**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB LAB MANUAL**



**Department of computer science and Engineering**

**Amrita School of Engineering**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

SUBMITTED FROM: SUBMITTED TO:

|  |  |  |  |
| --- | --- | --- | --- |
| NAME | **S.SOHAIL** | **NAME** | **RAJ KUMAR BATTU** |
| ROLL NO | **AV.SC.U4CSE**24319 | **DEPARTMAENT** | **OOPS** |
| SECTION | **CSE-B** | **DESIGNATION** | **PROFESSOR** |

|  |  |
| --- | --- |
| MARKS |  |
| SIGNATURE |  |
| DATE |  |

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Si.no | Name of programme | Date | Page no | Signature |
| 1. | Week 1  Inastalation of java |  | 3-5 |  |
| 2. | Printing student details |  | 6-7 |  |
| 3. | Week 2  Calculating area of rectangle |  | 8-9 |  |
| 4. | Temperature convertion from C0 to F0  and F0  to C0 |  | 10-13 |  |
| 5. | Calculating Simple Interest |  | 13-15 |  |
| 6. | Finding greatest of three numbers |  | 15-16 |  |
| 7. | Factorial of a number |  | 16-18 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**WEEK-1**

1.How to install java jdk?

A. **1.Steps for downloading.**

Step 1: Java is developed by oracle. so open your web browser and search for java oracle.

Step 2: Then go to the website <https://www.oracle.com/in/java/> .

Step 3: And navigate to the java downloads. Then some earlier versions are available like

* [**JDK 23**](https://www.oracle.com/in/java/technologies/downloads/#java23)
* [**JDK 21**](https://www.oracle.com/in/java/technologies/downloads/#java21)
* [**GraalVM for JDK 23**](https://www.oracle.com/in/java/technologies/downloads/#graalvmjava23)
* [**GraalVM for JDK 21**](https://www.oracle.com/in/java/technologies/downloads/#graalvmjava21)

Step 3: JDK 21 is the latest *Long-Term Support (LTS)* release of the Java SE Platform. So we use this version.

Step 4: According to your device operating system choose the product/file description.

Step 5: Then the version will be automatically downloaded.

**2.Steps for installation.**

Step 1: Go to the folder where it was downloaded.

Step 2: Then open and accept all terms and conditions.

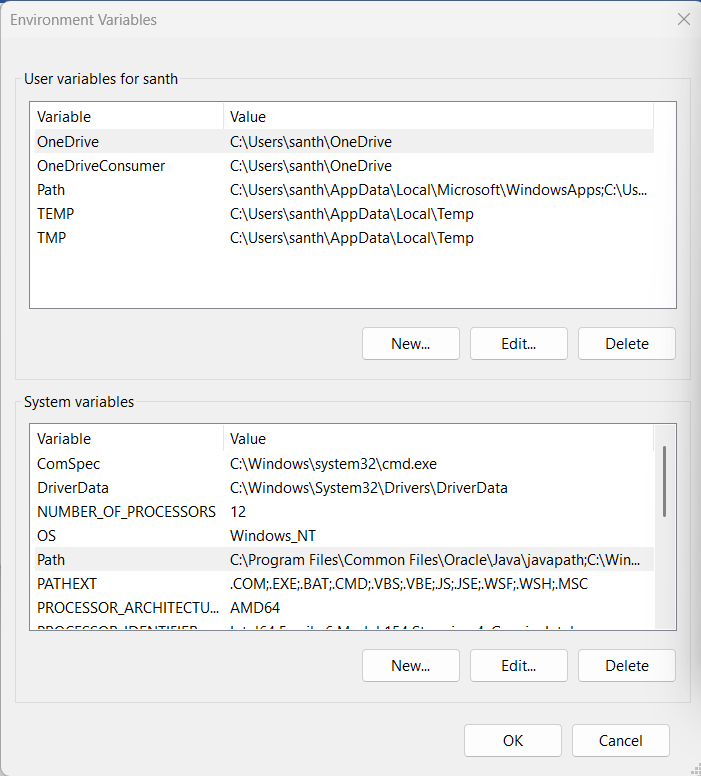
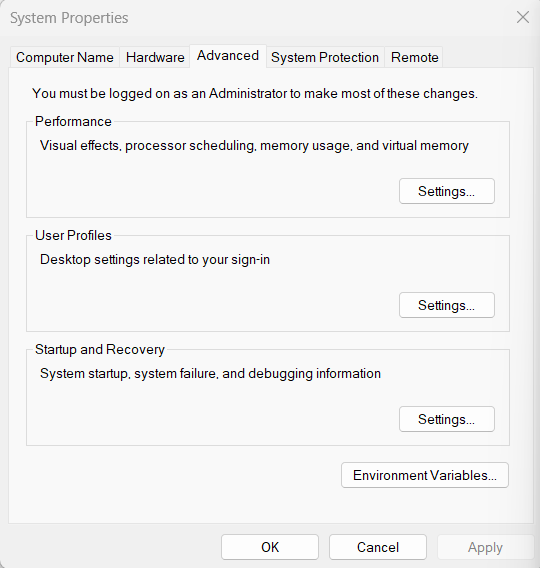
Step 3: And install it.

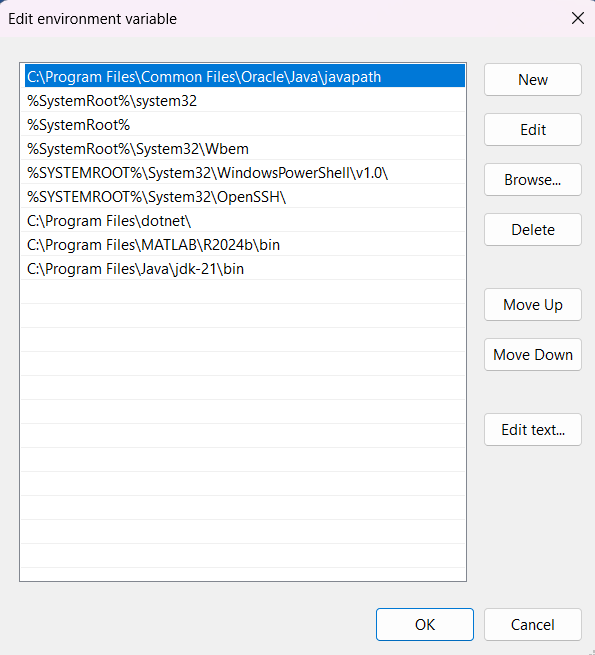
**3.Setting environmental variables.**

Step 1: Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on **Environment Variables**.

Step 2: Click **New** under **System Variables**:

* + 1. **Set Variable name as:** java\_home
    2. **Variable value:** The folder address where JDK is installed (like C:\Program Files\Java\jdk-21\bin)

Step 3: Find Path under **System Variables**, click **Edit**, and add the path of the jdk-21(C:\Program Files\Java\jdk-21\bin)

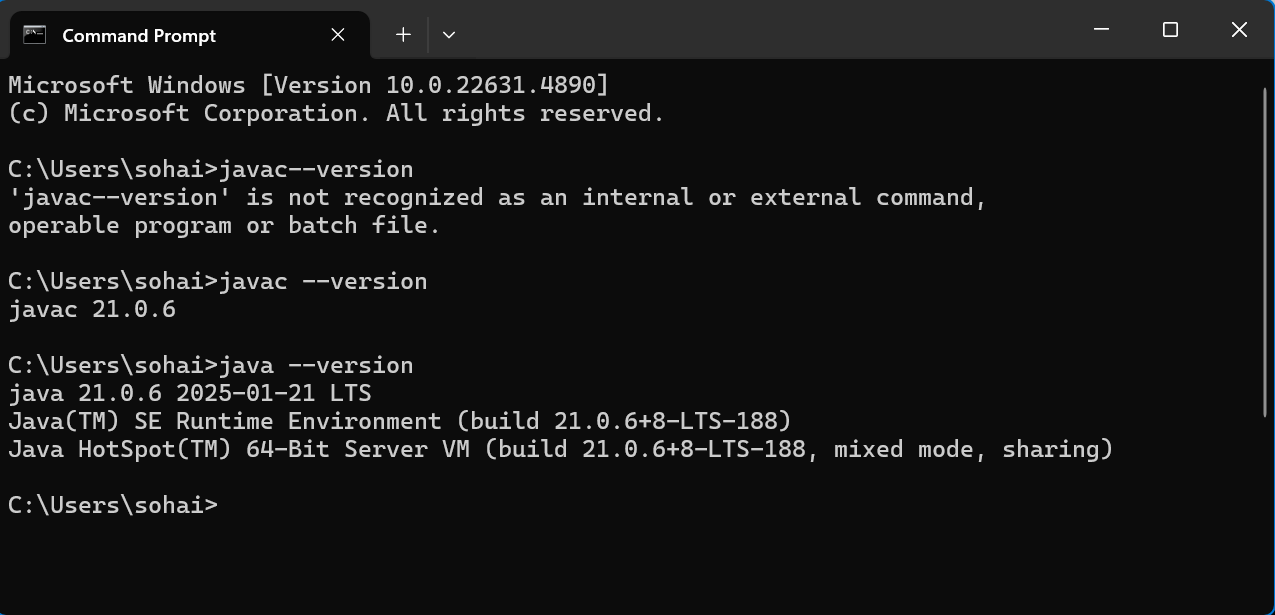


**4.Checking for jdk(java development kit) version.**

Step 1: Open command promt.

Step 2: Enter javac --version for version of jdk installed.

Step 3: Enter java –version for all details like when downloaded and path of environment variables.



**PROGRAM-1**

**Aim:** To print name, roll no class and section..

**Program:**

class student\_details{

public static void main(String[] args) {

System.out.println("NAME= Shaik Sohail");

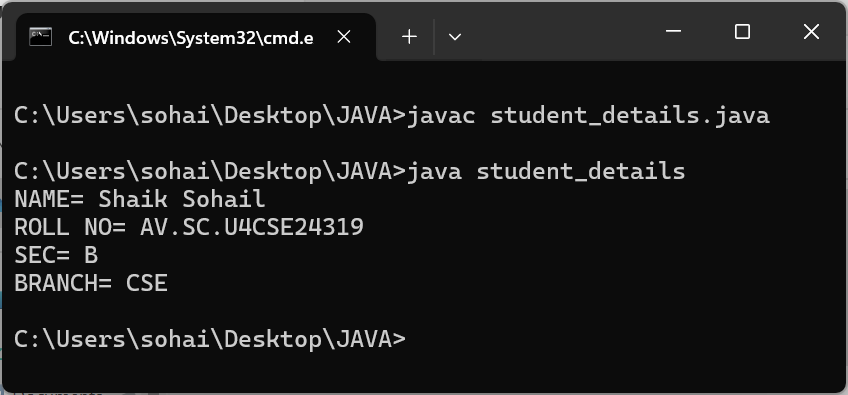
System.out.println("ROLL NO= AV.SC.U4CSE24319");

System.out.println("SEC= B");

System.out.println("BRANCH= CSE");

}

}

**Output:** 

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Placing ;(semi colon) |

**Important points:**

**WEEK-2**

**Aim:** Write a java program to calluclate the area of rectangle.

**Program:**

import java.util.Scanner;

public class area\_of\_rectangle {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the length of the rectangle: ");

double length = scanner.nextDouble();

System.out.print("Enter the width of the rectangle: ");

double width = scanner.nextDouble();

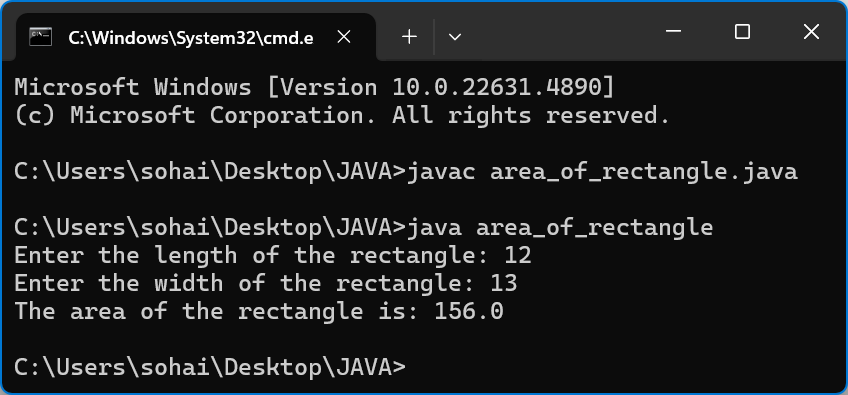
double area = length \* width;

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

scanner.close();

}

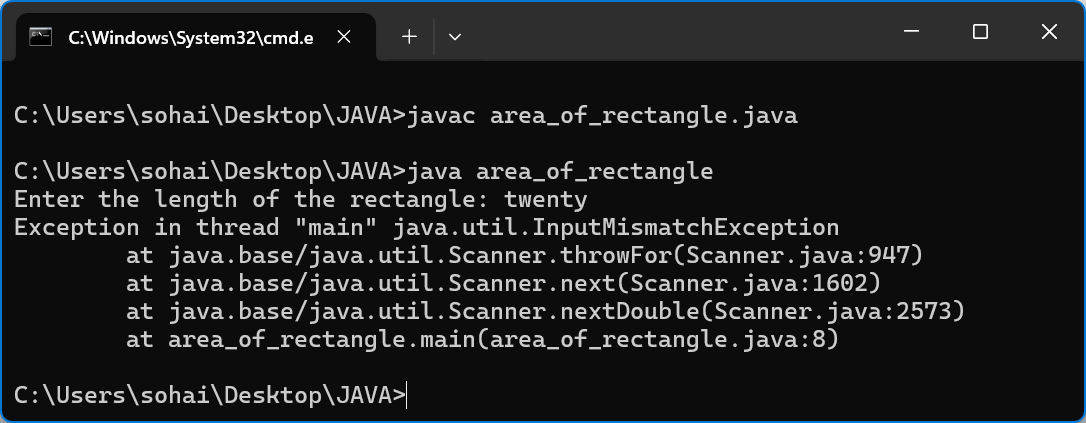
}

**Output:** ****

**Error:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Closing parentisis |

**Negative case:** The code dosent run when we give the input in text for



**Important points:**

1. Purpose: The program calculates the area of a rectangle based on the user-provided length and width.

2. User Input:

* It prompts the user to input the rectangle's length and width.
* The input is read using the Scanner class and stored as double values to allow for decimal measurements.

3.Modularity:

* The calculation logic is placed in a separate method, calculateArea(), which takes length and width

**2.Aim:** Write a java programee to calculate celcius to farenheat anf vice verce.

**Programee:**

import java.util.Scanner;

class TemperatureConverter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Choose conversion: ");

System.out.println("1. Celsius to Fahrenheit");

System.out.println("2. Fahrenheit to Celsius");

int choice = scanner.nextInt();

if (choice == 1) {

System.out.print("Enter temperature in Celsius: ");

double celsius = scanner.nextDouble();

double fahrenheit = (celsius \* 9.0 / 5.0) + 32;

System.out.println("Temperature in Fahrenheit: " + fahrenheit);

} else if (choice == 2) {

System.out.print("Enter temperature in Fahrenheit: ");

double fahrenheit = scanner.nextDouble();

double celsius = (fahrenheit - 32) \* 5.0 / 9.0;

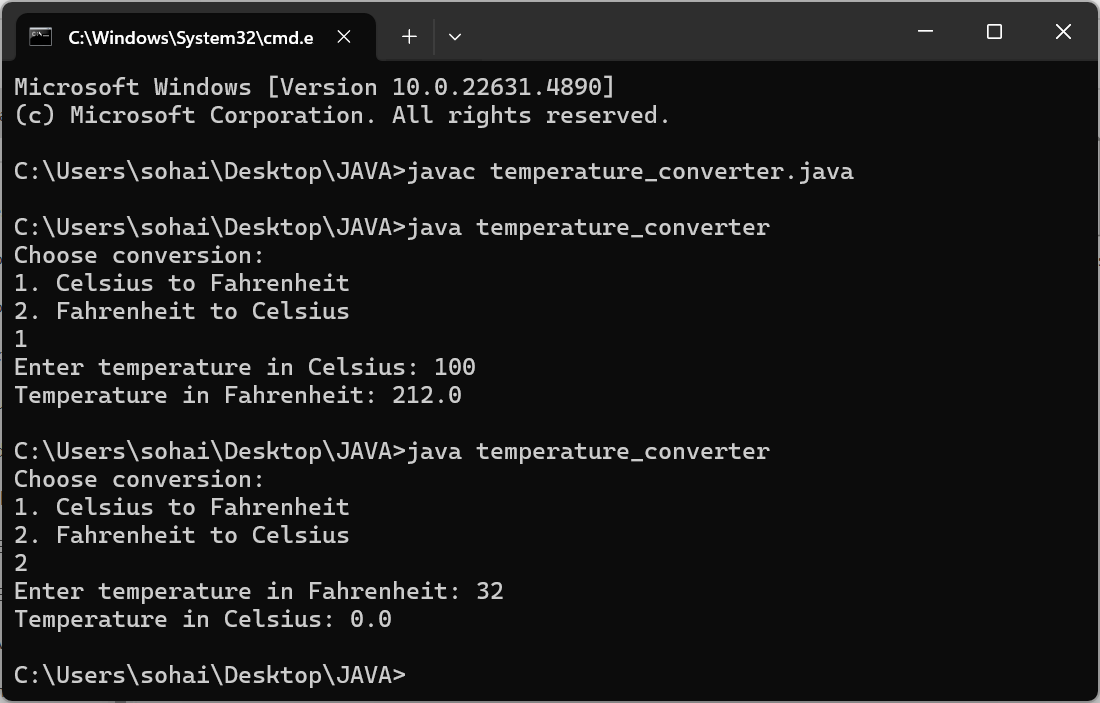
System.out.println("Temperature in Celsius: " + celsius);

} else {

System.out.println("Invalid choice! Please enter 1 or 2.");

}

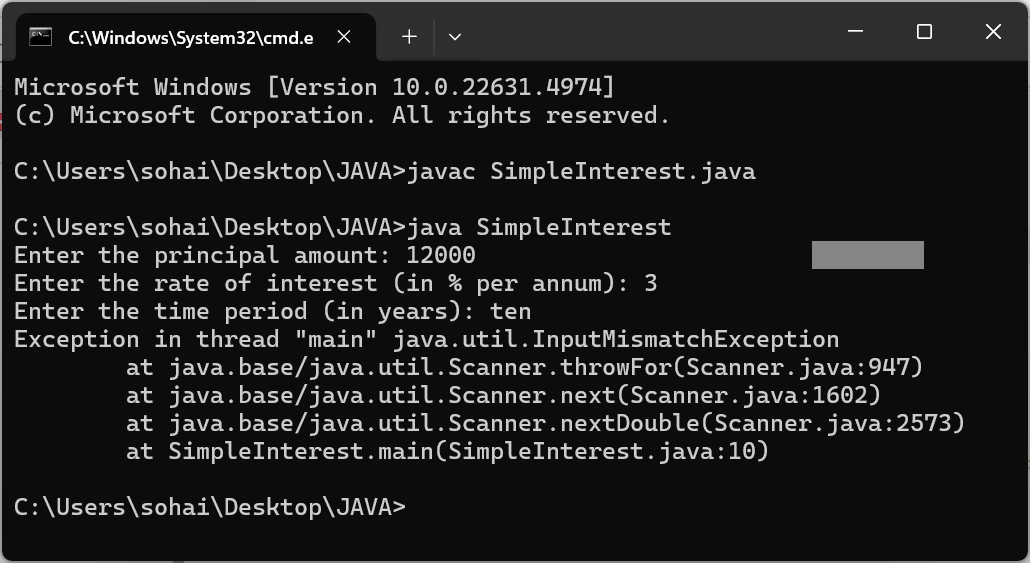
}}

**Output: **

**Errors:** 2 errors

|  |  |  |
| --- | --- | --- |
| SINO | Error name | Error rectifiction |
| 1 | Syntax error | Placing == in the place of = |

**Negative case:**

****

**Important points:**

1. The Scanner class is used to take user input from the keyboard.
2. The main method is the starting point of both programs where execution begins.
3. Each program converts temperature between Celsius and Fahrenheit using the correct formula and displays the result.

**3.**

**Aim:** Caluclate Simple Interest using java programme

**Programme:**

import java.util.Scanner;

public class SimpleInterest {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the principal amount: ");

double principal = scanner.nextDouble();

System.out.print("Enter the rate of interest (in % per annum): ");

double rate = scanner.nextDouble();

System.out.print("Enter the time period (in years): ");

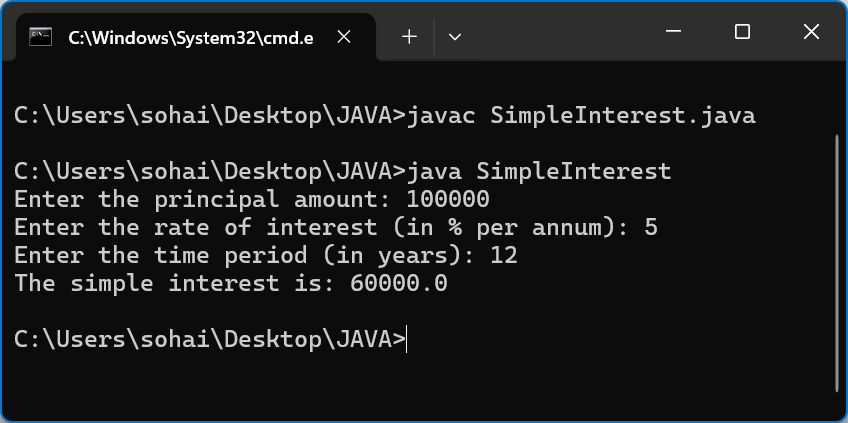
double time = scanner.nextDouble();

double simpleInterest = (principal \* rate \* time) / 100;

System.out.println("The simple interest is: " + simpleInterest);

}

}

**Output: **

**Errors:**

|  |  |  |
| --- | --- | --- |
| Si.no | Error name | Error correction |
| 1 | Syntax error | Placing ; at end of ling double rate. |

**Negative case:**

**Important points:**

1. =>We have used scanner method to give input to the code
2. =>Formula of the simple interest is (P\*T\*R)/100
3. =>We have assigned values for P,T,R

**4.**

**Aim:**To find largest number of there numbers.

**Programee:**

import java.util.Scanner;

class Gratest\_of\_Three{

Public void main(String[]args){

Scanner scn=new Scanner(System.in);

System.out.print(“enter three numbers ”);

int a=scn.nextInt();

int b=scn.nextInt();

int c=scn.nextInt();

if(a>b &a>c){

System.out.println(a+“ is gratest );

}

else if(b<c & b<a){

System.out.println(b+“ is gratest”);

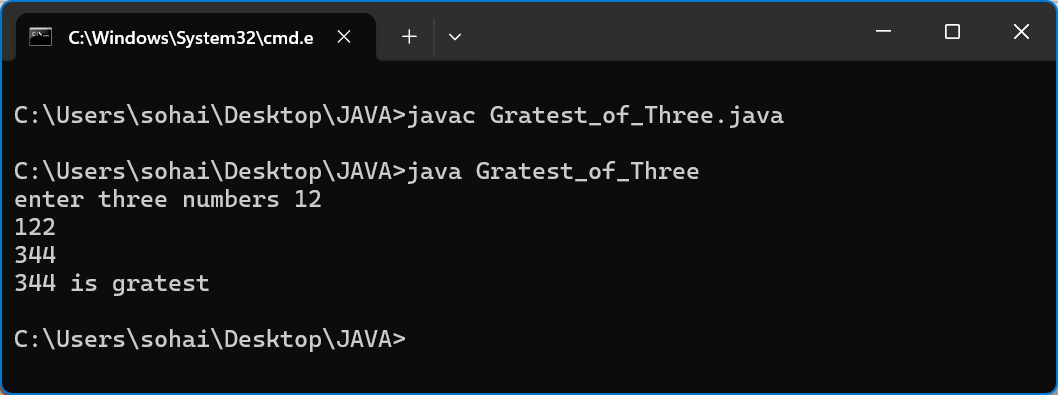
else{

System.out.println(c+ “ is gratest”):

}

}

}

**Output:** ****

**Errors:**

No errors found.

**Negative case:**

Error when we take strings.

**Important points:**

Calculate greatest of three numbers

**5.**

**Aim:** finding farctorial of a number using java.

**Programme:**

**Programme after rectification:**

import java.util.Scanner;

class Factorial{

public static void main(String[]args){

Scanner scn=new Scanner(System.in);

System.out.print("Enter the number: ");

int n=scn.nextInt();

int fact=1;

for(int i=1;i<=n;i++){

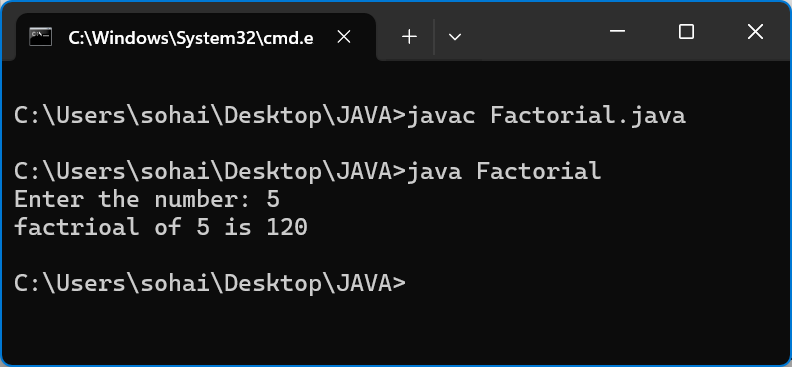
fact=fact\*i;

}

System.out.println("factrioal of "+n+" is "+fact);

}

}

**Output:** ****

O

**Errors:**

|  |  |  |
| --- | --- | --- |
| Si.no | Error name | Error rectification |
| 1 | Identifier error | Changing ststic to static |
| 2 | Error in string literal | Changimg = to + in print statment |

**Negative case:**

**Important points:**

sssssssssss

**WEEK-3**

**Aim**: To create java program with following instructions

1.Create a class with name car

2. Create four attributes named car colour ,Car brand, fuel type, mileage

3. Create three methods named start(), stop(). Service**()**

**4**. Create three objects named car1, car2 and car3

**Programme:**

import java.util.\*;

class Car{

    private String car\_color;

    private String Brand;

    private String fuel\_type;

    private String milage;

    Car() {

    }

    public void getCar() {

        System.out.println("car color: " + car\_color + "\n" +

                           "car Brand: " + Brand + " \n" +

                           "fuel Type: " + fuel\_type + " \n" +

                           "Milage: " + milage);

    }

    public void start() {

        System.out.println("Car started");

    }

    public void stop() {

        System.out.println("Car stopped");

    }

    public static void main(String[] args) {

        Car c1 = new Car();

        c1.car\_color = "Black";

        c1.Brand = "Rolls Royce";

        c1.fuel\_type = "Petrol";

        c1.milage = "10kmpl";

        c1.start();

        c1.getCar();

        c1.stop();

        System.out.println("\n");

        Car c2 = new Car();

        c2.car\_color = "Red";

        c2.Brand = "BMW";

        c2.fuel\_type = "Diesel";

        c2.milage = "14kmpl";

        c2.start();

        c2.getCar();

        c2.stop();

        System.out.println("\n");

        Car c3 = new Car();

        c3.car\_color = "White";

        c3.Brand = "Audi";

        c3.fuel\_type = "Petrol";

        c3.milage = "12kmpl";

        c3.start();

        c3.getCar();

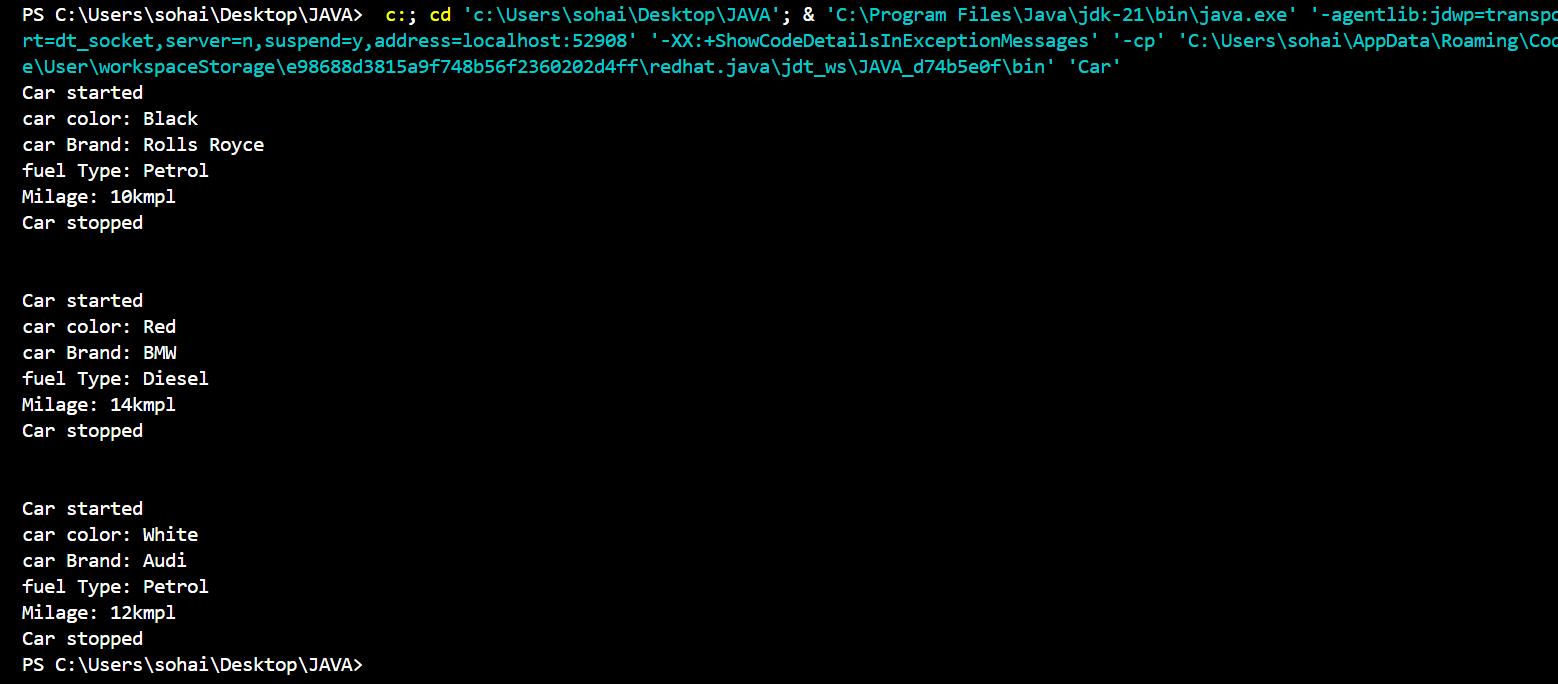
        c3.stop();

    }

}

**Class Diagram:**

|  |
| --- |
| car |
| - Car\_color: String  - Car\_brand: String  - fuel\_type: String  - mileage: int |
| + start(): void  + service(): void  + stop(): void |

**Output:** ****

**Errors:**

|  |  |  |
| --- | --- | --- |
| Si.no | Error name | Error rectification |
| 1 | Identifier error | Changing ststic to static |
| 2 | Error in main function | Changing arhs to args in print statment |

**Negative case:** No negative case

**Important Points:**

1. The car class has four attributes: Car\_color, Car\_brand, fuel\_type, and mileage.

2. It also has three methods: start(), service(), and stop().

3. The start(), service(), and stop() methods all print the same details about the car.

4. Each method prints the car's color, brand, fuel type, and mileage to the console

5. The main method creates three instances of the car class: car1, car2, and car3.

6. Each car object is assigned specific values for Car\_color, Car\_brand, fuel\_type, and mileage.

**2.**

**Aim:** Write a java class for a bank account with deposit() and withdraw() as methods.

**Programee:**

class Bank\_Account{

String Name;

int Account\_Number;

double Balance;

String IFSC;

String Branch;

double amount;

Bank\_Account(String Name, int Account\_Number, double Balance, String IFSC, String Branch){

this.Name = Name;

this.Account\_Number = Account\_Number;

this.Balance = Balance;

this.IFSC = IFSC;

this.Branch = Branch;

}

void Account\_details(){

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

System.out.println("Account Number: "+Account\_Number);

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

}

void deposit(double amount){

Balance += amount;

System.out.println("Deposit amount: "+amount);

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

}

void withdraw(double amount){

if(Balance < amount){

System.out.println("withwral Amount: "+amount);

System.out.println("Insufficient Balance");

}

else{

Balance -= amount;

System.out.println("withwral Amount: "+amount);

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

}

}

public static void main(String[] args){

Bank\_Account b1 = new Bank\_Account("Sohail", 1123323123, 10000, "SBIN080796","AMERICA");

b1.Account\_details();

b1.deposit(10000);

b1.withdraw(5000);

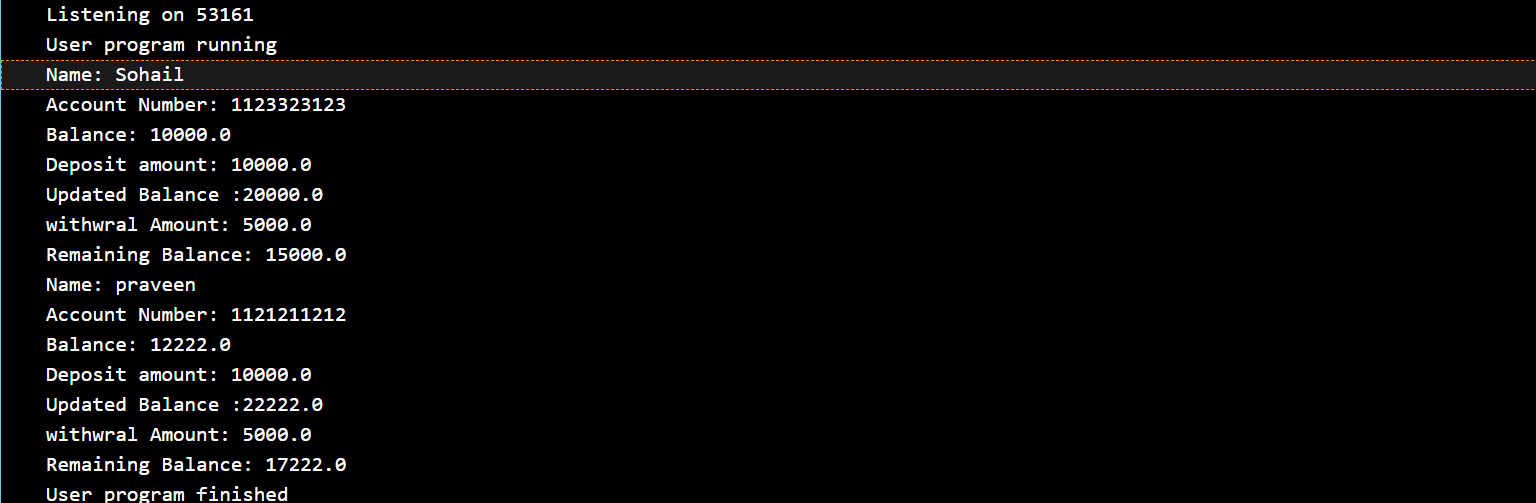
Bank\_Account b2 = new Bank\_Account("praveen", 1121211212,12222,"SBIN123321","Dharavi");

b2.Account\_details();

b2.deposit(10000);

b2.withdraw(5000);

}}

**Output: **

**Errors:**

|  |  |  |
| --- | --- | --- |
| Si.no | Error name | Error correction |
| 1 | Syntax error | Placing ; at end of print statment |

**Important Points:**

The Bank Account class has a private attribute balance to store the account balance.

* The class has a constructor, BankAccount(double initialBalance), which initializes the balance. If the initial balance is not positive, it sets the balance to 0.
* The deposit(double amount) method adds a positive amount to the balance and prints a message. If the deposit amount is not positive, it prints an error message.
* The getBalance() method returns the current balance of the account.
* The Main1 class contains the main method, which serves as the entry point of the program.
* In the main method, an instance of Bank Account is created with an initial balance of 1000.

**Class Diagram:**

|  |
| --- |
| BankAccount |
| -balance: double |
| +BankAccount(doubleinitialBalance  +deposit(doubleamount): void  + getBalance(): double |

**WEEK-4**

**1.**

**Aim**: Write a java program with class named “book”. The class should contain various attributes such as title, author, year of publication. It should also contain a constructor with parameters which initializes title, author, year of publication and create a method which displays the details of 2 books.

**Programme:**

import java.util.\*;

public class Book{

private String Title;

private String Author;

private String Year\_of\_publicition;

Book(){

}

public void getBook(){

System.out.println("Title: "+Title+"\n"+"Author: "+Author+" \n"+"Year of publicition: "+Year\_of\_publicition);

}

public static void main(String[]arga){

Book B1=new Book();

B1.Title="Missile Man of India";

B1.Author="APJ ABDUL KALAM";

B1.Year\_of\_publicition="2015";

B1.getBook();

System.out.println("\n");

Book B2=new Book();

B2.Title="Harry poter";

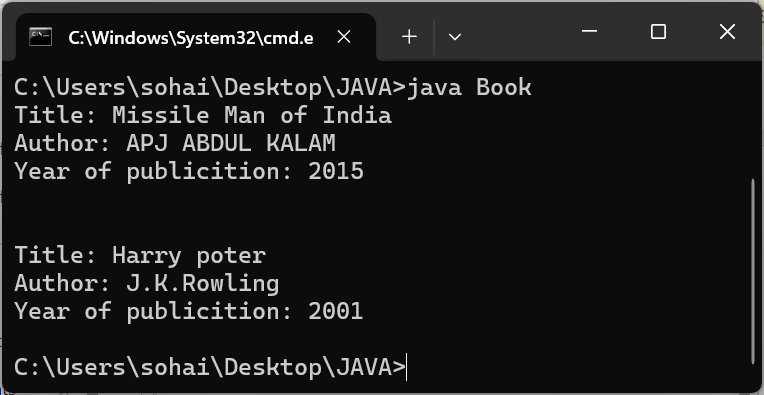
B2.Author="J.K.Rowling";

B2.Year\_of\_publicition="2001";

B2.getBook();

}

}

**Output:** 

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Closing parentices |

**Class Diagram:**

|  |
| --- |
| Book |
| -title: String  -author: String  -year: int |
| + Book(title: String, author:String, year: int) + displayDetails(): void |

**Important Points:**

1. **Constructor**:

* The constructor Book(String, String, int) is used to initialize the object when it is created.
* The keyword **this** is used to differentiate between class attributes and constructor parameters.

2.**Method**:

* The method displayDetails() is used to display the book details.
* The **System.out.println()** method prints the details to the console.

3. **Object Creation**:

* Two objects b1 and b2 are created using the constructor.

**2.Aim:** Write a java program with class named “myclass” with a static variable count of int type. Intialize it to zero and a constant variable “pi” of type double initialized to “3.14” as attributes of that class. Now define a constructor for “myclass”, that increments the count variable each time an object of “myclass” is created. Finally, print the final values of ‘count’ and ‘pi’ variables and create 3 objects.

**Programme:**

class My\_Class{

static int num=0;

final double pi=3.1415;

My\_Class(){

num++;

}

public static void main(String[] args){

My\_Class C1=new My\_Class();

My\_Class C2=new My\_Class();

My\_Class C3=new My\_Class();

My\_Class C4=new My\_Class();

My\_Class C5=new My\_Class();

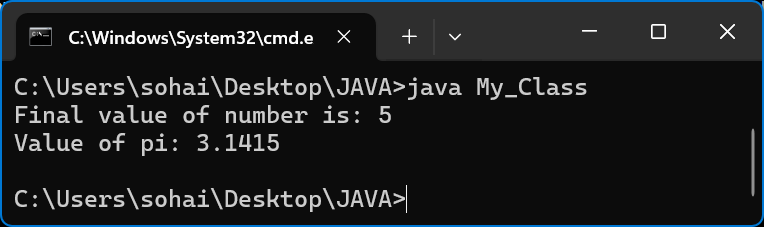
System.out.println("Final value of number is: "+num);

System.out.println("Value of pi: "+C1.pi);

}

}

**Output:**

****

**Negative case:**

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | No class name declared | Class created My\_Class |
| 2. | Syntax error | Key word new added |

**Class Diagram:**

|  |
| --- |
| MyClass |
| -count: int (static)  -pi: double (static, final) |
| +MyClass()  +main(args: String[]):void |

**Important points:**

**1.Static Keyword**

* Static members belong to the class, not to individual objects.
* Only one copy of the static variable is maintained for all objects.

**2.Static Variable**

* **static int count**:
  + Shared among all objects of the class.
  + It is initialized only once and not for every object.
  + It increments every time the constructor is called.

**3.Final Variable**

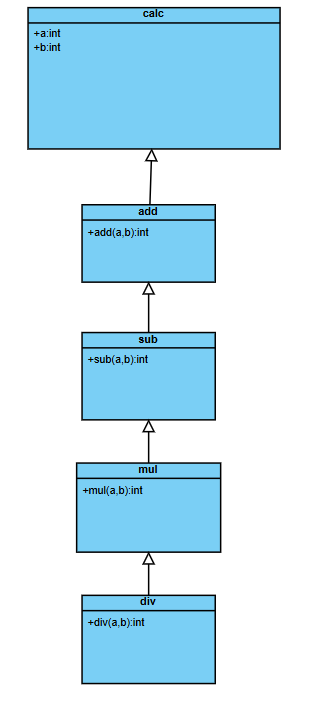
* **static final double pi**:
  + The final keyword makes the variable constant.
  + Its value cannot be changed once assigned.
  + It must be initialized at the time of declaration.

**WEEK-5**

**Aim:** Create a calculator using the operations including addition, subtraction, multiplication,

And division using multilevel inheritance and display the desiredoutput.

**Class Diagram:**

****

**Programme:**

import java.util.\*;

class Simplecaluclator{

public void add(float a, float b){

System.out.println("Addition : "+(a+b));

}

public void difference(float a, float b){

System.out.println("Subtraction : "+(a-b));

}

}

class supercaluclator extends Simplecaluclator{

public void multipication(float a, float b){

System.out.println("Multipicatio : "+(a\*b));

}

}

class superiorcaluclarot extends supercaluclator{

public void division(float a, float b){

if(b!=0){

System.out.println("Divison : "+(a/b));

}

else{

System.out.println("Divion cannot br done when denominator is zero");

}

}

}

class Final\_Caluclator {

public static void main(String[] args) {

Scanner scr=new Scanner(System.in);

System.out.println("Enter the two numbers : ");

float a=scr.nextFloat();

float b=scr.nextFloat();

superiorcaluclarot obj=new superiorcaluclarot();

obj.add(a,b);

obj.difference(a,b);

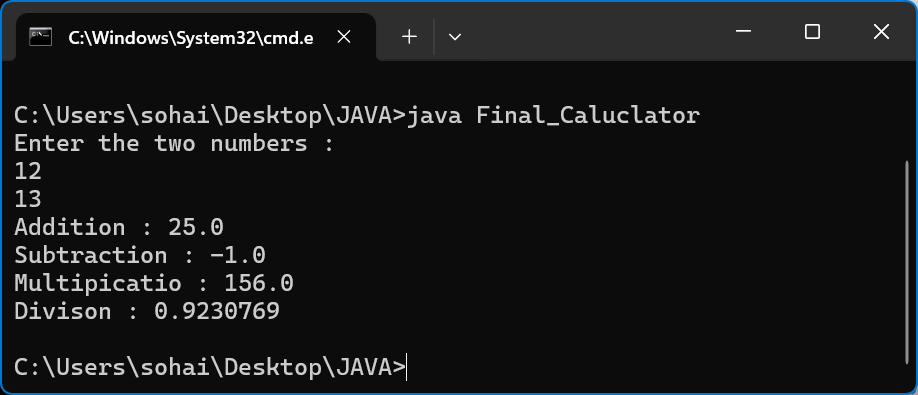
obj.multipication(a,b);

obj.division(a,b);

}

}

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Placing ;(semi colon) |

**Important Points:**

1. Multi-Level Inheritance
   * Classes form a chain: addition → subtraction → multiplication → division.
   * Each child inherits all methods from its parent.
2. Incremental Method Addition
   * Each class adds one new arithmetic operation (add, sub, mul, div).
3. Single Object Access to All Methods
   * A division object can perform all operations (add, sub, mul, div) due to inheritance.
4. Integer Division Limitation
   * div(4, 8) returns 0 (truncated). Should use double for accurate results.
5. Poor Real-World Design
   * Misuse of inheritance (operations aren’t hierarchical). Better to use a single class or interfaces

2.

**Aim:**A vehicle rental company wants to devlop a system that maintains information about different types of vehicles available for rent. The company rents out cars , bikes and trucks and they need a program to store details about each vehicle, such as brand and speed.

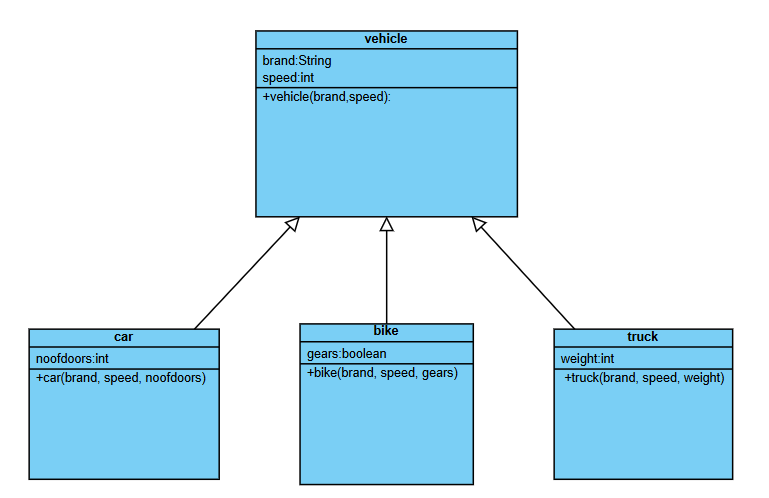
=>Cars should have an additional property such as number\_of\_doors .

=>Bikes should have a property indicating whether they have gears or not.

=>Truck should have a property of their capacity(in tons).

=>Every class should have a constructor.

**Class Diagram:**



**Programme:**

class TeesukoRentals{

}

class Bikes extends TeesukoRentals{

Bikes(){}

public void vehicle(String Brand,String Model,int Speed){

System.out.println("Brand : "+Brand);

System.out.println("Model : "+Model);

System.out.println("Speed : "+Speed);

}

}

class Cars extends TeesukoRentals{

Cars(){}

public void vehicle(String Brand,String Model,int Speed,int Doors){

System.out.println("Brand : "+Brand);

System.out.println("Model : "+Model);

System.out.println("Speed : "+Speed);

System.out.println("Doors : "+Doors);

}

}

class Truck extends TeesukoRentals{

Truck(){}

public void vehicle(String Brand,String Model,int Speed,double Load){

System.out.println("Brand : "+Brand);

System.out.println("Model : "+Model);

System.out.println("Speed : "+Speed);

System.out.println("Load : "+Load);

}}

class TeesukoRentalsMain{

public static void main(String[] args) {

Bikes bike=new Bikes();

bike.vehicle("Yamaha","FZ",120);

Bikes bike1=new Bikes();

bike1.vehicle("Hero Honda", "CD Dulex" , 100);

Bikes bike2=new Bikes();

bike2.vehicle("Royal Enfield","GT650",165);

Cars car=new Cars();

car.vehicle("Hyundai","Verna",180,4);

Cars car1=new Cars();

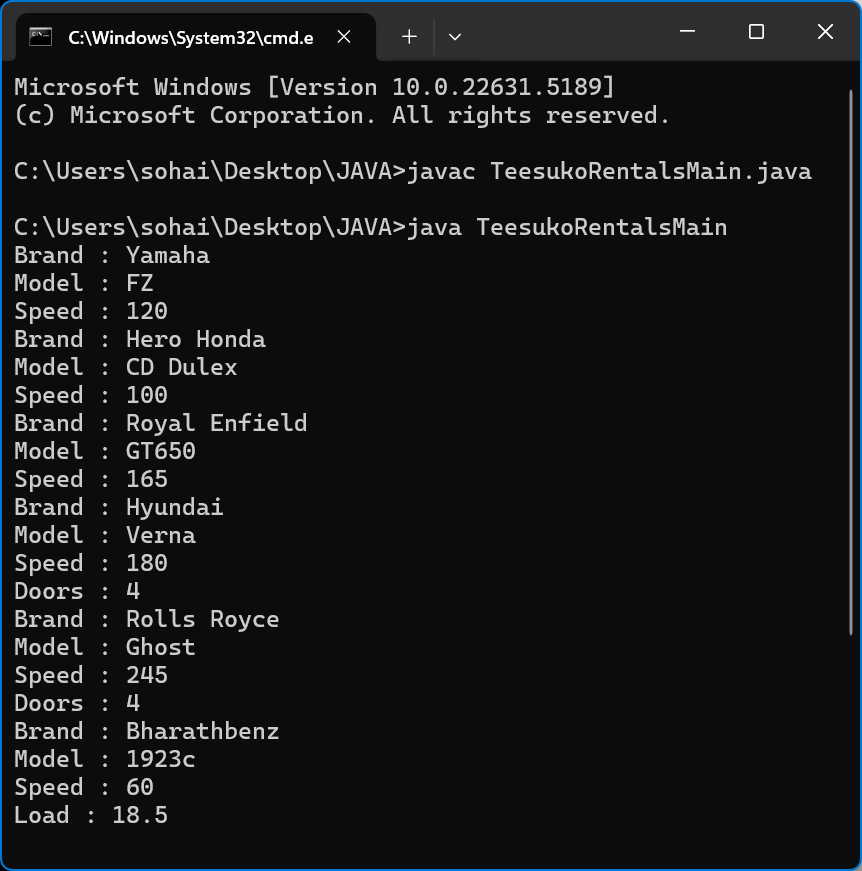
car1.vehicle("Rolls Royce","Ghost",245,4);

Truck truck=new Truck();

truck.vehicle("Bharathbenz", "1923c", 60, 18.5);

}

}

**Output: **

**Errors:**

|  |  |  |
| --- | --- | --- |
| Si.no | Error name | Error rectification |
| 1 | Identifier error | Changing ststic to static |
| 2 | Error in main function | Changing arhs to args in print statment |

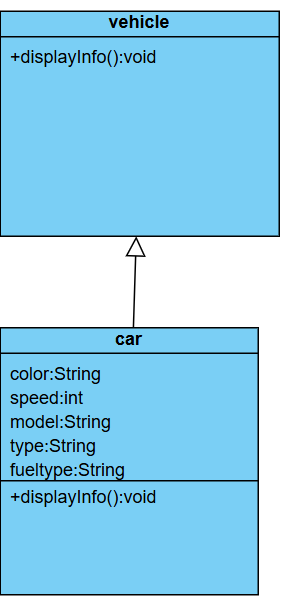
**Important points:**

1. Vehicle is the parent/super class
2. Car, Bike, and Truck are child/subclasses that inherit from Vehicle
3. Each class shows:
   * Class name at the top
   * Attributes in the middle section (with visibility: - for private)
   * Constructors in the bottom section (with visibility: + for public)
4. The inheritance relationship is shown with arrows from child to parent class
5. Each subclass adds its own specific attributes:
   * Car adds noofdoors

**WEEK-6**

**Aim:** Write a java program to create a Vehicle class with a method displayInfo().Override this method in the car subclass to provide specific information about a car.

**Class Diagram:**



**Programme:**

class vehicle1{

    public void displayInfo(){

        System.out.println("This is a vehicle.");

    }

}

class car1 extends vehicle1{

    String color = "red";

    int speed = 120;

    String model = "Toyota";

    String type = "SUV";

    String fuelType = "Petrol";

    public void displayInfo(){

        System.out.println("This is a car.");

        System.out.println("Color: " + color);

        System.out.println("Speed: " + speed + " km/h");

        System.out.println("Model: " + model);

        System.out.println("Type: " + type);

    }

}

class Override{

    public static void main(String[] args) {

        System.out.println("S.Sohail 24319 CSE-B");

        vehicle1 v = new vehicle1();

        v.displayInfo(); // Calls the method from the vehicle(super) class

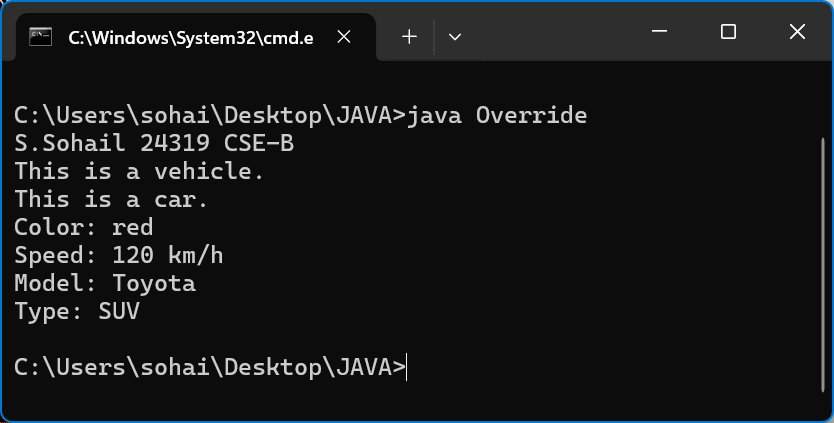
        car1 c = new car1();

        c.displayInfo(); // Calls the overridden method from the car(sub) class

    }

}

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | No class name declared | Class created Override |
| 2. | Syntax error | Key word new added |

**Important points:**

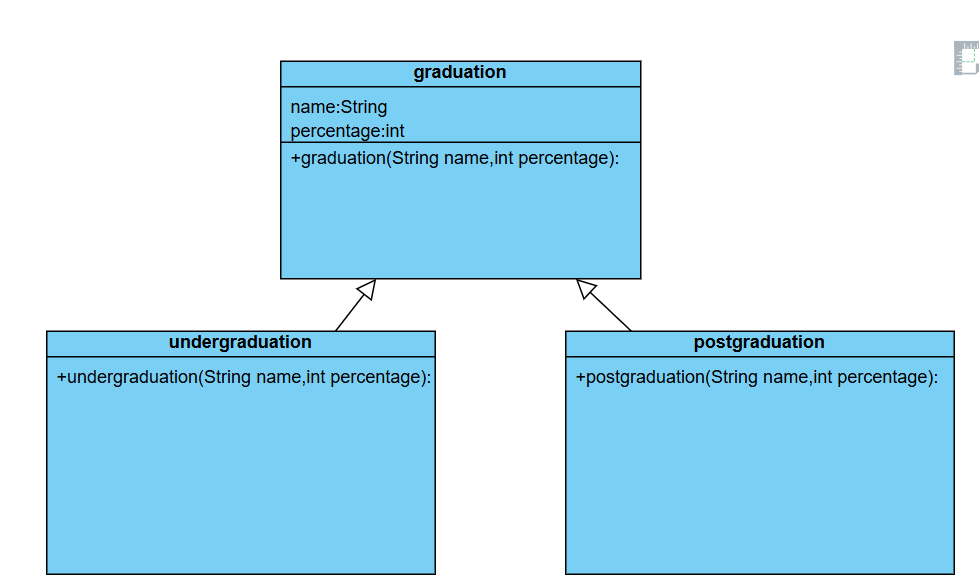
1. =>Here class vehicle and car share a single level inheritance between them.
2. =>Here in the above code is an example of method overriding in java where the method names and their signature are same in both super class and sub class.

**Aim:** A college is developing an automated admission system that verifies student’s eligibility for Undergraduate(UG) and Postgraduate(PG) programs. Each program has different eligibility criteria based on the student’s percentage in their previous qualification.

CONDITION:

1.UG admission require minimum of 60%.

2.PG admission require minimum of 70%.

**Class Daigram:** 

**Programme:**

class student{

    String name;

    double percentage;

    public student(String name,double percentage){

        this.name=name;

        this.percentage=percentage;

    }

    void eligibility(){

        System.out.println(name+" must have the following criteria");

    }

}

class UG extends student{

    UG(String name,double percentage){

        super(name,percentage);

    }

    void eligibility(){

        if(percentage>=60){

            System.out.println(name+" is eligible for ug");

        }

        else{

            System.out.println(name+"is not eligible for ug");

        }

    }

}

class PG extends student{

    PG(String name,double percentage){

        super(name,percentage);

     }

     void eligibility(){

        if(percentage>=70.0){

            System.out.println(name+" is not eligible for pg");

            }

            else{

                System.out.println(name+" is  eligible for pg");

     }

            }

}

class collegeadmission{

    public static void main(String[] args){

        System.out.println("S.Sohail 24319 CSE-B");

        student s1=new student("Ravi",80.0);

        s1.eligibility();

        student s2=new student("Raju",50.0);

        s2.eligibility();

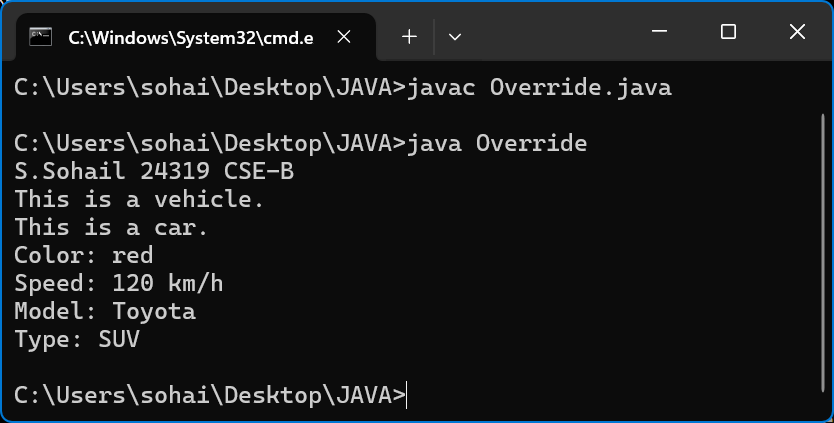
        PG s=new PG("ramya",70.0);

        s.eligibility();

        UG u=new UG("mahesh",60.0);

        u.eligibility();

    }}

**Output: **

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Closing parentices |

**Important Points:**

1. Here classes undergraduation and postgraduation share a Hierarchical inheritance with graduation class.
2. For each class there is a respective constructor used for the values name and percentage
3. super(name, percentage); Call the constructor of the superclass (graduation)

**3**

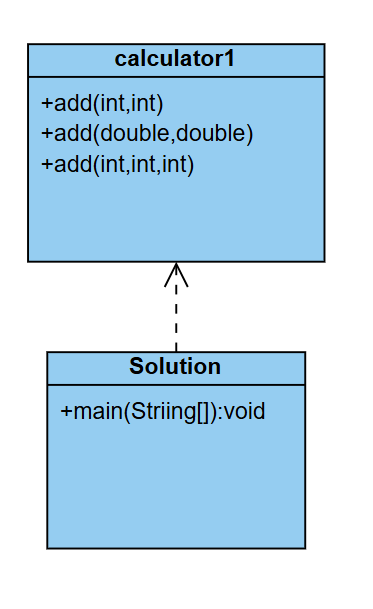
**Aim:** Create a calculator class with overloaded methods to perform addition:

1.add two integers

2.add 2 doubles

3.add 3 integers

**Class Diagram:**



**Programme:**

class calculator1{

    public void add(int num1, int num2){

        int num=num1+num2;

        System.out.println("sum of two integers is "+num);

    }

    public void add(double num1,double num2){

        double num=num1 +num2;

        System.out.println("sum of two floating integers  is "+num);

    }

    public void add(int num1,int num2,int num3){

        int num = num1+num2+num3;

        System.out.println("sum of three integers is "+num);

    }

}

public class Solution{

    public static void main(String[] args){

        System.out.println("S.Sohail,24319,CSE-B");

        calculator1 calculator= new calculator1();

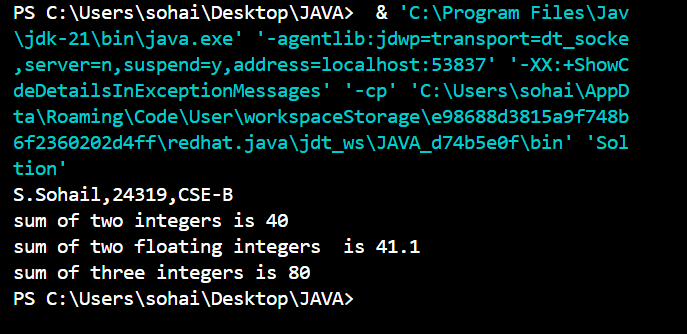
        calculator.add(10,30);

        calculator.add(10.5,30.6);

        calculator.add(10,30,40);

    }

}

**Output: **

**Errors:**

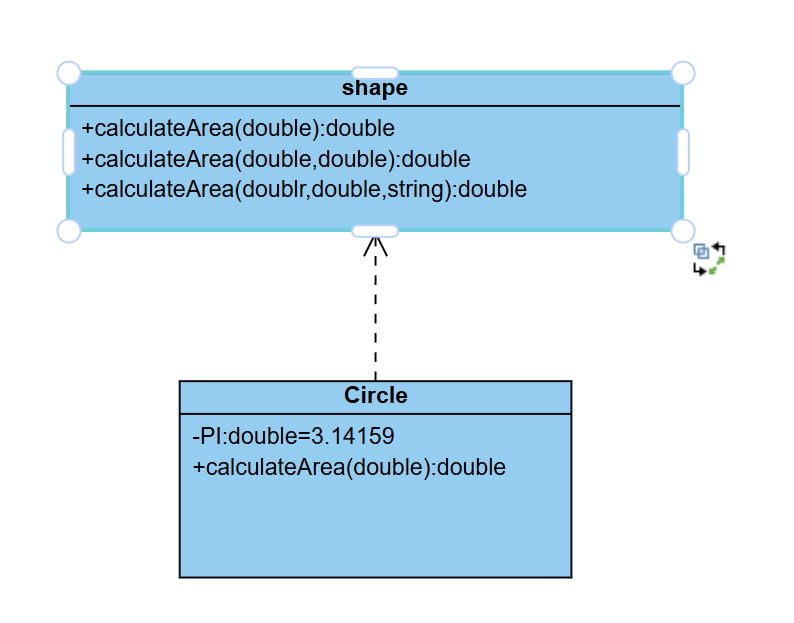
|  |  |  |
| --- | --- | --- |
| Si.no | Error name | Error correction |
| 1 | Syntax error | Placing ; at end of print statment |

**Important Points:**

1. The calculator1 class demonstrates method overloading by defining
2. multiple add() methods with the same name but different parameters.
3. Number of parameters (add(int, int) vs add(int, int, int))
4. Data types of parameters (add(int, int) vs add(double, double))
5. add(int num1, int num2) → Adds two integers.
6. add(double num1, double num2) → Adds two floating-point numbers.
7. add(int num1, int num2, int num3) → Adds three integers.
8. Each method prints a different message based on the type and number of arguments.

4.**Aim:** Create a shape class with a method “calculateArea()” that is overloaded for different shapes(ex:square,rectangle,triangle).then create a subclass “Circle” that overrides the calcArea() method for a circle.

**Class Diagram:**



**Programme:**

class shape {

     public double calculateArea(double side) {

        return side \* side;

    }

  public double calculateArea(double length, double width) {

        return length \* width;

    }

    public double calculateArea(double base, double height, String shape) {

        if (shape.equalsIgnoreCase("triangle")) {

            return 0.5 \* base \* height;

        }

        return 0;

    }

}

class Circle extends shape {

    private static final double PI = 3.14159;

     public double calculateArea(double radius) {

        return PI \* radius \* radius;

    }

}

public class AreaCalculator {

    public static void main(String[] args) {

        System.out.println("S.Sohail,24319,CSE-B");

        shape shape = new shape();

        Circle circle = new Circle();

        System.out.println("Area of square (side=5): " + shape.calculateArea(5));

        System.out.println("Area of rectangle (4x6): " + shape.calculateArea(4, 6));

        System.out.println("Area of triangle (base=3, height=4): " +

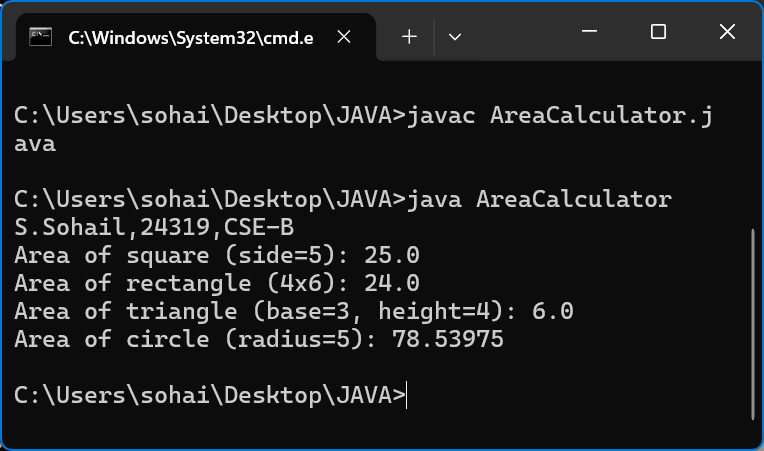
                          shape.calculateArea(3, 4, "triangle"));

        System.out.println("Area of circle (radius=5): " + circle.calculateArea(5));

    }

}

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Placing ;(semi colon) |

**Important Points:**

* The shape class demonstrates method overloading with three calculateArea() methods:
  1. calculateArea(double side) → Computes the area of a square (side \* side).
  2. calculateArea(double length, double width) → Computes the area of a rectangle (length \* width).
  3. calculateArea(double base, double height, String shape) → Computes the area of a triangle (0.5 \* base \* height).
* The Circle class inherits from shape and overrides calculateArea() to compute the area of a circle (π \* radius²).
* Uses a private static final constant PI for the value of π.
* The Circle class overrides the calculateArea() method to provide its own implementation.

WEEK-7

!.**Aim:** Write a java programme to create an abstract class Animal with an abstract method called sound.creaye a subclass lion and tiger which extends Animal class and implement the sound method to make a spefic sound for each animal.

**Class Diagram:**

**Programme:**

abstract class Animal{

    abstract void sound(); // Abstract method

}

class Tiger extends Animal{

    void sound(){ // Implementation abstract method named sound

        System.out.println("Roar");

    }

}

class Lion extends Animal{

    void sound(){

        System.out.println("Roar");

    }

    }

class liontiger{

    public static void main(String[] args) {

        System.out.println("S.Sohail, 24319, CSE-B");

        Tiger tiger1=new Tiger();

        tiger1.sound();

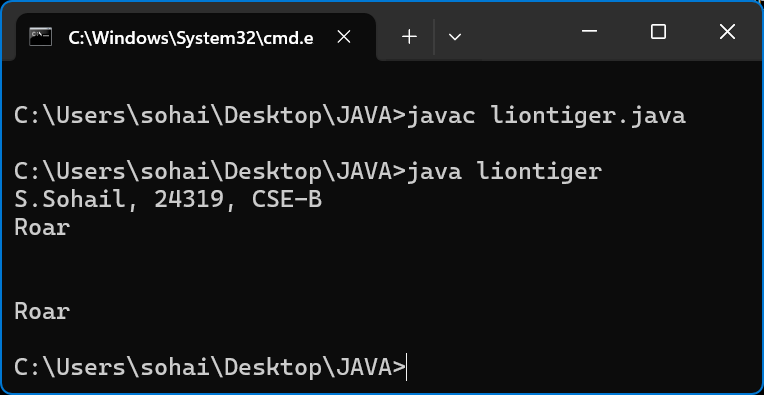
        System.out.println("\n");

        Lion lion1=new Lion();

        lion1.sound();

    }

}

**Output: **

**Errors:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Closing perentasis |

**Important Points:**

1. **Abstract Class Usage:**

Animal is declared as an **abstract class**.

It contains an **abstract method** sound() that must be implemented by subclasses.

1. **Method Overriding in Subclasses:**

Both Tiger and Lion **extend** Animal and **override** the sound() method to print "Roar".

1. **Polymorphism Concept:**

This demonstrates **runtime polymorphism** where the sound() method behaves differently depending on the object (Tiger or Lion).

**Aim:** Write a java programme and implement respective methods to caluclate each to create abstract class Shape 3D with abstract methods Caluclate volume and surface area of a sphere and cube that extends the class Shape3D

**Class Diagram:**

**Programme:**

import java.util.Scanner;

abstract class Shape3D {

    abstract void calculateVolume();

    abstract void calculateSurfaceArea();

}

class Sphere extends Shape3D {

    double radius;

    Sphere(double radius) {

        this.radius = radius;

    }

    void calculateVolume() {

        double volume = (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

        System.out.println("Sphere Volume: " + volume);

    }

    void calculateSurfaceArea() {

        double surfaceArea = 4 \* Math.PI \* Math.pow(radius, 2);

        System.out.println("Sphere Surface Area: " + surfaceArea);

    }

}

class Cube extends Shape3D {

    double side;

    Cube(double side) {

        this.side = side;

    }

    void calculateVolume() {

        double volume = Math.pow(side, 3);

        System.out.println("Cube Volume: " + volume);

    }

    void calculateSurfaceArea() {

        double surfaceArea = 6 \* Math.pow(side, 2);

        System.out.println("Cube Surface Area: " + surfaceArea);

    }

}

public class MainShape {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("S.Sohail, 24319, CSE-B");

        System.out.print("Enter radius of the sphere: ");

        double radius = scanner.nextDouble();

        Sphere sphere = new Sphere(radius);

        sphere.calculateVolume();

        sphere.calculateSurfaceArea();

        System.out.println();

        System.out.print("Enter side length of the cube: ");

        double side = scanner.nextDouble();

        Cube cube = new Cube(side);

        cube.calculateVolume();

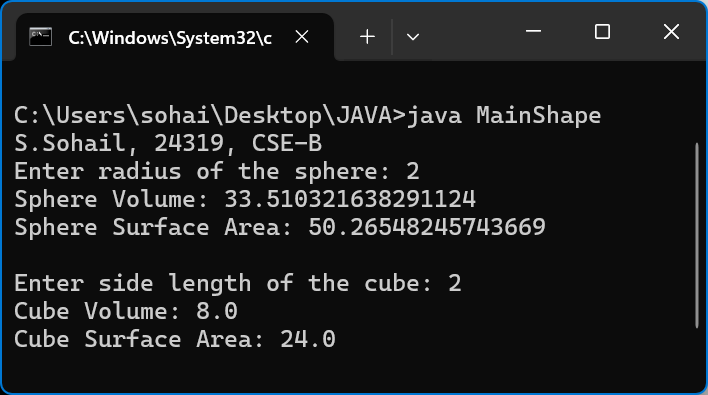
        cube.calculateSurfaceArea();

        scanner.close();

    }

}

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Logical Eror | Clrrection formula |

**Important Ponts:**

1. **Abstract Class Shape3D:**
   * Declares two abstract methods:
     + calculateVolume()
     + calculateSurfaceArea()
   * Forces all subclasses to implement these methods.
2. **Subclasses Sphere and Cube:**
   * Both extend Shape3D and provide their own implementations for volume and surface area calculations.
   * Formulas used:
     + **Sphere**
       - Volume = 43πr3\frac{4}{3} \pi r^334​πr3
       - Surface Area = 4πr24 \pi r^24πr2
     + **Cube**
       - Volume = a3a^3a3
       - Surface Area = 6a26a^26a2
3. **Main Method (MainShape Class):**
   * Takes user input for radius and side.
   * Creates Sphere and Cube objects.
   * Calls the respective methods to display results.

**Aim:** Write a java programme using an abstract class to define a method for pattern printing.

Create abstract class named printpattern with an abstract method printpattern(int n)

Create two sub classes

1.StarPattern(Right angled triangle of \*)

2.numberPattern(Right angled triangle of increasing numbers)

In main create objects of both subclasses and the [attern for a given number of rows.

**Class Diagram:**

**Programme:**

import java.util.\*;

// Abstract class

abstract class PatternPrinter {

    abstract void printpattern(int n);

}

// Star pattern class

class StarPattern extends PatternPrinter {

    @Override

    void printpattern(int n) {

        for (int i = 1; i <= n; i++) {

            for (int j = 1; j <= i; j++) {

                System.out.print(" \* ");

            }

            System.out.println();

        }

    }

}

// Number pattern class

class NumberPattern extends PatternPrinter {

    @Override

    void printpattern(int n) {

        for (int i = 1; i <= n; i++) {

            for (int j = 1; j <= i; j++) {

                System.out.print(j + " ");

            }

            System.out.println();

        }

    }

}

// Main class

public class Main {

    public static void main(String[] args) {

        Scanner scn = new Scanner(System.in);

        System.out.println("S.Sohail, 24319, CSE-B");

        System.out.print("Enter number of rows for the pattern: ");

        int n = scn.nextInt();

        System.out.println("\nStar Pattern:");

        PatternPrinter star = new StarPattern();

        star.printpattern(n);

        System.out.println("\nNumber Pattern:");

        PatternPrinter number = new NumberPattern();

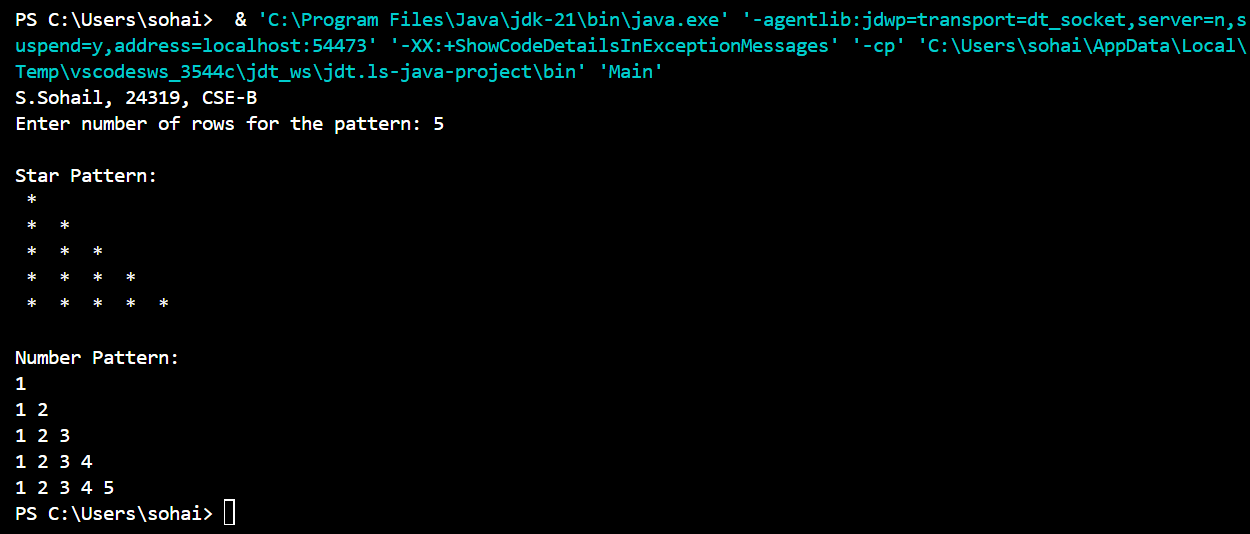
        number.printpattern(n);

        scn.close();

    }

}

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| SI.NO | Error Name | Error rectification |
| 1 | Syntax error | Placed static befour class number pattern |

**Important Points:**

**Abstract Class Usage:**

PatternPrinter is an abstract class that defines an abstract method printpattern(int n).

Enforces that all subclasses must provide their own implementation of the method.

**Subclass Implementation:**

StarPattern prints a right-angled triangle using \*.

NumberPattern prints a right-angled triangle using increasing numbers.

**Polymorphism in Action:**

Objects of subclasses (StarPattern, NumberPattern) are used to invoke the overridden method.

Demonstrates **runtime polymorphism** and abstraction together.