

**VISVESVARAYA TECHNOLOGICAL
UNIVERSITY** Belgaum -590014, Karnataka.



**LAB REPORT
on**

Object Oriented Java Programming

(23CS3PCOOJ) *Submitted by*

Mrunalini S M (1BM22CS228)

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
Dec-2024 to Mar-2024

12/12/23

Object oriented Java
1st Program

```
class hello {
    public static void main (String args[])
    {
        System.out.println ("helloworld");
    }
}
```

Output: helloworld.

Program 2

Two nos. are given) output. two. ans2

```
import java.util.Scanner;
```

```
(class Quadratic) {
    int a, b, c;
    double r1, r2, d;
```

{

```
int a, b, c;
```

```
double r1, r2, d;
```

```
void (Void) getd()
    {
        a = s.nextInt();
        b = s.nextInt();
        c = s.nextInt();
    }
```

```
getd();
    d = (b * b) - (4 * a * c);
    System.out.println (d);
```

System.out.println ("Enter the

Coefficients of a, b, c");

```
a = s.nextInt();
```

```
b = s.nextInt();
```

```
c = s.nextInt();
```

{

```
(a * a) / d;
```

```
Void computeC() {
    System.out.println ("Enter the value of a, b, c");
```

{

```
if (a == 0) {
    System.out.println ("a must not be zero");
}
```

{

```
if (a != 0) {
    System.out.println ("a must not be zero");
}
```

```
System.out.println ("Not a quadratic equation");
```

```
System.out.println ("Enter all 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 ("Root1=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");
```~~

```
}
```

```
else if (d < 0)
```

```
{
```

```
System.out.println ("Roots are imaginary");
```

```
r1 = (-b) / (2*a);
```

```
r2 = Math.sqrt (-d) / (2*a);
```

```
System.out.println ("Roots are  
real and distinct");
```

```
System.out.println ("Root1= " + 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);
```

```
}
```

```
else if(d==0)
```

```
{
```

```
    System.out.println("Roots are imaginary");
```

~~System.out.println("r1 = " + r1);~~~~r2 = Math.Sqrt(-d) / (2*a);~~~~System.out.println("Root1 = " + r1 + " + i" + r2);~~~~System.out.println("Root1 = " + r1 + " - i" + r2);~~

```
}
```

```
}
```

```
class Quadratic Main
```

```
{
```

```
    public static void main(String args[])
```

```
{
```

```
        Quadratic q = new Quadratic();
```

```
        q.getd();
```

```
        q.compute();
```

```
}
```

```
}
```

Output:

For Mounal 113M22CS228
Roots are real & distinct

Enter the coefficients of a, b, c

($x^2 + 18x^1 + 108x^0$) distinct, two roots

Root 1 = 4.561552180

Root 2 = 4.56155281296

Roots are real and distinct

Root 1 = 4.561552180

Root 2 = 4.56155281296

Output 2

For Mounal 113M22CS228

Enter the coefficients of a, b, c

($x^2 + 10x^1 + 25x^0$) distinct, two roots

($x^2 + 10x^1 + 25x^0$) distinct, two roots

Roots are real & equal

Root 1 = Root 2 = 1.0

J.D
10/26/15

Output 3
Printed by Dinesh Kulkarni

10/23/2015 with 2015/10/23

10/23/2015

10/23/2015

> Write a Program In Java to find the area of a rectangle and Verify the same with Various inputs (length, breadth)

```
class RectangleArea {
    public static void main (String args[]) {
        int length, breadth;
        length = Integer.parseInt(args[0]);
        breadth = Integer.parseInt(args[1]);
        int area = length * breadth;
        System.out.println ("length of rectangle = " + length);
        System.out.println ("breadth of rectangle = " + breadth);
        System.out.println ("area of rectangle = " + area);
    }
}
```

Output :

length of rectangle = 10
breadth of rectangle = 8
area of rectangle = 80;

> Write a Program to find factorial of given number:

```
class Factorial {
    public static void main (String args[]) {
        int fac = 1;
        System.out.println ("Enter a number:");
        Scanner sc = new Scanner (System.in);
        int n = sc.nextInt();
        for (int i=1; i<=n; i++) {
            fac = fac * i;
        }
        System.out.println ("The factorial is " + fac);
    }
}
```

{ write code for a program which takes a 5 digit integer as input and checks whether it is a palindrome or not.

Output: System.out.

Scanner sc = new Scanner(System.in);

Palindrome: What to find - the given 5 digit integer is it a palindrome or not.

Closest Palindrome: { found & output to screen }

Public static void main (String args[])

{ Scanner sc = new Scanner (System.in); }

int n, t, rem, rev = 0;

Scanner sc = new Scanner (System.in);

System.out.println ("Enter a 5 digit number");

n = sc.nextInt();

t = n; { explanation for original }

while (t > 0) { { explanation for original }

rem = t % 10; { explanation for saved }

rev = rev * 10 + rem;

t = t / 10; { explanation for original }

} { if (rev == n) { { explanation for original }

System.out.println ("Palindrome");

} { explanation for original } else { { explanation for original }

else { { explanation for original }}

System.out.println ("not palindrome");

} { explanation for original }

} { explanation for original }

```
else
{
    for (i=2 ; i<=m; i++)
    {
        if (n % i == 0)
            break;
    }
    if (flag == 0)
    {
        System.out.println("n is not a prime number.");
    }
}
```

```
public static void main (String args[])
{
    int n;
}
```

```
Scanner sc = new Scanner (System.in);
System.out.println ("Enter the value of i:");
i = sc.nextInt();
if (isPrime(i))
    System.out.println ("i is prime");
else
    System.out.println ("i is not prime");
}
```

```
int i;
Scanner sc = new Scanner (System.in);
System.out.println ("Enter the value of i:");
i = sc.nextInt();
if (isPrime(i))
    System.out.println ("i is prime");
else
    System.out.println ("i is not prime");
}
```

Conversion:

C loss Conversion {format specifier}

Public Static Void main (String args [])

{

byte b = 1; short s1 = 1000, s2;

int i1 = 100000, i2;

long l1 = 10000000, l2;

Char c = 't'

float f1 = 25.69f, f2;

double d1 = 536987.125, d2;

System.out.println(b + " " + s1 + " " + i1 + "
" + c + " " + f1 + " " + d1);

s2 = b;

i2 = s1;

l2 = i1;

System.out.println(s2 + " " + i2 + " " + l2);

}

}

Output:

1 1000 100000 10000000 +25.695369871

1 1000 100000 10000000 +25.695369871

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 CGPA of student.

```
import java.util.Scanner;
```

```
class Student {
```

```
    private String USN;
```

```
    private String name;
```

```
    private int[] Credits;
```

```
    private int[] marks;
```

```
    public Student(String USN, String name,
```

```
        int[] Credits, int[] marks) {
```

```
        this.USN = USN;
```

```
        this.name = name;
```

```
        this.Credits = Credits;
```

```
        this.marks = marks;
```

```
}
```

```
public void acceptDetails() {
```

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

```
    System.out.println("Enter USN:");
```

```
    USN = Sc.nextLine();
```

```
    System.out.println("Enter Name:");
```

```
    name = Sc.nextLine();
```

```
    System.out.println("Enter Number of Subjects:");
```

```
    System.out.print("Enter Number of Subjects:");
```

```
    numSubjects = Sc.nextInt();
```

```
    Credits = new int[numSubjects];
```

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

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

```
System.out.println("Enter credit for Subject");
credits[i] = sc.nextInt();
System.out.println("Enter marks for Subject " + (i+1));
marks[i] = sc.nextInt();
}
```

```
Public void displayDetails() {
    System.out.println("USN : " + USN);
    System.out.println("Name : " + name);
    for (int i = 0; i < credits.length; i++) {
        System.out.println("Subject " + (i+1) + " : ");
        credits[i] + " , mark : " + marks[i];
    }
}
```

```
Public double calculateSGPA() {
    double totalCreditPoints = 0;
    double totalCredits = 0;
    for (int i = 0; i < credits.length; i++) {
        totalCredits += credits[i];
        Get Grade Points (marks[i]);
        totalCreditPoints += credits[i] * gradePoints;
    }
    return totalCreditPoints / totalCredits;
}
```

```

private double getGradePoints (int marks) {
    if (marks >= 90) {
        return 10.0;
    } else if (marks >= 80) {
        return 9.0;
    } else if (marks >= 70) {
        return 8.0;
    } else if (marks >= 60) {
        return 7.0;
    } else if (marks >= 50) {
        return 6.0;
    } else if (marks >= 40) {
        return 5.0;
    } else {
        return 0.0;
    }
}

```

```

public static void main (String [] args) {
    int [] credits = {3, 4, 2, 3};
    int [] marks = {85, 78, 92, 88};
    Student student = new Student
        ("BM22CS228", "John", credits, marks);
    student.acceptDetails ();
    student.displayDetails ();
    System.out.println ("SGPA: " + student.
        calculateSGPA ());
}

```

output:

Enter Name: Mrunalini

Enter USN: IBM02CS028

Enter number of Subjects: 3

Enter credit for Subject S1:

3 marks for Subject S1:

Enter credit for Subject S2:

90 marks for Subject S2:

Enter mark for Subject S2:

90 marks for Subject S2:

Enter credit for Subject S3:

4 marks for Subject S3:

85 marks for Subject S3:

USN: IBM02CS028

Name: Mrunalini

Subject credits: 3, Marks: 90

Subject credits: 6, Marks: 90

Subject credits: 4 Marks: 85

SGPA: 9.636363

~~Mrunalini~~

Lab Program - 3

→ Create a class book which contains four members : name, author, Price, numPages. Include a constructor to set the values for the members. include methods to set and get the details of the object. that could display the complete details of book. Develop a Java program to create n book objects.

```
import java.util.Scanner;  
class Books {  
    String name;  
    String author;  
    int Price;  
    int numPages;  
    Books (String name, String author, int price,  
           int numPages)  
    {  
        this.name = name;  
        this.author = author;  
        this.Price = Price;  
        this.numPages = numPages;  
    }
```

• Public String toString () {
 String name, author, Price, numPages;
 name = "Book name:" + this.name + "\n";
 author = "Author name:" + this.author + "\n";
 Price = "Price:" + this.Price + "\n";
}

numPages = "Number of Pages;" + this.numPages + "

return name + author + Price + numPages;

}

Author book[]

Class Main

{

 Public Static void main (String args[]) {

 Scanner s = new Scanner(System.in);

 int n;

 String name;

 String author;

 int Price;

 int numPages;

 System.out.println ("Enter the number of books");

 n = s.nextInt();

 Books b[];

 b = new Books[n];

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

 System.out.println ("Enter the name of book");

 name = s.next();

 System.out.println ("Enter the author of book");

 author = s.next();

System.out.println ("Enter the number of
Pages of book");

num Pages = S.nextInt();

b[i] = new Books (name, author, Price, numPages);

}

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

{

System.out.println (b[i]);

{

Author ? name, author

{

Pages ? numPages

{

Output: Enter the number of books

2

Enter the name of the book

Physics

Enter the author of book

Einstein

Enter the Price of book

100

Enter the Number of Pages of book

189

Enter the name of book

Chemistry

Enter the author of book

Joseph

Enter the Price of book

200

Enter the no. number of Pages of book

390

Book name : Physics

Author name : Einstein

Price : 100

Number of Pages : 189

Book name : Chemistry

Author name : Joseph

Price : 390 & 00

Number of Pages : 390

Additional Program

import java.util.Scanner;

Class Fast

```
{    public void main (String args) {  
    int a, b;
```

```
    Test (int i, int j) {  
        a = i; b = j;
```

```
    boolean equals (Test t) {  
        if (a == t.a && b == t.b) return true;
```

```
        else return false;  
    }
```

```
}
```

Close Page 08

{ Public static void main (String args[]) {

Test ob1 = new Test (100, 22);

Test ob2 = new Test (100, 22);

Test ob3 = new Test (-1, -1);

System.out.println("ob1 == ob2 " + ob1.

equals (ob2));

System.out.println("ob1 == ob3 " +

ob1.equals (ob3));

}

Output:

ob1 == ob2 : false

ob1 == ob3 : false

~~ob1 == ob1 : true~~

Java Example 10: String Comparison Method

The standard equals method works

Object's equals method works

Test class's equals method works

String class's equals method works

```
import java.util.Scanner;
```

```
class InputScanner {
```

```
    public static Scanner scanner =
```

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

```
}
```

abstract class Shape extends InputScanner {

```
    protected int side1;
```

```
    protected int side2;
```

```
    public Shape (int side1, int side2) {
```

```
        this.side1 = side1;
```

```
        this.side2 = side2;
```

```
    public Shape (int side1, int side2) {
```

```
        this.side1 = side1;
```

```
        this.side2 = side2;
```

```
}
```

```
public abstract void printArea();
```

```
}
```

```
class Rectangle extends Shape {
```

```
    public Rectangle (int length,  
                     int width) {
```

```
        super (length, width);
```

```
}
```

@Override

public void

Lab Prg - 54

import java.util.Scanner;

class InputScanner

{

 protected Scanner Scanner;

 public InputScanner()

{

 Scanner s = new Scanner(System.in);

}

{

abstract class Shape extends InputScanner

{

 double a, b, r;

 Scanner s = new Scanner (System.in);

 System.out.println ("Enter the value of
 a : " + a);

 a = s.nextDouble();

 System.out.println ("Enter the value of
 b : " + b);

 b = s.nextDouble();

 PrintArea();

{

class Rectangle extends Shape

{

2

Rectangle()

{
double width, height, area;
Subroutine();
}

}

Print Area()

2

$$\text{double area} = \text{a} * \text{b};$$

System.out.println("Area of the
rectangle is " + area);

}

{
and then I made a mistake and I wrote
class Triangle extends shape;

2.

Triangle()

{
double width, height, area;
Subroutine();
}

{
double area = 0.5 * width * height;
}

PrintArea()

{
System.out.println("Area of the triangle is " + area);
}

$$\text{double area} = 3.14 * r * r;$$

System.out.println("Area of the triangle is " + area);

}

{
and then I made a mistake and I wrote

```
class Circle extends Shape  
{  
    circle (double a)  
    {  
        area(a);  
    }  
    void area()  
    {  
        System.out.println ("area of circle" +  
            (3.14 * (radius) * (radius)));  
    }  
    triangle (double a, double b)  
    {  
        super;  
    }  
}
```

```
class AbstractAreaMain  
{  
    public static void main (String args[])  
    {  
        Scanner s = new Scanner (System.in);  
        System.out.print ("enter the length and  
            breadth of rectangle");  
        double l = s.nextInt();  
        double b = s.nextInt();  
        System.out.println ("enter the base and  
            height");  
        double h1 = s.nextInt();  
        double h2 = s.nextInt();  
    }  
}
```

System.out.println ("Enter the circle").

double r = s.nextInt();

Shape sh;

Rectangle rect = new Rectangle (l, b);

Triangle tri = new Circle (r);

sh = rect;

- sh = area();

sh = tri;

sh = area();

sh = cir;

sh = area();

}

{

Output:

Enter the length and breadth of rectangle

9

5

Enter the base and height

3

4

Enter the circle

9

area of rectangle : 45.0

area of triangle : 6.0

area of circle : ~~254.34~~
~~3.141592653589793~~
~~0.2~~

{ more prob. }

{ same as }

{ right prob. }

{ rotated shapes }

(choose left, mixed prob.)

(rotated shapes, right prob.)

?

{ more or less, diff. }

{ same or same, diff. }

{ right or right, diff. }

{ rotated or rotated, diff. }

?

(invent of prob) difficult level

?

{ fractions & decimal }

?

(invent of prob) hard level

?

Easy (normal + rotated) 10

Lab - 5

```
import java.util.Scanner;  
class account {  
    String name;  
    int accno;  
    String type;  
    double balance;  
    account (String name, int accno,  
             String type, double balance)  
{
```

```
    this.name = name;
```

```
    this.accno = accno;
```

```
    this.type = type;
```

```
    this.balance = balance;
```

```
}
```

```
void deposit (double amount)
```

~~```
 balance += amount;
```~~~~```
void withdrawal (double amount)
```~~~~```
if ((balance - amount) >= 0)
```~~~~```
    balance -= amount;
```~~

else

{

System.out.println("Insufficient balance,
can't withdraw");

} // account class

}

class Bank {

void display() {

{

System.out.println("Name: " + name)
+ " Accno: " + accno + " Type: " + type
+ " Balance: " + balance);

{

class SavAcct extends Account {

{

private static double rate = 5;

SavAcct(String name, int accno, double balance)

{ super(name, accno, "Savings", balance); }

} // Savings class

(Customer

void interest()

{

balance += balance * (rate) / 100;

System.out.println("Balance: " + balance);

}

Class CurrAcct Extends account

Private double minBal = 500;

Private double ServiceChrgs = 50;

CurrAcct (String name, int accno,
double balance)

{

 Parameters String name, int accno, "Current",
 double balance); // constructor

} // class definition (continued)

- void checkIn()

{

 if (balance < minBal)

 System.out.println ("Balance is less than min-");

 {

 System.out.println ("Balance is less than min-");

 balance = balance + ServiceChrgs;

 balance = balance - ServiceChrgs;

 System.out.println ("balance is : " +

 balance);

}

} // class definition (continued)

} // class definition (continued)

Creates account Main
Method() shows a bank account

{
 public static void main (String acj)

L. int name = s.next();
Scanner s = new Scanner (System.in);

System.out.println ("Enter the name:");
String name = s.next();

System.out.println ("Enter the type");
String type = s.next();

System.out.println ("Enter the account number:");
(current (savings) :);

String type = s.next();
System.out.println ("Enter the account number:");

int accno = s.nextInt();

System.out.println ("Enter initial balance");
double balance = s.nextDouble();

int ch;

double amount1, amount2;

account ac = new account (name, accno,
type, balance);

SavAcct sa = new SavAcct (name,
accno, balance);

CurrAcct ca = new CurrAcct (name, accno,
balance);

while (true) {

{ is (acc. type - equals ("savings"))

System.out.println ("In menu 1 n1.

System.out.println ("1. withdraw 2. Computed interest
deposit 3. withdraw 4. display");

Scanner s = new Scanner (System.in);

System.out.println ("Enter your choice?");

ch = s.nextInt ();

switch (ch) {

case 1: Sa.withdraw (); System.out.println ("After withdrawal");

{ if ("yes".equals (s.nextLine ()) System.out.println ("Enter the");

Case 1: System.out.println ("Amount: ");

amount1 = s.nextInt ();

Sa.deposit (amount1);

break; case 2: Sa.withdraw ();

Case 2: System.out.println ("Enter the
amount: ");

amount2 = s.nextInt ();

Sa.withdraw (amount2);

break; case 3: Sa.interest ();

break;

case 4: Sa.display ();

break;

Case 5 : System.out.println("exit");

default : System.out.println("invalid input");

break;

}

 } // switch case 4 through

else

{ default

}

 System.out.println("\n menu\n1.

 deposit 2. withdraw 3. display");

System.out.println("Enter choice");

ch = S.nextInt();

switch(ch)

{

 Case 1 : System.out.println("enter amount");

 amount1=S.nextInt();

 ca.deposit(amount1);

 break;

Case 2 : System.out.println("enter amount");

 amount2=S.nextInt();

 ca.withdraw(amount2);

 ca.checkmin();

 break;

case 3: ~~char. display (0);~~
break; ~~and. display (0);~~

case 4: System.Exit(0);

default : System.out.println
("invalid input");

break;

start menu {
3 addition, two numbers}

{
3 multiplication or division

{
3 subtraction and remainder}

{
3 factorial, 0 <= n <= 1000

output.

Enter the name:
maurodini

Enter the type (current / savings)

current

Enter initial balance
4500

- menu
1) deposit 2 withdrawal
3) display

Enter the choice

Enter amount
23000

menu

1. deposit

2. withdraw

3. display

3

name: neenu

acc no: 1234

type: current

balance: 27500

reqd 30000 to withdraw
in 10000 sum of amount

1) Demonstrate

various

constructors In String

Str1 : Hello

Str2 : world

Str3 : or '

Str4 : HELLO

Str5 : Java

2) Str = "Hello. World"

String length = 12

3) demonstrate

toString()

Output:

101. mrunalini Bangalore

102. mrunal Mylore.

4) Using getchars(), Extract Bmsce from
"welcome to Bmsce college"Output : Bmsce5) Demonstrate getbytes(), tochar Array C)
with Proper Java ProgramsOutput : (i) getbytes()

Hello , world!

(ii) tochar Array()

Java programming

6) Check the following output and write
the java programs using Stringfunction

i. Bmsce equals Bmsce → true

ii. Bmsce equals College → false

Bmsec Equals BMSCE → false

Bmsec Equals IgnoreCase BMSCE → true

Output :- Public class StringComparison &

Public static void main (String [] args) {

String Str1 = "Bmsec";

String Str2 = "College";

String Str3 = "BMSCE";

System.out.println ("Str1 + " + Str2 + " equals " + Str3);

System.out.println ("Str1.equals (Str2));");

System.out.println ("Str1.equals (Str3));");

System.out.println ("Str1.equals (Str2 + " + Str3 + "));");

System.out.println ("Str1.equals (Str3 + " + Str2 + "));");

System.out.println ("Str1.equals (Str4 + " + Str3 + "));");

System.out.println ("Str1.equals (Str4 + " + Str2 + "));");

if matches

{ new StringTokenizer (str1, " ") . nextToken () . equals ("Bmsec") }

{ new StringTokenizer (str1, " ") . nextToken () . equals ("College") }

7) Using ~~regionMatches ()~~ final Substring "Bmsec"

College" from string "welcome to Bmsec College of Engineering",
it matches whole Substring it is matched otherwise displayed

Output : Substring is matched

6) ~~String~~ ~~String~~ ~~String~~

if a.equals (b + c) a.equals (b) & a.equals (c)

8) Demonstrate `StartWith()` to give output true & false.

Sol:-

```
import java.util.*;
public class StartWith {
    public static void main() {
        String str = "Welcome to Java";
        System.out.println("Check whether String Starts with Welcome at Pos 11: " + str.startsWith("Welcome"));
        System.out.println("Check whether String Starts with 'o' at pos 11: " + str.startsWith("o"));
        System.out.println("Check whether String Starts with 'Java' at pos 11: " + str.startsWith("Java"));
    }
}
```

Output: Check whether String Starts with Welcome at Pos 11: true
Check whether String Starts with 'o' at pos 11: false
Check whether String Starts with 'Java' at pos 11: false

9) Demonstrate `EndsWith()` to give output true and false.

Output: True
False

10) Demonstrate a java Program to show output for `equals()` values ==

Output using == with String literals:

```
Str1 == Str2 : true
```

```
Str1 != Str3 : false
```

Calling `equals()` with String literals:

```
Str1.equals(Str2) : true
```

```
Str1.equals(Str3) : false
```

`using == with new String Objects:`

`Str4 == Str5 : false`

`using equals() with new String Object:`

`Str4.equals(Str5) : true`

~~String str1 = "Hello world";
String str2 = "Hello world";
System.out.println(str1 == str2);
System.out.println(str1.equals(str2));~~

11) apple, ball, cat, dog, ant, tree, gun,
Boat, ice, jug, lift man, Orange, Parrot,
Green ring, Star, tree, umbrella

Van → watch, Ambulance [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

12) Sorted numbers
it was a test. It was 100

13) it was a test.

Hello world

Commerce

15) Hello friends

16) Hello friends

17) Enter the no of students : 3

Enter the name: Rani munalini

Enter the regtner no : 567228

Enter the name : Anurica kushi

Enter the cgpa : -9.8

Sorted by cgpa : 230

name: Anurica kushi regno: 228

cgpa: 8.1

name: kushi regno: 230

cgpa : 9.8

18) Set length (Hello how are you?)

User at index 4 : c

After set char at : Hello = l o

After set char : Hello

Get char : Hello

After append : Hello How are you?

After append : Hello How awesome

After insert : Hello How are you?

After delete : Hello How awesome

After delete : Hello How are you?

16-01-14

EE-Workshop

for out, out

return (E)

Week - 6

```

package CIE;
import java.util.Scanner;
public class Student {
    protected String USN = new String();
    protected String name = new String();
    protected int Sem;
    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();
    }
    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]
```

```
Public void input CIEmarks () {
```

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

```
for (int i = 0; i < 5; i++) {
```

```
System.out.print ("Enter CIE marks for
```

```
Subject " + (i+1) + ":" );
```

```
marks [i] = s.nextInt();
```

```
Package SEE;
```

```
import CIE. Internals;
```

```
import java.util.Scanner;
```

```
Class Externals extends Internals {
```

```
Protected int final_marks [ ] = new int [5];
```

```
Protected int super_marks;
```

```
Public void input SEE marks () {
```

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

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

```
for (int i = 0; i < 5; i++) {
```

```
System.out.print ("Enter SEE marks for
```

```
Subject " + (i+1) + ":" );
```

```
super_marks [i] = s.nextInt();
```

```
}
```

```
Public void calculate Final Marks () {
```

```
for (int i = 0; i < 5; i++) {
```

```
final Marks [i] = super_marks [i] / 2 +
```

```
super_marks [i];
```

```
}
```

```
public void displayFinalMarks() {
    displayStudentDetails();
    for (int i = 0; i < 5; i++) {
        System.out.println("Subject " + (i+1) + ",",
                           + finalMarks[i]);
    }
}
```

import SEE.Externals;

class Main

```
public static void main (String args[]) {
    int numofStudent = 2;
    Externals finalMarks [ ] = new Externals [numofStudent];
    for (int i = 0; i < numofStudent; i++) {
        finalMarks[i] = new Externals();
    }
}
```

finalMarks[i].inputStudentDetails();

finalMarks[i].inputCSEmarks();

System.out.println("Enter CSE marks");

finalMarks[i].inputCSEmarks();

System.out.println("Enter SEE marks");

finalMarks[i].inputSEEmarks();

}

System.out.println("Displaying data:\n");

```
for (int i = 0; i < numofStudent; i++) {
    finalMarks[i].calculateFinalMarks();
}
```

finalMarks[i].displayFinalMarks();

}

}

SEE:

Sub 1 marks : 30
Guru 2 marks : 70
Guru 3 marks : 90
Sub 4 marks : 80
Guru 5 marks : 30

C.F.

Sub 1 marks = 70
Guru 2 marks = 8
Guru 3 marks = 1
Sub 4 marks = 1
Sub 5 marks = 1

S.F.

Sub 1 marks = 80
Sub 2 marks = 60
Guru 3 marks = 70

Sub 4 marks = 80
Sub 5 marks = 1

Jitendra

(got 2 marks)

SEE i

Sub 1 marks : 30

Sub 2 marks : 70

Sub 3 marks : 90

Sub 4 marks : 80

Sub 5 marks : 30

C 7 1/2

Sub 1 marks = 70

Sub 2 marks = 80

Sub 3 marks = 90

Sub 4 marks = 80

Sub 5 marks = 20

STEF

Sub 1 marks : 80

Sub 2 marks : 60

Sub 3 marks : 70

Sub 4 marks : 80

Sub 5 marks : 30

✓ 126-01 M

✓ cannot change will add

✓ softball not started

(find in a separate file) add to add

Write a program that demonstrate handling of exceptions in inheritance tree. Create a base class called "father" and derived class called "Son", which extends the base class, implements a constructor which takes the age and throws the exception WrongAge > when the input age < 0. In Son class implement a constructor that calls both Father & Son's age & throws an exception if Son's age is \geq father's age.

```
import java.util.Scanner;
```

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

```
class Father {
    protected int fatherAge;
}
```

```
public Father (int age) {
    fatherAge = age;
    if (fatherAge < 0)
        throw new WrongAge ("Fathers age cannot be negative");
}
```

```
class Son extends Father {
    private int sonAge;
}
```

```
public Son (int fatherAge, int sonAge) {
    throws WrongAge {
        super (fatherAge);
    }
}
```

this . SonAge = SonAge;

if (SonAge <= SonAge)

if (Son Age <= 0) {

throw new WrongAge ("Son's age cannot
be zero or negative");

}

if (SonAge >= FatherAge) {

throw new WrongAge ("Son's age
cannot be greater than or
equal to father's age");

}

}

Main25 {

public class Main25 {

 static void main (String [] args) {

 public class Main25 {

 Scanner Scanner = new Scanner (System.in);

 try {

 System.out.print ("Enter father's age: ");

 int FatherAge = Scanner.nextInt ();

 System.out.print ("Enter son's age: ");

 int SonAge = Scanner.nextInt ();

 int SonAge = Son (FatherAge, SonAge);

 Son Son = new Son (FatherAge + SonAge);

 System.out.println ("Son's age: " + SonAge);

 System.out.println ("Son's age: " + SonAge);

 } catch (WrongAge e) {

 System.out.println ("Exception caught: " + e);

 System.out.println ("Exception caught: " + e.getMessage());

 System.out.println ("Exception caught: " + e.getLocalizedMessage());

```
Catch (Exception e) {
```

```
    System.out.println ("Error: " + e);
```

```
    System.out.println ("Error: " + e.getMessage());
```

```
}
```

```
finally {
```

```
    Scanner.close();
```

```
} // finally block
```

```
} // try block
```

```
}
```

```
}
```

Output: Enter father's age: 50

Enter son's age: -10

Exception Caught: Wrong Age: Son's age Cannot be
negative or zero

Enter father's age: 30

Enter son's age: 40

Exception Caught: Wrong Age: Son's age Cannot
be greater than or equal to
father's age.

30.0

30.0

30.0

30.0

30.0

30.0

Lab Program-8

```

class BMS extends Thread
{
    public void run()
    {
        for (int i=1; i<=5; i++)
        {
            System.out.println("BMS college of
                Engineering " + i);
            Thread.sleep(1000);
        }
    }
}

```

Catch (InterruptedException e)

```

    e.printStackTrace();
}
}

```

```

class CS extends Thread
{
    public void run()
    {
        for (int i=1; i<=5; i++)
        {
            System.out.println("CSE " + i);
            Thread.sleep(2000);
        }
    }
}

```

Catch (InterruptedException e)

```

    {
        eprintStackTrace();
    }

}

class threadMain
{
    public static void main (String args[])
    {
        BMS b1 = new BMS();
        CS c1 = new CS();

        b1.start();
        c1.start();
    }
}

```

Output: BMS College of Engineering 1

CSE 1

CSE 2

CSE 3

CSE 4

CSE 5

BMS College of Engineering 2

BMS College of Engineering 3

BMS College of Engineering 4

BMS College of Engineering 5

~~Output: BMS College of Engineering 1~~

~~BMS College of Engineering 2~~

~~BMS College of Engineering 3~~

~~BMS College of Engineering 4~~

~~BMS College of Engineering 5~~

lab program - q

```
class A {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while (!valueSet)
            ; // do nothing
        try {
            System.out.println("In Consumer waiting(n)");
            wait();
        } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
        }
        System.out.println("Got:" + n);
        valueSet = false;
        System.out.println("Producer In");
        notify();
        return n;
    }
    void put(int n) {
        synchronized {
            while (valueSet)
                ; // do nothing
            try {
                System.out.println("Producer waiting(n)");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
            System.out.println("Put:" + n);
            valueSet = true;
        }
    }
}
```

this, n = n;
valuen = true;

System.out.println("Put: " + n);
System.out.println("In Intimate Consumer");
notify();

} class Producer implements Runnable {

with Producer implements Runnable {
a q; } (a method declaration) static
Producer (a q) { int i = 0; }
int q = q;
new Thread (this, "Producer").start();

;} (a method declaration) static
public void run () {
int i = 0;
while (i < 15) {
q.put (i++); } } (a method declaration)

} (a method declaration) static
class Consumer implements Runnable {

a q; } (a method declaration) static
Consumer (a q) { int i = 0; }
this.q = q;
new Thread (this, "Consumer").start();

}

```
public void suncl()
```

```
    int i = 0;  
    while (i < 15) {  
        int r = q.get();  
        System.out.println("consumed: " + r);  
        i++;  
    }
```

```
3  
3  
public class PCFixed {  
    public static void main (String args[]) {  
        public static  
        Or q = new A();  
        new Producer (q);  
        new Consumer (q);  
        System.out.println ("Press Control-C to  
                           STOP.");  
    }  
}
```

Output

| | |
|-------|--------|
| put 9 | |
| put 1 | got 9 |
| got 1 | put 10 |
| put 2 | got 10 |
| got 2 | put 11 |
| put 3 | got 11 |
| got 3 | put 12 |
| put 4 | got 12 |
| got 4 | put 13 |
| put 5 | got 13 |
| got 5 | put 14 |
| put 6 | got 14 |
| got 6 | |
| put 7 | |
| got 7 | |
| put 8 | |
| got 8 | |

ANSWER

Lab Program - 10

13/10/2024

class A {

Synchronized void foo(B b) {
String name = Thread.currentThread().
getName();
System.out.println(name + " entered A.foo");

try {

Thread.sleep(1000);

} catch (Exception e) {

System.out.println("A interrupted");

}

System.out.println(name + " trying to call
B.wait()");

b.wait();

{ b.notify(); }
(stop)

void bar() {

System.out.println("inside A.bar");

} {
}

Class B {

Synchronized void bar(A a) {

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

getname();
a.b

System.out.println(name + " entered B.
bar");

try {

Thread.sleep(1000);

} catch (Exception e) {

```
System.out.println("B interrupted");
```

}

```
System.out.println(name + " trying to  
call A.wait()");  
a.wait();
```

void wait() {

```
System.out.println("inside A.wait");
```

}

class Deadlock implements Runnable {

```
A a = new A();
```

```
B b = new B();
```

Deadlock() {

```
Thread t1 = new Thread(this, "main thread");
```

```
Thread t2 = new Thread(this, "Raunq Thread");
```

```
t1.start();
```

```
a.foo(b);
```

```
System.out.println("Back in main  
thread");
```

public void run() {

```
b.bar(a);
```

```
System.out.println("Back in other  
thread");
```

public void run() {

```
b.bar(a);
```

```
System.out.println("Back in other  
thread");
```

Public static void main (String args[])

{

System.out.println ("new Deadlock (2);

3. Thread A

2. Thread B

Output: main Thread entered A - box

Racing 1 Thread entered B, box
main Thread trying to call B. last()

Inside A - last

Back in main Thread (was)

Racing Thread trying to call A.last()

Inside A.last() - 1

Back in other thread

Procon.java (Multi-threaded program)

class SynchronizationExample

int n; { }

boolean need; Set = false;

Synchronized int get () { }

while (!value Set) { }

for { }

System.out.println ("new thread

waiting, ");

wait();

} catch (InterruptedException) { }

System.out.println (e); }

```
System.out.println("got " + n);
```

value set = false;

```
System.out.println(" tell producer");
```

```
notify();
```

```
} return "
```

```
? }
```

```
Synchronized void put (int n) {
```

```
(value set) {
```

```
while
```

```
try {
```

```
System.out.println("a producer waiting")
```

```
System.out.println
```

```
wait();
```

```
catch (InterruptedException e) {
```

```
} catch
```

```
System.out.println(e);
```

```
int i = 0;
```

```
this.n = n;
```

```
value set = true};
```

```
System.out.println("PA: " + n);
```

```
System.out.println("The consumer");
```

```
notify();
```

```
Producer
```

```
implements Runnable
```

```
Or,
```

```
Producers (or q) {
```

```
this.q = q;
```

```
new Thread (this. "Producer").start();
```

```
new Thread (this. "Producer").start();
```

```
}
```

```
public void C1 {
```

```
int i = 0;
```

```
while (i < 3) {
```

Q1. Put `(i++)`; in `main()`
} `if (empty) { do something }`
} `return value of variable, i.e., n.`
} `return value of variable, i.e., n.`
class consumer implements Runnable {
 int v;
}

Consumer `(C. consumer)` & `Runnable`
this.v = v; `else`

new Thread (this, "Consumer").start();

public void run() {
 int i = 0; `intial`

while (i < 3) {
 i++; `intial`

int n = g.get();

System.out.println("Consumed: " + i);

i++; `intial`

if (i == 3) `intial`

} `intial`

if (i == 3) `intial`

Output - 1

Put: 1

Got: 1

Put: 2

Got: 2

Put: 3

Got: 3

19/02/2024

Week = 10

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
  
class SwingDemo extends JFrame {  
    public static void main(String args[]) {  
        SwingDemo s = new SwingDemo();  
    }  
  
    SwingDemo() {  
        JFrame jfrm = new JFrame("Divide a/b");  
        jfrm.setSize(275, 150);  
        jfrm.setLayout(new FlowLayout());  
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
        JLabel jlab = new JLabel("Enter the divisor  
and dividend :");  
        JTextField aJtf = new JTextField(8);  
        JTextField bJtf = new JTextField(8);  
        JButton buttonA = new JButton("calculate");  
  
        jfrm.add(jlab);  
        jfrm.add(aJtf);  
        jfrm.add(bJtf);  
        jfrm.add(buttonA);  
  
        jfrm.setLayout(null);  
        jlab.setBounds(10, 10, 200, 30);  
        aJtf.setBounds(10, 40, 200, 30);  
        bJtf.setBounds(10, 70, 200, 30);  
        buttonA.setBounds(10, 100, 200, 30);  
  
        buttonA.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent e) {  
                int a = Integer.parseInt(aJtf.getText());  
                int b = Integer.parseInt(bJtf.getText());  
                if (b == 0) {  
                    JOptionPane.showMessageDialog(jfrm, "Division by zero is not allowed");  
                } else {  
                    int ans = a / b;  
                    String ansStr = String.valueOf(ans);  
                    JOptionPane.showMessageDialog(jfrm, "The result is " + ansStr);  
                }  
            }  
        });  
    }  
}
```

```
ActionListener1 = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        System.out.println("Action Event from a
text field");
    }
}
```

```
ajtf.addActionListener1();
bjtf.addActionListener1();
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {

```

```
        if (bjtf.getText().length() > 0) {
            int a = Integer.parseInt(ajtf.getText());
            int b = Integer.parseInt(bjtf.getText());
            int ans = a / b;
        }
    }
});
```

```
    else {
        aLab.setText("In A = " + a);
        bLab.setText("In B = " + b);
        ansLab.setText("In Ans = " + ans);
    }
}
```

```
} catch (NumberFormatException e) {
    aLab.setText(" " + a);
    bLab.setText(" " + b);
    ansLab.setText(" " + ans);
    err.setText("Enter only Integers!");
}
```

```
} catch (ArithmException e) {
    aLab.setText(" " + a);
    bLab.setText(" " + b);
    ansLab.setText(" " + ans);
    err.setText("B Should be Non zero!");
}
```

```
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

frame.setVisible(true);
```

```
public static void main (String args[]) {
    SwingUtilities.invokeLater (new Runnable() {
        public void run() {
            new SwingDemo ();
        }
    });
}
```

Output: Enter the divisor and dividend:

Input:
Divisor : 25 Dividend : 5
Result: 5

$$A = 25 \quad B = 5 \quad \text{Ans} = 5$$

Report

① JFrame is a class defined in javax.swing package that represents a window or a frame in a GUI application. JFrame serves as a container to hold various GUI components such as buttons, text fields, labels etc.

SetSize: This method is used to set size of frame in pixels.
(int width, int height)

SetLayout: This method is used to set the layout manager for the frame, which determines how components are arranged with frame's size.

SetDefaultCloseOperation: This method sets the default operation that will be performed when the user closes the frame, such as exiting the application or hiding the frame.

JLabel: Is a component used to display text or an image on GUI. It can be instantiated with 'JLabel(String text)'.

getText(): This method retrieves the text entered by user in text field.

Adding Component to JFrame:
'Action Listener' is an interface used to handle events triggered by user actions, such as clicking on a button.

'add action listener(ActionListener listener)'. This method adds an Actionlistener to component.

'getText()': Method is commonly used to get
content of various components in
graphical user interface (GUI) Programming,
particularly with components like JLabel,
JTextField, javax.swing.JTextField component
and javax.swing.JLabel.

See

WEEK 1

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

Program:

```
import java.util.Scanner;

class Quadratic {
    int a, b, c;
    double r1, r2, d;

    void getCoefficients() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the coefficients of a, b, and c:");
        a = scanner.nextInt();
        b = scanner.nextInt();
        c = scanner.nextInt();
    }

    void computeRoots() {
        while (a == 0) {
            System.out.println("Not a quadratic equation");
            System.out.println("Enter a non-zero value for a:");
            Scanner scanner = new Scanner(System.in);
            a = scanner.nextInt();
        }

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

        if (d == 0) {
            r1 = -b / (2.0 * a);
            System.out.println("Roots are real and equal");
            System.out.println("Root1 = Root2 = " + r1);
        } else if (d > 0) {
            r1 = (-b + Math.sqrt(d)) / (2.0 * a);
            r2 = (-b - Math.sqrt(d)) / (2.0 * a);
            System.out.println("Roots are real and distinct");
            System.out.println("Root1 = " + r1);
            System.out.println("Root2 = " + r2);
        } else {
            System.out.println("Roots are complex and conjugate");
        }
    }
}
```

```
r2 = (-b - Math.sqrt(d)) / (2.0 * a);
System.out.println("Roots are real and distinct");
System.out.println("Root1 = " + r1 + " Root2 = " + r2);
} else {
    System.out.println("Roots are imaginary");
    r1 = -b / (2.0 * a);
    r2 = Math.sqrt(-d) / (2.0 * a);
    System.out.println("Root1 = " + r1 + " + i" + r2);
    System.out.println("Root2 = " + r1 + " - i" + r2);
}
}

class QuadraticMain {
    public static void main(String[] args) {
        System.out.println("My Name is Rohit Gandhi");
        System.out.println("My USN is 2023BMS02599");
        Quadratic quadratic = new Quadratic();
        quadratic.getCoefficients();
        quadratic.computeRoots();
    }
}
```

WEEK2

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.

Program:

```
import java.util.Scanner;
import java.io.*;

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

    // Constructor
    public Student(String usn, String name, int numSubjects) {
        this.usn = usn;
        this.name = name;
        this.credits = new int[numSubjects];
        this.marks = new int[numSubjects];
    }

    // Method to accept details
    public void acceptDetails() {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter details for student " + name + " (USN: " + usn +
        ")");
        for (int i = 0; i < credits.length; i++) {
            System.out.print("Enter credits for subject " + (i + 1) + ": ");
            credits[i] = scanner.nextInt();

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

```
// Method to display details
public void displayDetails() {
    System.out.println("Details for student " + name + " (USN: " + usn + ")");
    for (int i = 0; i < credits.length; i++) {
        System.out.println("Subject " + (i + 1) + " - Credits: " + credits[i] + ", "
Marks: " + marks[i]);
    }
}

// Method to calculate SGPA
public double calculateSGPA() {
    double totalCredits = 0;
    double totalGradePoints = 0;

    for (int i = 0; i < credits.length; i++) {
        totalCredits += credits[i];
        totalGradePoints += calculateGradePoints(marks[i]) * credits[i];
    }

    return totalGradePoints / totalCredits;
}

// Helper method to calculate grade points based on marks
private double calculateGradePoints(int marks) {
    if (marks >= 90) {
        return 10.0;
    } else if (marks >= 80) {
        return 9.0;
    } else if (marks >= 70) {
        return 8.0;
    } else if (marks >= 60) {
        return 7.0;
    } else if (marks >= 50) {
        return 6.0;
    } else {
        return 0.0;
    }
}
```

```
public class StudentSGPA {  
    public static void main(String[] args) {  
        System.out.print("my name is Rohit Gandhi");  
        System.out.print("my USN is 2023BMS02599");  
  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Enter the number of subjects: ");  
        int numSubjects = scanner.nextInt();  
  
        System.out.print("Enter the USN: ");  
        String usn = scanner.next();  
  
        System.out.print("Enter the name: ");  
        String name = scanner.next();  
  
        Student student = new Student(usn, name, numSubjects);  
        student.acceptDetails();  
  
        // Displaying details and SGPA  
        student.displayDetails();  
        System.out.println("SGPA: " + student.calculateSGPA());  
    }  
}
```

WEEK3

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.

Program:

```
import java.util.Scanner;

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

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

    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 BookStore{

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

        System.out.print("Enter the number of books: ");
    }
}
```

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

Book[] books = new Book[n];

for (int i = 0; i < n; i++) {
    System.out.print("Enter name of the book: ");
    scanner.nextLine(); // consume the newline character
    String name = scanner.nextLine();

    System.out.print("Enter author of the book: ");
    String author = scanner.nextLine();

    System.out.print("Enter the price of the book: ");
    int price = scanner.nextInt();

    System.out.print("Enter the number of pages of the book: ");
    int numPages = scanner.nextInt();

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

System.out.println("\nBook Details:");
for (int i = 0; i < n; i++) {
    System.out.println("Book " + (i + 1) + ":" + books[i]);
}
```

WEEK4

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.

Program:

```
import java.util.Scanner;

abstract class Shape {
    protected int side1;
    protected int side2;

    public Shape(int side1, int side2) {
        this.side1 = side1;
        this.side2 = side2;
    }

    public abstract void printArea();
}

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

    @Override
    public void printArea() {
        int area = side1 * side2;
        System.out.println("Rectangle Area: " + area);
    }
}

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

```
@Override
public void printArea() {
    double area = 0.5 * side1 * side2;
    System.out.println("Triangle Area: " + area);
}
}

class Circle extends Shape {
    public Circle(int radius) {
        super(radius, 0);
    }
    @Override
    public void printArea() {
        double area = Math.PI * side1 * side1;
        System.out.println("Circle Area: " + area);
    }
}

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

        System.out.println("Enter length and width for Rectangle:");
        int rectLength = scanner.nextInt();
        int rectWidth = scanner.nextInt();
        Rectangle rectangle = new Rectangle(rectLength, rectWidth);
        rectangle.printArea();

        System.out.println("Enter base and height for Triangle:");
        int triBase = scanner.nextInt();
        int triHeight = scanner.nextInt();
        Triangle triangle = new Triangle(triBase, triHeight);
        triangle.printArea();

        System.out.println("Enter radius for Circle:");
        int circleRadius = scanner.nextInt();
        Circle circle = new Circle(circleRadius);
        circle.printArea();
        scanner.close();
    }
}
```

WEEK5

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.

Program:

```
import java.util.Scanner;

abstract class Account {
    String customerName;
    long accountNumber;
    String accountType;
    double balance;

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

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

```

// Abstract method for withdrawal
public abstract void withdraw(double amount);
}

class CurrAcct extends Account {
    double minimumBalance;
    double serviceCharge;

    public CurrAcct(String customerName, long accountNumber, double balance) {
        super(customerName, accountNumber, "Current Account", balance);
        this.minimumBalance = 1000; // Set minimum balance
        this.serviceCharge = 50;   // Set service charge
    }

    @Override
    public void withdraw(double amount) {
        if (balance - amount >= minimumBalance) {
            balance -= amount;
            System.out.println("Withdrawal successful. Remaining balance: $" +
balance);
        } else {
            System.out.println("Insufficient funds. Service charge of $" +
serviceCharge + " applied.");
            balance -= serviceCharge;
            System.out.println("Remaining balance after service charge: $" + balance);
        }
    }
}

class SavAcct extends Account {
    double interestRate;

    public SavAcct(String customerName, long accountNumber, double balance) {
        super(customerName, accountNumber, "Savings Account", balance);
        this.interestRate = 0.05; // Set interest rate (5%)
    }

    @Override
    public void withdraw(double amount) {
        if (balance - amount >= 0) {

```

```

        balance -= amount;
        System.out.println("Withdrawal successful. Remaining balance: $" +
balance);
    } else {
        System.out.println("Insufficient funds. Cannot complete withdrawal.");
    }
}

public void depositInterest() {
    double interest = balance * interestRate;
    balance += interest;
    System.out.println("Interest deposited. Updated balance: $" + balance);
}
}

public class Bank1 {
    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 number: ");
        long accountNumber = scanner.nextLong();

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

        System.out.print("Enter account type (Current/Savings): ");
        String accountType = scanner.next();

        Account account;
        if (accountType.equalsIgnoreCase("Current")) {
            account = new CurrAcct(customerName, accountNumber, initialBalance);
        } else if (accountType.equalsIgnoreCase("Savings")) {
            account = new SavAcct(customerName, accountNumber, initialBalance);
        } else {
            System.out.println("Invalid account type. Exiting program.");
            return;
        }
    }
}

```

```
int choice;
do {
    System.out.println("\n1. Deposit");
    System.out.println("2. Display Balance");
    System.out.println("3. Deposit Interest for Savings Account");
    System.out.println("4. Withdraw");
    System.out.println("5. Exit");
    System.out.print("Enter your choice: ");
    choice = scanner.nextInt();

    switch (choice) {
        case 1:
            System.out.print("Enter deposit amount: ");
            double depositAmount = scanner.nextDouble();
            account.balance += depositAmount;
            System.out.println("Deposit successful. Updated balance: $" +
account.balance);
            break;

        case 2:
            account.displayBalance();
            break;

        case 3:
            if (account instanceof SavAcct) {
                ((SavAcct) account).depositInterest();
            } else {
                System.out.println("This option is applicable for Savings Account
only.");
            }
            break;

        case 4:
            System.out.print("Enter withdrawal amount: ");
            double withdrawalAmount = scanner.nextDouble();
            account.withdraw(withdrawalAmount);
            break;

        case 5:
```

```
        System.out.println("Exiting program. Goodbye!");
        break;

    default:
        System.out.println("Invalid choice. Please enter a valid option.");
    }

} while (choice != 5);

scanner.close();
}
```

WEEK6

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

Program:

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

public class MainProgram {

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

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

        Internals[] internalsArray = new Internals[numStudents];
        External[] externalsArray = new External[numStudents];

        for (int i = 0; i < numStudents; i++) {
            System.out.println("Enter details for Student " + (i + 1));
            System.out.print("USN: ");
            String usn = scanner.next();

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

            System.out.print("Semester: ");
            int sem = scanner.nextInt();
        }
    }
}
```

```

System.out.println("Enter Internal Marks for 5 courses:");
int[] internalMarks = new int[5];
for (int j = 0; j < 5; j++) {
    System.out.print("Course " + (j + 1) + ": ");
    internalMarks[j] = scanner.nextInt();
}

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

System.out.println("Enter SEE Marks for 5 courses:");
int[] seeMarks = new int[5];
for (int j = 0; j < 5; j++) {
    System.out.print("Course " + (j + 1) + ": ");
    seeMarks[j] = scanner.nextInt();
}

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

// Calculate and display final marks
System.out.println("\nFinal Marks of Students:");
for (int i = 0; i < numStudents; i++) {
    int[] internalMarks = internalsArray[i].getInternalMarks();
    int[] seeMarks = externalsArray[i].getSeeMarks();

    int[] finalMarks = new int[5];
    int totalMarks = 0;

    System.out.println("Student " + (i + 1) + " - " +
internalsArray[i].getName());

    for (int j = 0; j < 5; j++) {
        finalMarks[j] = internalMarks[j] + seeMarks[j];
        totalMarks += finalMarks[j];
        System.out.println("Course " + (j + 1) + ": " + finalMarks[j]);
    }

    System.out.println("Total Marks: " + totalMarks + "\n");
}
}

```

```
}
```

```
// CIE/Internals.java
package CIE;

public class Internals extends Student {
    protected int[] internalMarks;

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

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

```
// CIE/Student.java
package CIE;

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

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

    // Default constructor
    public Student() {
        this("", "", 0);
    }

    // Getter for name
}
```

```
public String getName() {
    return name;
}
}

// SEE/External.java
package SEE;

import CIE.Student;

public class External extends Student {
    protected int[] seeMarks;

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

    public int[] getSeeMarks() {
        return seeMarks;
    }
}
```

WEEK7

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.

Program:

```
import java.util.Scanner;

class WrongAge extends Exception {
    public WrongAge() {
        super("Age Error");
    }

    public WrongAge(String message) {
        super(message);
    }
}

class InputScanner {
    protected Scanner scanner;

    public InputScanner() {
        scanner = new Scanner(System.in);
    }

    public int nextInt() {
        return scanner.nextInt();
    }
}

class Father extends InputScanner {
    protected int fatherAge;

    public Father() throws WrongAge {
        System.out.println("Enter father's age:");
    }
}
```

```

fatherAge = super.nextInt();
if (fatherAge < 0) {
    throw new WrongAge("Age cannot be negative");
}
}

public void display() {
    System.out.println("Father's age: " + fatherAge);
}
}

class Son extends Father {
    private int sonAge;

    public Son() throws WrongAge {
        super();
        System.out.println("Enter son's age:");
        sonAge = super.nextInt();

        if (sonAge >= fatherAge) {
            throw new WrongAge("Son's age cannot be greater than or equal to father's
age");
        } else if (sonAge < 0) {
            throw new WrongAge("Age cannot be negative");
        }
    }

    public void display() {
        super.display();
        System.out.println("Son's age: " + sonAge);
    }
}

public class ExceptionHandlingDemo {
    public static void main(String[] args) {
        try {
            Son son = new Son();
            son.display();
        } catch (WrongAge e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

```

WEEK8

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.

Program:

```
class DisplayThread extends Thread {  
    private String message;  
    private int interval;  
  
    public DisplayThread(String message, int interval) {  
        this.message = message;  
        this.interval = interval;  
    }  
  
    public void run() {  
        try {  
            while (true) {  
                System.out.println(message);  
                Thread.sleep(interval);  
            }  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
    }  
}  
  
public class week8 {  
    public static void main(String[] args) {  
        DisplayThread thread1 = new DisplayThread("BMS College of Engineering",  
10000); // 10 seconds  
        DisplayThread thread2 = new DisplayThread("CSE", 2000); // 2 seconds  
  
        thread1.start();  
        thread2.start();  
    }  
}
```

WEEK9

9. Write a program that creates a user interface to perform integer divisions.

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

Program:

```
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 divider and dividend:");

        // 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 message
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) {
        // No need to handle action event from text fields separately
        // because we handle it in the button's action listener
    }
};

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());
            if (b == 0) {
                throw new ArithmeticException();
            }
            int ans = a/b;

            alab.setText("A = " + a);
            blab.setText("B = " + b);
            anslab.setText("Ans = " + ans);
            err.setText(""); // Clear any previous error message
        }
        catch(NumberFormatException e){
            clearLabels();
            err.setText("Enter Only Integers!");
        }
        catch(ArithmeticException e){
            clearLabels();
        }
    }
}

```

```
        err.setText("B should be NON zero!");
    }
}

private void clearLabels() {
    alab.setText("");
    blab.setText("");
    anslab.setText("");
}
});

// 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();
        }
    });
}
}
```

WEEK10

10. Demonstrate Inter process Communication and deadlock

Program:

A. Deadlock

```
public class DeadlockExample {  
  
    public static void main(String[] args) {  
        final SharedResource sharedResource = new SharedResource();  
  
        Thread process1 = new Thread(() -> {  
            try {  
                sharedResource.method1();  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        });  
  
        Thread process2 = new Thread(() -> {  
            try {  
                sharedResource.method2();  
            } catch (InterruptedException e) {  
                e.printStackTrace();  
            }  
        });  
  
        process1.start();  
        process2.start();  
    }  
}  
  
class SharedResource {  
    private final Object lock1 = new Object();  
    private final Object lock2 = new Object();  
  
    public void method1() throws InterruptedException {  
        synchronized (lock1) {  
            System.out.println("Method 1 acquired lock 1");  
            synchronized (lock2) {  
                System.out.println("Method 1 acquired lock 2");  
            }  
        }  
    }  
}
```

```

System.out.println("Method 1 acquired lock1");
Thread.sleep(1000);

    synchronized (lock2) {
        System.out.println("Method 1 acquired lock2");
        // Perform some task using both lock1 and lock2
    }
}

public void method2() throws InterruptedException {
    synchronized (lock2) {
        System.out.println("Method 2 acquired lock2");
        Thread.sleep(1000);

        synchronized (lock1) {
            System.out.println("Method 2 acquired lock1");
            // Perform some task using both lock1 and lock2
        }
    }
}

```

B. InterprocessCommunication:

```

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("\nNotify 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("\nNotify 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++);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

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++;
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

public class InterprocessCommunication {
    public static void main(String args[]) {
        Q q = new Q();
        new Producer(q);
        new Consumer(q);
        System.out.println("Press Control-C to stop.");
    }
}
```

