**23CSE111**

**Object Oriented Programming**



**Department of Computer Science Engineering**

**Amrita School of Engineering**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: Bhargavi.CH**

**Verified By : Roll No: AV.SC.CSE24101**

**JAVA:**

Java is a high level, class based, object oriented programming language that is widely used across various operating systems.

**Installation of JAVA in Computer:**

**Aim:**

Solving computational programming using JAVA.

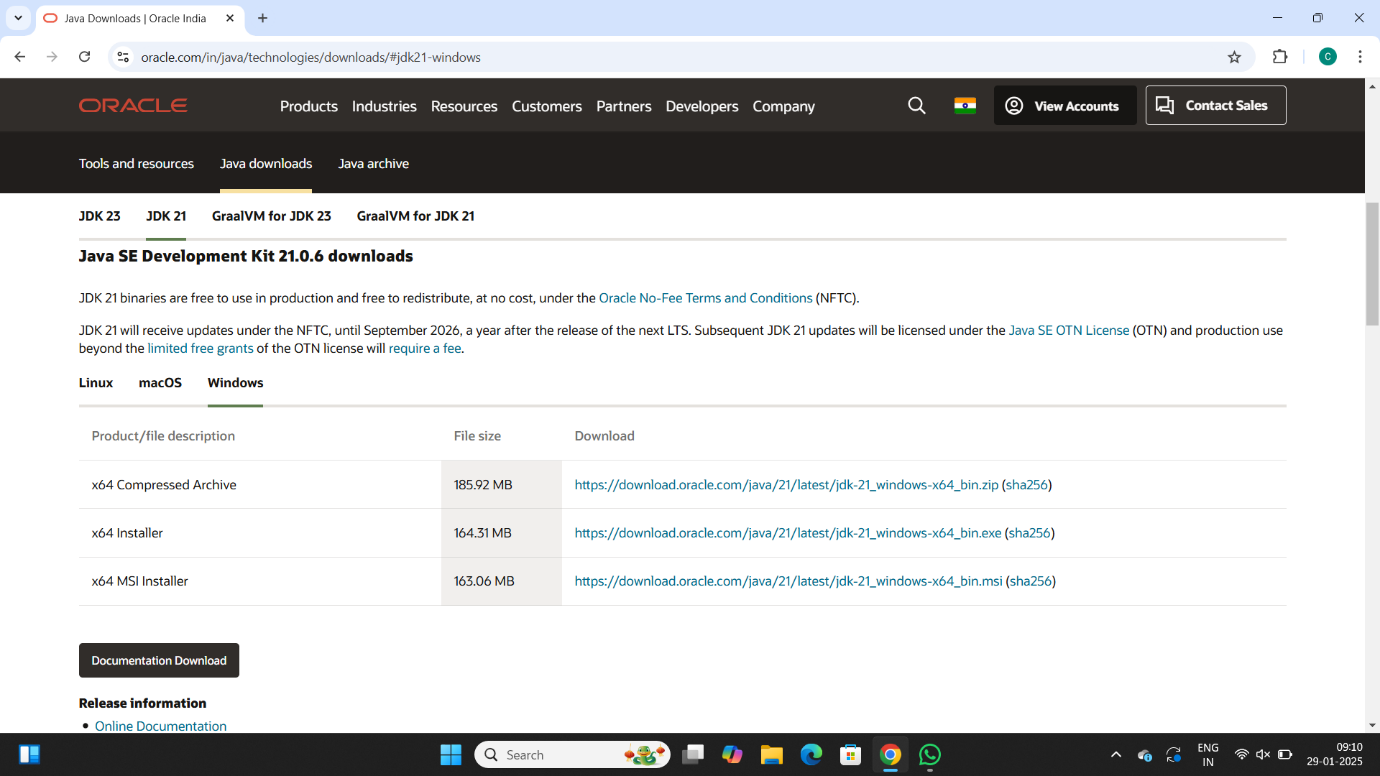
**Procedure:**

Follow the steps below to install Java on Windows:

1. Download JDK(Java Development Kit)
2. Run the Installer
3. Configure Environment Variables
4. Verify Installation

1.Download JDK:

Go to the official oracle website to download the JDK. Choose x64 MSI Installer on the windows tab and click on download link.



2. Run the Installer:

Now, go to your downloads folder and run the installer you just downloaded.

3. Configure Environment Variables:

After installation, you will need to tell your system where to find Java. This is done by setting environment variables.

Locate JDK Path:

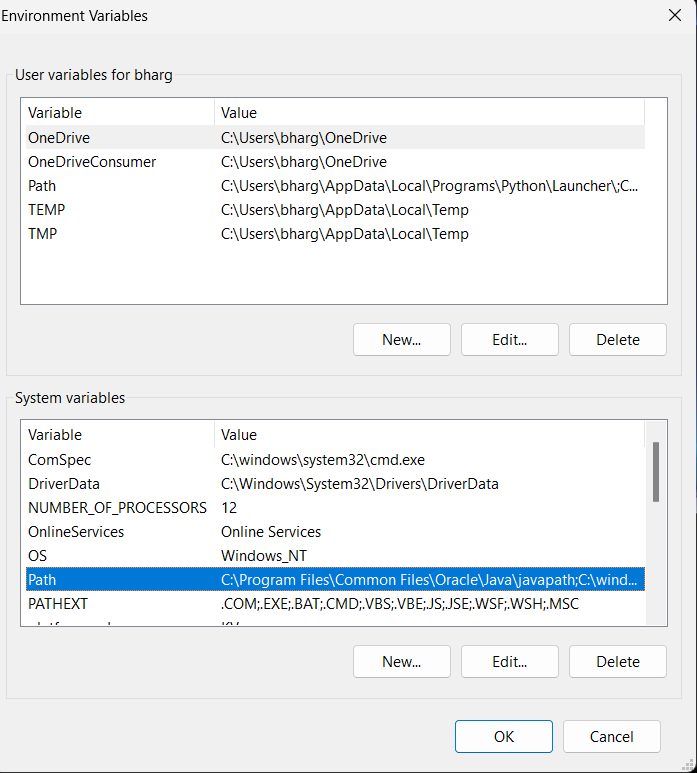
Navigate through your file explorer to reach the JDK installation directory. Normally, it is located at -

C:\Program Files\Java\jdk-22\bin

Copy this path.

Access Environment Variables:

Search environment variable on the terminal. In system properties, click on environment variables. You will be prompted to the screen below.



Update the Path Variable:

Find the Path variable in the System variables section and click on edit. Then, click New and paste your JDK bin path.

Finally, click Ok to close each window.

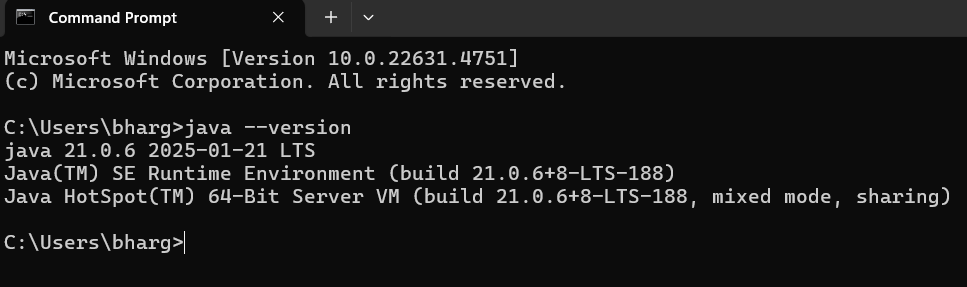
Set JAVA\_HOME Variable:

Back in the environment variables window, under the system variables section, click New to create a new variable.

Now, name the variable  and set its value to the path of the JDK folder directory. Close all the dialogues with the Ok button.

1. Verify Installation:

After the installation, you can verify whether Java is installed by using the following command in the command prompt.



If Java is installed successfully, it will print the version information; otherwise, it will produce an error message indicating that the command is not recognized.

**JAVA Program:**

**INPUT:**

Class program{

Public static void main(string[] args){

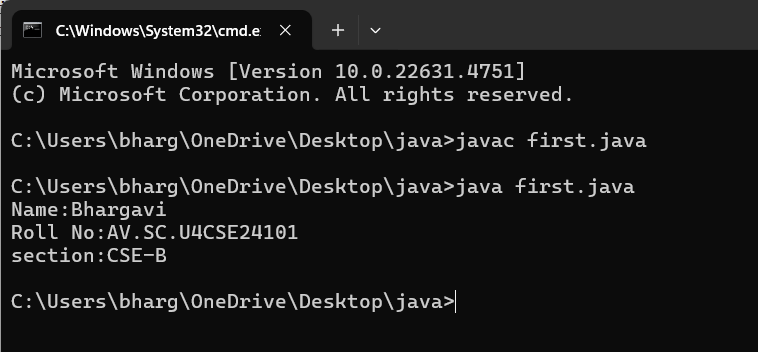
System.out.println(“Name:Bhargavi.ch”);

System.out.println(“Section:CSE-B”);

System.out.println(“Roll No:AV.SC.U4CSE24101”);

}

}



**WEEK 02**

**PROGRAM-1:**

**AIM:** Write a Java program to calculate area of rectangle.

**CODE:**

import java.util.Scanner;

class Area {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter length: ");

float l = input.nextFloat();

System.out.print("Enter width: ");

float b = input.nextFloat();

input.close();

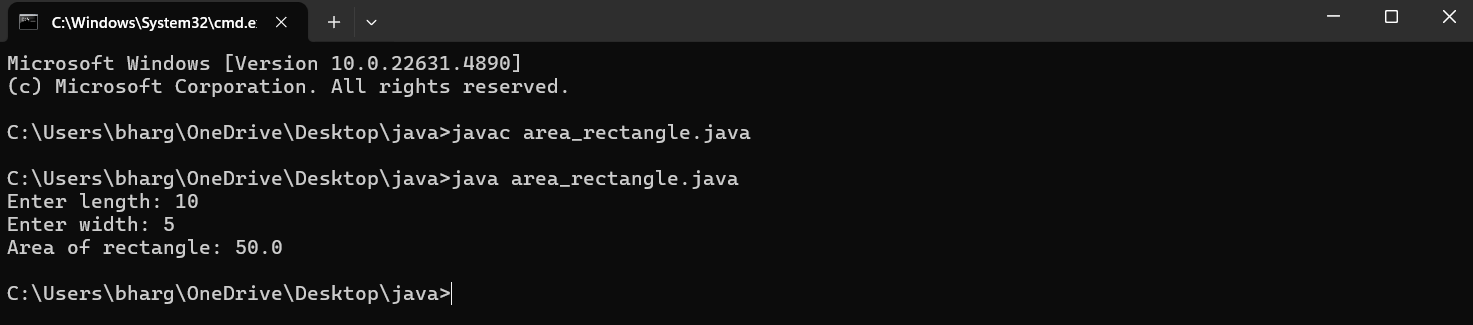
float area = l \* b;

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

}

}

**OUTPUT:**



**PROGRAM-2:**

**AIM:** Write a Java program to convert temperature from Fahrenheit to Celsius and vice versa.

**CODE:**

import java.util.Scanner;

class Temperature {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

float F = input.nextFloat();

input.close();

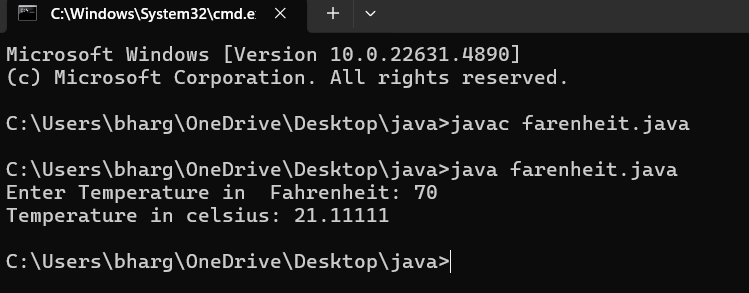
float C = (F - 32)\*5/9;

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

}

}

**OUTPUT:**



**CODE:**

import java.util.Scanner;

class temperature2 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

float C = input.nextFloat();

input.close();

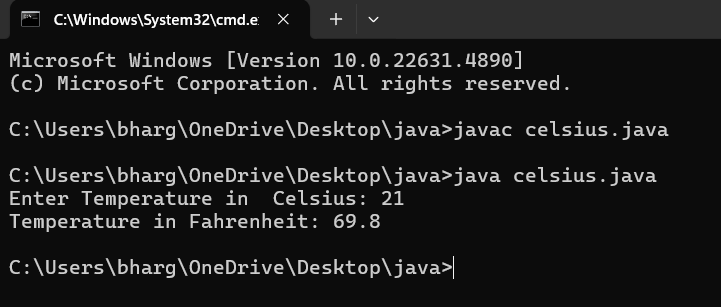
float F = (C \* 9/5) + 32;

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

}

}

**OUTPUT:**

****

**PROGRAM-3:**

**AIM:** Write a Java program to calculate simple intrest.

**CODE:**

import java.util.Scanner;

class simple\_interest {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter principle: ");

float P = input.nextFloat();

System.out.print("Enter time: ");

float T = input.nextFloat();

System.out.print("Enter rate: ");

float R = input.nextFloat();

input.close();

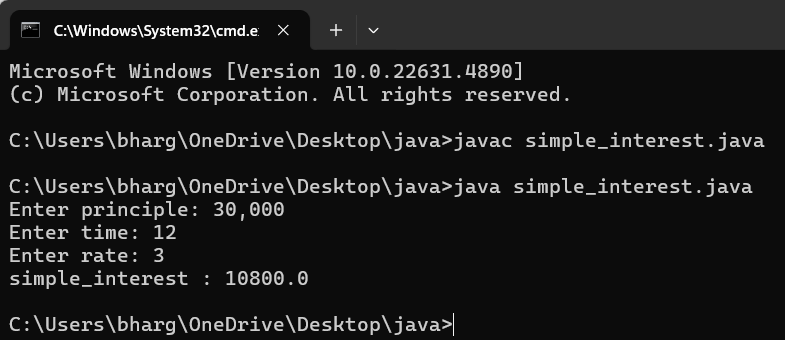
float SI = (P\*T\*R)/100;

System.out.println("simple\_interest : " + SI );

}

}

**OUTPUT:**

****

**PROGRAM-4:**

**AIM:** Write a Java program to calculate largest of 3 numbers using ternary operators.

**CODE:**

import java.util.Scanner;

class largest\_num {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter number1: ");

float n1 = input.nextFloat();

System.out.print("Enter number2: ");

float n2 = input.nextFloat();

System.out.print("Enter number3: ");

float n3 = input.nextFloat();

input.close();

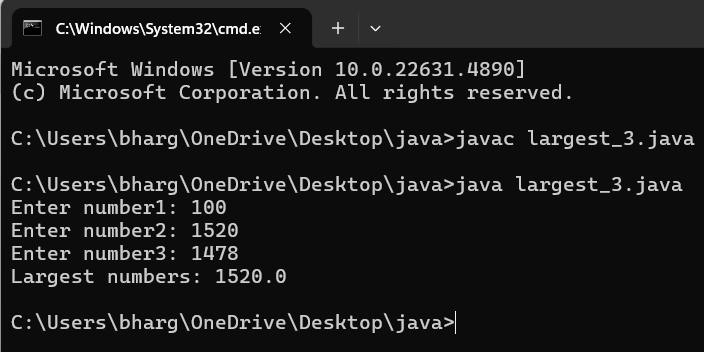
float largest = (n1 >= n2) ? ((n1 >= n3) ? n1 : n3) : ((n2 >= n3) ? n2 : n3);

System.out.println("Largest numbers: " +largest);

}

}

**OUTPUT:**

****

**PROGRAM-5:**

**AIM:** Write a Java program to calculate factorial of a number.

**CODE:**

import java.util.Scanner;

class factorial {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

input.close();

long factorial = 1;

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

factorial \*= i;

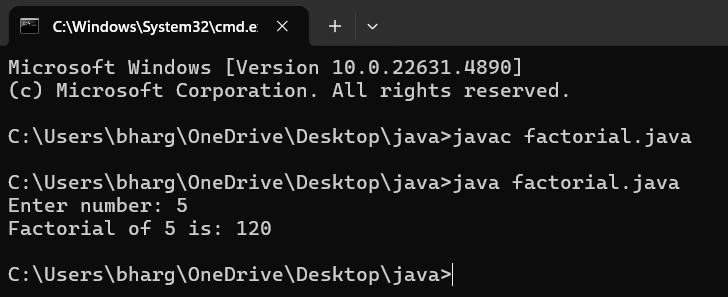
}

System.out.println("Factorial of " + n + " is: " + factorial);

}

}

**OUTPUT:**



**WEEK – 3:**

**PROGRAM- 1:**

**AIM:** To create a class BankAccount with methods deposit() and withdraw() . create two subclasses savingsaccount and checkingaccount override the withdraw () method in each subclass to impose different withdrawal limits and fees.

**CODE:**

public class Bankaccount {

public String accountHolder;

public double balance;

public int accountNumber;

public Bankaccount(String accountHolder, int accountNumber, double balance) {

this.accountHolder = accountHolder;

this.accountNumber = accountNumber;

this.balance = balance;

}

public void withdrawal(double amount) {

if (amount <= balance) {

balance -= amount;

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

} else {

System.out.println("Invalid withdrawal amount");

}

}

public void deposit(double amount) {

balance += amount;

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

}

public static void main(String[] args) {

Bankaccount ba = new Bankaccount("Ch.Bhargavi", 24151, 1000);

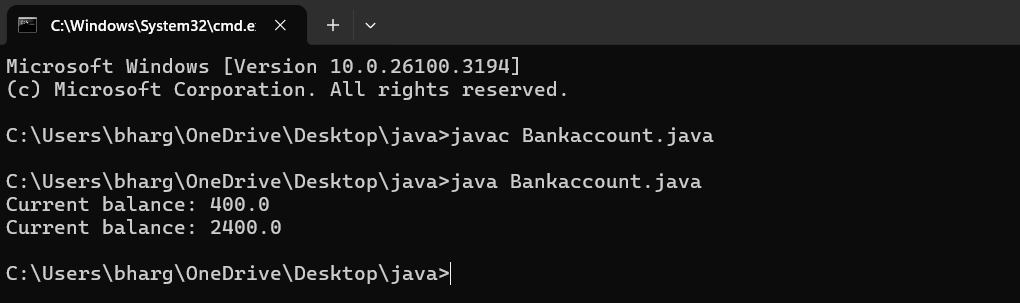
ba.withdrawal(600);

ba.deposit(2000);

}

}

Output:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Withdrawal,deposit not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

IMPORTANT POINTS:

1. The condition inside the if statement must be correct.
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount.

**Class diagram:**

|  |
| --- |
| **BankAccount**  **----------------------------------------------------------**  **-balance: double**  **----------------------------------------------------------**  **+BankAccount(intialBalance: double)**  **+deposit(amount: double):void**  **+withdraw(amount: double):void** |

**PROGRAM- 2:**

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

1. Create a class with name Car

2. Create four attributes named car\_color, car\_brand, fuel\_type, mileage

3. Create these methods named start(),stop(),service()

4. Create the objects named car, car1,car2.

**CODE:**

public class car {

private String car\_color;

private String car\_brand;

private String fuel\_type;

private String mileage;

public void start() {

System.out.println("car is started");

}

public void stop() {

System.out.println("car is stopped");

}

public void service() {

System.out.println("car is for service");

}

public static void main(String args[]) {

car car = new car();

car.car\_color = "blue";

car.car\_brand = "wolkswagen";

car.fuel\_type = "petrol";

car.mileage = "40";

car.start();

System.out.println("car\_color: " + car.car\_color + " car\_brand: " + car.car\_brand + " fuel\_type: " + car.fuel\_type + " mileage: " + car.mileage);

car car\_one = new car();

car\_one.car\_color = "grey";

car\_one.car\_brand = "maruti suzuki";

car\_one.fuel\_type = "petrol";

car\_one.mileage = "75";

car\_one.stop();

System.out.println("car\_color: " + car\_one.car\_color + " car\_brand: " + car\_one.car\_brand + " fuel\_type: " + car\_one.fuel\_type + " mileage: " + car\_one.mileage);

car car\_two = new car();

car\_two.car\_color = "white";

car\_two.car\_brand = "Mercedes benz";

car\_two.fuel\_type = "diesel";

car\_two.mileage = "45";

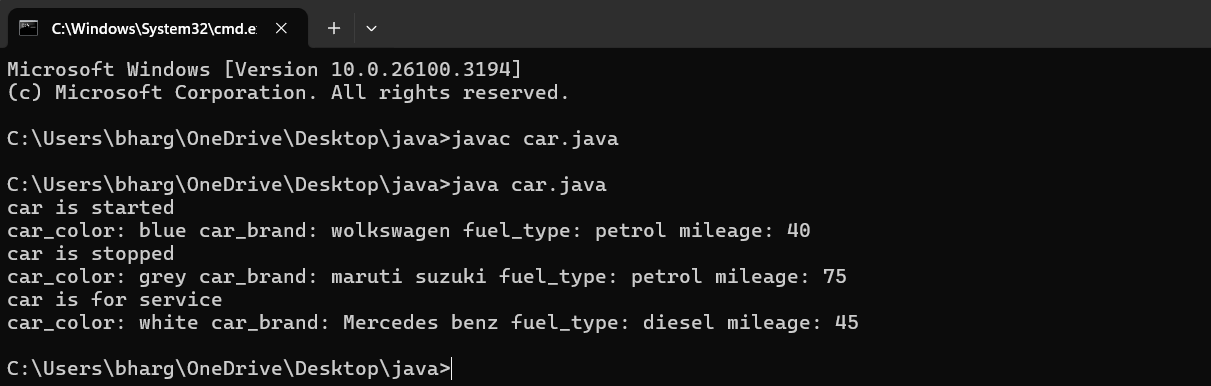
car\_two.service();

System.out.println("car\_color: " + car\_two.car\_color + " car\_brand: " + car\_two.car\_brand + " fuel\_type: " + car\_two.fuel\_type + " mileage: " + car\_two.mileage);

}

}

Output:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not putting the semi-colon; after calling the function. 2. After Start, Stop, Service not giving the parenthesis ( ). | 1. Put the semi-colon after the writing the code. 2. After every method, put the parenthesis ( ). |

IMPORTANT POINTS:

1. Before calling the function we should write the method properly.
2. Here, the “public void start( )” indicates that we are writing a method to call the function.
3. When we call a certain method, the process inside it will be printed as an output of the code.
4. Here the details inside the function are called objects, we can give any objects

**Class diagram:**

|  |
| --- |
| **car**  **-----------------------**  **-car\_color:string**  **-car\_brand:string**  **-fuel\_type:string**  **-milage:double**  **----------------------**  **+start():void**  **+stop():void**  **+service():void** |

**WEEK -4**

**PROGRAM – 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, and year of publication.

Create a method which displays the details of the book and display the details of two books.

**CODE:**

public class Book{

public String Title;

public String Author;

public String Year\_of\_publication;

public Book(String Title, String Author, String Year\_of\_publication){

this.Title=Title;

this.Author=Author;

this.Year\_of\_publication=Year\_of\_publication;

}

public void Bookdetails()

{

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

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

System.out.println("Year\_of\_publication:"+Year\_of\_publication);

}

public static void main (String[] args){

Book book1=new Book("Mahabaratam","Vyasa","3rd century");

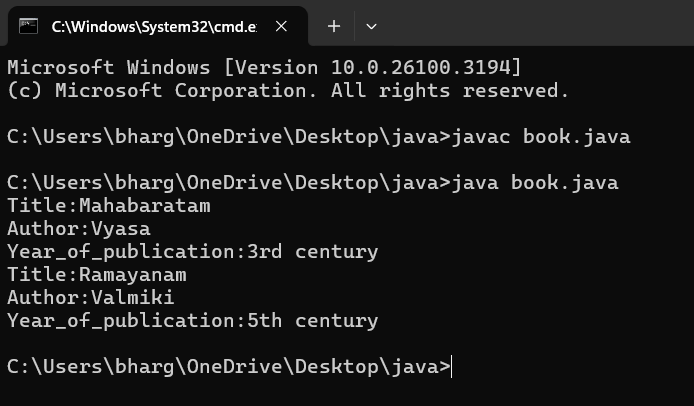
Book book2=new Book("Ramayanam","Valmiki","5th century");

book1.Bookdetails();

book2.Bookdetails();

}

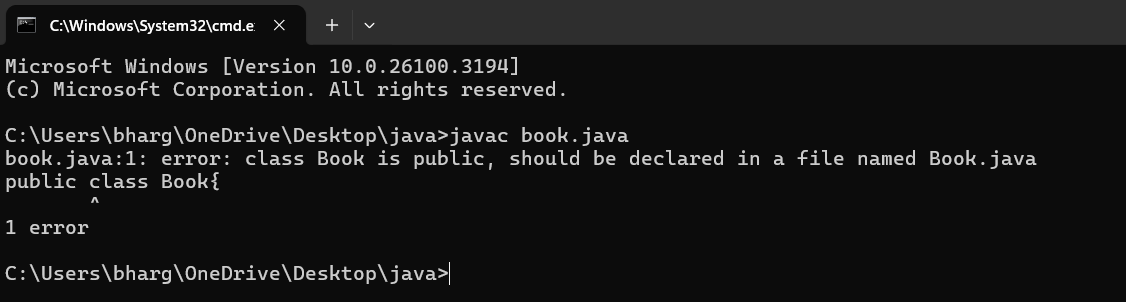
}



**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not defining the function in a file. 2. Two public class files should not be saved in the same file. | 1. To call the method we must define a function in a file. 2. Two public class files should be saved in different files. |

**NEGATIVE CASE:**



**IMPORTANT POINTS:**

1. While defining two classes for a code, we must be sure that we save both the classes in separate files.
2. While defining a method we should also define a function to call that method.

**CLASS DIAGRAM:**

|  |
| --- |
| Book   * Title: String * Author: String * Year of publication: int   + Book(title: String,  Author: String;  Year of publication: int  + displayDetails( ): void |

**PROGRAM – 2:**

**AIM**: Create a java Program with class named myclass with static variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of the class, ow define a constructor for “myclass” that increments the count variable each time an object of my class is created (count++), finally print the final values of count and pi variables create three objects.

**CODE:**

public class myclass {

static int count = 0;

final double pi = 3.14;

public myclass() {

count++;

}

public static void main(String[] args) {

myclass a = new myclass();

myclass b = new myclass();

myclass c = new myclass();

myclass d = new myclass();

myclass e = new myclass();

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

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

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

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

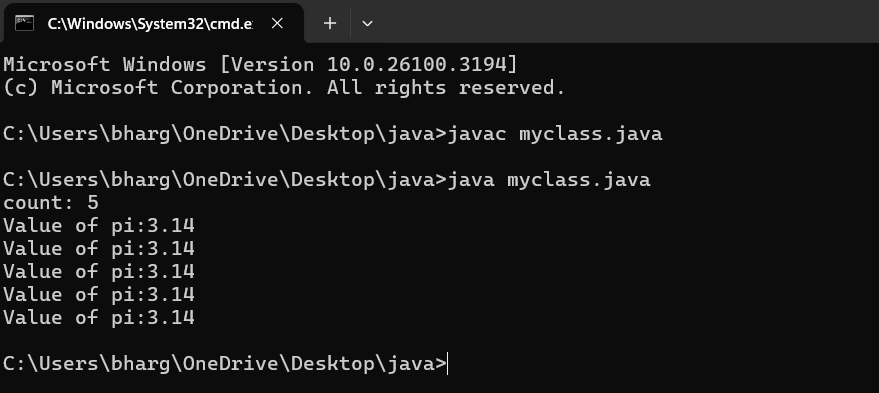
System.out.println("Value of pi:"+d.pi);

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

}

}

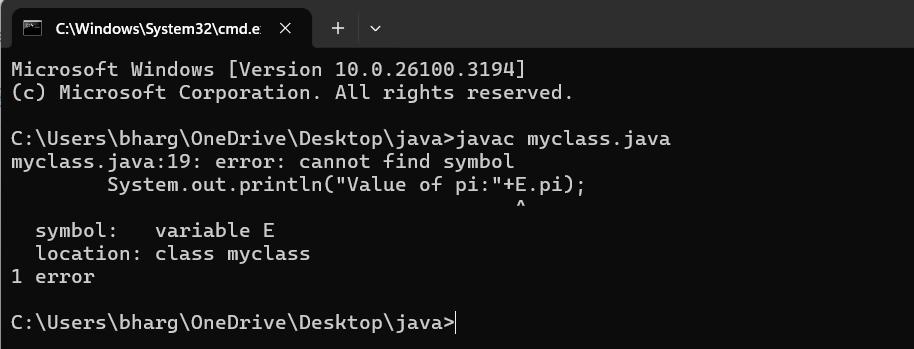
OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not Putting the semi-colon after calling a function, 2. Not giving the indentation properly. | 1. Put the semi-colon after calling a function. 2. All the indentation must be correct to run the code correct. |

**NEGATIVE CASE:**



**IMPORTANT POINTS:**

1. We must declare the initial value of the variable before declaring the final one.
2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

**CLASS DIAGRAM:**

|  |
| --- |
| Myclass   * Count: int * Pi: double   + myclass( )  + main(args: String[]): void |

**Week – 5**

**Aim:** To create a Java Program of a calculator using the operations including addition Subtraction, multiplication and division using multilevel inheritance & dis Play the desired out Put

**CODE:**

class SimpleCalculator{

//attributes, objects

int a;

int b;

//initialization

public static void add(int a, int b) {

System.out.println(a+ " + " + b + " = " + (a+b));

}

public static void diff(int a, int b) {

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

}

}

class AdvCalculator extends SimpleCalculator {

// Initialization

public static void mul(int a, int b) {

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

}

}

class UltiCalculator extends AdvCalculator{

//initialization

public static void div(int a, int b) {

if(b == 0) {

System.out.println("Denominator should not be a zero ");

}

else{

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

}

}

}

class Calc{

public static void main(String[] args) {

UltiCalculator u = new UltiCalculator();

System.out.println("The calculated values are: ");

u.add(5,5);

u.diff(6,5);

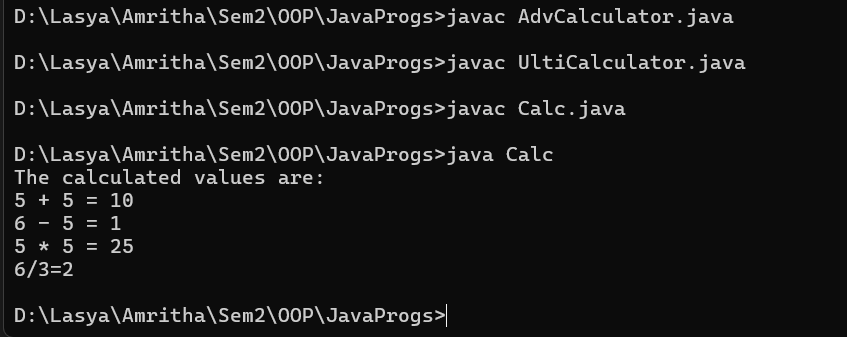
u.mul(5,5);

u.div(6,3);

} // end of the main function

} // end of the class

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: unclosed string literal System.out.println(a+ " - " + b +" = + (a-b)); | Add a '”' after the ‘=’. |
| 2. | error: '(' or '[' expected UltiCalculator u = new UltiCalculator; | Add a '()' after class name. |

**Class Diagram:**

|  |
| --- |
| **SimpleCalculator** |
| a : double  b : double |
| + add (a,b) : void  + diff (a,b) : void |

|  |
| --- |
| **AdvCalculator** |
| + mul (a,b) : void |

|  |
| --- |
| **UltiCalculator** |
| + div (a,b) : void |

|  |
| --- |
| **Calc** |
| + main(String[]) |

**Concepts to be known:**

1. We must declare the initial value of the variable before declaring the final one.

2. here, the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

* **Aim:** A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed( should be in super class)

1. cars should have an additional property: no.of doors
2. Bikes should have a property indicating whether they have gears or not.
3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.
4. Every class should have a constructor

Question:

1. Which oops concept is used in the above program
2. If the company decides to add a new type of vehicle, Truck, how would you modify the program?
3. Truck should include an additional property capacity (in tons)
4. Create a showTruckdetails() method to display the truck’s capacity.
5. Write a constructor for Truck that initializes all properties
6. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details.

**CODE:**

class Vehicle {

    String brand;

    double speed;

    public Vehicle(String brand, double speed) {

        this.brand = brand;

        this.speed = speed;

    }

    public void displayDetails() {

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

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

    }

    public void start() {

        System.out.println(brand + " vehicle is starting...");

}

}

class Car extends Vehicle {

    int numberOfDoors;

    int seatingCapacity;

    public Car(String brand, double speed, int numberOfDoors, int seatingCapacity) {

        super(brand, speed);

        this.numberOfDoors = numberOfDoors;

        this.seatingCapacity = seatingCapacity;

    }

    public void displayCarDetails() {

        super.displayDetails();

        System.out.println("Number of doors: " + numberOfDoors);

        System.out.println("Seating capacity: " + seatingCapacity);

    }

    public void startCar() {

        super.start();

        System.out.println("Car is ready to go!");

    }

}

class Bike extends Vehicle {

    boolean hasGears;

    public Bike(String brand, double speed, boolean hasGears) {

        super(brand, speed);

        this.hasGears = hasGears;

    }

    public void displayBikeDetails() {

        super.displayDetails();

        System.out.println("Has gears: " + (hasGears ? "Yes" : "No"));

    }

    public void startBike() {

        super.start();

        System.out.println("Bike is ready to go!");

    }

}

class Truck extends Vehicle {

    double cargoCapacity;

    public Truck(String brand, double speed, double cargoCapacity) {

        super(brand, speed);

        this.cargoCapacity = cargoCapacity;

    }

    public void displayTruckDetails() {

        super.displayDetails();

        System.out.println("Cargo capacity: " + cargoCapacity + " tons");

    }

    public void startTruck() {

        super.start();

        System.out.println("Truck is ready to go!");

    }

}

public class VehicleRentalSystem {

    public static void main(String[] args) {

        Car car = new Car("Toyota", 150, 4, 5);

        Bike bike = new Bike("Yamaha", 120, true);

        Truck truck = new Truck("Volvo", 90, 10);

        System.out.println("Car Details:");

        car.displayCarDetails();

        car.startCar();

        System.out.println("\nBike Details:");

        bike.displayBikeDetails();

        bike.startBike();

        System.out.println("\nTruck Details:");

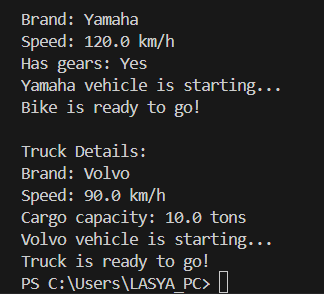
        truck.displayTruckDetails();

        truck.startTruck();

    }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: Declaring two super classes inside the same file. | Make two separate files to save the two super classes |
| 2. | error: Not declaring the variable using ‘this’ keyword inside the constructor. | Declare the variable using this keyword to run the program. |

**Concepts to be known:**

1. a constructor helps in initializing an object that doesn't exist.

2. a method performs functions on pre-constructed or already developed objects.

3. a double method can represent more decimal point numbers than float method.

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| - brand: string  - speed: double |
| +Vehicle()  + displayDetails() : void  + start() : void |

|  |
| --- |
| **Car** |
| - noOfDoors: int  - seatingCapacity: int |
| +Car()  + displayCarDetails() : void  + startCar() : void |

|  |
| --- |
| Bike |
| - hasGears: Boolean |
| +Bike()  + displayBikeDetails() : void  + startBike() : void |

|  |
| --- |
| **Truck** |
| - cargoCapacity: Boolean |
| + Truck()  +displayTruckDetails() : void  + startTruck() : void |

**Answer:**

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

To add a new vehicle type truck we need to create a truck class that will:

* Include an additional property capacity (in tons).
* Implement a showTruckdetials() method to display the truck's capacity.
* Implement a constructor for the truck class to initialize all its properties.

**Week – 6**

* **Aim:** Write a Java Program to create a Vehicle class with a method display(). Override this method in the Car subclass. Print car model, brand, petrol type, car color and provide the information about the car.

**Code :**

class Vehicle {

String brand;

String petrolType;

String color;

public void display() {System.out.println("This is a vehicle.");

}

}

class Car extends Vehicle {

Car(String brand, String petrolType, String color) {

this.brand = brand;

this.petrolType = petrolType;

this.color = color;

}

public void display() {

System.out.println("Car Details:");

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

System.out.println("Petrol Type: " + petrolType);

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

}

}

public class Info {

public static void main(String[] args) {

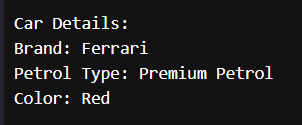
Vehicle myCar = new Car("Ferrari", "Premium Petrol", "Red");

myCar.display();

}

}

**Output :**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: class Main is public, should be declared in a file named Main.java  public class Main { | Save the file name as the name of the main class |

**Class Diagram:**

|  |
| --- |
| **Vehicle** |
| + brand : String  + petrolType : String  + color : String |
| + display() : void |

|  |
| --- |
| **Car** |
| Car(String brand, String petrolType, String color) |

**Concepts to be known:**

1. We use the concept of method overriding where the names of the methods in the different classes. The method of the parent class is overridden by the method of the child class.

**Aim:** A college is developing an automated admission system that verifies students’ eligibility for under-graduation and post-graduation. Each program has different eligibility criteria base on the percentage of students in their provided qualifications

* Ug requires 60%
* PG requires 70%

**Code :**

class Student{

String name;

double percentage;

Student(String name, double percentage){

this.name = name;

this.percentage = percentage;

}

public void Eligibility(){

System.out.println(name + " must meet the general admission criteria");

}

}

class UG extends Student{

UG(String name, double percentage){

super(name, percentage);

}

public void Eligibility(){

if (percentage>59){

System.out.println(name + " is elligible for UG admission");

}

else {

System.out.println(name + " is Not elligible for UG admission");

}

}

}

class PG extends Student{

PG(String name, double percentage){

super(name, percentage);

}

public void Eligibility(){

if (percentage>69){

System.out.println(name + " is elligible for PG admission");

}

else {

System.out.println(name + " is Not elligible for PG admission");

}

}

}

public class Admission {

public static void main (String[] args){

UG ug = new UG("Laila", 69);

PG pg = new PG("Majnu", 59);

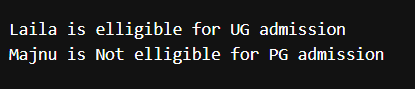
ug.Eligibility();

pg.Eligibility();

}

}

**Output :**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: ';' expected System.out.println(name + " is elligible for UG admission") | Add a ‘;’ after the print statement. |

**Class Diagram:**

|  |
| --- |
| **UG** |
| UG(String name, double percentage) |

|  |
| --- |
| **Student** |
| + name : String  + percentage : double |
| + Student(String name, double percentage): void  + Eligibility(): void |

|  |
| --- |
| **PG** |
| PG(String name, double percentage) |

**Concepts to be known:**

1. The variables once declared in the super class need not be declared twice in any of the sub classes.

2. super keyword is used in sub classes to access the methods of super classes, they are basically the reverse of overriding.

* **Aim:** To create a Java Program with class named “my class” with a Static Variable Count int type and initialize to 0 and A Constant Variable "pi" of type double initialized to 3.1415 has attributes of that class. Now defi a Constructor for my class that increments the Count Variable each time an object of my class is created. Finaly Print the final values of count.

**CODE:**

class AddCalculator{

    AddCalculator(){

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

    }

    public int add(int a, int b){

        return a+b;

    }

    public double add(double a, double b){

        return a+b;

    }

    public int add(int a, int b, int c){

        return a+b+c;

    }

}

public class AddCalc{

    public static void main (String [] args){

    AddCalculator calc = new AddCalculator();

    System.out.println("The sum of the 2 numbers: 7 and 18 is "+ calc.add(7, 18));

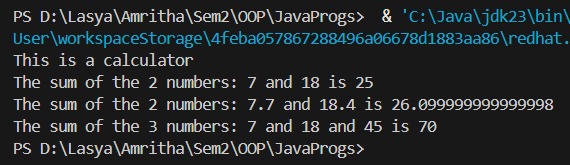
    System.out.println("The sum of the 2 numbers: 7.7 and 18.4 is "+ calc.add(7.7, 18.4));

    System.out.println("The sum of the 3 numbers: 7 and 18 and 45 is "+ calc.add(7, 18, 45));

    }

}

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: Main method not defined | Add public static void main (String [] args){ |
| 2. | error: ';' return type is not expected | Change return type from int to double in its case. |

**Class Diagram:**

|  |
| --- |
| **AddCalculator** |
| + AddCalculator ()  + add(int a, int b) : int  + add(double a, double b) : double  + add(int a, int b, int c) : int |

**Concepts to be known:**

1. We use the concept of method overloading where the names of the methods in the same class are same but the parameters are given different.

**Aim:** Write a Java Program and create a Shape class with a method calcArea(). That is overloaded for different shapes like square and rectangle. Create a sub class circle that overrides the calcArea() for a circle.

**Code :**

class Shape{

int calcArea(int a){

return a\*a;

}

int calcArea(int b, int h){

return b\*h;

}

}

class Circle extends Shape{

double r;

double pi = 3.141592653589793;

Circle(double r){

this.r = r;

}

double calcArea(double r){

return pi\*r\*r;

}

}

public class AreaCalc {

public static void main(String[] args) {

Circle c = new Circle(7);

System.out.println("The area of circle is " + c.calcArea(7.7));

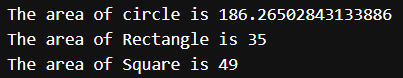
System.out.println("The area of Rectangle is " + c.calcArea(5, 7));

System.out.println("The area of Square is " + c.calcArea(7));

}

}

**Output :**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sl. no** | **Error** | **Error rectification** |
| 1. | error: invalid method declaration; return type required  calcArea(int a){ | Enter the return type as per required. Here it is int |
| 2. | error: incompatible types: possible lossy conversion from double to int  return pi\*r\*r; | For calculating area of circle, we need to give return type double. |

**Class Diagram:**

|  |
| --- |
| **Circle** |
| + r : double  + pi : double |
| + calcArea(int r) : double |

|  |
| --- |
| **Shape** |
| + calcArea(int a) : int  + calcArea(int b, int h) : int |

**Concepts to be known:**

1. We use the concept of method overloading to calculate the area of square and rectangle in the parent class Shape.

2 we use method overriding in the child class Circle to calculate it’s area.

**WEEK-7**

**PROGRAM-1**

**AIM:**

Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion and Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal.

**CODE:**

abstract public class Animal1 {

    abstract String sound();

}

class Lion extends Animal1 {

    @Override

    String sound() {

        return "Roar Roar..";

    }

}

class Tiger extends Animal1 {

    @Override

    String sound() {

        return "Raww Raww..";

    }

}

class Make {

    public static void main(String[] args) {

        System.out.println("Name:Bhargavi.ch, RollNo:AV.SC.U4CSE24101, Section:CSE = B");

        Animal1 l = new Lion();

        System.out.println("Lion makes the sound " + l.sound());

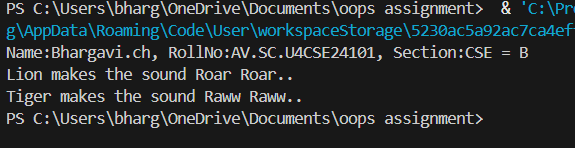
        Animal1 t = new Tiger();

        System.out.println("Tiger makes the sound " + t.sound());

    }

}

**OUTPUT:**



**CLASS DIAGRAM:**

|  |
| --- |
| **Animal**  + sound(): void |

|  |
| --- |
| Lion  + sound (): void |

|  |
| --- |
| Tiger    + sound(): void |

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Error while printing the variables. 2. Incorrect declaration of integer. | 1. Give the plus sign while printing. 2. Give input.nextInt(), where I should be capital. |

**IMPORTANT POINTS:**

1. We override the methods in the superclass.

Here we are using the heirarchial inheritance.

**AIM:**Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

**CODE:**

abstract class Shape3D {

abstract double volume();

abstract double surfaceArea();

}

class Sphere extends Shape3D {

double radius;

Sphere(double radius) {

this.radius = radius;

System.out.println("Sphere created with radius: " + radius);

}

@Override

double volume() {

return (4.0/3.0) \* 3.14 \* radius \* radius \* radius;

}

@Override

double surfaceArea() {

return 4 \* 3.14 \* radius \* radius;

}

}

class Cylinder extends Shape3D {

double radius;

double height;

Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

System.out.println("Cylinder created with radius: " + radius + " and height: " + height);

}

@Override

double volume() {

return 3.14 \* radius \* radius \* height;

}

@Override

double surfaceArea() {

return (2 \* 3.14 \* radius \* height) + (2 \* 3.14 \* radius \* radius);

}

}

class ShapeTest {

public static void main(String[] args) {

System.out.println("Name: Bhargavi.ch, Rollno: AV.SC.U4CSE2401, Section: B");

Sphere s1 = new Sphere(5.0);

System.out.println("Sphere Volume: " + s1.volume());

System.out.println("Sphere Surface Area: " + s1.surfaceArea());

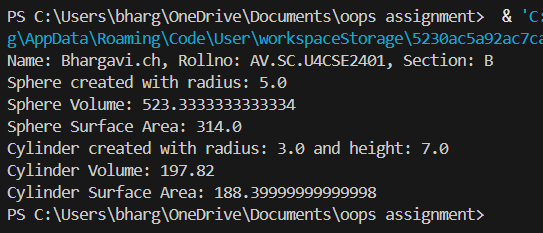
Cylinder c1 = new Cylinder(3.0, 7.0);

System.out.println("Cylinder Volume: " + c1.volume());

System.out.println("Cylinder Surface Area: " + c1.surfaceArea());

}}

**OUTPUT:**



**CLASS DIAGRAM:**

|  |
| --- |
| Shape3D  +calculatevolume(): double  +calculatesurfacearea(): double |

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Wrong datatype entered. 2. Object not defined. | 1. Enter the correct datatype i.e double instead of int. 2. Enter the correct object and if not create new one. |

1. Here we used the abstract to declare an abstract class.
2. Abstract classes and methods help us to declare the methods without declaring the return type in them.

To get the values, we declared a constructor for each subclass and initialized values for them

**AIM:**Write a Java program using an abstract class to define a method for pattern printing.

Create an abstract class named PatternPrinter with:an abstract method printPattern(int n)and a concrete method to display the pattern title

Implement two subclasses:

1. StarPattern – prints a right-angled triangle of stars

2. NumberPattern – prints a right-angled triangle of increasing numbers

In the main() method, create objects of both subclasses and print the patterns for a given number of rows.

Example Output for n = 5:

Star Pattern

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Number Pattern

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**CODE:**

abstract class PatternPrinter {

int rows;

PatternPrinter(int rows) {

this.rows = rows;

}

abstract void printPattern();

void displayTitle(String title) {

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

}

}

class StarPattern extends PatternPrinter {

StarPattern(int rows) {

super(rows);

}

void printPattern() {

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

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

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

}

System.out.println();

}

}

}

class NumberPattern extends PatternPrinter {

NumberPattern(int rows) {

super(rows);

}

void printPattern() {

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

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

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

}

System.out.println();

}

}

}

public class Teststar {

public static void main(String[] args) {

System.out.println("Name:Bhargavi.ch,Section:B,Roll NO:AV.SC.U4CSE24101");

int numberOfRows = 5;

PatternPrinter star = new StarPattern(numberOfRows);

star.displayTitle("Star Pattern");

star.printPattern();

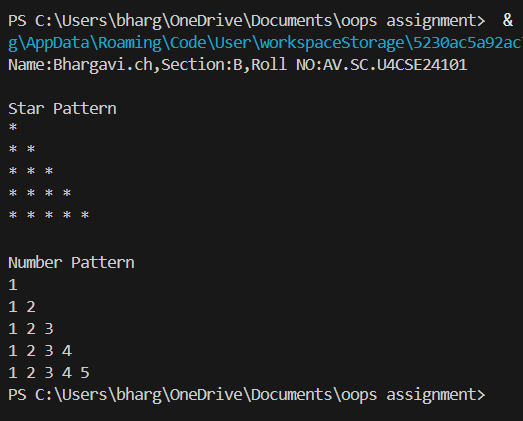
PatternPrinter number = new NumberPattern(numberOfRows);

number.displayTitle("Number Pattern");

number.printPattern();

}}

**OUTPUT:**



**CLASS DAIGRAM:**

PatternPrinter

- rows: int

+displayTitle()

+printPattern()

StarPattern

+printPattern()

NumberPattern

+printPattern()

**ERROR TABLE:**

|  |  |
| --- | --- |
| CODE ERROR:   1. Class name and file name should match 2. Subclass doesn’t override abstract method | ERROR RECTIFICATION   1. Save file as main.java   2)implement printpattern()in all subclasses |

**Important Points:**

Use abstract classes to enforce a common structure for pattern printing.

PatternPrinter is the abstract class defining the common template.

Subclasses (StarPattern, NumberPattern) provide specific implementations.

displayTitle() is a concrete method shared by all subclasses.