        double internal[] = {43, 45, 47, 44, 41};
        double external[] = {90, 87, 65, 98, 43};
        Student s1 = new Student ("1Bm22CS163", "Monish", 3);
        Internals i1 = new Internals ("1Bm22CS163", "Monish", 3, internal);
        Externals e1 = new Externals ("1Bm22CS163", "Monish", 3, external);
        S.O.P ("usn:" + s1.usn + "name:" + s1.name + "sem:" + s1.sem);
        S.O.P ("internal marks:");
        for (int i = 0; i < 5; i++) {
            S.O.P ("internal marks %d : %f", i+1, i1.imarks[i]);
        }
        S.O.P ();
        S.O.P ("external marks:");
        for (int i = 0; i < 5; i++) {
            S.O.P ("External marks %d : %f", i+1, e1.emarks[i]);
        }
    }
}
```

Output:

```
name: Monish  sem: 3
internal marks:
internal marks 1: 50.000
internal marks 2: 48.000
internal marks 3: 49.000
internal marks 4: 46.000
internal marks 5: 45.000
```

external marks :

external marks 1 : 60.000

external marks 2 : 80.000

external marks 3 : 90.000

external marks 4 : 95.000

external marks 5 : 100.000

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

```
import java.util.Scanner;
class WrongAgeException extends Exception {
public WrongAgeException(String message) {
super(message);
}
}
class Father {
private int fatherAge;
public Father(int age) throws WrongAgeException {
if (age < 0) {
throw new WrongAgeException("Age cannot be negative");
}
this.fatherAge = age;
}
}
class Son extends Father {
private int sonAge;
public Son(int fatherAge, int sonAge) throws WrongAgeException {
super(fatherAge);
if (sonAge >= fatherAge) {
throw new WrongAgeException("Son's age should be less than Father's age");
}
this.sonAge = sonAge;
System.out.println("Father's Age: " + fatherAge);
System.out.println("Son's Age: " + sonAge);
}
}
public class ExceptionInheritanceDemo {
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
try {
System.out.print("Enter Father's Age: ");
int fatherAge = scanner.nextInt();
Father father = new Father(fatherAge);
System.out.print("Enter Son's Age: ");
int sonAge = scanner.nextInt();
Son son = new Son(fatherAge, sonAge);
} catch (WrongAgeException e) {
System.out.println("Exception: " + e.getMessage());
}
}}
```

6) EXCEPTION HANDLING

```
class MyException
import java.util.Scanner;
class WrongAgeException extends Exception{
    public WrongAgeException(String msg){
        super(msg);
    }
Class Father{
    private int fatherAge;
    public Father(int age) throws WrongAgeException{
        if (age<0){
        throw new WrongAgeException("Age cannot be negative");
    }
    this.fatherAge = age;
}

class Son extends Father{
    private int sonAge;
    public Son (int fatherAge, int sonAge) throws WrongAgeException
    {
      super(fatherAge);
      if (sonAge>=fatherAge){
        throw new WrongAgeException("Son's age should be less
        than father's Age");
    }
    this.sonAge=sonAge;
    S.O.P("father's age:" + fatherAge);
    S.O.P("Son's age:" + sonAge);
}
```

```
public class ExceptionInheritanceDemo {
    psvm (String args[]) {
    Scanner s = new Scanner (System.in);
    try {
        S.O.P ("Enter father's Age:");
        int fatherAge = s.nextInt();
        Father f = new father (fatherAge);
        S.O.P ("Enter Son's Age:");
        int SonAge = s.nextInt();
        Son s = new Son (fatherAge, SonAge);
    }
    Catch (WrongAgeException e) {
        S.O.P ("Exception:" + e.getMessage());
    }
}
```

Output:

Enter father's age: -6
Enter Son's age: 10
Exception: Age cannot be negative.

Enter father's age: 6
Enter Son's age: 10
Exception: Son's Age should be less than father's Age.

Enter father's age: 10
Enter Son's age: 6
Father's Age: 10
Son's Age: 6

8. Write a java program containing 2 threads, one thread displaying "BMS College of Engineering" once every 10 seconds and another displaying "CSE" every 2 seconds.

```
class newThread implements Runnable{
Thread t;
newThread(){
t = new Thread(this,"NThread");
System.out.println("CT: "+t);
t.start();
}
public void run(){
try{
for(int i=0;i<5;i++){
System.out.println("CSE");
Thread.sleep(100);
}
}
catch(InterruptedException ie){
System.out.println("Child thread interrupted");

}
System.out.println("Child thread quitting");
}
}
class ThreadsMain3{
public static void main(String args[]){
new newThread();
System.out.println("Back in main");
try{
for(int i=0;i<5;i++){
System.out.println("BMS COLLEGE OF

ENGINEERING");
Thread.sleep(100);
}
}
catch(InterruptedException ie){
System.out.println("Main Thread Interrupted");
}
System.out.println("Main thread quitting");
}
}
```

3) WAP containing 2 threads, one thread displaying "Bms College of Engineering" once every 10 seconds & another displaying "CSE" once every 2 seconds.

```
class BmsThread ~~extends Thread~~ implements Runnable
{
  Thread t;
  newThread(){
        t = new Thread (this, "Nthread");
        S.o.P("CT:"+t);
        t.start();
    }
  public void run(){
        try { ~~for(;;)~~ while(true){
                S.o.P("CSE");
                Thread.sleep(2000);
            }}
```

```
        catch (InterruptedException ie){
            S.o.P("Child thread interrupted");
        }
        S.o.P("Child thread quitting");
    }
}

class ThreadsMain3{
    psvm (String args[]){
        new newThread();
        S.o.P("Back in main");
        try{
            while(true){
            S.o.P("BMS College of Engineering");
            Thread.sleep(10000);
            }
        }
        catch(InterruptedException ie){
            S.o.P("Main thread interrupted");
        }
        S.o.P("Main thread quitting");
    }
}
```

O/P

CT: Thread[NThread,5,main] Back in main.
BMS College of Engineering
CSE
CSE
CSE
CSE
CSE
;
;
}

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.

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

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

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

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

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

		num1.addActionListener(this);
		num2.addActionListener(this);
		dResult.addActionListener(this);
		addWindowListener(new WindowAdapter()
		{
			public void windowClosing(WindowEvent we)
			{
				System.exit(0);
			}
		});
	}

	public void actionPerformed(ActionEvent ae)
	{
		double n1,n2;
		try
```

```
		{
			if (ae.getSource() == dResult)
			{
				n1=Double.parseDouble(num1.getText());
				n2=Double.parseDouble(num2.getText());

				/*if(n2==0)
					throw new ArithmeticException();*/
				out=n1+" "+n2;
				resultNum=n1/n2;
				out+=String.valueOf(resultNum);
				repaint();

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

	}

	public void paint(Graphics g)
	{
		if(flag==0)

	g.drawString(out,outResult.getX()+outResult.getWidth(),outResult.getY()+outResult.getHeight(
)-8);
		else
		g.drawString(out,100,200);
		flag=0;
	}

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

}
```

9) AWT Program:

```
import java.awt.*;
import java.awt.event.*;
public class DivisionMain extends Frame implements
 action listener
{
   TextField num1, num2;
   Button dResult; label out Result;
   String out = " ";
   double ResultNum;
   int flag = 0;
   public DivisionMain()
   {
      SetLayout (New FlowLayout());
      dResult = new Button ("Result");
      Label number1 = new Label ("Number 1: ", Label.RIGHT);
      Label number2 = new Label ("Number 2: ", Label.RIGHT);
      num1 = new TextField(5);
      num2 = new Text Field (5);
      OutResult = new Label ("Result" , Label.Right);
      add (number1);
      add (num1);
      add (number2);
      add (num2);
      add (dResult);
      add (Out Result);
      num1. add Action Listner (this);
      num2 : add Action Listener (this);
      dResult. add Action Listner (this);
      add WindowListener (new WindowAdapter ()
      { public void window (losing ( WindowEvent We)
            { System. exit(0); }
      });
}
```

```
public void action performed (ActionEvent ae)
{
    double n1, n2;
    try {
        if (ae.getSource() == dResult) {
            n1 = Double.parseDouble (num1.getText());
            n2 = Double.parseDouble (num2.getText());
            out = n1 + " " + n2;
            resultNum = n1/n2;
            out += String.valueOf(resultNum);
            repaint();
        }}
        Catch (ArithmeticException e2)
        {
            flag = 1;
            out = "Divide By 0 exception!" + e2;
            repaint();
        }
        Catch (NumberFormatException e)
        {
            flag = 1;
            out = "Number Format Exception!" + e;
            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;
    }
}
```

```
psvm (String args[]){
    Division main dm = new Division main();
    dm. setSize (new Dimension (800,400));
    dm. setTitle ("Division of Integers");
    dm. setVisible (true);
}
}
```

O/P

Number 1 : [12] Number 2 : [0]

[Result] Result: 12 0 0 infinity.

Number 1 : [12] Number 2 : [a]

[Result] Result:

Number Format Exception!

'java. lang. Number Format Exception. for input string 'a'.

Number 1 : [0] Number 2 : [12]

[Result] Result: 0 0 12.000

Number 1 : [1] Number 2 : [2]

[Result] Result: 1 2 0.5000