**B.M.S COLLEGE OF ENGINEERING BENGALURU**
Autonomous Institute, Affiliated to VTU


# OBJECT ORIENTED JAVA PROGRAMMING
## LAB RECORD

*Submitted by:*

**Siddharth HG**
**1BM22CS276**

Department of Computer Science and Engineering
B.M.S College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019

**Lab 1:**

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

```
import java.util.Scanner;
class Quadratic {
   int a, b, c;
   double r1, r2, d;
   void getd() {
      Scanner s = new Scanner(System.in);
      System.out.println("Enter the coefficients of a, b, c:");
      a = s.nextInt();
      b = s.nextInt();
      c = s.nextInt(); }
   void compute() {
      while (a == 0) {
         System.out.println("Not a quadratic equation");
         System.out.println("Enter a non-zero value for a:");
         Scanner s = new Scanner(System.in);
         a = s.nextInt(); }
      d = b * b - 4 * a * c;
      if (d == 0) {
         r1 = (-b) / (2 * a);
         System.out.println("Roots are real and equal");
         System.out.println("Root1 = Root2 = " + r1);
      } else if (d > 0) {
         r1 = ((-b) + (Math.sqrt(d))) / (2 * a);
         r2 = ((-b) - (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);
            System.out.println("Root2 = " + r2 + "i");
        }
    }
}

class QuadraticMain {
    public static void main(String[] args) {
        System.out.println("USN = 2023BMS02586 NAME = SANTHOSH S");
        Quadratic q = new Quadratic();
        q.getd();
        q.compute();
    }
}
```

**Lab 2:**

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

```
import java.util.*;
 class Subject{
    int subjectMarks;
    int credits;
    int grade;
}

class Student{
    Subject subject[];
    String name;
    String usn;
    double SGPA;
    Scanner sc=new Scanner(System.in);

 Student(){
int i;
subject = new Subject[8];
for(i=0;i<8;i++)
subject[i] = new Subject();
}

public void getStudentDetails(){
    System.out.println("enter your name");
    name=sc.nextLine();
    System.out.println("usn");
    usn=sc.nextLine();
 }
```

```
public void getMarks(){
    for(int i=0;i<8;i++){
        System.out.println("enter the marks for "+(i+1)+" subject");
        subject[i].subjectMarks=sc.nextInt();
        System.out.println("enter the credits of "+(i+1)+" subject:");
        subject[i].credits=sc.nextInt();
        subject[i].grade=(subject[i].subjectMarks/10)+1;
        if(subject[i].grade==11){
            subject[i].grade=10;
        }
        if(subject[i].grade<4){
            subject[i].grade=0;
        }
    }
}

public void computeSGPA(){
    int effectiveScore=0;
    int total=0;
    float SGPA=0;
    for(int i=0;i<8;i++){
        effectiveScore+=(subject[i].credits*subject[i].grade);
        total+=subject[i].credits;

    }
     System.out.println("effective score:"+effectiveScore);
        System.out.println("total credits:"+total);
        SGPA=effectiveScore/total;
        System.out.println("SGPA;"+SGPA);
}
}

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

```
        Student s=new Student();
        s.getStudentDetails();
        s.getMarks();
        s.computeSGPA();
        System.out.println("name:"+s.name);
        System.out.println("usn:"+s.usn);
    }
}
```

**Lab3:**

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

```
import java.util.Scanner;
class Books{
String name,author;
int price,numPages;
Books(String name,String author,int price,int numPages){
this.name=name;
this.author=author;
this.price=price;
this.numPages=numPages;
}
String tostring(){
   String author,name,price,numPages;
   name="This Book is  "+this.name+"\n";
   author="Author is "+this.author+"\n";
   price="price ="+this.price+"\n";
   numPages="No of pages "+this.numPages+"\n";
   return name+author+price+numPages;
}
}

public class Main{
   public static void main(String[] args) {
      System.out.println("USN = 2023BMS02586 NAME = SANTHOSH S");
      Scanner s=new Scanner(System.in);
      int n,numPages,price;
      String name,author;
      System.out.println("Enter No of Books ");
      n=s.nextInt();
      Books arr[];
      arr=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 price of Book ");
           price=s.nextInt();
            System.out.println("Enter the no of pages of the Book ");
            numPages=s.nextInt();
            arr[i]=new Books(name,author,price,numPages);
        }

        for(int i=0;i<n;i++){
            System.out.println(arr[i].tostring());
        }
    }
}
```

**Lab 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.**

```
import java.util.Scanner;


class InputScanner{
Scanner s;
InputScanner(){
   s=new Scanner(System.in);
}
}


abstract class Shape extends InputScanner{
   double a;
   double b;
   abstract void getDetails();
   abstract void printArea();
}


class Rectangle extends Shape{
   void getDetails(){
      System.out.println("Enter length and breadth");
      a=s.nextDouble();
      b=s.nextDouble();
   }
   void printArea(){
      System.out.println("Area of Rectangle = "+(a*b));
   }


}


class Triangle extends Shape{
   void getDetails(){
```

```
        System.out.println("Enter a and b");
        a=s.nextDouble();
        b=s.nextDouble();
    }
    void printArea(){
        System.out.println("Area of Triangle = "+(a*b/2));
    }

}

class Circle extends Shape{
 void getDetails(){
    System.out.println("Enter radius");
    a=s.nextDouble();
    }
    void printArea(){
        System.out.println("Area of Circle = "+(3.14*a*a));
    }
}

class AbstractMain{
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.getDetails();
        r.printArea();
        t.getDetails();
        t.printArea();
        c.getDetails();
        c.printArea();
        System.out.println("Name = Santhosh S USN = 2023BMS02586"); }}
```

**Lab 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.**

```
import java.util.*;
import java.util.*;

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 withdraw(double amount){
if((balance-amount)>=0){
balance -=amount;
}
else{
System.out.println("Insufficient balance");
}
}
void display(){
System.out.println("Name : "+name+"\n"+
  "AccountNo : "+accno+"\n"+
  "Type : "+type+"\n"+
  "balance: "+balance+"\n");
}
}

class SavingAccount extends Account{
private static int rate = 5;
SavingAccount(String name,int accno,String type,double balance){
super(name,accno,type,balance);
}
void balanceWithInterest(){
balance +=balance*rate/100;
System.out.println("balance: "+balance);
}

}
```

```
class CurrentAccount extends Account{
   private static int rate = 5;
   CurrentAccount(String name,int accno,String type,double balance){
   super(name,accno,type,balance);
   }
   void balanceWithInterest(){
   balance +=balance*rate/100;
   System.out.println("balance: "+balance);
   }

   }

public class Main{
public static void main(String args[]){
Scanner s = new Scanner(System.in);
System.out.println("Enter your name: ");
String name = s.nextLine();

System.out.println("Enter the account type (current or savings)");
String type = s.next();

System.out.println("Enter the account number: ");
int accno = s.nextInt();

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

Account acc = new Account(name,accno,type,balance);
SavingAccount sa =  new SavingAccount(name,accno,type,balance);
CurrentAccount cc=  new CurrentAccount(name,accno,type,balance);
double amount;
while(true){
if(acc.type.equals("savings")){
System.out.println("\n-------MENU------\n");
System.out.println("1. Deposit \n2.Withdraw \n 3.compute interest for SavingsAccount \n 4.Display
Account Details\n 5.Exit\n");

System.out.println("Enter your choice");
int choice = s.nextInt();


switch(choice){
case 1:System.out.println("Enter the deposit amount");
amount = s.nextDouble();
sa.deposit(amount);
break;
case 2: System.out.println("Enter the withdrawl amount ");
amount = s.nextDouble();
sa.withdraw(amount);
break;
case 3:sa.balanceWithInterest();
break;
case 4:System.out.println("Details: ");
sa.display();
break;
```

```
case 5: return;
default: System.out.println("Invalid choice ");
}

}

else if(acc.type.equals("current")){
   System.out.println("\n-------MENU------\n");
   System.out.println("1. Deposit \n2.Withdraw \n 3.compute interest for SavingsAccount \n
4.Display Account Details\n 5.Exit\n");

   System.out.println("Enter your choice");
   int choice = s.nextInt();


   switch(choice){
   case 1:System.out.println("Enter the deposit amount");
   amount = s.nextDouble();
   cc.deposit(amount);
   break;
   case 2: System.out.println("Enter the withdrawl amount ");
   amount = s.nextDouble();
   cc.withdraw(amount);
   break;
   case 3:cc.balanceWithInterest();
   break;
   case 4:System.out.println("Details: ");
   cc.display();
   break;
   case 5: return;
   default: System.out.println("Invalid choice ");
   }

   System.out.println("Santhosh S 2023BMS02586");
}
}
}
}
```

**Lab 6:**

**Create a package CIE which has two classes- Student and Internals. 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. 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.**

Cie/Student.java:

```
package cie;

import java.util.Scanner;

public class Student{
    public int sem;
   public String usn,name;
   Scanner sc=new Scanner(System.in);
   public void setStudentDetails(){
    System.out.println("Enter your Name :");
    name=sc.nextLine();
    System.out.println("Enter your USN :");
    usn=sc.nextLine();
    System.out.println("Enter your Sem :");
    sem=sc.nextInt();
   }

   public void getDetails(){
    System.out.println("Name :"+name);
    System.out.println("USN :"+usn);
    System.out.println("sem :"+sem);
   }



}
```

Cie/Internals.java:

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


public class Internals extends Student{
```

```
public int internalsMarks[]=new int[5];
Scanner sc=new Scanner(System.in);
public void setCIE(){
for(int i=0;i<5;i++){
    System.out.println("Enter cie marks of Subject "+(i+1));
    internalsMarks[i]=sc.nextInt();
}

}
}
```

See/External.java:

```
package see;
import cie.Internals;
import java.util.Scanner;
public class External extends Internals{
  public  int seeMarks[]=new int[5];
  public int finalMarks[]=new int[5];
  Scanner sc=new Scanner(System.in);
  public void setSEE(){
    for(int i=0;i<5;i++){
      System.out.println("Enter see marks of Subject "+(i+1));
      seeMarks[i]=sc.nextInt();
  }
  }

public void computeFinal(){
  for(int i=0;i<5;i++){
    finalMarks[i]=internalsMarks[i]+seeMarks[i]/2;
}

}
public void displayMarks(){
  for(int i=0;i<5;i++){
```

```
    System.out.println("Subject "+(i+1)+" = "+finalMarks[i]);
}

}}
```

DemoMain.java:

```
import see.External;
public class DemoMain{
    public static void main(String[] args) {
      External obj=new External();
      obj.setStudentDetails();
      obj.getDetails();
      obj.setCIE();
      obj.setSEE();
      obj.computeFinal();
      obj.displayMarks();
}}
```

**Lab 7:**

**Write a program that demonstrates handling of exceptions in inheritance tree. Create a base class called "Father" and derived class called "Son" which extends the base class. In Father class, implement a constructor which takes the age and throws the exception WrongAge( ) when the input age=father's age.**

```
import java.util.Scanner;


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


class Father{
int fatherAge;
Scanner sc=new Scanner(System.in);
public void validAge() throws WrongAge{
System.out.println("Enter Fathers age");
fatherAge=sc.nextInt();
if(fatherAge<=0){
   throw new WrongAge("Invalid fathers age");
}
}
}


class Son extends Father{
int sonAge;
Scanner sc=new Scanner(System.in);
public void validAge() throws WrongAge{
   System.out.println("Enter sons age");
   sonAge=sc.nextInt();
   super.validAge();
   if (sonAge>=fatherAge) {
      throw new WrongAge("Sons age cant be greater than Fathers age");
   }
```

```
    else if(sonAge<0){
        throw new WrongAge("Invalid son age");
    }
}

}

public class MyMain{
    public static void main(String[] args) {
        Son obj = new Son();
        try{
            obj.validAge();
        }
        catch(WrongAge e){
            System.out.println("Exception "+e.getMessage());
        }

    }
}
```

**Lab 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.**

```
class Bmsce extends Thread{
   public void run(){
      for(int i=0;i<10;i++){
         System.out.println("Bmsce");
         try{
            this.sleep(10000);
         }
        catch(InterruptedException e){System.out.println(e);}
        System.out.println(i);
      }
   }
}
class Cse extends Thread{
   public void run(){
      for(int i=0;i<10;i++){
         System.out.println("CSE");
         try{
            this.sleep(2000);
         }
        catch(InterruptedException e)
        {System.out.println(e);}
      }
   }
}
public class MyMain{
public static void main(String[] args) {
   Bmsce obj1 = new Bmsce();
   Cse obj2 = new Cse();
   obj1.start();
   obj2.start();}}
```

**Lab 9:**

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

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

class SwingDemo {
   SwingDemo() {
      // create jframe container
      JFrame jfrm = new JFrame("Divider App");
      jfrm.setSize(275, 150);
      jfrm.setLayout(new FlowLayout());
      // to terminate on close
      jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // text label
      JLabel jlab = new JLabel("Enter the divider and 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
jfrm.add(jlab);
jfrm.add(ajtf);
jfrm.add(bjtf);
jfrm.add(button);
jfrm.add(alab);
jfrm.add(blab);
jfrm.add(anslab);

ActionListener listener = new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
      System.out.println("Action event from a text field");
   }
};
ajtf.addActionListener(listener);
bjtf.addActionListener(listener);

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("\nA = " + a);
         blab.setText("\nB = " + b);
         anslab.setText("\nAns = " + ans);
         err.setText("");
      } catch (NumberFormatException e) {
```

```
                err.setText("Enter Only Integers!");
                alab.setText("");
                blab.setText("");
                anslab.setText("");
            } catch (ArithmeticException e) {
                err.setText("B should be NON zero!");
                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() {
        System.out.println("Santhosh 2023BMS02586");
      public void run() {
        new SwingDemo();
      }
    });
  }
}
```

**Lab 10:**

**Demonstrate Inter process Communication and deadlock.**

a) IPC

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

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

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

   synchronized void put(int n) {
      while (valueSet)
         try {
            System.out.println("Producer waiting");
            wait();
         } catch (InterruptedException e) {
            System.out.println("InterruptedException caught");
         }

      this.n = n;
      valueSet = true;
```

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

class Producer implements Runnable {
    Q q;

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

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

class Consumer implements Runnable {
    Q q;

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

    public void run() {
        int i = 0;
        while (i < 15) {
            int r = q.get();
```

```
            System.out.println("consumed: " + r);
            i++;
        }
    }
}

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

b) DeadLock

```
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 {
```

# INDEX

Name Siddharth H.S Std Sec

Roll No. Subject School/College

School/College Tel. No. Parents Tel. No.

| Sl. No. | Date | Title | Page No. | Teacher Sign / Remarks |
|---|---|---|---|---|
| 1 | 12/12/23 | Lab-1 Quadratic Equation | | |
| 2 | 19/12/23 | Lab-2 Student database | | |
| 3 | 26/2/24 | Lab-3 | | |
| 4 | 2/1/24 | Lab-4 Abstraction | | |
| 5 | 9/1/24 | Lab-5 Bank | | |
| 6 | 16/1/24 | Lab-6 Strings & Generics | | |
| 7 | 23/1/24 | Lab-7 Exception | | |
| 8 | 30/1/24 | Lab-8 Multithreading | | |
| 9 | 6/2/24 | Lab-10 IPC & Deadlock | | |
| 10 | 13/2/24 | Lab-9 Applet | | |
| 11 | 23/1/24 | Packages | | |

12-12-23

LRA 12-12-23

Date / /

Page

# Lab program

```
import java.util.scanner;
class quadratic
 {
   int a,b,c;
   double r1, r2, d;
   void getd()
  {
    scanner s = new scanner (System.in);
    System.out.println("Enter the values
                 of a, b, c");
     a = s.nextInt();
     b = s.nextInt();
     c = s.nextInt();
   }
    void compute()
  {
     while (a==0)
   {
     System.out.println("Not a quadratic
          equation");
     System.out.println("Enter a non zero
          value for a;");
     scanner s= new scanner (System.in);
     a = s.nextInt();
  }
d= b*b - 4*a*c;
```

```
if (d == 0)
  {
    r1 = (-b)/(2*a);
    System.out.println("Roots are real
      and equal");
    System.out.println("Root 1 = Root 2 =",
      + 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 &
      distinct");
  System.out.println("Root 1 =" + r1 + "Root 2
      =" + r2);
}
else if (d < 0)
{
  System.out.println("Roots are imaginary");
  r1 = (-b)/(2*a);
  r2 = Math.sqrt(-d)/(2*a);
  System.out.println("Root 1 =" + r1 + "+i" + r2);
  System.out.println("Root 1 =" + r1 + "-i" + r2);
}
}
}
```

```
class Quadratic main
{
 public static void main (String args[])
{
  Quadratic q = new Quadratic ();
   q.getd ();
   q.compute ();
  System.out.println ("USN: 1BM22CS276 ,
                Name : Siddharth");
 }
}
```

o/p

Enter the coefficients of a, b, c

8

10

11

Roots are Imaginary

Root1 = 0.0 + i0.9921567416492215

Root2 = 0.0 - i0.9921567416492215

USN : 1BM22CS276 , Name : Siddharth.

19/12/23 MRA 19-12-23 Date / / Page

// 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 method to calculate SGPA of a Student.

```
import java.util.Scanner;

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

        Subject() {
            subjectMarks = 0;
            credits = 0;
            grade = 0;
        }
}

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

        Student() {
            int i;
            subjects = new Subjects[8];
            for(i=0; i<8; i++)
                subjects[i] = new Subject();
```

```
    s = new Scanner(System.in);
}
void getstudentDetails(){
        System.out.print(s:"Enter Student
                NAME:");
        name = s.nextLine();
        System.out.print(s:"Enter USN:");
        usn = s.nexLine();
}
void getMarks(){
    for(int i=0; i<8; i++){
        System.out.println("Enter details
        for Subject" + (i+1));
        System.out.print(s:"Enter Marks:");
        subjects[i].subjectMarks = s.nextInt();
        System.out.print(s:"Enter Credits:");
        subjects[i].credits = s.nextInt();

        if(subjects[i].subjectMarks >=90){
            subjects[i].grade=10;
        } else if(subjects[i].subjectMarks>=75)
        {
            subjects[i].grade=9;
        }
        else if(subjects[i].subjectMarks>=60)
        {
            subjects[i].grade=8;
        }
        else if(subjects[i].subjectMarks>=50)
        {
            subjects[i].grade=7;
        }
```

```
else if (subjects[i].subjectMarks >= 40)
    {
        subjects[i].grade = 6;
    }
else {
        subjects[i].grade = 0;
    }
}
}

Void computeSGPA () {
    double totalCredits = 0;
    double weightedSum = 0.
    for (int i = 0; i < 8; i++) {
        totalCredits += subjects[i].credits;
        weightedSum += subjects[i].grade *
                subjects[i].credits;
    }
    SGPA = weightedSum / totalCredits;
}
Void displayResult () {
    System.out.println("Student
            Details:");
    System.out.println("Name:" + name);
    System.out.println("USN:" + usn);
    System.out.println("SGPA:" + SGPA);
}
}
```

```
Public class Main{
    public static void main (string[] args)
    {
        Student s1 = new Student();
        s1.getStudentDetails();
        s1.getMarks();
        s1.computeSGPA();
        s1.displayResult();
    }
}
```

2/1/24 URA-2-1-23

Lab-4

Develop a Java programming 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

1. [illegible]
2. [illegible]

```
import java.util.Scanner;
class InputScanner{
 Scanner s;
InputScanner(){
        s= new Scanner(System.in);
 }
}
abstract class shape extends InputScanner{
        double a;
        double b;
        abstract void getDetails();
        abstract void printArea();
}

class Rectangle extends Shape{
        void getDetails(){
```

```
        System.out.println("Enter length and
                breadth:");
        a=s.nextDouble();
        b=s.nextDouble();
    }
    void printArea(){
        System.out.println("Area of Rectangle="
                + (a*b));
    }
}
class Triangle extends Shape{
    void getDetails(){
        System.out.println("Enter a and b:");
        a=s.nextDouble();
        b=s.nextDouble();
    }
    void printArea(){
        System.out.println("Area of a Triangle
            Rectangle="+(a*b/2));
    }
}
class Circle extends Shape{
    void getDetails(){
        System.out.println("Enter radius:");
        a=s.nextDouble();
    }
    void printArea(){
        System.out.println("Area of Circle
    Rectangle="+(3.14*a*a));
    }
}
```

```
class AbstractMain {
    public static void main(String[] args) {
        Rectangle r = new Rectangle();
        Triangle t = new Triangle();
        Circle c = new Circle();
        r.getDetails();
        r.printArea();
        t.getDetails();
        t.printArea();
        c.getdetails();
        c.printArea();
    }
}
```

O/P

```
Enter length and breadth:
2
4
Area of Rectangle = 8.0
Enter a and b
2
6
Area of Triangle = 6.0

Enter radius
6
Area of Circle = 113.039999
```

lab-5

Date __/__/__
Page _____

Develop a Java program which has class Bank-account and saving-account which extends account

```
import java.util.*;
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) {
            dou balance += amount;
    }
    void withdraw (double amount) {
        if((balance.amount) > 0) {
                amount == amount;
        }
    else {
        System.out.println ("Insufficient balance");
    }
}
    void display () {
            System.out.println ("Name: " + name +
                "\n" + "Accounts: " +
            account + "\n" + "type: " + type +
            "\n" + "balance: " + balance +
                "\n");
        }
```

```
class SavingAccount extends Account {
    private static int rate = 5;
    SavingAccount (String name, int accno,
        String type, double balance) {
        super (name, accno, "savings", balance);
    }
    void balanceWithInterest() {
        balance += balance * rate/100;
        System.out.println("balance:" +bal
    }
}
class CurrentAccount extends Account {
    private double min_Bal = 500;
    private double serviceCharges = 50;
    CurrentAccount (String name, int accno,
        double balance)
    {
        super (name, accno, "current", balan
    }
    void checkMin()
    {
        if (balance < min_Bal)
        {
            System.out.println("Balance is less
                than min balance.
                Service charges imposed:" + service
                charges);
            balance -= serviceCharges;
            System.out.println("balance is:" + balan
        }
    }
}
```

```
class account.main
{
public static void main (String args[])
{
Scanner s = new Scanner(System.in);
System.out.println("Enter the name:");
string name = s.nextline();
system.out.println("Enter the type
(savings/current):");
string type = s.next();
System.out.println("Enter Accnt No.:");
int accno = s.nextInt();
System.out.println("Enter Initial Balance:");
double balance = s.nextDouble;
int ch;
double amount1, amount2;
Account acc = new Account(name,
accno, type, balance);
Saving_Account sa = new
Saving_Account(name, accno,
balance);
current_account ca = new current_Account
(name, accno, balance);
while(true)
{
if(acc.type.equals("Savings"))
{
System.out.println("\n Menu\n 1. Deposit
2. withdraw 3. compute Interest
4. display");
System.out.println("Enter the choice:");
ch = s.nextInt();
```

```
switch(ch)
{
case 1: System.out.println("Enter the
        amount :");
        amount1 = s.nextInt();
        sa.deposit(amount1);
        break;

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.exit(0);
}
}
else
{
System.out.println("In Menu In 1.deposit
        2 withdraw 3 display");
System.out.println("Enter the choice:");
ch = s.nextInt();
switch(ch)
{ case 1: System.out.println("Enter the
        amount:");
        amount1 = s.nextInt();
        ca.deposit(amount1);
        break;
```

```
case 2: System.out.println("Enter the amount:");
        amount2 = s.nextInt();
        ca.withdraw(amount2);
        ca.checkBan();
        break;
case 3: ca.display();
        break;
case 4: system.exit(0);
}
```

O/P

Enter the name: Virat
Enter the type (current/savings):
current
Enter the account number:
1001
Enter the Initial balance:
1000

Menu:
1 Deposit 2 withdraw 3. display
1
Enter the amount : 1000

Menu:
1. Deposit 2 withdraw 3. display
2
Enter the amount : 500

Menu:
1 Deposit 2 withdraw 3 display
3
name: Virat acno: 1001 type: current balance: 1500

16/01/24

# Lab-6

Date __/__/__
Page ____

① Demonstrate startswith() to give output true and false.

```
public class startsWithExample {
    public static void main(String[] args) {
        String str1 = "Hello, World!";
        String prefix1 = "Hello";
        boolean result1 = str1.startsWith(prefix1);
        System.out.println("Example 1 - Result: "
                + result1);

        String str2 = "Java Programming";
        String prefix2 = "Python";
        boolean result2 = str2.startsWith(prefix2);
        System.out.println("Example 2 - Result: "
                + result2);
    }
}
```

o/p

Example 1 - Result: true
Example 2 - Result: false

Date __/__/____
Page _____

(2) Demonstrate endsWith() to given output true and false

```
public class StartsWithExample {
    public static void main(String[] args)
    {
        String str1 = "Hello, World!";
        String suffix1 = "World!";
        boolean result1 = str1.endsWith(suffix1);
        System.out.println("Example 1 - Result: " +
                result1);

        String str2 = "Java Programming";
        String suffix2 = "Python";
        boolean result2 = str2.endsWith(suffix2);
        System.out.println("Example 2 - Result: " +
                result2);

    }
}
```

Example 1: Result: true
Example 2: Result: false

Date / /
Page

③ Write Java program to create an abstract class Bird with abstract methods fly() and makeSound(). Create Subclass Eagle & Hawk that extends the Bird class and implement the respective methods to describe how each bird flies and makes a sound.

```
abstract class Bird {
    abstract void fly();
    abstract void makeSound();
}

class Eagle extends Bird {
    void fly() {
        System.out.println("Eagle soars
            high in the sky.");
    }
    void makeSound() {
        System.out.println("Eagle makes
            screeching sound.");
    }
}

class Hawk extends Bird {
    void fly() {
        System.out.println("Hawk glides
        gracefully in the air.");
    }
    void makeSound() {
        System.out.println("Hawk emits a
            distinctive cry");
    }
}
```

Date / /
Page

```
class BirdMain {
    public static void main(String[] args)
    {
        Eagle e = new Eagle();
        e.fly();
        e.makesound();
        Hawk h = new Hawk();
        h.fly();
        h.makesound();
    }
}
```

o/p

Eagle soars high in the sky.
Eagle makes screeching sound.
Hawk glides gracefully in the air.
Hawk emits a distinctive cry.

16/1/24

Date __/__/__
Page

(4) Write a Java program to creati a generic class Stack which hold 5 integers and 5 double values

```
public class GenericStack<T> {
    private Object[] stackArray;
    private int top;
    private static final int MAX_SIZE = 10;

    public GenericStack() {
        stackArray = new Object[MAX_SIZE];
        top = -1;
    }

    public void push(T element) {
        if (top < MAX_SIZE - 1) {
            stackArray[++top] = element;
            System.out.println("Pushed: " + element);
        } else {
            System.out.println("Stack is full.
            Cannot push more elements.");
        }
    }

    public T pop() {
        if (!isEmpty()) {
            @SuppressWarnings("unchecked")
            T element = (T) stackArray[top--];
            System.out.println("Popped: " + element);
            return element;
        }
        else
        {
            System.out.println("Stack is empty. Cannot
            Pop elements.");
```

```
        return null;
    }
}
public boolean isEmpty() {
        return top == -1;
}

public static void main (String[] args) {
    GenericStack<Integer> IntegesStack =
            new GenericStack<>();
    InteperStack.push(element:1);
    Integersstack.push(element:2);
    Integerstack.push(element:3);
    IntegerStack.pop();

    GenericStack<Double> doubleStack =
            new GenericStack<>();
    doublestack.push(element:1.5);
    doublestack.push(element:2.5);
    doublestack.push(element:3.5);
    doubleStack.pop();
}
}
```

o/p

16/1/24

```
Pushed : 1
Pushed : 2
Pushed : 3
Pushed :
Popped : 3
Pushed : 1.5
Pushed : 2.5
Pushed : 3.5
Popped : 3.5
```

23/01/24

# Lab-6

Date / /
Page

Create a package CIE which has two classes - Students & Internals. 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.

CIE/Student.java

```
package cie;
import java.util.Scanner;
public class student {
    Scanner sc = new Scanner (System.in);
    public String name, USN;
    public int sem;
    public void setDetails () {
        name = sc.nextLine ();
        USN = sc.nextLine ();
        sem = sc.nextInt ();
    }
public void getDetails () {
    System.out.println("Name:" +name);
    System.out.println ("USN:" +USN);
    System.out.println ("Sem:" +sem);
}
```

cie/Internals.java

```
package cie;
import java.util.*;

public class Internals extends students {
    public int internalMarks[] = new int[5];
    Scanner sc = new Scanner(System.in);
    public void setcie(){
        for (int i=0, i<5 ; i++) {
            System.out.println("Enter marks
                of " + (i+1));
            internalMarks[i] = sc.nextInt();
    }
}
```

see/External

```
package see;

import java.util.*;
import cie.Internals;

class Public class Extends Internals {
        public int seeMarks[] = new Int[5];
        public int finalMarks[] = new Int[5];
        Scanner sc = new Scanner(System.in);

    public void setsee(){
        for(int i=0; i<5; i++) {
        System.out.println("Enter see marks of
            " + (i+1));
        seeMarks[i] = sc.nextInt();
}
```

```
public void computeFinal(){
    for(int i=0; i<5; i++){
        finalMarks[i] = seeMarks[i]/2 +
                externalMarks[i];
}

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

DemoMain.Java

```
import see.External
public class DemoMain{
    public static void main(String[] args){
        External obj = new External();
        obj.setDetails();
        obj.getDetails();
        obj.setCie();
        obj.getSee();
        obj.computeFinal();
        obj.displayMarks();
    }
}
```

O/P

Enter the cie marks of 5 subjects

50
40
35
40
50

Enter see marks of 5 subjects

150

95

65

85

90

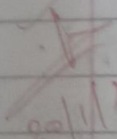
Subject 1 = 100

subject 2 = 87

subject 3 = 67

subject 4 = 82

Subject 5 = 95



30/01/24 Lab-7

Date / /
Page

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.

```
import java.util.Scanner;

class WrongAge extends Exception {
    public WrongAge(String A) {
        super(A);
    }
}
class Father {
    int fatherAge;
    Scanner sc=new Scanner(System.in);
    public void validAge() throws WrongAge {
        System.out.println("Enter Fathers age:");
        fatherAge = sc.nextInt();
        if(fatherAge<=0) {
            throw new WrongAge("Invalid
                fathers age");
        }
    }
}
class Son extends Father {
    int sonAge;
    Scanner sc=new Scanner(System.in);
    public void validAge() throws WrongAge {
        System.out.println("Enter sons age");
        sonAge = sc.nextInt();
        Super.validAge();
```

```
if (sonAge >= FatherAge){
    throw new WrongAge("Sons age cant be
        greater than Fathers age");
  }
else if (sonAge<0){
    throw new WrongAge("Invalid son age");
  }
}
}
public class ExceptionMain{
    public static void main (String[] args){
        Son obj= new Son()
        try{
            obj.validAge();
        }
        catch(wrongAge a){
            System.out.println("Exception "
                    +e.getMessage());
        }
  }
}
```

o/p 1

Enter sons age
10
Enter Fathers age
30
o/p 2
Enter Sons age
20
Enter Fathers age
15
Exception Sons age cant be greater than
Fathers age.

6/02/24

Lab-8 40

WAP which creates two threads, one thread displaying "BMSCE" once every ten seconds and another displaying "CSE" once every two seconds.

```
class Brisce extends Thread {
    public void run() {
        for(int i=0; i<10; i++) {
            System.out.println("BMSCE");
            try {
                this.sleep(millis:10000);
            }
            catch(InterruptedException e)
                System.out.println(e); }
            System.out.println(i);
        }
    }
}

class Cse extends Thread. {
    public void run() {
        for (int i=0; i<10; i++) {
            System.out.println("CSE");
            try {
                this.sleep(millis:2000);
            }
            catch (InterruptedException e)
            { System.out.println(e); }
        }
    }
}
```

```
public class MyMain {
    public static void main (String[] args) {
        BMSCE obj1 = new Bmsce();
        Cse obj2 = new Cse();
        obj1.start();
        obj2.start();
    }
}
```

o/p

BMSCE
CSE
CSE
CSE
CSE
CSE
0
BMSCE
CSE
CSE
CSE
CSE
CSE
1
BMSCE
2
BMSCE
3
Bmsce
4
BMSCE
5
BMSCE
6

BMSCE
7
BMSCE
8
BMSCE
9

6/02/24

Lab-10

```
// Demonstrate Inter process Communication & deadlock.

class Q {
    int n;
    boolean valueSet = false;

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

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

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

O/P

```
Put: 0
Intimate Consumer
Producer waiting
Got: 0
Intimate ~~Consumer~~ Producer
Put: 1
Intimate consumer
Producer waiting
Consumed: 0
Got: 1
Intimate Producer
consumed: 1
Put: 2
Intimate Consumer
Producer waiting
Got: 2
Intimate Producer
Consumed: 2
Put: 3
Intimate Consumer
Producer waiting
Got: 3
```

Intimate Consumer
consumed : 3
Put : 4
Intimate consumer
Producer waiting
Got : 4
Intimate Producer
consumed 4
Put : 5
Intimate Consumer
Producer waiting
Got : 5
Intimate Producer
Consumed : 5
Put : 6
Intimate consumer
producer waiting
~~Put~~ Got : 6
Intimate Producer
consumed : 6
Put : 7
Intimate consumer
Producer waiting

6/2/24

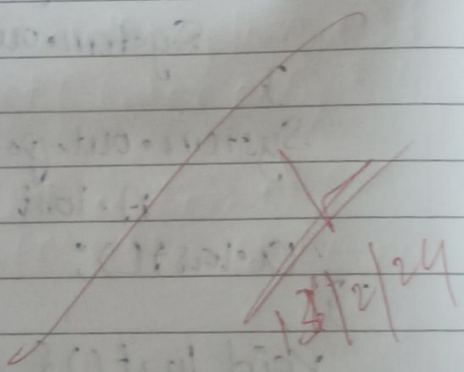
13/02/23

12. Dead Lock:

```
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("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 tocall
            A.last()");
    a.last();
}
void last() {
    System.out.println("Inside A.last");
}
}
class Deadlock implements Runnable{
    A a = new A();
    B b = new B();
    Deadlock() {
        Thread.currentThread().setName
            ("MainThread");
        Thread t = new Thread(this,
            "RacingThread");
```

```
        t.start();
        a.foo(b);
        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();
    }
}
```



Lab-9

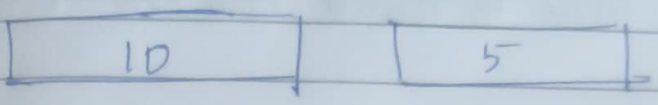
Write a program that creates a user interface to perform integer divisions.

```
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 dividend:");
        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();
        JLabel ansLab = new JLabel();
        jfrm.add(jlab);
        jfrm.add(ajtf);
        jfrm.add(bjtf);
        jfrm.add(button);
        jfrm.add(alab);
        jfrm.add(blab);
        jfrm.add(ansLab);
        jfrm.add(err);
```

```
button.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        try{
            int a = Integer.parseInt(aff.getText());
            int b = Integer.parseInt(aff.getText());
            if(b==0){
                throw new ArithmeticException
                    ("B Should be non-zero!");
            }
            int ans = a/b;
            alab.setText("A=" +a);
            blab.setText("B=" +b);
            anslab.setText("Ans=" +ans);
            err.setText("");
        }
        catch(NumberFormatException e){
            err.setText("Enter only Integers!");
        }
        catch(ArithmeticException e){
            err.setText("B should be non-zero");
        }
    }
});
frm.setVisible(true);
}

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

o/p

Enter the divider and divident:




Calculate  A=10 B=5 Ans=2

## FUNCTIONS

① JFrame:
Represents the main window of a GUI application. It can provide functionalities to create, manipulate and manage top-level containers. We can add buttons Text field using this.

② SetSize :-
SetSize (int width, int height) is a method of the JFrame class used to set size of the frame window in pixels.

③ Set layout:
Sets layout is a method of the container which chooses how components inside the window are arranged.

④ JLabel:
Displays the text or images on the window.

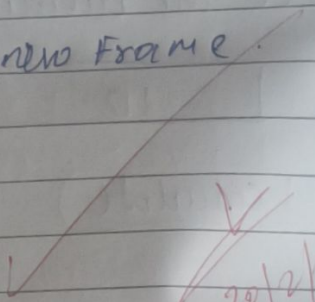
⑤ JField : Provides an editable text box

Date __/__/__
Page _____

for user input.

add Frame:

Used to add new Frame.



20/2/24