VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Object Oriented Java Programming

(23CS3PCOOJ)

Submitted by

Jiya Nitturkar (1BM23CS128)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Object Oriented Java Programming (23CS3PCOOJ)" carried out by **Jiya Nitturkar (1BM23CS128),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Object Oriented Java Programming (23CS3PCOOJ) work prescribed for the said degree.

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

Index

Sl. No.	Date	Experiment Title	Page No.
1	23-09-2024	Quadratic Equations	1-09
2	30-09-2024	SGPA Calculations	10-17
3	07-10-2024	ToString Program	18-23
4	14-10-2024	Shape area	24-29
5	21-10-2024	Bank Account	30-41
6	28-10-2024	Package (Internal and External)	42-48
7	28-11-2024	Exceptions (Age Problem)	49-54
8	28-11-2024	Threads (BMCE and CSE)	55-59
9	28-11-2024	Interfaces	60-66
10	28-12-2024	Interprocess Communication and deadlock	67-78

Github Link:
https://github.com/jiyanitturkar/Java

Program 1
Quadratic Equation

Algorithm:

LAB PROGRAM-1

4

```
import java.util.Scanner;
class Quadratic
{
 int a,b,c;
 double r1,r2,d;
 void get()
 {
 Scanner in= new Scanner (System in);
 System out.println("enter the value of a");
 a= in.nextInt();
 system.out.print ("enter the value of b");
 b= in. nextInt();
 System. out_ println ("Enter the value of c");
 c= in. nextInt();
 }
void compute()
 {
    if (a==0){
        system. out. println("the value of 'a' cannot be 0");
        }

    else {
        d= (b* b)- (4* a *c);
        if (d>=0){
            r1= (-b) / (2* a);
            system. out. print("The roots are real & equa");
            System. out. print ("The first root is "+r1);
```

```
System.out.print("The second root is "+r2);

else if (d>0) {
        r1 = (-b+ Math.sqrt(d)) / (2*a);
        r2 = (-b - Math.sqrt(d)) / (2*a);
        System.out.println("The first root "+r1);
        System.out.println("The second root" + r2);
        }

    else {
        System.out.println("Both the roots are not real
        and are imaginary");
        r1 = (-b) / (2*a);
        r2 = (Math.sqrt(-d) / (2*a));
        System.out.println("The first root is"+r1);
        System.out.println("The second root is"+r2);
        }

    }

}

class QuadraticMain
{
  public static void main (String Arg[])
  {
    Quadratic quad = new Quadratic();
    quad.get();
    quad.compute();
    System.out.println("Name: Tuja");
    System.out.println("USN: 1BM22CS128");
  }
```

① Enter the value of a

1

Enter the value of b

-3

Enter the value of c

6

Roots are real & distinct

3 and - 1

② Enter the value of 'a'

1

Enter the value of 'b'

-6

Enter the value of 'c'

9

Roots are real and equal

3

o/pre

Bear

3, -4-24

Code:

```
import java.util.Scanner;
import java.lang.Math;

class Quadratic {
int a, b, c;
void scan() {

Scanner in = new Scanner(System.in);
System.out.println("Enter the value of a:");
a = in.nextInt();
if (a == 0) {
System.out.println("Value of a is invalid. Enter a non-zero value:");
a = in.nextInt();
}
System.out.println("Enter the value of b:");
b = in.nextInt();
System.out.println("Enter the value of c:");
c = in.nextInt();
}
void display() {
double d = b * b - 4 * a * c;
double r1, r2;
if (d == 0) {
r1 = (-b) / (2.0 * a);
System.out.println("Roots are real and equal.");
System.out.println("Roots are " + r1 + " and " + r1);
} else if (d &gt; 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("Roots are " + r1 + " and " + r2);
```

```
} else {
r1 = (-b) / (2.0 * a);
r2 = Math.sqrt(-d) / (2.0 * a);
System.out.println("Roots are imaginary.");

System.out.println("Real part is " + r1 + " and imaginary part is " + r2);
}
}
public static void main(String[] args) {
Quadratic q = new Quadratic();
q.scan();
q.display();
System.out.println("Name: Krithika");
System.out.println("USN: 1BM23CS128");
}
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 2
SGPA Calculation

Algorithm:

7th Oct 2024.

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

```
import java.util.Scanner;

class student {

    Scanner sc = new Scanner(System.in);
    String name, usn;
    double marks[] = new double[8];
    double credits[] = new double[8];

    void stuinfo() {

        System.out.print("Enter the name of the student");
        name = sc.next();
        System.out.print("Enter the USN of the student");
        usn = sc.next();
        System.out.print("Enter the marks of student");
        for (int i=0; i<8; i++) {
            System.out.print("subject"+(i+1)+":");
            marks[i] = sc.nextdouble();
        }
        for (int i=0; i<8; i++) {
            System.out.print("subject"+(i+1)+":");
            credit[i] = sc.nextDouble();
        }
```

```
void printinfo() {
    S.O.P (" Name of student is "+ name);
    S.O.P (" USN : "+ usn);
    S.O.P (" The marks and credits of the subjects are :");
    for (int i=0; i<8; i++) {
        S.O.P ("subject" + (i+1) + ":"+ marks[i] + "Credit:"+
        credit [i] );
    }
}

double sgpa() {
    double sum=0;
    double cred=0;
    for (i=0, i<8; i++) {
        sum += calculateGrade (marks[i]) * credit [i];
        cred += credit [i];
    }
    return cred == 0? 0: sum / cred;
}

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

```
        return 0;
    }
}
class Display {
    public static void main (String [] args) {
        Student s1 = new student [];
        s1. stdinfo();
        s1. printinfo ();
        double sgn = s1. sgpa();
        S.O.P.(" The SGPA of the student is :"+ sgn);
        S.O.P (" Name : Jiya Nittiurkar");
        S.O.P (" USN : 1BM23CS128");
    }
}
```

OUTPUT:

```
>> javac Display.java
>> java Display
Enter the name of the student : Jiya
Enter the usn of the student : 1BM23CS128
Enter the marks of the student subject 1, 91
Subject 2: 86
Subject 3, 92
Subject 4: 76
```

Enter the credits of the subject respectively subject 1-4.

Subject 2: 4

Subject 3: 3

Subject 4: 3

Subject 5: 3

Subject 6: 1

Subject 7: 1

Subject 8: 1

The SGPA of the student is 9.4.

Name: Jiya Nittuekar

USN: 1BM23CS128

9-10-24

Code:

```
import java.util.Scanner;

class Subject {
int credits[] = new int[8];
int grades[] = new int[8];

public void getMarksAndCredits(Scanner s) {
System.out.println("Enter the subject Marks:");
for (int j = 0; j &lt; 8; j++) {
System.out.print("Subject " + (j + 1) + ": ");
subjectMarks[j] = s.nextInt();
}

System.out.println("Enter the subject Credits:");
for (int j = 0; j &lt; 8; j++) {
System.out.print("Credits for Subject " + (j + 1) + ": ");
credits[j] = s.nextInt();
}
}

public void computeGrades() {
for (int j = 0; j &lt; 8; j++) {
grades[j] = subjectMarks[j] / 10 + 1;
if (grades[j] &gt; 10) {
grades[j] = 10;
}
if (grades[j] &lt; 4) {
grades[j] = 0;
}
}
}
```

```
}

class Student {
String name;
String usn;
double sgpa;
Subject subject;

public Student() {
subject = new Subject();
}

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

public void computeSGPA() {
double totalCredits = 0;
double totalPoints = 0;

for (int j = 0; j &lt; 8; j++) {
totalCredits += subject.credits[j];
totalPoints += subject.grades[j] * subject.credits[j];
}
sgpa = totalPoints / totalCredits;
}

public void display() {
System.out.println("Name: " + name);
```

```
System.out.println("USN: " + usn);
System.out.println("SGPA: " + sgpa);
}

public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
Student[] students = new Student[3];
System.out.println("Name: Jiya NitturkarH Kotian");
System.out.println("USN: 1BM23CS128");

for (int i = 0; i &lt; 3; i++) {
System.out.println("Entering details for Student " + (i + 1) + ":");
students[i] = new Student();
students[i].getStudentDetails(sc);
students[i].subject.getMarksAndCredits(sc);
students[i].subject.computeGrades();
students[i].computeSGPA();
students[i].display();
System.out.println();
}
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 3
ToString Program

Algorithm:

14th October 2024 LAB-PROGRAM-3

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

```
import java.util.Scanner;
class Books {
String name;
String author;
int price;
int num_pages;
Books (String name, String author, int price, int num_pages) {
this.name = name;
this.author = author;
this.price = price;
this.num_pages = num_pages;
}
}
class Main {
public static void main (String args[]) {
Scanner in = new Scanner (System.in);
int n;
int i; }
```

```
public String toString(){
String name, author, price, num_pages;
name = "Book name" + this.name + "\n");
author = " Author name" + this.author + "\n");
price = " Price:" + this.price + "\n");
num_pages = " Number of pages:" + this.num_pages" + "\n");

return name + author + price + num_pages;
}

}

System.out.println("number of Books");
n = in.nextInt();
Books b[] = new Books[n];
for(i=0; i<n; i++){
System.out.print("Name of books "+ (i+1));
String name = in.next();

system.out.print(" Author :"+ (i+1));

String author = in.next();

system.out.print(" Price :" + (i+1));
int price = in.nextIn();

System.out.print(" number of pages"+ (i+1));

int num_pages = in.nextInt();

b[i] = new Books(name, author, price, num_pages);
}
```

```
for(i=0;i<n;i++){
System.out.print(b[i]);
}
System.out.print("Name: Jiya Nittukar");
System.out.print("USN: 1RM23CS128");
}
}
```

Output:

Enter the number of books: 1

Enter the details of book 1 :

Name: Arch

Author: Satya_Harsh_Chandra

Price: 420

Name of pages: 500

Details of the book:

Book name: Arch

Author: Satya_Harsh_Chandra

Price: 420

Number of pages: 500

Done
14-10-24

Code:

```
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() {
return “Book name: “ + this.name + “\n” +
“Author name: “ + this.author + “\n” +
“Price: “ + this.price + “\n” +
“Number of pages: “ + this.numPages + “\n”;
}
}

class BookDemo {
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();
s.nextLine();

Books b[] = new Books[n];
for (int i = 0; i &lt; n; i++) {

System.out.println("Enter details for book " + (i + 1) + ":");
System.out.println("Name:");
name = s.nextLine();
System.out.println("Author:");
author = s.nextLine();
System.out.println("Price:");
price = s.nextInt();
System.out.println("Number of pages:");
numPages = s.nextInt();
s.nextLine();

b[i] = new Books(name, author, price, numPages);
}
System.out.println("Name: Krithika\nUSN:1BM23CS128");
System.out.println("Book Details:");
System.out.println();
for (int i = 0; i &lt; n; i++) {
System.out.println("Details for book " + (i + 1) + ":");
System.out.println(b[i].toString());
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 4
Shape Area

Algorithm:

21st October 2024 LAB PROGRAM- 4 14

Develop a Java program to create an abstract class named ~~Using generics, write a Java program that has a method~~ Shape that contains two integers and an empty method ~~named design~~ named printArea(). Provide three classes extend the Shape. Each one of the classes contain only the method printArea() that gives area of the given shape.

```
import java.util. Scanner;
abstract class shape{
double l, b, result
void PrintArea(){
Scanner in= new scanner (System.in);
System.out.print("Enter the length of the shape");
l=in.nextDouble();
System.out.print(" Enter the breadth of the shape (enter 0
                for circle");
b= in.nextDouble();
}
abstract void calc();
}

class Rectangle extends shape{
void calc(){
result= l*b;
System.out.print(" The area of rectangle is" + result); }}
```

```
class Triangle extends Shape {
void calc() {
result = 0.5*l*b;
System.out.print("The area of triangle is"+result);
}
}

class circle extends Shape {
void calc() {
result = 3.14*l*l;
System. out. print("The area of the circle" + result);
}
}

class Main {
public static void main (String args[]) {
Rectangle r = new Rectangle ();
r. PrintArea();
r. calc();
Triangle t = new Triangle ();
t. PrintArea();
t. calc();
Circle c = new Circle();
c. printArea();
```

OUTPUT

Enter the length of shape

2

Enter the breadth of shape

5

The area of rectangle is 10.0.

Enter the ~~length~~ height of shape

3

Enter the ~~length~~ breath of shape

2

The area of triangle is 3.0.

Enter the length of shape

2

Enter the 'o' for circle.

0

The area of circle is 12.56.

21.10

Code:

```
import java.util.Scanner;
abstract class Shape {
double a, b;
abstract void printArea();
}

class Rectangle extends Shape {
Rectangle(double le, double br) {
a = le;
b = br;
}
void printArea() {
double area = a * b;
System.out.println("Area of Rectangle is " + area);
}
}

class Triangle extends Shape {
Triangle(double le, double br) {
a = le;
b = br;
}
void printArea() {
double area = 0.5 * a * b;
System.out.println("Area of Triangle is " + area);
}
}

class Circle extends Shape {
Circle(double le) {
```

```
a = le;

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

class ShapeDemo {
public static void main(String s[]) {
Scanner sc = new Scanner(System.in);
System.out.println("Name:Krithika\nUSN:1BM23CS128");

System.out.println("Enter the dimensions of Rectangle");
double a = sc.nextDouble();
double b = sc.nextDouble();
Rectangle r = new Rectangle(a, b);

System.out.println("Enter the dimensions of Triangle");
a = sc.nextDouble();
b = sc.nextDouble();
Triangle t = new Triangle(a, b);

System.out.println("Enter the radius of Circle");
a = sc.nextDouble();
Circle c = new Circle(a);

r.printArea();
t.printArea();
c.printArea();
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 5
Bank Account

Algorithm:

28th October 2024 LAB PROGRAM - 5 17

- Develop a Java program to create a class Bank that maintains two kinds of account for its customers, one called saving account and the other current account. The savings account provides compound interest with withdrawal facilities but no cheque book facility. The current account provides check book facility but no interest. Current account holders should also maintain a minimum balance and if the balance falls below this level, service charge is imposed.

```
import java.util.Scanner;
class Account {
    String customerName;
    int accountnumber;
    String account_name;
    double balance;

    Account (String name, int accNumber, String accType) {
    customerName = name;
    accountNumber = accNumber;
    accountType = accType;
    balance = 0;
}

public void deposit ( double amount ) {
    balance += amount;
    S.O.P ("Deposited : " + amount + " Updated balance" + balance);
}

public void displayBalance () {
        S.O.P ("Account Balance : " + balance);
}
```

```
public void withdraw (double amount) {
    S.O.P ("This operation is specific to account type");
}
}

class SavingAccount extends Account {
    double interestRate = 0.12;
    public void computeInterest () {
        double interest = balance * interestRate;
        balance += interest;
        S.O.P ("Interest added:" + interest + "updated balance" +
                balance);
    }

public void withdraw (double amount) {
    if (balance >= amount) {
        balance = - amount;
        S.O.P ("withdraw " + amount + " updated balance" +
                balance); }
    else {
        S.O.P ("Insufficient balance");
    }
}
}
```

```
public void check_minimum (){
    if (balance < minBalance) {
        balance -= service charge;
        S.O.P ("Balance below minimum - penalty " + penalty +"
            updated balance " + balance);
    }
}

public class Bank {
public static void main (String args[]) {
    S.O.P ("Name: Jiya Nittur");
    S.O.P ("USN: 1BM23CS128");

    Scanner sc = new Scanner (System.in);
    S.O.P ("Enter customer name")
    name = in.next();
    S.O.P ("Enter account number");
    account_number = in.next Int();

    while (true) {
        S.O.P. (" 1. Deposit 2. Withdraw 3. Compute
            interest for saving amount 4. Display
            5. Exit ");

        S.O.P ("Enter your choice");
```

```
if (accType.equals("savings")){
    switch (choice){
        case 1:
            S.O.p("Enter the deposit amount");
            double deposit = sc.nextDouble();
            break;

        case 2:
            savings.Acount.computeInterest();
            break;

        case 3:
            S.O.p("customer name" + savingsAcount.customerName);

            S.O.p("Account number" + savingsAcount.number);
            S.O.p("Type of Account" + savingsAcount.accountType);
            break;

        case 4:
            System.exit(0);
            break;
        default:
            S.O.P("Invalid choice");
```

Output:

choose Account type

1

Enter name: Jiya

Enter Account number: 128

Enter Account type Saving

Enter the initial balance

6500

Menu

1. Deposit
2. Withdraw
3. Compute interest
4. Display
5. Exit

Enter your choice: 3

Enter rate, Times compounded & hour: 223

Compound interest 5042.2207.

28-11-24

Code:

```
import java.util.Scanner;

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


    Account(String name, int accNumber, String accType) {
        customerName = name;
        accountNumber = accNumber;
        accountType = accType;
        balance = 0;
    }


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

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


    public void withdraw(double amount) {
        System.out.println("This operation is specific to account type.");
    }
```

```
}


class SavAccount extends Account {
    double interestRate = 0.12;
    SavAccount(String name, int accNumber) {
        super(name, accNumber, "Savings");
    }



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

    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}
class CurAccount extends Account {
    double minBalance = 500.0;
    double serviceCharge = 50.0;

    CurAccount(String name, int accNumber) {
        super(name, accNumber, "Current");
    }
```

```
    public void checkMinBalance() {
        if (balance < minBalance) {
            balance -= serviceCharge;
            System.out.println("Balance below minimum. Service charge imposed: " + serviceCharge + ".
Updated balance: " + balance);
        }
    }
    public void withdraw(double amount) {
        if (balance >= amount) {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ". Updated balance: " + balance);
            checkMinBalance();
        } else {
            System.out.println("Insufficient balance.");
        }
    }
}

public class Bank {
    public static void main(String[] args) {
            System.out.print("Name:Jiya Nitturkar\n");
            System.out.print("USN: 1BM23CS128\n");
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter customer name:");
        String name=sc.next();
        System.out.println("Enter account number:");
        int accountnumber=sc.nextInt();
        SavAccount savingsAccount = new SavAccount(name, accountnumber);
System.out.println("Enter customer name:");
        String name1=sc.next();
        System.out.println("Enter account number:");
```

```
        int accountnumber1=sc.nextInt();
        CurAccount currentAccount = new CurAccount(name1, accountnumber1);

        while (true) {
            System.out.println("\n-----MENU-----");
            System.out.println("1. Deposit\n2. Withdraw\n3. Compute Interest for Savings Account\n4.
Display Account Details\n5. Exit");
            System.out.print("Enter your choice: ");
            int choice = sc.nextInt();

            System.out.print("Enter the type of account (saving/current): ");
            String accType = sc.next();

            if (accType.equals("saving")) {
                switch (choice) {
                    case 1:
                        System.out.print("Enter the deposit amount: ");
                        double depositAmount = sc.nextDouble();
                        savingsAccount.deposit(depositAmount);
                        break;
                    case 2:
                        System.out.print("Enter the withdrawal amount: ");
                        double withdrawalAmount = sc.nextDouble();
                        savingsAccount.withdraw(withdrawalAmount);
                        break;
                    case 3:
                        savingsAccount.computeInterest();
                        break;
                    case 4:
                        System.out.println("Customer name: " + savingsAccount.customerName);
                        System.out.println("Account number: " + savingsAccount.accountNumber);
                        System.out.println("Type of Account: " + savingsAccount.accountType);
```

```
            savingsAccount.displayBalance();
            break;
        case 5:
            System.exit(0);
            break;
        default:
            System.out.println("Invalid choice.");
    }
} else if (accType.equals("current")) {
    switch (choice) {
        case 1:
            System.out.print("Enter the deposit amount: ");
            double depositAmount = sc.nextDouble();
            currentAccount.deposit(depositAmount);
            break;
        case 2:
            System.out.print("Enter the withdrawal amount: ");
            double withdrawalAmount = sc.nextDouble();
            currentAccount.withdraw(withdrawalAmount);
            break;
        case 3:
            System.out.println("Current accounts do not earn interest.");
            break;
        case 4:
            System.out.println("Customer name: " + currentAccount.customerName);
            System.out.println("Account number: " + currentAccount.accountNumber);
            System.out.println("Type of Account: " + currentAccount.accountType);
            currentAccount.displayBalance();
            break;
        case 5:
            System.exit(0);
            break;
```

```
                default:
                    System.out.println("Invalid choice.");
            }
        } else {
            System.out.println("Invalid account type.");
        }
    }
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 6
Package (Internals and Externals)

Algorithm:

11th ~~Oct~~ Nov 2024 LAB - PROGRAM - 6

Create a package CIE which has two classes - Student & Internal. The class Student has members like usn, name, sem. The class internals derived from Student 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. Import the two packages in a file that declares the final marks of 'n' students in all five courses.

```
package CIE;
public class Student {
public String usn;
public String name;
public int sem;
}

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

```
this.internal_marks= internal_marks;
}
public int [] getInternal_marks(){
return internal_marks;
}
}

package SEE;
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;

public class Main 1 {
    public static void main(String args[]){
        Scanner scanner= new Scanner(System.in);
        S.o.p ("Enter the no. of students:");
        int n= scanner.nextInt();
        scanner.nextline();
            for (int i=0; i<n; i++){
                S.o.p (" Enter details for student" + (i+1);

                S.o.p (" Enter USN:");
                String usn= scanner.next line();

                S.o.p ("Enter name");
                String name= scanner.nextLine();

                S.o.p (" Enter semester");
```

```
S.O.p ("Course"+(i+1)+": ");
for (int j=0; j<5; j++) {
    S.O.p ("Course"+(j+1)+": ");
    external_marks [j]= scanner.nextInt();
}
scanner.nextLine();
Internal student 1 = new Internals ();
External student 2 = new Externals ();
C.O.p ("Final marks for"+ student1.name+": ");
for (j=0; j<5; j++) {
    int finalMark = student1.getInternalMarks()[j]+
                    student2.external_marks()[j];
    S.O.p ("Course"+(j+1)+": ");
}
}
S.O.p ("Name: Riya");
S.O.p ("USN: 128");
}
}
```

OUTPUT

Name: Riya

USN: 125

Code:

```
package CIE;
public class Student{
public String usn;
public String name;
public int sem;

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

//PACKAGE CIE IN WHICH WE HAVE INTERNALS.JAVA
package CIE;
public class Internals extends Student{
public int internal_marks[]=new int[5];
public Internals (String usn, String name, int sem,int internal_marks[]){
super(usn,name,sem);
this.internal_marks=internal_marks;
}
public int[] getInternalMarks(){
return internal_marks;
}
}

//PACKAGE SEE IN WHICH WE HAVE EXTERNALS.JAVA
package SEE;
import CIE.Student;
```

```
public class Externals extends Student{
public int external_marks[]=new int[5];
public Externals (String usn,String name,int sem,int external_marks[]){
super(usn,name,sem);
this.external_marks=external_marks;
}
public int[] getexternal_marks(){
return external_marks;
}
}

//WE HAVE MAIN FILE DIRECTLY INTO THE USN FOLDER
import CIE.Internals;
import SEE.Externals;
import java.util.Scanner;

public class Main1 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of students: ");
        int n = scanner.nextInt();
        scanner.nextLine();
              for (int i = 0; i < n; i++) {
            System.out.println("\nEnter details for student " + (i + 1));

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

            System.out.print("Enter Name: ");
            String name = scanner.nextLine();
```

```
        System.out.print("Enter Semester: ");
        int sem = scanner.nextInt();
        scanner.nextLine();
                int[] internal_marks = new int[5];
        System.out.println("Enter internal marks for 5 courses: ");
        for (int j = 0; j < 5; j++) {
            System.out.print("Course " + (j + 1) + ": ");
           internal_marks[j] = scanner.nextInt();
        }
        scanner.nextLine();

                int[] external_marks = new int[5];
        System.out.println("Enter external marks for 5 courses: ");
        for (int j = 0; j < 5; j++) {
            System.out.print("Course " + (j + 1) + ": ");
            external_marks[j] = scanner.nextInt();
        }
        scanner.nextLine();
                 Internals student1 = new Internals(usn, name, sem, internal_marks);
        Externals student2 = new Externals(usn, name, sem, external_marks);

                System.out.println("\nFinal marks for " + student1.name + ":");
        for (int j = 0; j < 5; j++) {
            int finalMark = student1.getInternalMarks()[j] + student2.external_marks[j];
            System.out.println("Course " + (j + 1) + ": " + finalMark);
        }
    }
     System.out.println("Name:Jiya Nitturkar");
     System.out.println("USN:1BM23CS128");
    scanner.close();
  }
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 7
Exceptions (Age problem)

Algorithm:

LAB-PROGRAM-7

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

```
import java.util.Scanner;
class WrongAge extends Exception {
        public WrongAge () {
                (" Age Error");
        }
        public WrongAge (String message) {
                super (message);
        }
}
class Father {
    int FatherAge;

    public Father () throws WrongAge {
        System.out.Println(" Enter father's age:");
        fatherage = in.nextInt();
        if (fatherage <0) {
            throw new WrongAge ("Age cannot be -ve");
        }
```

```
        super();
        System.out.print("Enter son's age");
        sonAge = in.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 -ve");
        }
    }

    public void display() {
        super.display();
        S.O.P("son's age" + sonAge);
        S.O.P("father's age" + fatherAge);
    }
}

public class Main {
    public static void main (String [] args) {
        try {
            Son son = new Son();
```

Output

Enter father's Age : 56

Enter son's Age : -5

exception : Age cannot be -ve

Father's Age : 45

Enter son's age : 15

Father's Age : 45

Son's Age : 15

29-11-24

Code:

```
import java.util.Scanner;
class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message); // Pass message to the superclass (Exception)
    }
}
class InputScanner {
    public static Scanner s = new Scanner(System.in); // Scanner object for input
}

class Father extends InputScanner {
    protected int fatherAge;
    public Father() throws WrongAge {
        System.out.print("Enter father's age: ");
        fatherAge = s.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.print("Enter son's age: ");
        sonAge = s.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() {
        System.out.println("Son's Age: " + sonAge);
            System.out.println("Father's Age: " + fatherAge);
    }
}
public class Main {
    public static void main(String[] args) {
        try {
            Son son = new Son();
            son.display();  // This will display son's age

        } catch (WrongAge e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 8
Threads (BMSCE and CSE)

Algorithm:

LAB_PROGRAM_8

Write a program which creates two threads one thread displaying "BMS College of Eng" once every 10 seconds & another "CSE" once every '2' seconds.

```
class CollegeThread extends Thread {
    public void run() {
        try {
            while (true) {
                S.O.P ("BMS College of Engineering");
                Thread.sleep(10000);
            }
        }
        catch (InterruptedException e) {
            S.O.P ("CollegeThread Interrupted");
        }
    }
}

class DepartmentThread extends Thread {
    public void run() {
        try {
            while (true) {
                S.O.P ("CSE");
                Thread.sleep(2000);
            }
```

```
public class Main {
    public static void main (String [] args) {
        CollegeThread college = new CollegeThread();
        DepartmentThread dept = new DepartmentThread();

        college.start();
        dept.start();
    }
}
```

output:

```
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
BMS college of Engineering
CSE
CSE
CSE
CSE
CSE
```

28-11-24

Code:

```
import java.util.Scanner;
class WrongAge extends Exception {
    public WrongAge(String message) {
        super(message); // Pass message to the superclass (Exception)
    }
}
class InputScanner {
    public static Scanner s = new Scanner(System.in); // Scanner object for input
}

class Father extends InputScanner {
    protected int fatherAge;
    public Father() throws WrongAge {
        System.out.print("Enter father's age: ");
        fatherAge = s.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.print("Enter son's age: ");
        sonAge = s.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() {
        System.out.println("Son's Age: " + sonAge);
            System.out.println("Father's Age: " + fatherAge);
    }
}
public class Main {
    public static void main(String[] args) {
        try {
            Son son = new Son();
            son.display();  // This will display son's age

        } catch (WrongAge e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

Github Link:
https://github.com/jiyanitturkar/Java

Program 9
Interface

Algorithm:

LAB - PROGRAM - 7

Write a program that creates a user interface to perform integer divisions. The user enters two inputs input 1 & input 2. Result divided is displayed and the result is input 1 / input 2. If input 2 = 0, throw an Arithmetic Exception Display.

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

class SwingDemo {
    SwingDemo() {
        JFrame jfrm = new JFrame("Divider App");
        jfrm.setSize(275, 150);
        jfrm.setLayout(new FlowLayout());
        jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel jlab = new JLabel("Enter the divider and
                    divider:");

        JTextField ajtf = new JTextfield(8);
        JTextField bjtf = new JTextfield(8);
        JButton button = new JButton("Calculate");

        JLabel err = new JLabel();
        JLabel alab = new JLabel();

        JLabel blab = new JLabel();
```

```
jform.add (button);
jform. (alas);
jform. add (blas);
jform. add (anslas);

Action listener l = new Action listener();
public void action performed (Action Event e) {
  S.O.P ("Action event from a text field"); };
ajtf. addAction listener (l);
bjtf. add Action listener (l);
button. add Action. listener (new Action listener() {
public void action performed (Action event evt) {
  try {
      int a = Integer . parseInt (ajtf. get Text());
      int b = Integer.parseInt(bjtf.getText());
      int ans = a/b;
  catch (Arithmatic Exception e) {
      alab .set Text ("");
      blas . set Text ("");
      anslab. set Text ("");
      err.set Text ("Enter only Integer");
  }
});
public static void main (String args[]) {
```

```
new SwingDemo();
    }
    });
    }
}
```

Output:

28-11-24

Code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class SwingDemo {
    SwingDemo() {
        // Create jframe container
        JFrame jfrm = new JFrame("Divider App");
        jfrm.setSize(300, 200);
        jfrm.setLayout(new GridLayout(6, 2, 5, 5)); // Using GridLayout for better arrangement

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

        // Text label
        JLabel jlab = new JLabel("Enter the divisor 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 for displaying results and error
        JLabel err = new JLabel("");
        JLabel alab = new JLabel("");  // A label for the divisor
        JLabel blab = new JLabel("");  // B label for the dividend
        JLabel anslab = new JLabel(""); // Answer label
```

```
// Add components in a structured manner using GridLayout
jfrm.add(jlab);
jfrm.add(new JLabel(""));  // Empty label for layout spacing

jfrm.add(new JLabel("Divisor (A):"));
jfrm.add(ajtf);

jfrm.add(new JLabel("Dividend (B):"));
jfrm.add(bjtf);

jfrm.add(button);
jfrm.add(err);  // Error message label

jfrm.add(alab);  // A value label
jfrm.add(blab);  // B value label

jfrm.add(anslab);  // Answer label

// Add action listener to the Calculate button
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        try {
            int a = Integer.parseInt(ajtf.getText());  // Read divisor
            int b = Integer.parseInt(bjtf.getText());  // Read dividend

            if (b == 0) { // Handle division by zero
                throw new ArithmeticException("B should be NON-zero!");
            }

            int ans = a / b;  // Perform division
            alab.setText("A = " + a);  // Display A value
            blab.setText("B = " + b);  // Display B value
```

```
            anslab.setText("Ans = " + ans);  // Display result
            err.setText("");  // Clear error if valid input

        } catch (NumberFormatException e) {
            // Handle invalid integer input
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText("Enter Only Integers!");
        } catch (ArithmeticException e) {
            // Handle division by zero
            alab.setText("");
            blab.setText("");
            anslab.setText("");
            err.setText(e.getMessage());
        }
    }
  });

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

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

Github Link:
https://github.com/jiyanitturkar/Java

Program 10
Interprocess Communication and Deadlock

Algorithm:

LAB_PROGRAM_10

A. Implementation of a producer and consumer.

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

```
        this.n=n;
        valueSet = true;
        System.out.print("Put:" + n);
        System.out.print("Intimate consumer\n");
        notify();
    }
}

class Producer implements Runnable {
    Q q;
    Producer(Q q) {
        this.q=q;
        new Thread(this,"Producer").start();
    }
    public void run() {
        int i=0;
        while (i<15) {
            q.put(i++);
        }
    }
}

class Consumer implements Runnable {
```

```
while (i<15){
  int r=q.get();
  system.out.print ("consumed"+ r);
  i++;
  }
}
}

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

Output:

Put: 1

Got: 1

Put: 2

Got: 2

Put: 3

Got: 3

Put: 4

Code:

```
class Q {
    int n;
    boolean valueSet = false;
    synchronized int get() {
        while (!valueSet) {
            try {
                System.out.println("\nConsumer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        System.out.println("Got: " + n);
        valueSet = false;
        System.out.println("\nIntimate Producer\n");
        notify();
        return n;
    }
    synchronized void put(int n) {
        while (valueSet) {
            try {
                System.out.println("\nProducer waiting\n");
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        this.n = n;
        valueSet = true;
        System.out.println("Put: " + n);
```

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

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

LAB_PROGRAM_10

B. Deadlock

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

class B {
    synchronized void bar (A a) {
        Thread currentThread().getName()
```

```
catch(Exception e){
    system.out.println(" B interrupted");
  }
  system.out.print(name+ "trying to call A.last()");
  a.last();
 }

void last(){
    system.out.print(" Inside A.last");
  }
}

class Deadlock implements Runnable {
        A a= new A();
        B b= new B();
        Deadlock(){
          Thread.currentThread().setName(" MainThread");
          Thread t= new Thread(this," RacingThread");
          t.start();
          a.foo(b);
          system.out.print(" back in main thread");
        }

public void run(){
    b.bar(a);
    ...int(" Back in other thread");
```

Output :

Main thread entered A.foo

Racing Thread entered B.bar

Main Thread trying to call B.last()

Inside A.last

Back in main thread

Racing Thread trying to call A.last()

Inside A.last

Back in other thread

28-11-24

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

```
}
class Deadlock implements Runnable {
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName("MainThread");
        Thread t = new Thread(this, "RacingThread");
        t.start();
        a.foo(b);
        System.out.println("Back in main thread");
    }
    public void run() {
        b.bar(a);
        System.out.println("Back in other thread");
    }
    public static void main(String args[]) {
        new Deadlock();
    }
}
```