**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**

****

**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: NANDINI RAVULA**

Roll No: **AV.SC.U4CSE24310**

Branch: CSE

Section: C [D]

**Verified By**

**INDEX**

|  |  |  |
| --- | --- | --- |
|  | EXPERIMENTS | REMARKS |
| WEEK-1 |  |  |
| 1) | downloading and installation of java |  |
| 2) | write a java program that prints name, roll no, section of a student. |  |
| WEEK-2 |  |  |
| 1) | write a java program to find the simple interest where all the inputs are taken from the user |  |
| 2) | write a java program to find the fibonacci sequence of a given number |  |
| 3) | write a java program to find the area of rectangle |  |
| 4) | write a java program to find the area of triangle |  |
| 5) | write a java code to convert the temperature from celsius to fahrenheit and from fahrenheit to celsius. |  |
| 6) | write a java code to convert the temperature from celsius to fahrenheit and from fahrenheit to celsius. |  |
| WEEK-3 |  |  |
| 1) | 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 three methods named start(), stop(), service(). 4. create three methods named car1, car2, car3. |  |
| 2) | to create a class bankaccount with methods deposit() and withdrawl |  |
| WEEK-4 |  |  |
| 1) | write a java program with class named “book”. the class should contain various attributes such as “title of the book”, “author”, year of publications”, it should also contain a constructor with parameter which initializes “title of the book”, “author”, “year of publication”. create a method which displays the details of the book. title of the book (), author (), year of publication ().display the details of two book, by creating 2 objects |  |
| 2) | to create a java program with class named “myclass” with a static variable count of “int” type, in sized to “zero” and a constant variable “pi” of type “double” initialized to 3.1415 as attributes of that class. define a contractor for “myclass” is created finally print the final values “count” and “pi” variables. |  |
| WEEK-5 |  |  |
| 1) | Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.Hint: collect required variables using super class, Create each class for a parameter and each class must contain a method. |  |
| 2) | 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?  a. Truck should include an additional property capacity (in tons)  b. Create a showTruckdetails() method to display the truck’s capacity.  c. Write a constructor for Truck that initializes all properties  3. 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 |  |
| WEEK-6 |  |  |
| 1) | Write a java program to create a vehicle class with a method displayInfo().  Override this method in the car subclass to provide specific information a About car. |  |
| 2) | A college is developing an automated admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.  1. UG admission require min of 60%  2. PG admission require min of 70% |  |
| 3) | 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. |  |
| 4) | 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. |  |
| WEEK-7 |  |  |
| 1) | Write a java program to create an abstract class animal with an abstract method called sound(). Create subclasses Lion Tiger that extend the Animal class and implement the sound() method to make a specific sound for each animal. |  |
| 2) | 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. |  |
| 3) | 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 stars  2.NumberPateern-Prints a right angled triangle increasing number  IN the main() method create objects of both subclasses and print the pattern for a given number of rows. |  |
| WEEK-8 |  |  |
| 1) | Write a JAVA program to create an interface Shape with the getPerimeter() method. Create three classes Rectangle, Circle and Triangle that implement the  shape interface. Create three getPerimeter() method for each of the three classes. |  |
| 2) | Write a JAVA program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football , Volleyball and Basketball that implement  the Playable interface and override the play() method to play the respective sports. |  |
|  |  |  |

**WEEK-1**

**Program-1**

**AIM:** To download and install Java (JDK 21)

**PROCEDURE:**

This is the process for installation of JDK on windows.

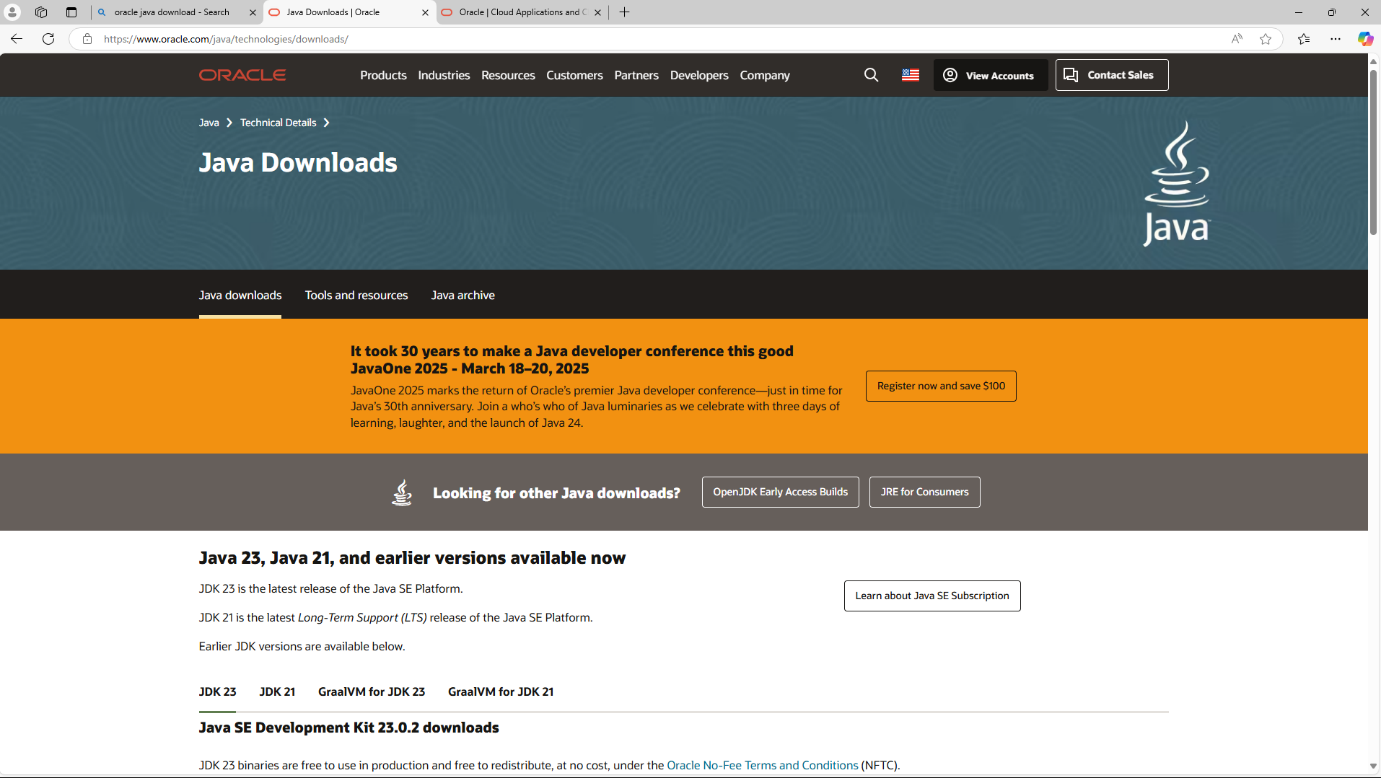
Follow the steps below to install Java on Windows:

* Download JDK (Java Development Kit)
* Run the Installer
* Configure Environment Variables
* Update the path variable
* Verify Installation in Command Prompt

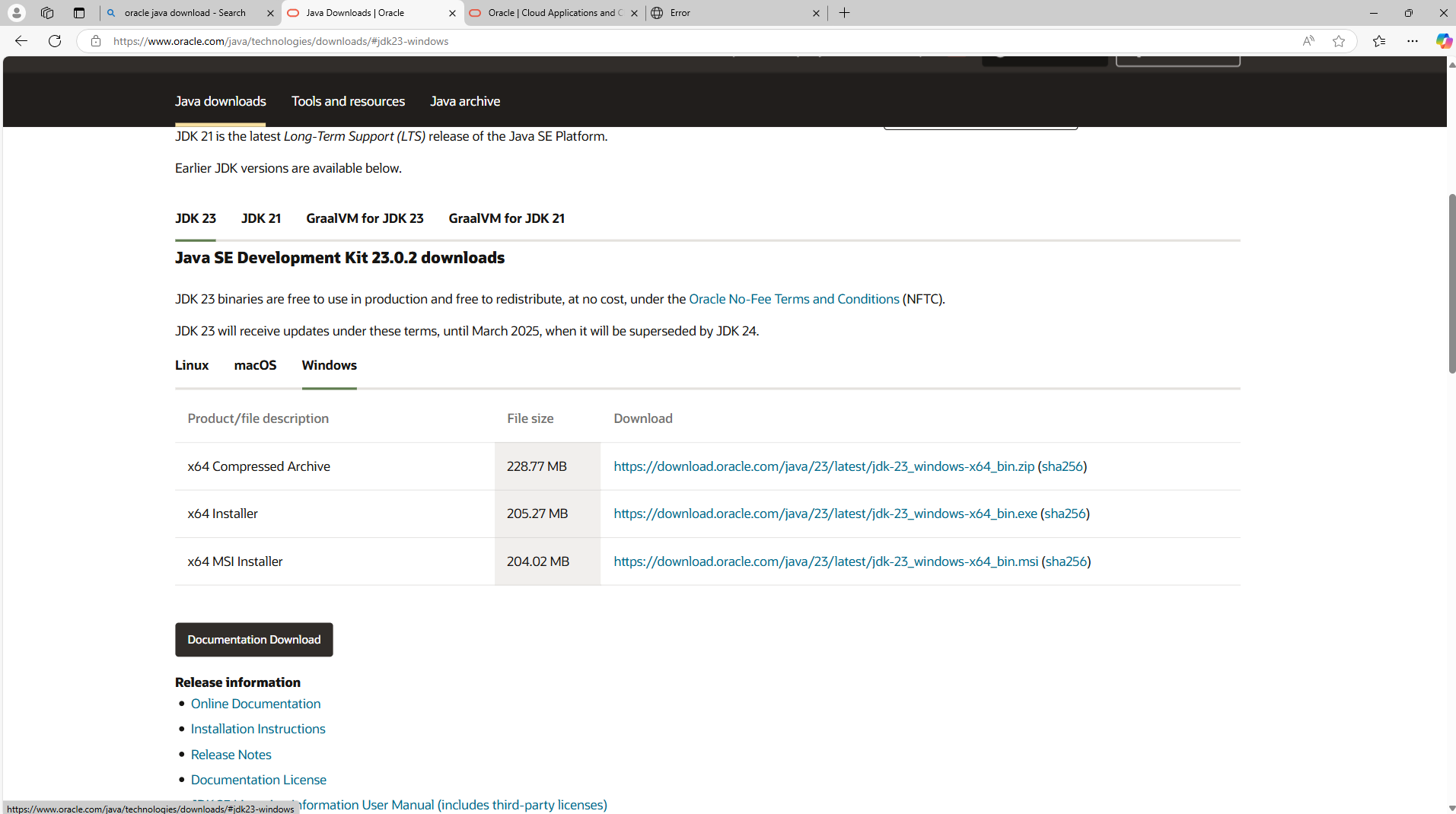
Here's a detailed explanation of each of the steps.

**Step 1**: Download JDK

1. Go to the official oracle website in the google search to download the JDK.
2. Locate the downloaded jdk-21\_windows-x64\_bin.exe file.
3. Double-click to launch the installer.
4. Click Next on the setup wizard.



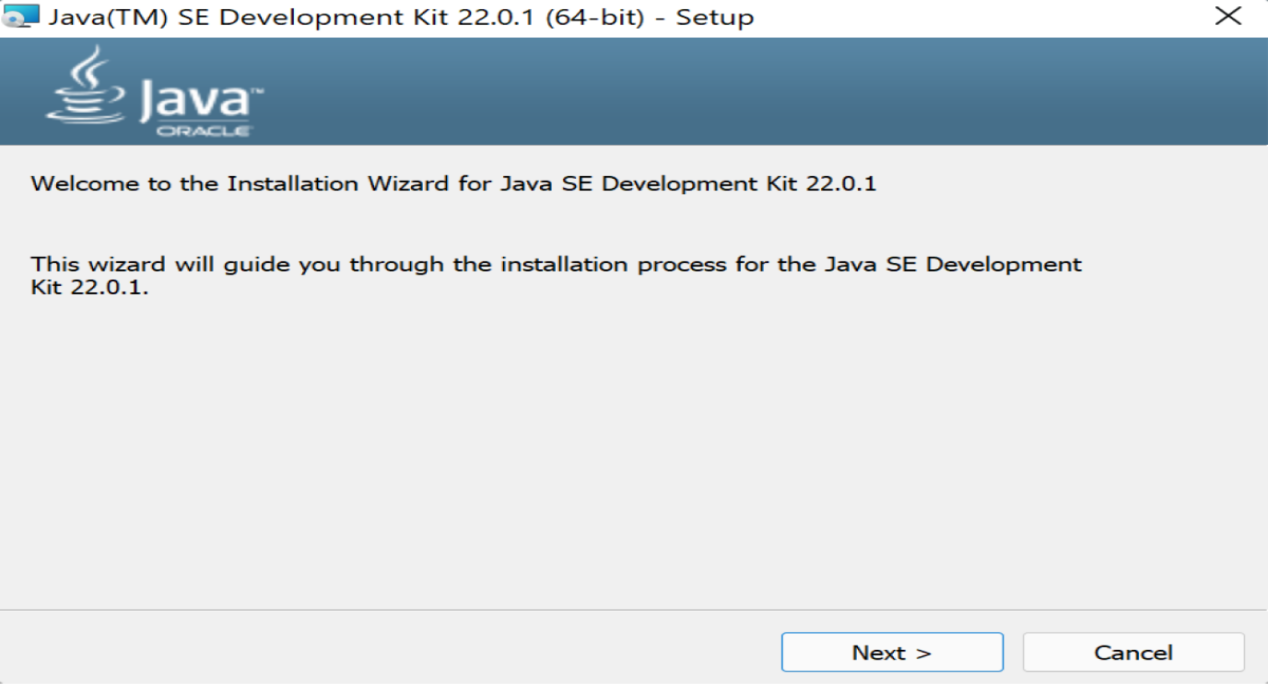
1. Choose the installation path (default is C:\Program Files\Java\jdk-21).
2. Click Next, then click Install.
3. Wait for the installation to complete.
4. Click Close once the installation is finished
5. Choose **x64 MSI Installer** on the windows tab and click on download link.



Step 2: Run the Installer

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

The screen below will be seen.



Simply click **Next** to proceed. Next you will be prompted another screen simply click next on that also.

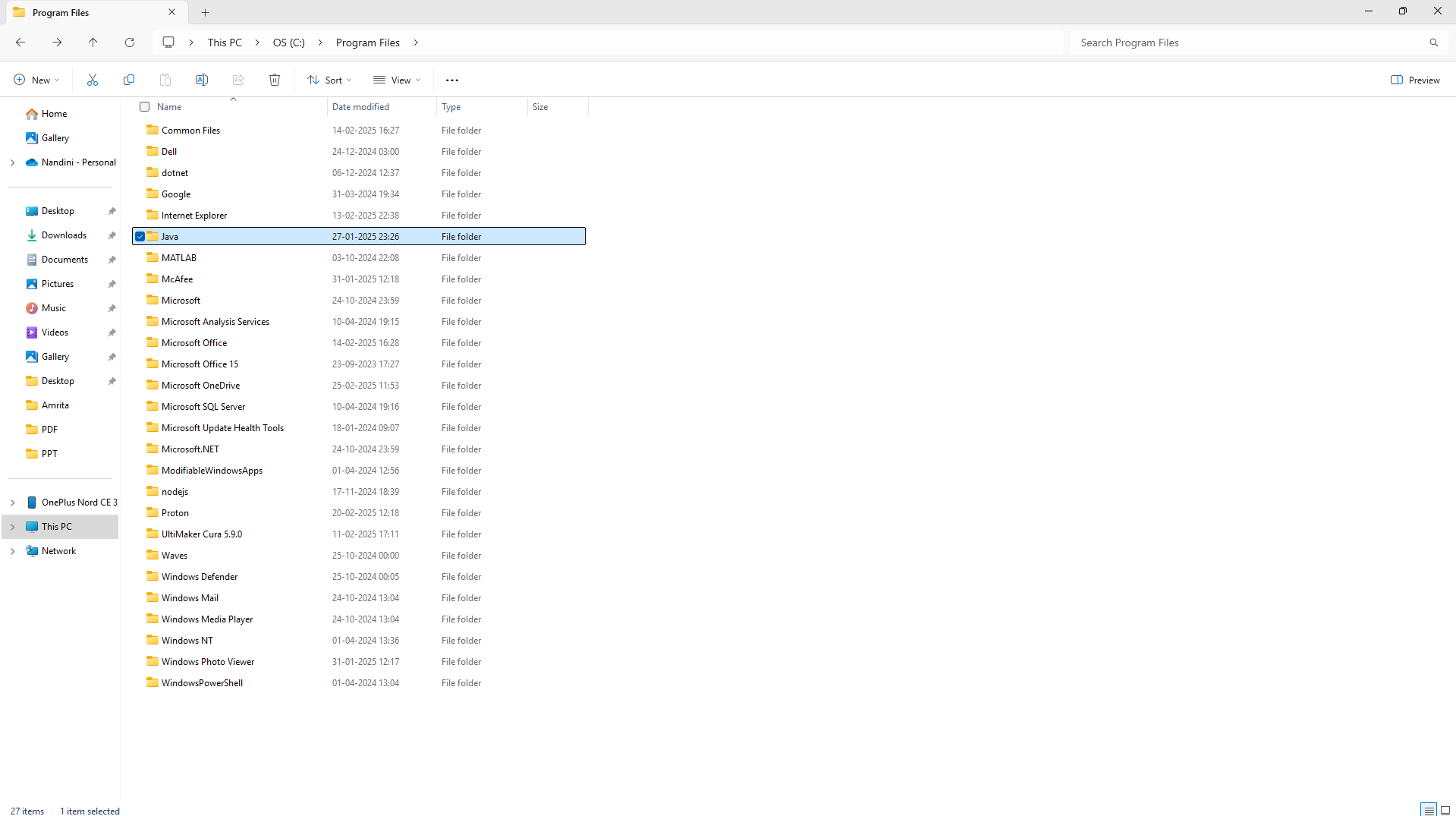
Step 3: Configure Environment Variables

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

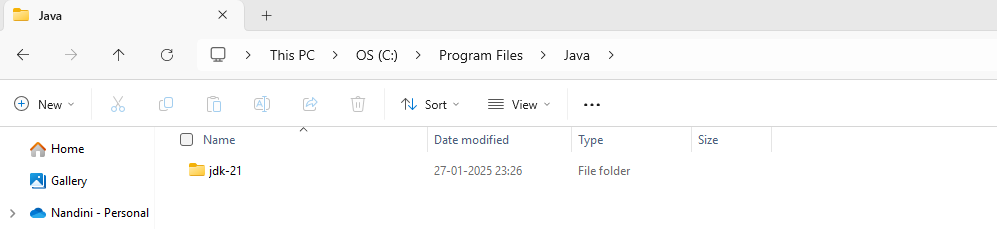
These are the ways to follow:

1)Go to file manager on your laptop or pc

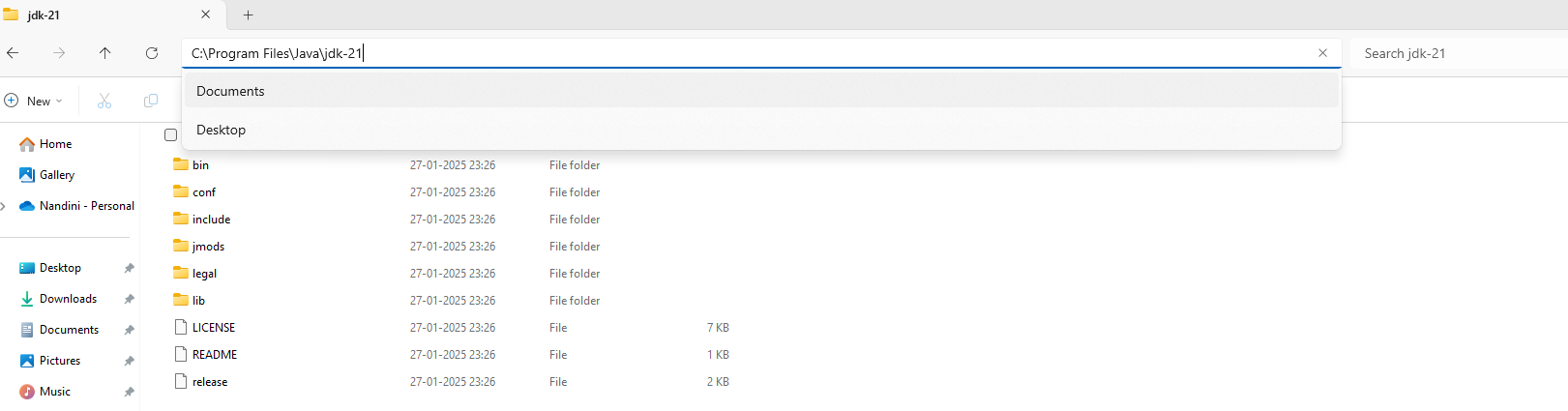
2) Go to “Windows C” Drive in File manager



3) Choose Program Files, select Java, then JDK 22, then select Bin.

4) Select and copy the path at the address bar 

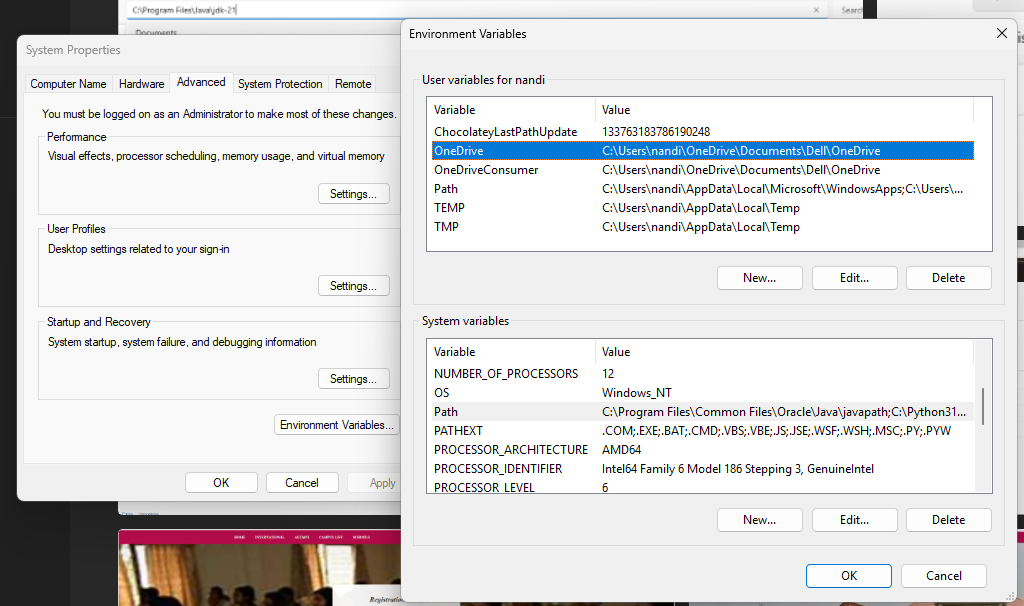
**Locate JDK Path**: Navigate through your file explorer to reach the JDK installation directory. Normally, it is located at

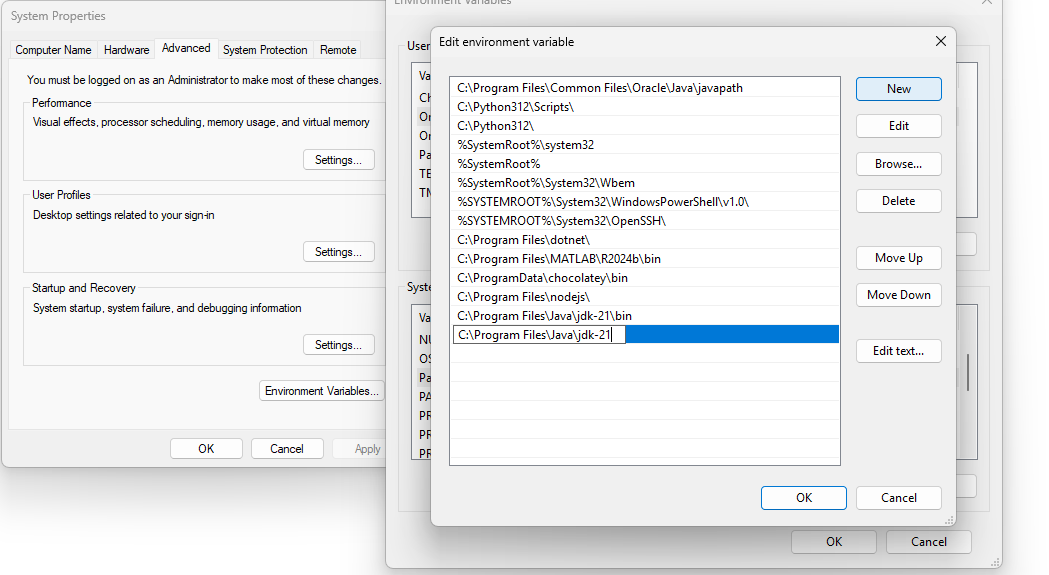


C:\Program Files\Java\jdk-22\bin

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.





**Step 4: Update the Path Variable:**

Find the **Path** variable in the System variables section and click on **Edit**.

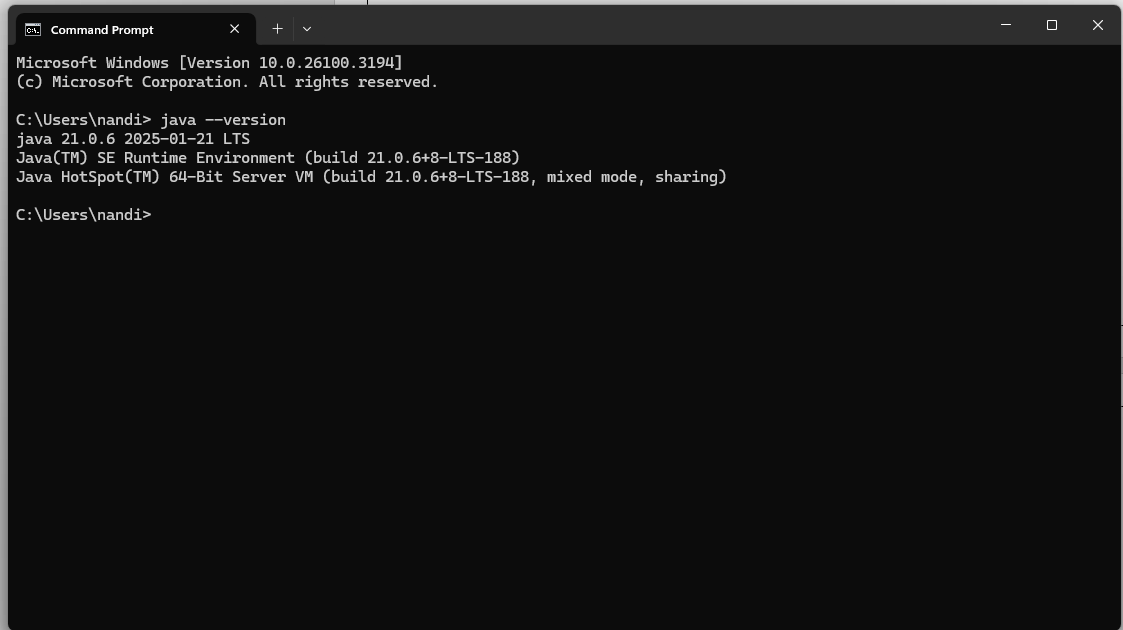
Then, click **New** and paste your JDK bin path (i.e. C:\Program Files\Java\jdk-22\bin).

Finally, click **Ok** to close each window.

**Step 5**: Verify your Installation

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

java --version



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.

**PROGRAM-2:**

**AIM:** Write a Java program to print the message “Hello World”.

**Code**:

public class Hello{

public static void main(String[] args){

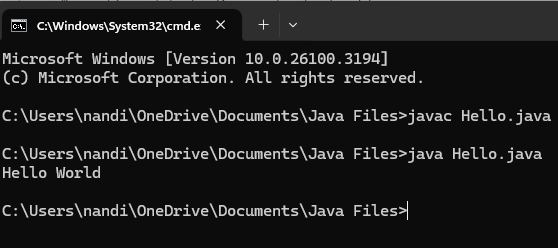
System.out.println("Hello World");

System.out.println("NandiniRavula");

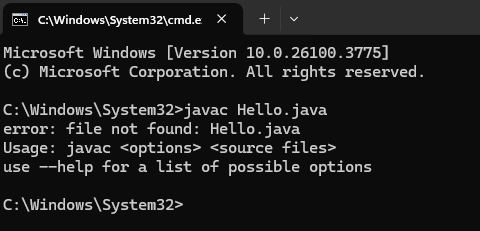
}

}

**POSITIVE OUTPUT**:



NEGATIVE OUTPUT;



**ERROR TABLE:**

|  |  |
| --- | --- |
| ERRORS | RECTIFICATION |
| S in string is written in lowercase letter | The error is rectified by writing s in uppercase letter |

**PROGRAM-3:**

**AIM:** Write a Java Program that prints Name, Roll No,Course

**CODE:**

class studentdetails{

public static void main(String[] main){

System.out.println("Name:R.Nandini");

System.out.println("Roll no:AV.SC.U4CSE24310");

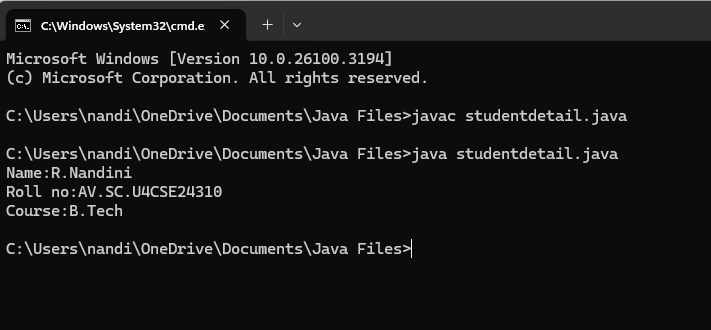
System.out.println("Course:B.Tech");

System.out.println("NandiniRavula");

}

}

**OUTPUT:**



**ERROR TABLE:**

|  |  |
| --- | --- |
| ERROR | RECTIFICATION |
| In the statement at the end ; is not mentioned | Rectified by keeping ; at the end of the ststement |

**WEEK-2**

**Program-1**

**AIM:** Write a java program to find the simple interest where all the inputs are taken from the user**.**

**CODE:**

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(P): ");

double principal = scanner.nextDouble();

System.out.print("Enter the rate of interest (R): ");

double rate = scanner.nextDouble();

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

double time = scanner.nextDouble();

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

System.out.println("Principal: " + principal);

System.out.println("Interest Rate: " + rate + "%");

System.out.println("Time Duration: " + time + " years");

System.out.printf("Simple Interest: %.2f\n", simpleInterest);

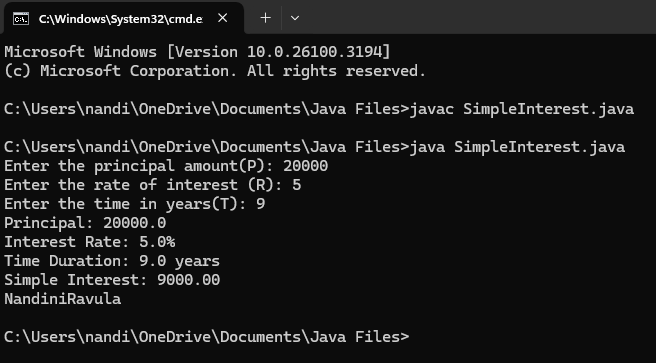
scanner.close();

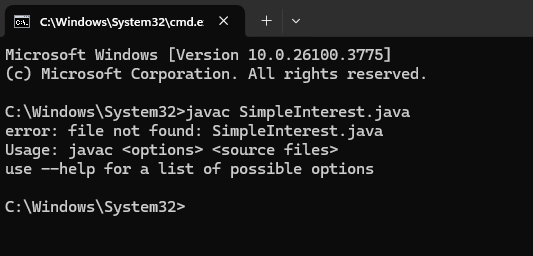
System.out.println("NandiniRavula");

}

}

**POSITIVE OUTPUT:**

****

**NEGATIVE OUTPUT;**

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | error: ';' expected          System.out.print("Enter the rate of interest (R) in percentage: ") | Insert: ‘;’          System.out.print("Enter the rate of interest (R) in percentage: "); |

**PROGRAM-2:**

**AIM:** Write a java program to find the Fibonacci sequence of a given number

**CODE:**

import java.util.Scanner;

public class FibonacciSequence {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter n = ");

int n = scanner.nextInt();

int num1 = 0, num2 = 1;

System.out.println("Fibonacci sequence up to " + n + " terms:");

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

System.out.print(num1 + " ");

int sum = num1 + num2;

num1 = num2;

num2 = sum;

}

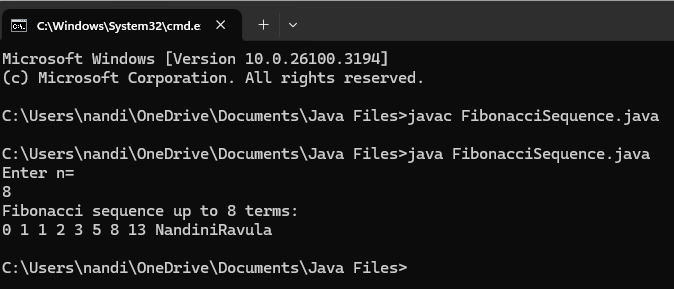
System.out.println("\nNandiniRavula");

scanner.close();

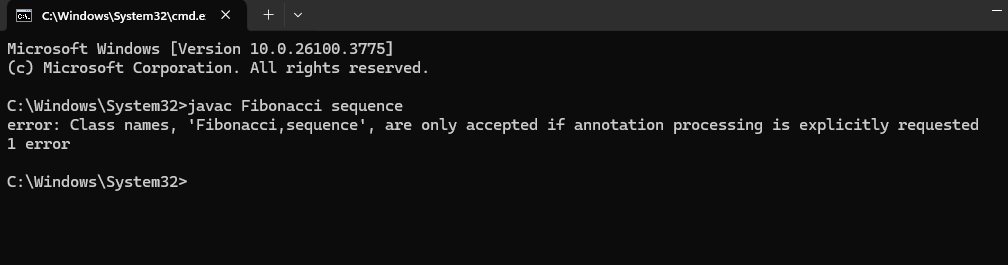
}

}

**POSITIVE OUTPUT:**

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAAGE | ERROR RECTIFICATION |
| 1. | Enter the number: 5  The factorial of 5 is 1  Error: factorial \*=1; | Replace i in 1 place   Replace:factorial \*=i; |

**Program-3:**

**AIM:** Write a java program to find the area of rectangle and triangle.

**CODE:**

**a)**import java.util.Scanner;

public class RectangleArea {

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(" area of the rectangle="+area);

scanner.close();

System.out.println("NandiniRavula");

}

}

**b)** import java.util.Scanner;

public class TriangleArea {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the base of the triangle: ");

double base = scanner.nextDouble();

System.out.print("Enter the height of the triangle: ");

double height = scanner.nextDouble();

double area = (base \* height) / 2;

System.out.printf("The area of the triangle is: %.2f\n", area);

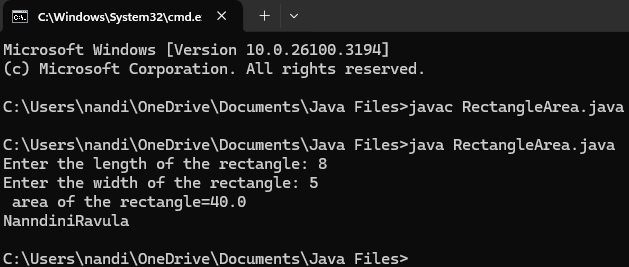
System.out.println("NandiniRavula");

scanner.close();

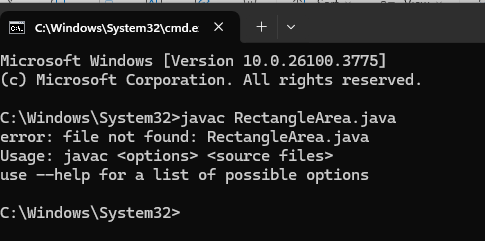
}

}

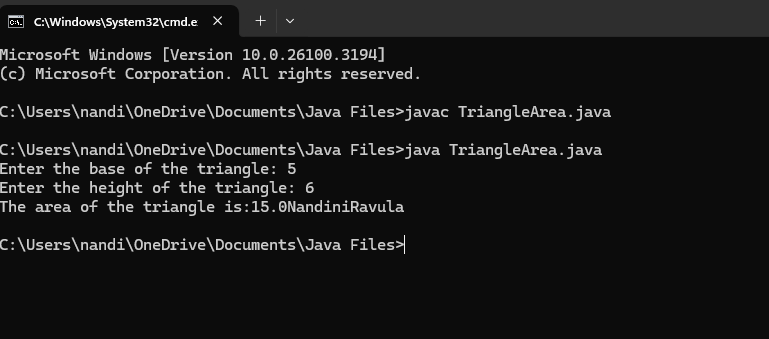
**POSITIVE OUTPUT:**

****

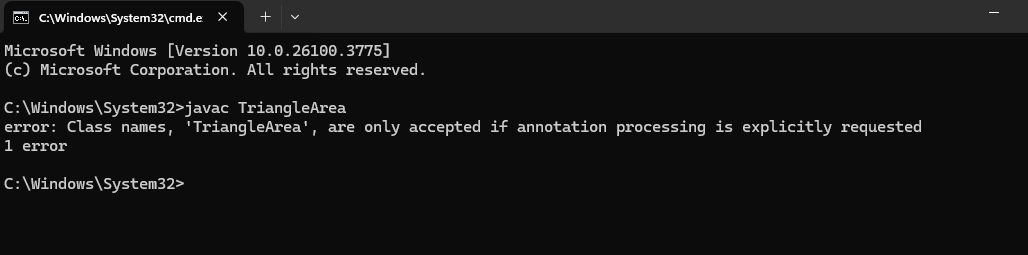
**NEGATIVE OUTPUT:**

****

**POSITIVE OUTPUT:**

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error: float fahrenheit = celsius \* 9 / 5 + 32;  Reason:Formula mistake. | It should be ((celsius \* 9) / 5) + 32 |

**PROGRAM-4:  
AIM:** Write a java code to convert the temperature from Celsius to Fahrenheit and from Fahrenheit to Celsius.

**CODE:**

**a)** import java.util.Scanner;

public class Fahrenheit {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double celsius = scanner.nextDouble();

double fahrenheit = (celsius \* 9 / 5) + 32;

System.out.printf("The temperature in Fahrenheit is: %.2f\n", fahrenheit);

System.out.println("NandiniRavula");

scanner.close();

}

}

b) import java.util.Scanner;

public class Celsius {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double fahrenheit = scanner.nextDouble();

double celsius = (fahrenheit - 32) \* 5 / 9;

System.out.printf("The temperature in Celsius is: %.2f\n", celsius);

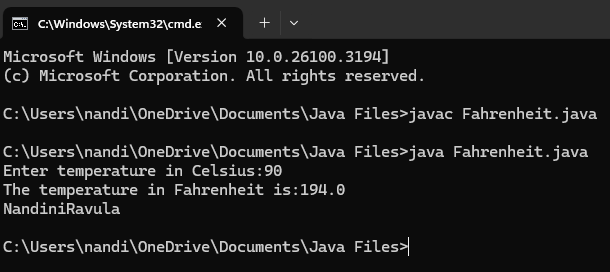
System.out.println("NandiniRavula");

scanner.close();

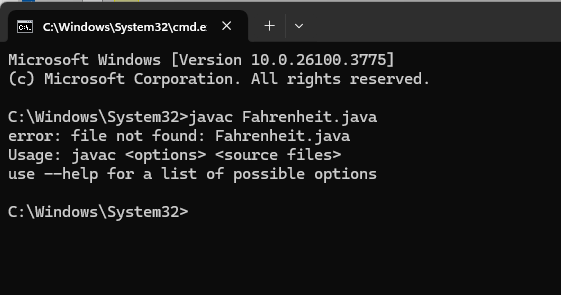
}

}

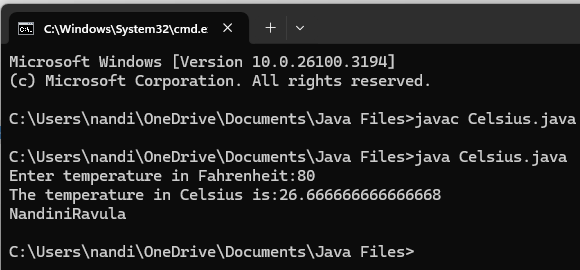
**POSITIVE OUTPUT:**

****

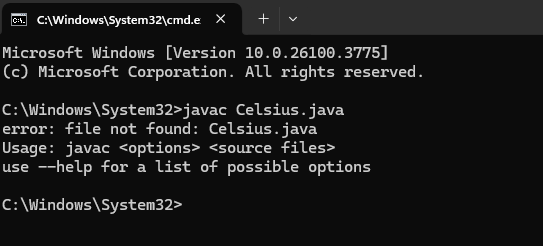
**NEGATIVE OUTPUT:**

****

**POSITIVE OUTPUT:**

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Error: Print statement with incorrect variable name System.out.println(fahrenheit + "°F is equal to " + Celsius + "°C") | 'Celsius' should be lowercase  It should be ‘celsius’. |

**PROGRAM-5**

**AIM:** Write a java code to find factorial of a number by taking input.

**CODE:**

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

if (number < 0) {

System.out.println("Factorial is not defined for negative numbers.");

} else {

long factorial = 1;

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

factorial \*= i;

}

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

}

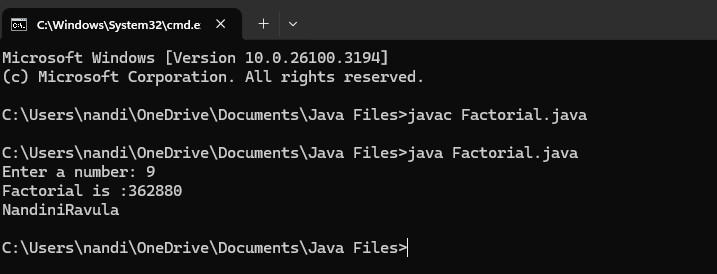
System.out.println("NandiniRavula");

scanner.close();

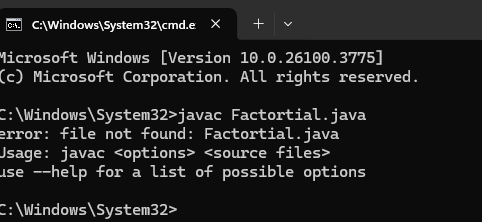
}

}

**POSITIVE OUTPUT:**

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | int firstTerm;  Error: variables not initialized properly | should be initialized          int n = 10; |

**WEEK-3**

**Program-1**

**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 three methods named start (), stop (), service ().
4. Create three methods named Car1, Car2, Car3.
5. **CLASS DIAGRAM:**

|  |
| --- |
| CLASS CAR |
| +Car\_color: string  +Car\_brand: string  + fuel\_type:string  +mileage:int |
| +void start()  +void stop()  +void service() |

**CODE:**

public class Car {

public String carColor;

public String carBrand;

public String fuelType;

public int mileage;

public void start() {

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

showDetails();

}

public void stop() {

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

showDetails();

}

public void service() {

System.out.println("Car Sent for Service");

showDetails();

}

public void showDetails() {

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

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

System.out.println("Fuel Type : " + fuelType);

System.out.println("Mileage : " + mileage + " km/l");

System.out.println();

}

public static void main(String[] args) {

System.out.println("=== Nandini Ravula ===\n");

Car car1 = new Car();

car1.carColor = "Blue";

car1.carBrand = "Audi";

car1.fuelType = "Diesel";

car1.mileage = 100;

car1.start();

Car car2 = new Car();

car2.carColor = "Red";

car2.carBrand = "Tesla";

car2.fuelType = "EV";

car2.mileage = 200;

car2.stop();

Car car3 = new Car();

car3.carColor = "Yellow";

car3.carBrand = "BMW";

car3.fuelType = "Petrol";

car3.mileage = 300;

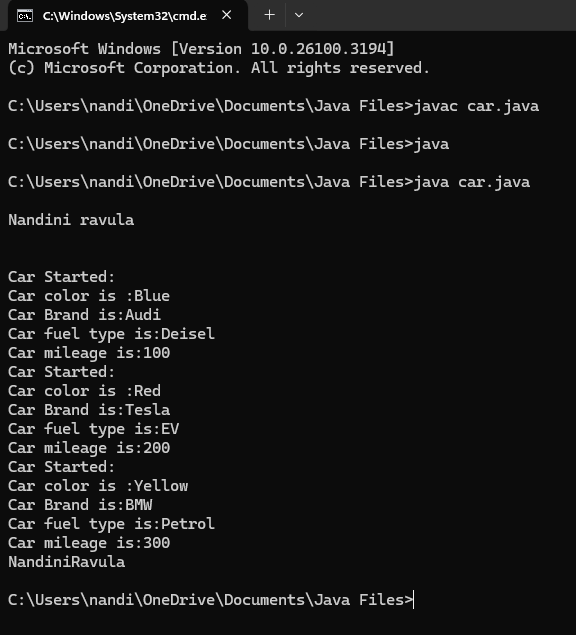
car3.service();

System.out.println;

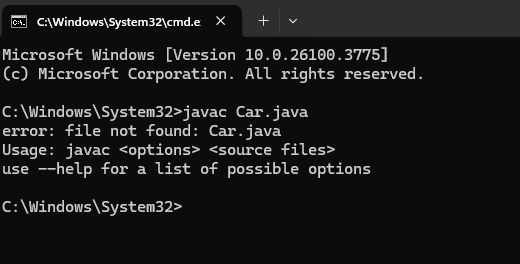
}

}

**POSITIVE OUTPUT:**

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| **1.** | **Error: car\_Color is undefined, should be car\_color** | **Replace: car\_Color with car\_color** |

**Program-2**

**Aim:** To create a class bank account with method deposit () and withdrawal ().

**CODE:**

**CLASS DIAGRAM:**

|  |
| --- |
| CLASS BANKACCOUNT |
| -balance:double |
| +void deposit()  +void withdrawal() |

**POSITIVE OUTPUT:**

public class BankAccount {

private String accountNumber;

private double balance;

public BankAccount(String accountNumber, double initialBalance) {

this.accountNumber = accountNumber;

this.balance = initialBalance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("₹" + amount + " deposited. New balance: ₹" + balance);

} else {

System.out.println("Deposit amount must be positive!");

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("₹" + amount + " withdrawn. Remaining balance: ₹" + balance);

} else {

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

}

}

public double getBalance() {

return balance;

}

public static void main(String[] args) {

BankAccount account = new BankAccount("12345", 1000);

account.deposit(500);

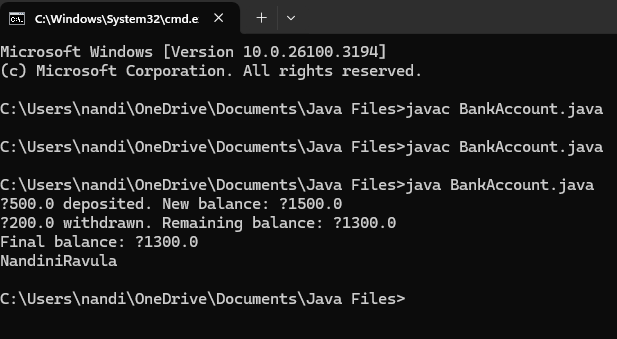
account.withdraw(200);

System.out.println("Final balance: ₹" + account.getBalance());

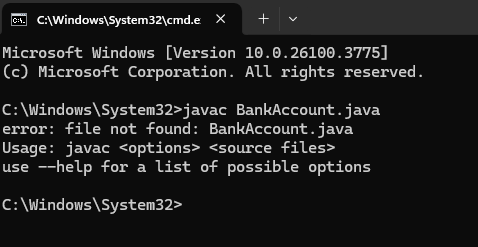
System.out.println("NandiniRavula");

}

}

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | this.existing = int.nextFloat(); | this.existing = input.nextFloat(); |

**WEEK-4**

**Program-1**

**AIM:** Write a java program with class named “book”. The class should contain various attributes such as “Title of the book”, “Author”, Year of publications”, it should also contain a constructor with parameter which initializes “Title of the book”, “Author”, “Year of publication”. Create a method which displays the details of the book. Title of the book (), Author (), Year of publication ().Display the details of two book, by creating 2 objects.

**CLASS DIAGRAM:**

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

**CODE:**

class Book {

// Declaring attributes

String title;

String author;

int yearOfPublication;

// Constructor to initialize values

Book(String title, String author, int yearOfPublication) {

this.title = title;

this.author = author;

this.yearOfPublication = yearOfPublication;

System.out.println("Your book: " + this.title);

}

// Creating a method

public void getBookDetails() {

System.out.println("Title of the book: " + this.title);

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

System.out.println("Year of publication: " + this.yearOfPublication);

}

public static void main(String[] args) {

// Creating objects for class Book

Book book1 = new Book("The Lord of Rings", "J.R.R. Tolkien", 1954);

book1.getBookDetails();

Book book2 = new Book("1984", "The Harry Potter seriesA", 1949);

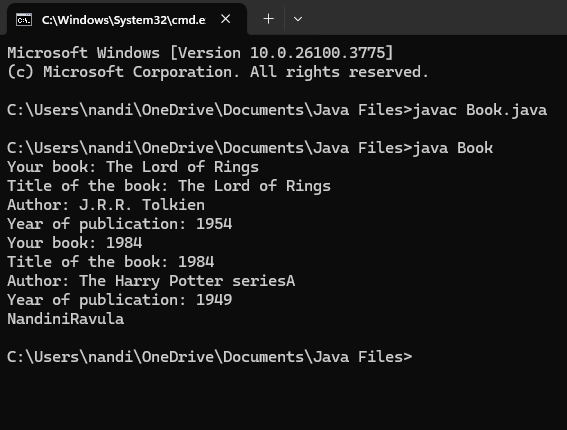
book2.getBookDetails();

System.out.println("NandiniRavula");

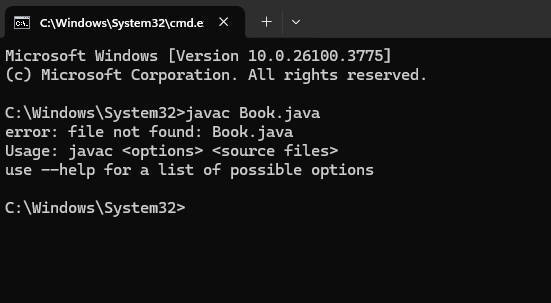
}

}

**POSITIVE OUTPUT:**



**NEGATIVE OUTPUT:**



**Error:**

|  |  |  |
| --- | --- | --- |
| SI.NO | ERROR MESSAGE | ERROR RECTIFICATION |
| 1. | Not defining the function in a file. | To call the method we must define a function in a file. |

**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.

**Program-2**

**AIM:** To create a java program with class named “Myclass” with a static variable count of “int” type, in sized to “zero” and a constant variable “pi” of type “double” initialized to 3.1415 as attributes of that class. define a contractor for “Myclass” is created finally print the final values “count” and “pi” variables.

**CLASS DIAGRAM:**

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

**CODE:**

class MyClass {

static int count = 0; //

final double pi = 3.1415;

MyClass() {

count = count + 1;

}

public void display() {

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

System.out.println("double is: " + pi);

System.out.println();

}

public static void main(String[] args) {

MyClass Asec = new MyClass();

Asec.display();

MyClass Bsec = new MyClass();

Bsec.display();

System.out.println("The final count is: " + count);

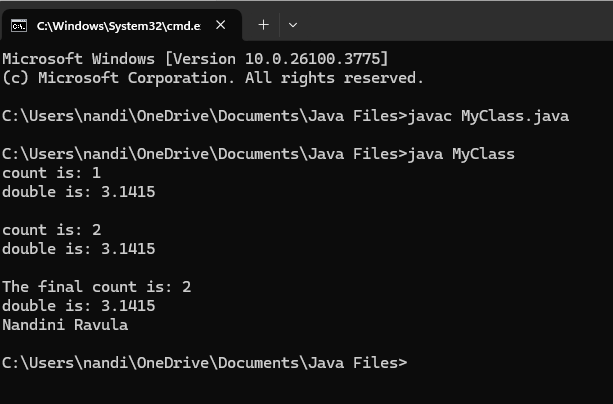
System.out.println("double is: " + Bsec.pi);

System.out.println("Nandini Ravula");

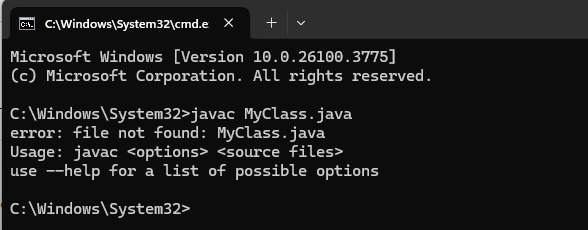
}

}

**POSITIVE OUTPUT:**



**NEGATIVE OUTPUT:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| **1.** | Not Putting the semi-colon after calling a function, | Put the semi-colon after calling a function. |

**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.

**WEEK-5**

**Program-1**

**AIM:: Create a calculator using the operations including addition, subtraction, multiplication and division using Multilevel Inheritance and display the desired output.**

**CLASS DIAGRAMS:**

|  |
| --- |
| **Calculator** |
| **- a: double**  **- b: double** |
| **+ add(): double**  **+ subtract(): double**  **+ multiply(): double**  **+ divide(): double**  **+ displayResults(): void** |

**CODE:**

import java.util.Scanner;

class Calculator {

private double a, b;

public Calculator(double a, double b) {

this.a = a;

this.b = b;

}

public double add() {

return a + b;

}

public double subtract() {

return a - b;

}

public double multiply() {

return a \* b;

}

public double divide() {

if (b == 0) {

throw new ArithmeticException("Error: Division by zero is not allowed.");

}

return a / b;

}

public void displayResults() {

System.out.println("Addition: " + add());

System.out.println("Subtraction: " + subtract());

System.out.println("Multiplication: " + multiply());

try {

System.out.println("Division: " + divide());

} catch (ArithmeticException e) {

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

}

}

}

public class AllCalculator {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

double a = input.nextDouble();

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

double b = input.nextDouble();

Calculator calc = new Calculator(a, b);

calc.displayResults();

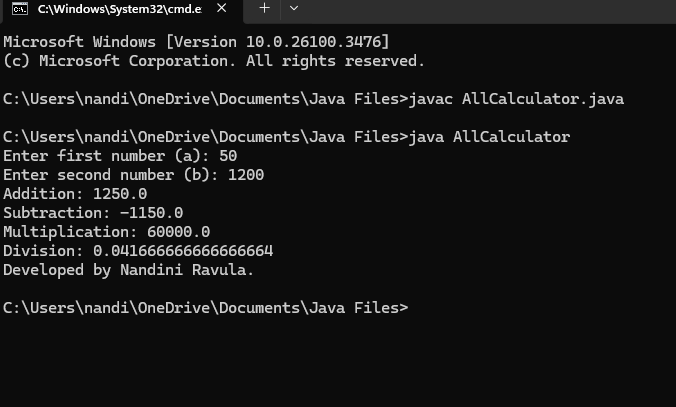
System.out.println("Developed by Sonali");

input.close();

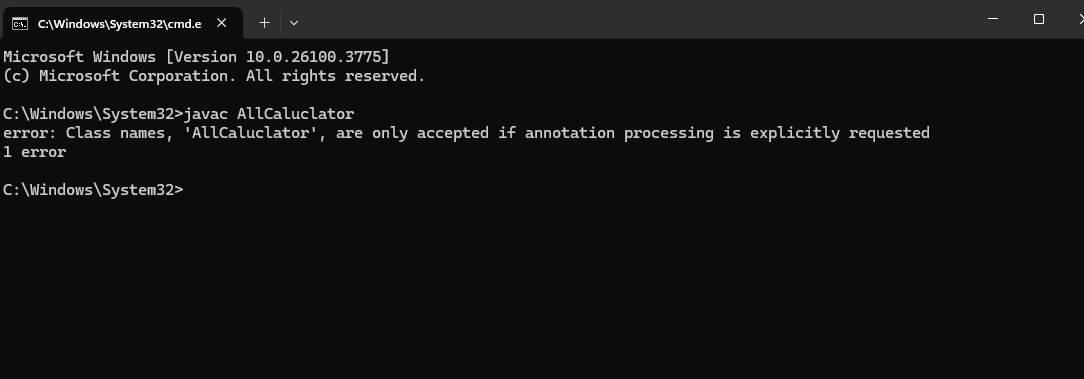
}

}

**POSITIVE OUTPUT:**



**NEGATIVE OUTPUT:**

****

**Important Points:**

Demonstrates **inheritance** with a single Calculator class

**ERROR TABLE**-

|  |  |  |
| --- | --- | --- |
| **Error** | **Possible Cause** | **Solution** |
| **Main method not found** | Class name does not match file name (Rent instead of rent) | Ensure class name and file name match (Rent.java) |

**Program-2**

**AIM:**

**A vehicle rental company wants to develop a system that maintains information about different types of vehicles 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**

**• Cars should have an additional property: number of doors**

**• Bike should have a property indicating whether they have gears or not**

**• The system should also include a function to display details about each vehicle and indicate when a vehicle is starting**

**• Every class should have a constructor**

**1. Which OOP concept is used in the above program? Explain why it is useful in this scenario.**

**2. If the company decides to add a new type of vehicle truck, how would you modify the program? Truck should include an additional property capacity(in tons). Create a showTruckDetails() method to display the truck’s capacity. Write a constructor for truck that initializes all properties.**

**3. Implement the truck class and update the main method to create a truck object, also create an object for car and bike subclassed. Finally display its details.**

**CLASS DIAGRAMS:**

|  |  |  |
| --- | --- | --- |
| **Vehicle** | | |
| **- brand: String**  **- speed: int** | | |
| **+ start(): void**  **+ showDetails(): void** | | |
| **Car** | **Bike** | **Truck** |
| **- noOfDoors: int** | **- hasGears: boolean** | **- capacity: int** |
| **+ showDetails(): void** | **+ showDetails(): void** | **+ showDetails(): void** |

**CODE:**

import java.util.Scanner;

class Vehicle {

private String brand;

private int speed;

public Vehicle(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

public void start() {

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

}

public void showDetails() {

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

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

}

}

class Car extends Vehicle {

private int noOfDoors;

public Car(String brand, int speed, int noOfDoors) {

super(brand, speed);

this.noOfDoors = noOfDoors;

}

@Override

public void showDetails() {

super.showDetails();

System.out.println("Number of Doors: " + noOfDoors);

}

}

class Bike extends Vehicle {

private boolean hasGears;

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

super(brand, speed);

this.hasGears = hasGears;

}

@Override

public void showDetails() {

super.showDetails();

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

}

}

class Truck extends Vehicle {

private int capacity;

public Truck(String brand, int speed, int capacity) {

super(brand, speed);

this.capacity = capacity;

}

@Override

public void showDetails() {

super.showDetails();

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

}

}

public class Rent {

public static void main(String[] args) {

Car car = new Car("Mercedes", 180, 4);

Bike bike = new Bike("Ducati", 140, true);

Truck truck = new Truck("Scania", 110, 15);

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

car.start();

car.showDetails();

System.out.println("\n===== Bike Details =====");

bike.start();

bike.showDetails();

System.out.println("\n===== Truck Details =====");

truck.start();

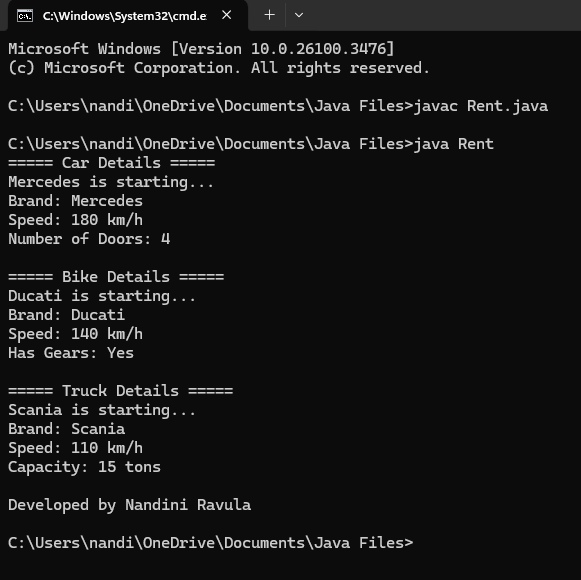
truck.showDetails();

System.out.println("\nDeveloped by Nandini Ravula");

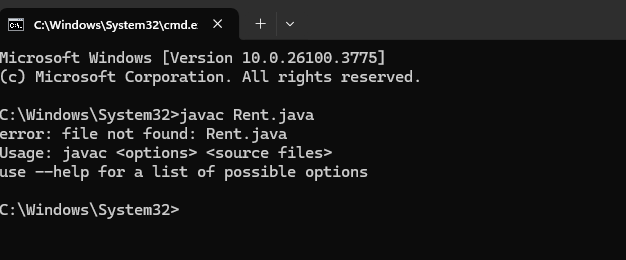
}

}

**POSITIVE OUTPUT:**



**NEGATIVE OUTPUT:**

****

Important Points:

Demonstrates **inheritance** (Car, Bike, Truck inherit from Vehicle).

Uses **method overriding** (showDetails() method in each subclass).

### **Error Table:**

| **Error** | **Possible Cause** | **Solution** |
| --- | --- | --- |
| **Method not found error** | showTruck() used in Truck, but other classes use showDetails() | Rename showTruck() to showDetails() in Truck |

**WEEK-6**

**Program-1-** **Write java program to create a vechical class with a method dispaly info().override this method in the car subclass to provide specific information about the car.**

**Class Diagram:**

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

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

**CODE**-

class Vehicle {

String name;

int wheels;

Vehicle(String name, int wheels) {

this.name = name;

this.wheels = wheels;

}

void displayInfo() {

System.out.println("This is a vehicle named: " + name + " with " + wheels + " wheels.");

}

}

class Car extends Vehicle {

Car(String name, int wheels) {

super(name, wheels);

}

@Override

void displayInfo() {

System.out.println("This is a car named: " + name + " with " + wheels + " wheels. It runs smoothly!");

}

}

class Vehicle1 {

public static void main(String[] args) {

Vehicle myVehicle = new Vehicle("Generic Vehicle", 2);

myVehicle.displayInfo();

Car myCar = new Car("Honda Civic", 4);

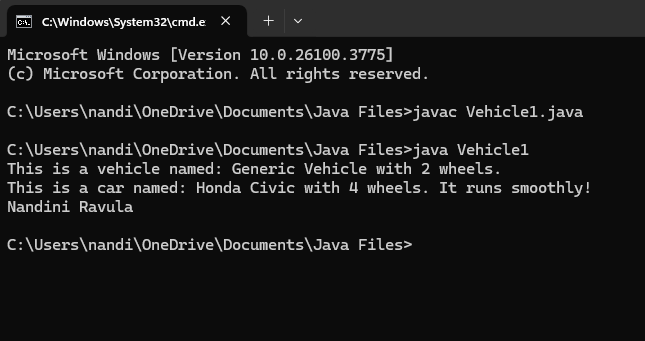
myCar.displayInfo();

System.out.println("Nandini Ravula");

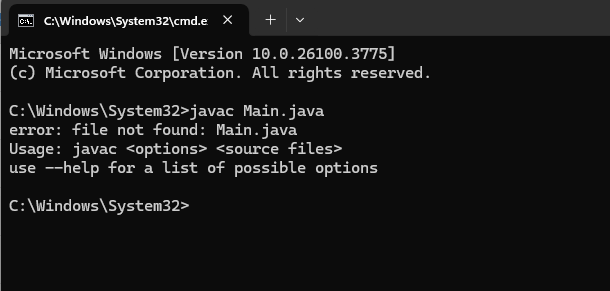
}

}

**POSITIVE OUTPUT-**



**NEGATIVE 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 |

**IMPORTANT POINTS**-

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

**PROGRAM-2:**

A college is developing an automated admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.

1. UG admission require min of 60%

2. PG admission require min of 70%

**CLASS DIAGRAM:**

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

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

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

**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 eligible for UG admission");

} else {

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

} else {

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

}

}

}

public class Admission {

public static void main(String[] args){

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

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

ug.Eligibility();

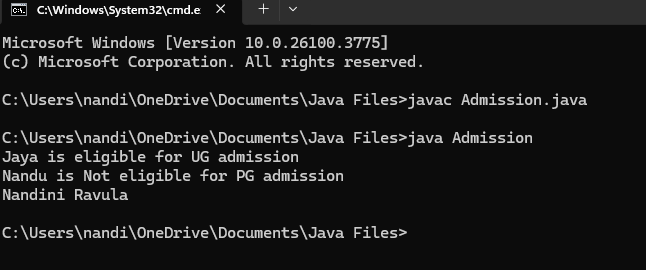
pg.Eligibility();

System.out.println("Nandini Ravula");

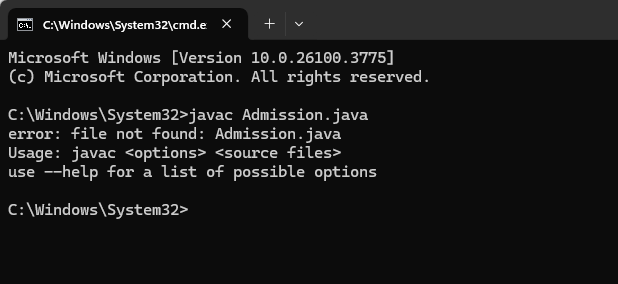
}

}

**POSITIVE OUTPUT-**



**NEGATIVE OUTPUT:**

****

IMPORTANT POINTS-

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.

**PROGRAM-3:**

3. Create a calculator class with overloaded methods to perform addition.

A. Add two integers

B. Add two double

C. Add three integer

**CLASS DIAGRAM:**

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

**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: 12 and 25 is " + calc.add(12, 25));

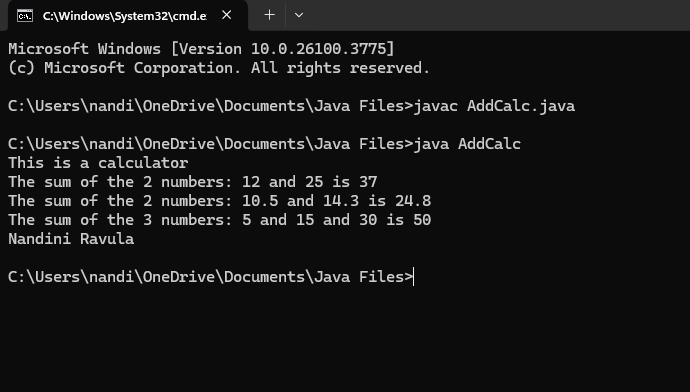
System.out.println("The sum of the 2 numbers: 10.5 and 14.3 is " + calc.add(10.5, 14.3));

System.out.println("The sum of the 3 numbers: 5 and 15 and 30 is " + calc.add(5, 15, 30));

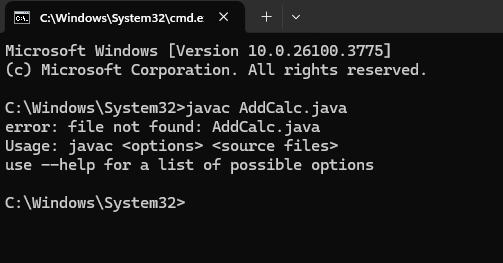
System.out.println("Nandini Ravula");

}

}

**POSITIVE OUTPUT- **

**NEGATIVE 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. |

**IMPORTANT POINTS-**

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.

3.Create a shape class with a method CalculateArea() that is overloaded for different shpaes (e.g square, rectangle) then, create a subclass circle that overrides the calculatearea() method for a circle.

**PROGRAM-4**

4)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.

**Class Diagram:**

|  |
| --- |
| **Circle** |
| + r : double  + pi : double |
| + calcArea(int r) : double |
| **Shape** |
| + calcArea(int a) : int  + calcArea(int b, int h) : int |

**CODE-**

class Shape {

public double calculatearea(double s) {

return s \* s; // Area of square

}

public int calculatearea(int l, int b) {

return l \* b; // Area of rectangle

}

}

class Circle extends Shape {

@Override

public double calculatearea(double r) {

double area = 3.14 \* r \* r;

return area;

}

}

public class Calculator1 {

public static void main(String[] args) {

Shape s = new Shape();

System.out.println("Area of square: " + s.calculatearea(3.5));

System.out.println("Area of a rectangle: " + s.calculatearea(6, 3));

Circle c = new Circle();

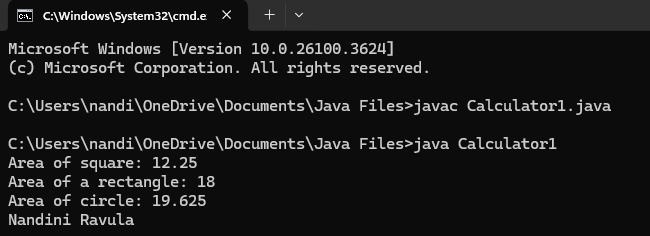
System.out.println("Area of circle: " + c.calculatearea(2.5));

System.out.println("Nandini Ravula");

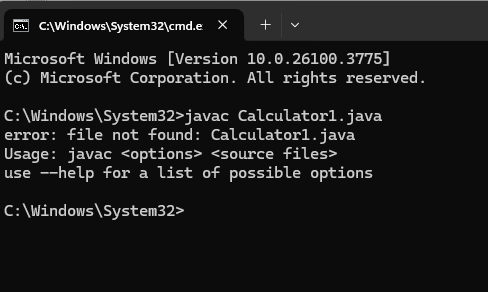
}

}

**POSITIVE OUTPUT-**

****

**NEGATIVE 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. |

**IMPORTANT POINTS**-

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**

1.Write a java program to create an abstract class animal with an abstract method called sound() .create sub class lion and tiger that extend the animal class and implement the sound method method to make a specifi sound for each animal.

**CLASS DIAGRAM**

|  |
| --- |
| abstract class |
| Animal |
| +sound(): void |

/\

/ \

|  |  |
| --- | --- |
| Lion | Tiger |
| +sound() | +sound() |

**CODE-**

abstract class Animal {

public abstract void sound();

}

class Lion extends Animal {

@Override

public void sound() {

System.out.println("Rabbit runs");

}

}

class Tiger extends Animal {

@Override

public void sound() {

System.out.println("Zebra zigzags");

}

}

public class AnimalMain {

public static void main(String[] args) {

Animal lion = new Lion();

Animal tiger = new Tiger();

lion.sound();

tiger.sound();

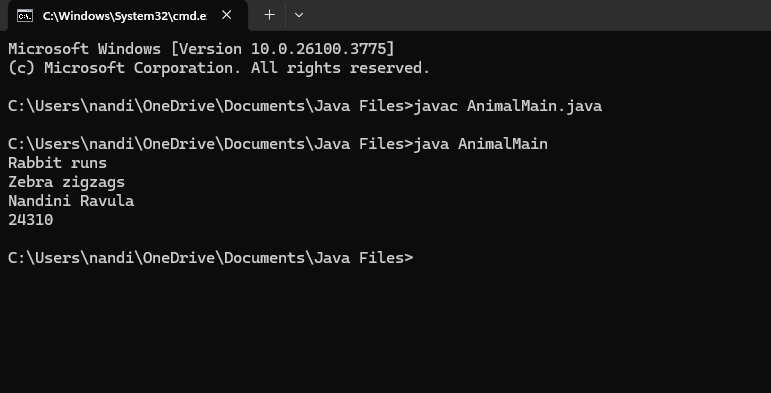
System.out.println("Nandini Ravula");

System.out.println("24310");

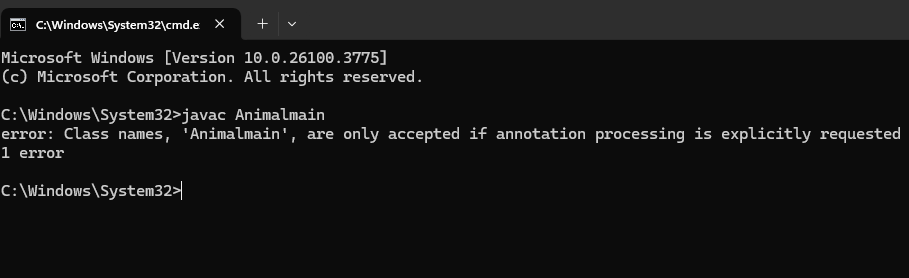
}

}

**POSITIVE OUTPUT-**

****

**NEGATIVE OUTPUT-**

****

**ERROR TABLE-**

|  |  |  |
| --- | --- | --- |
| **Error** | **Cause** | **Solution** |
| Animal is abstract... | Instantiating abstract class | Don't create object of Animal |
| Method sound() not defined | Missing @Override or method body | Implement sound() in child class |

**IMPORTANT POINTS-**

Abstract classes cannot be instantiated.

Abstract methods have no body in the abstract class.

**PROGRAM-2**

2) 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.

**CLASS DIAGRAM**

|  |
| --- |
| abstract Shape3D |
| +calculateVolume(): double  +calculateSurfaceArea(): double |

/ \

/ \

|  |  |
| --- | --- |
| Sphere | Cube |
| +calculateVolume() | |
| +calculateSurfaceArea() | |

**CODE-**

abstract class Shape3D {

// Abstract methods to calculate volume and surface area

abstract double calculateVolume();

abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

double radius;

Sphere(double radius) {

this.radius = radius;

}

@Override

double calculateVolume() {

return (4.0 / 3) \* Math.PI \* Math.pow(radius, 3);

}

@Override

double calculateSurfaceArea() {

return 4 \* Math.PI \* Math.pow(radius, 2);

}

}

class Cube extends Shape3D {

double side;

Cube(double side) {

this.side = side;

}

@Override

double calculateVolume() {

return Math.pow(side, 3);

}

@Override

double calculateSurfaceArea() {

return 6 \* Math.pow(side, 2);

}

}

public class Shape3DTest1 {

public static void main(String[] args) {

// Create Sphere and Cube objects

Sphere sphere = new Sphere(5);

Cube cube = new Cube(4);

// Calculate and print volume and surface area

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

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

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

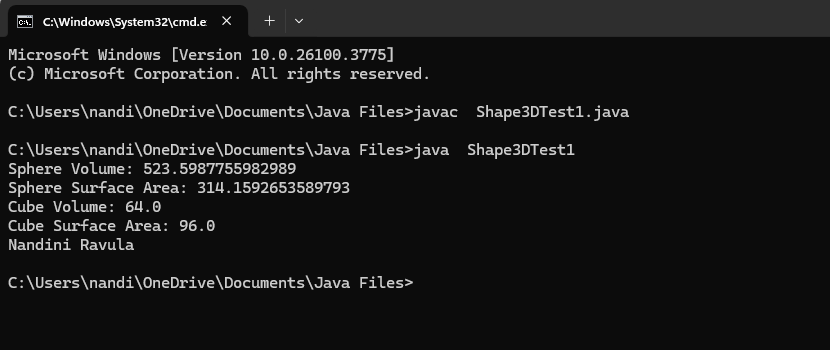
System.out.println("Cube Surface Area: " + cube.calculateSurfaceArea());

System.out.println("Nandini Ravula");

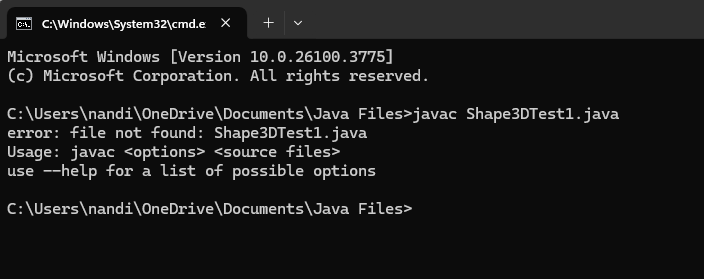
}

}

**POSITIVE OUTPUT-**

****

**NEGATIVE OUTPUT-**

****

**ERROR TABLE-**

|  |  |  |
| --- | --- | --- |
| **Error** | **Cause** | **Solution** |
| Cannot instantiate Shape3D | Shape3D is abstract | Use Sphere or Cube instead |
| calculateVolume not found | Method not implemented in subclass | Implement all abstract method |

**IMPORTANT POINTS-**

Use Math.PI for π in Java

**PROGRAM-3**

3) 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 stars

2.NumberPateern-Prints a right angled triangle increasing number

IN the main() method create objects of both subclasses and print the pattern for a given number of rows.

**CLASS DIAGRAM-**

|  |
| --- |
| abstract PatternPrinter |
| +printPattern(int n): void  +showTitle(): void |

/ \

/ \

|  |  |
| --- | --- |
| StarPattern | NumberPattern |
| +printPattern()| | +printPattern() |

**CODE-**

abstract class PatternPrinter {

abstract void printPattern(int n);

void showTitle() {

System.out.println("Pattern Printing:");

}

}

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();

}

}

}

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();

}

}

}

public class PatternTest {

public static void main(String[] args) {

// Create objects of StarPattern and NumberPattern

PatternPrinter starPattern = new StarPattern();

PatternPrinter numberPattern = new NumberPattern();

// Print pattern with 5 rows

starPattern.showTitle();

starPattern.printPattern(5);

System.out.println();

numberPattern.showTitle();

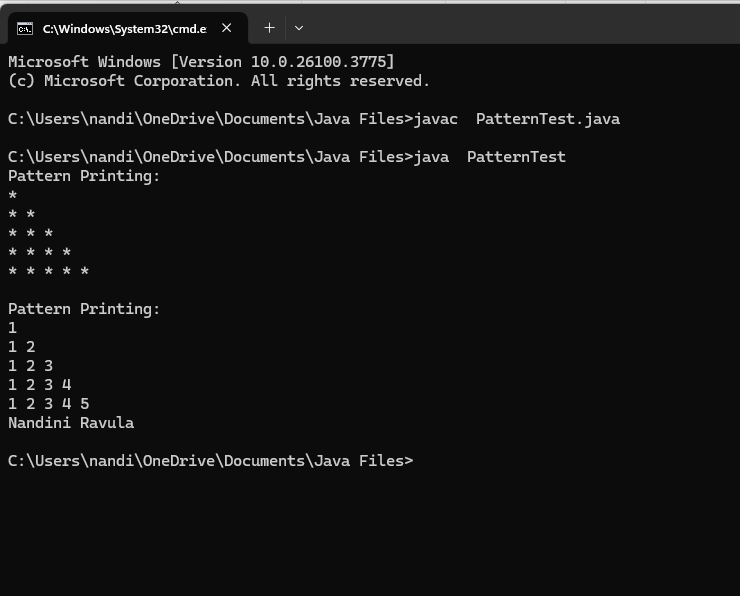
numberPattern.printPattern(5);

System.out.println("Nandini Ravula");

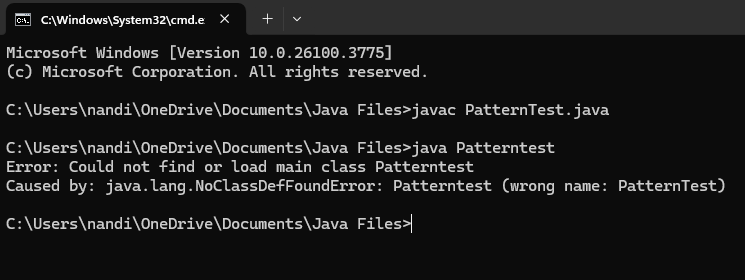
}

}

**POSITIVE OUTPUT-**

****

**NEGATIVE OUTPUT-**

****

**ERROR TABLE-**

|  |  |  |
| --- | --- | --- |
| **Error** | **Cause** | **Solution** |
| printPattern() missing | Subclass didn't override method | Use @Override and implement it |
| Wrong pattern output | Incorrect loop logic | Trace loops properl |

**IMPORTANT POINTS-**

Abstract classes can contain concrete methods (like showTitle()).

Loop structure is important for pattern printing:

**WEEK-8**

**PROGRAM-1**

1)Write a JAVA program to create an interface Shape with the getPerimeter() method. Create three classes Rectangle, Circle and Triangle that implement the

shape interface. Create three getPerimeter() method for each of the three classes.

**CLASS DIAGRAM-**

|  |
| --- |
| Shape |
| +getPerimeter() : double |

|  |
| --- |
| Rectangle |
| -length  -width |

|  |
| --- |
| Circle |
| -radius |

|  |
| --- |
| Triangle |
| -side1  -side2  -side3 |

|  |
| --- |
| Main |
| +main() |

**CODE-**

interface Shape {

double getPerimeter();

}

class Rectangle implements Shape {

double length, width;

Rectangle(double length, double width) {

this.length = length;

this.width = width;

}

public double getPerimeter() {

return 2 \* (length + width);

}

}

class Circle implements Shape {

double radius;

Circle(double radius) {

this.radius = radius;

}

public double getPerimeter() {

return 2 \* Math.PI \* radius;

}

}

class Triangle implements Shape {

double side1, side2, side3;

Triangle(double s1, double s2, double s3) {

side1 = s1;

side2 = s2;

side3 = s3;

}

public double getPerimeter() {

return side1 + side2 + side3;

}

}

public class InterfaceMain {

public static void main(String[] args) {

Shape r = new Rectangle(4, 5);

Shape c = new Circle(3);

Shape t = new Triangle(3, 4, 5);

System.out.println("Rectangle Perimeter: " + r.getPerimeter());

System.out.println("Circle Perimeter: " + c.getPerimeter());

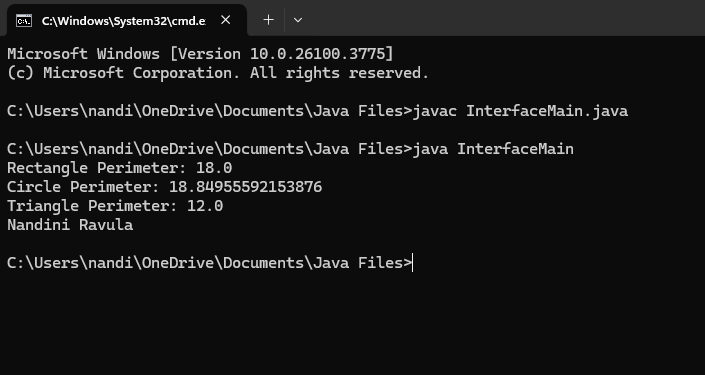
System.out.println("Triangle Perimeter: " + t.getPerimeter());

System.out.println("Nandini Ravula");

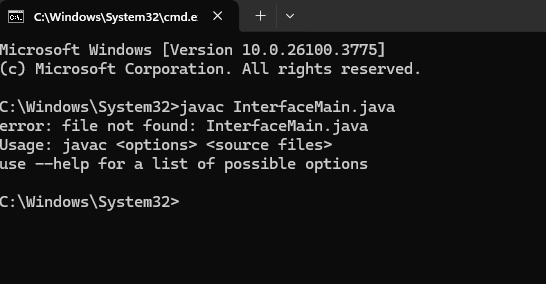
}

}

**POSITIVE OUTPUT-**

****

**NEGATIVE OUTPUT-**

****

**ERROR TABLE-**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Error | Cause | Fix |
|  | Cannot find symbol | Method/class not named properly | Check spelling |
|  | Incompatible types | Using int where double is expected | Use double consistently |
|  | No main method | Main class missing main method | Add public static void main() |

**IMPORTANT POINTS-**

An interface defines method signature only.

Classes implement interface and override the method.

**PROGRAM-2**

2) Write a JAVA program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football , Volleyball and Basketball that implement

the Playable interface and override the play() method to play the respective sports.

**CLASS DIAGRAM-**

|  |
| --- |
| Playable |
| +play() |

|  |
| --- |
| Football |
| +play() |

|  |
| --- |
| Volleyball |
| +play() |

|  |
| --- |
| Basketball |
| +play() |

|  |
| --- |
| Main |
| +main() |

**CODE-**

interface Playable {

void play();

}

class Football implements Playable {

public void play() {

System.out.println("Playing Football");

}

}

class Volleyball implements Playable {

public void play() {

System.out.println("Playing Volleyball");

}

}

class Basketball implements Playable {

public void play() {

System.out.println("Playing Basketball");

}

}

public class PlayMain {

public static void main(String[] args) {

Playable p1 = new Football();

Playable p2 = new Volleyball();

Playable p3 = new Basketball();

p1.play();

p2.play();

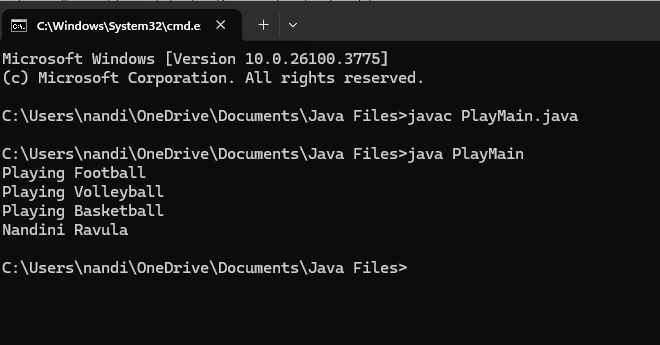
p3.play();

System.out.println("Nandini Ravula");

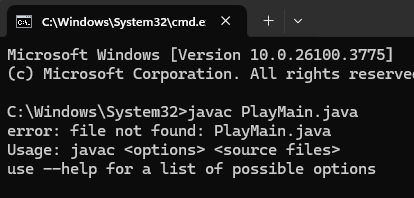
}

}

**POSITIVE OUTPUT-**

****

**NEGATIVE OUTPUT-**

****

**ERROR TABLE-**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error | Cause | Fix |
| 1 | Missing method | Didn’t override play() | Use public void play() correctly |
| 2 | Interface not implemented | Forgot implements Playable | Add implements keyword |
| 3 | Class not found | Misspelling in class name | Double-check names |

**IMPORTANT POINTS-**

Interface Playable is used for defining standard action play().

Each class defines its own version of the play() method.