

(23CSE111) object-OrenteD Programing LAB MANUAL

CSE-1st YEAR I SEMESTER (2024-2025)

SUBMITTED From: SUBMITTED To:

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | G. Chaitanya Raja  Reddy | **NAME** | **Dr.Raj Kumar Batchu** |
| **ROLL NO** | **AV.SC.U4CSE24112** | **DEPARTMAENT** | **Ogject-Orented\_programing** |
| **SECTION** | **CSE-B** | **DESIGNATION** | **PROFESSOR** |

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

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TITLE** | **PAGES** | **SIGN** |
| week1. | Installing jdk  And running the basic information about a student | 1-5 |  |
| week2. | 1.Factorial series.  2.Simple Intreset.  3.Fibonacci Series.  4.Area of Triangle using herons, Formula and Area of Rectangle.  5.convertion from celcius to farenheit. | 5-11 |  |
| week3. | 1.About Car Class.  2.About BankAccounts class | 11-18 |  |
| week4. | 1.About Book.  2.About Myclass. |  |  |
| week5. | 1.Calculator  2.Vechicles |  |  |
| week6. | . 1.Vechiles override.  2.PG and UG admission  3.Calculator.  4.Area Calculation |  |  |
| week7. |  |  |  |
| week8. |  |  |  |
| week9. | . |  |  |
| week10. |  |  |  |

**How to download and install java software in your computer.**

**1.Steps for downloading.**

Step 1: java is downloaded from the official Oracle website,open web browser and search jdk download

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

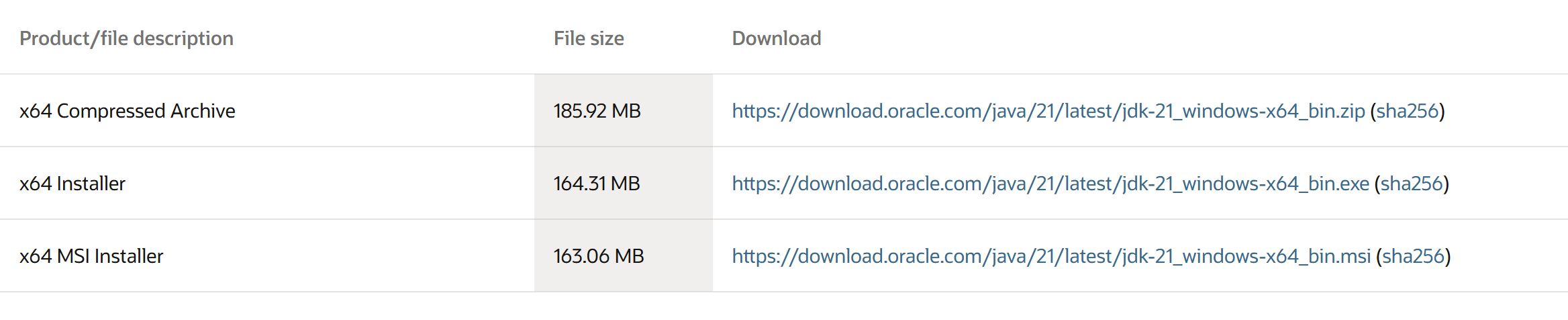
Step 3: And navigate to the java downloads. Then some earlier versions are available like

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

Step 4:Now click on JDK 21 .this is the latest Long-Term Support (LTS) release of the Java SE Platform

Step 5: then select the type of operating system of your device

Step 6:now press the second link as shown in the image

****

Click on x64 installer link

**2.Steps for installation.**

Step 1 : after downloading that link go to the folder where it was downloaded

Step 2: then open it and accept all terms and conditions

Step 3: then instal it

**3.Setting environmental variables.**

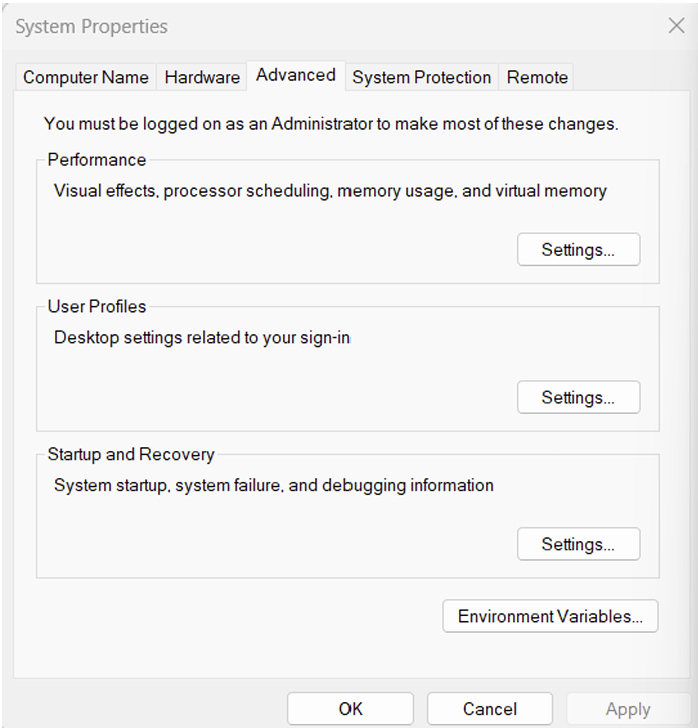
Step 1: Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on Environment Variables.

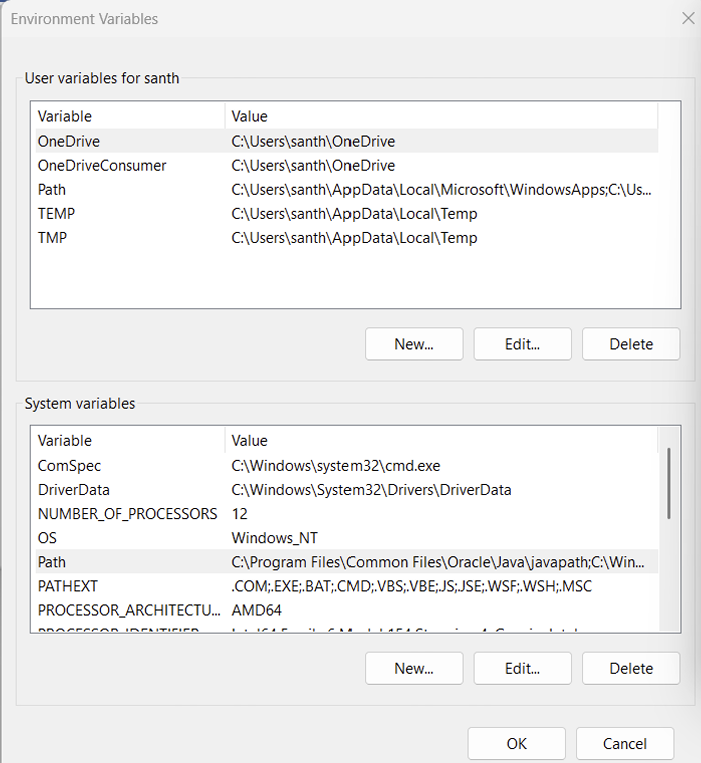
Step 2: Click New under System Variables:

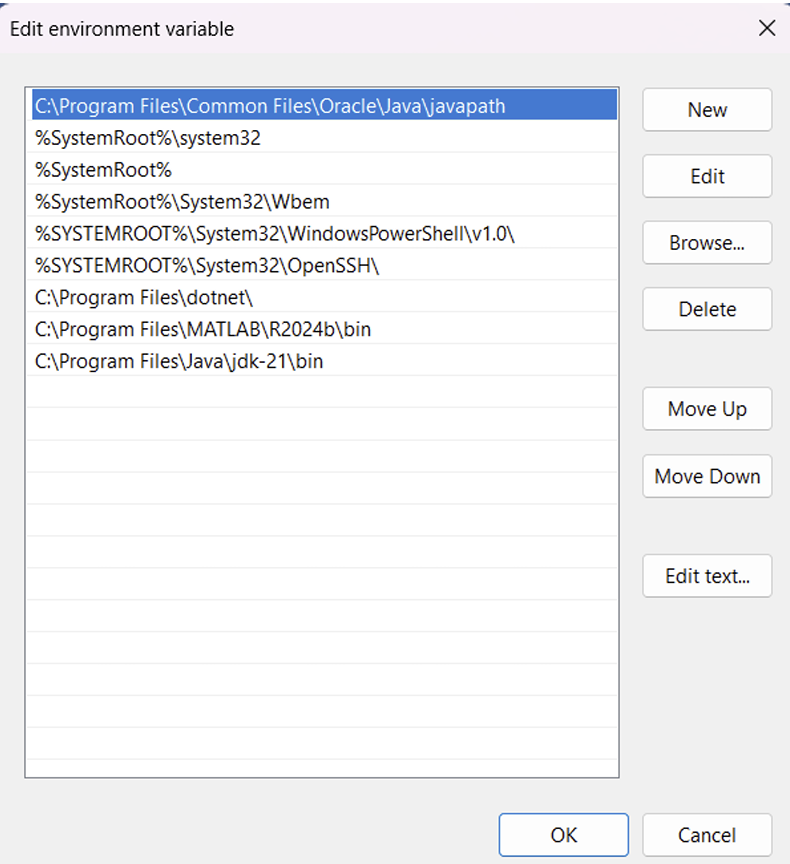
Set variable name as:java\_

Variable value: The folder address where JDK is installed (like C:\Program Files\Java\jdk-21\bin)

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

****

****

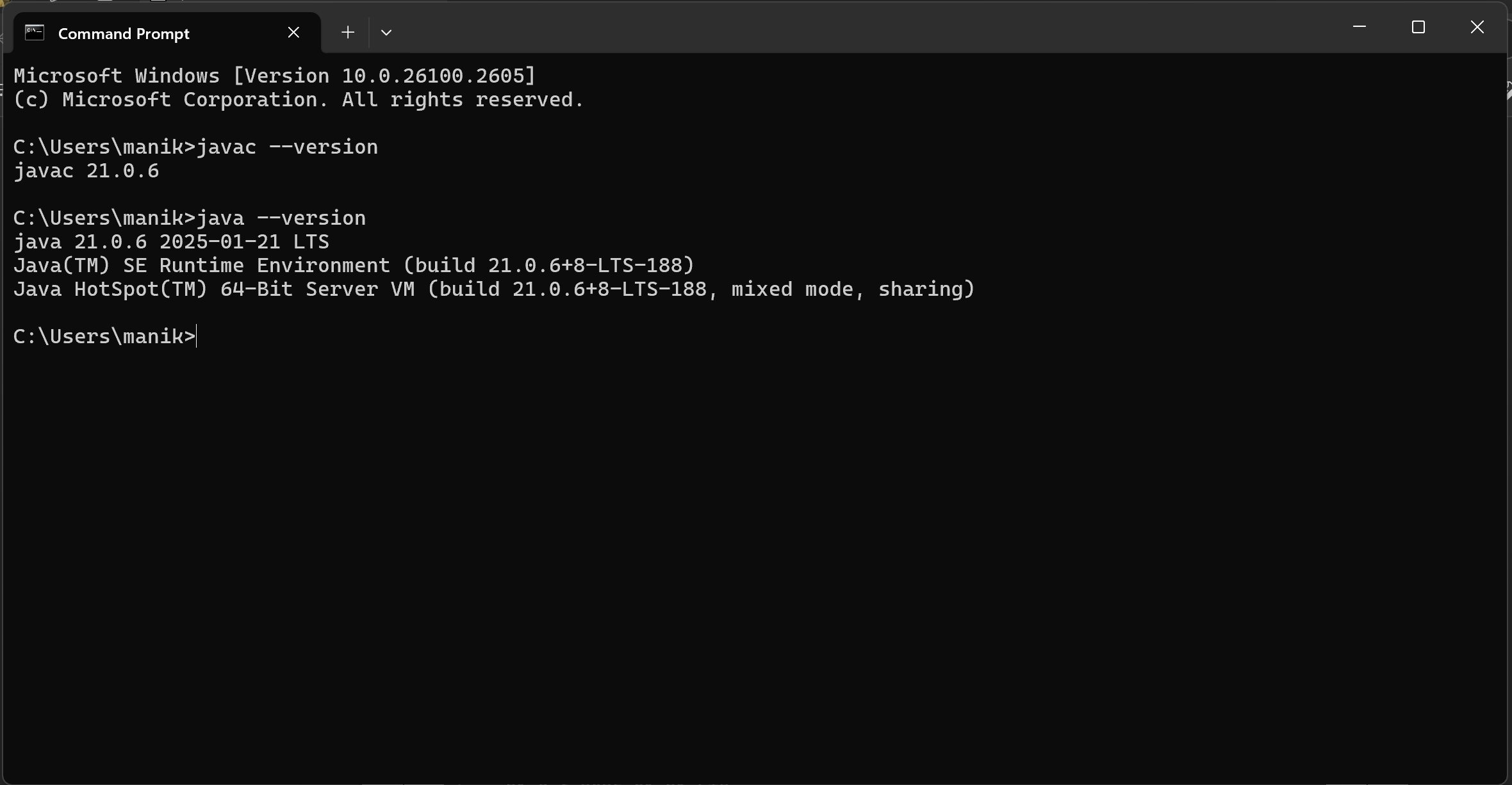
****

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

Step 1: Open command promt.

Step 2: Enter javac --version in the command promt for version of jdk installed

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

****

**WRITE A SIMPLE JAVA CODE FOR YOUR NAME,SECTION,ROLL NO AND BRANCH**

**class Me**

**{**

**public static void main(String[] args)**

**{**

**System.out.println("NAME=G.Chaitanya Raja Reddy");**

**System.out.println("SECTION=b");**

**System.out.println("ROLL NO=av.sc.u4cse24145");**

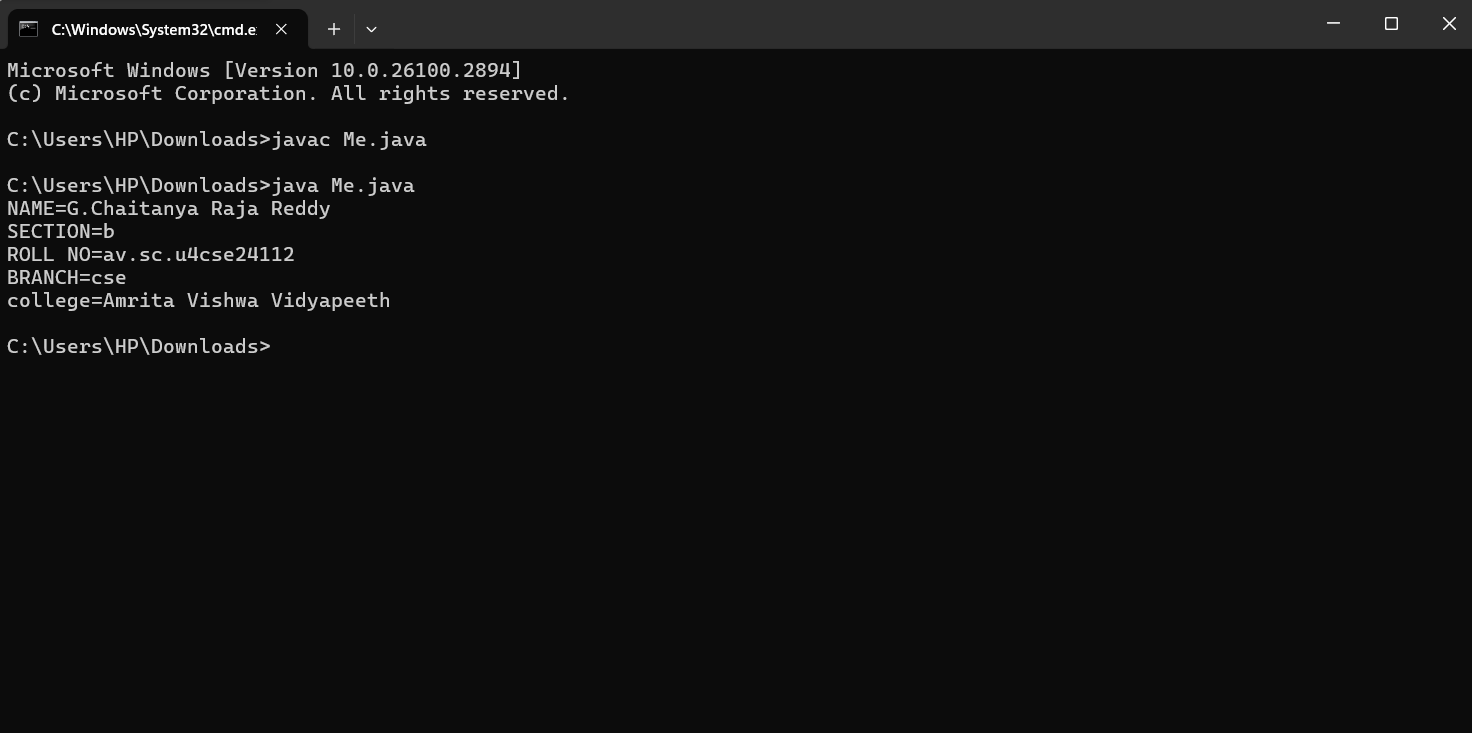
**System.out.println("BRANCH=cse");**

**System.out.println(“college=Amrita Vishwa Vidyapeeth”)**

**}**

**}**

**Output:**

****

**Week-2**

**1.Factrioal series**

**Code:**

**import java.util.Scanner;**

**class FactorialSeries {**

**public static void main(String[] args) {**

**Scanner input = new Scanner(System.in);**

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

**int n = input.nextInt();**

**int fact = 1;**

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

**fact \*= i;**

**System.out.println(i + "! = " + fact);**

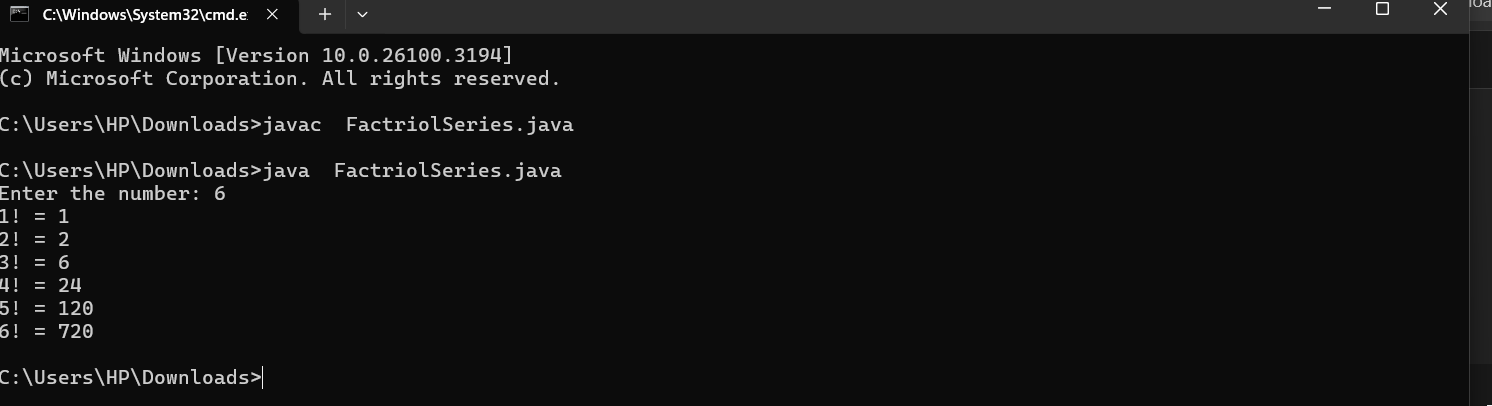
**}**

**input.close();**

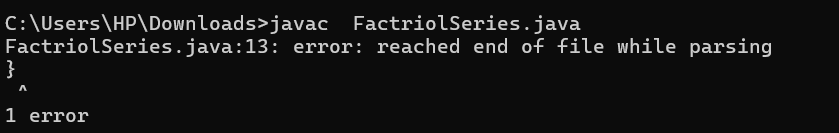
**}**

**}**

**Output:**

****

**Errors:**



**2.Simple Interest**

**Code:**

**import java.util.Scanner;**

**class SimpleInterest {**

**public static void main(String[] args) {**

**Scanner input = new Scanner(System.in);**

**// Simple Interest Calculation**

**System.out.print("Enter Principal amount: ");**

**double principal = input.nextDouble();**

**System.out.print("Enter Rate of interest: ");**

**double rate = input.nextDouble();**

**System.out.print("Enter Time (in years): ");**

**double time = input.nextDouble();**

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

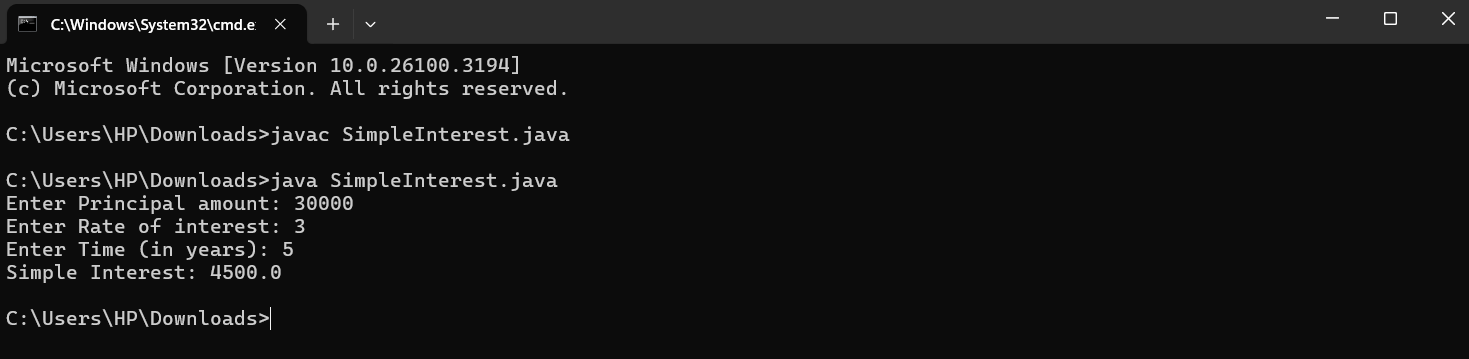
**System.out.println("Simple Interest: " + simpleInterest);**

**input.close();**

**}**

**}**

**Output:**

****

(No errors occurred in this code )

**3.Fibnocci Series:**

**Code:**

**import java.util.Scanner;**

**class FibonacciSeries {**

**public static void main(String[] args) {**

**Scanner input = new Scanner(System.in);**

**System.out.print("Enter number of terms for Fibonacci series: ");**

**int n = input.nextInt();**

**int a = 0, b = 1;**

**System.out.print("Fibonacci Series: " + a + " " + b);**

**for (int i = 3; i <= n; i++) {**

**int next = a + b;**

**System.out.print(" " + next);**

**a = b;**

**b = next;**

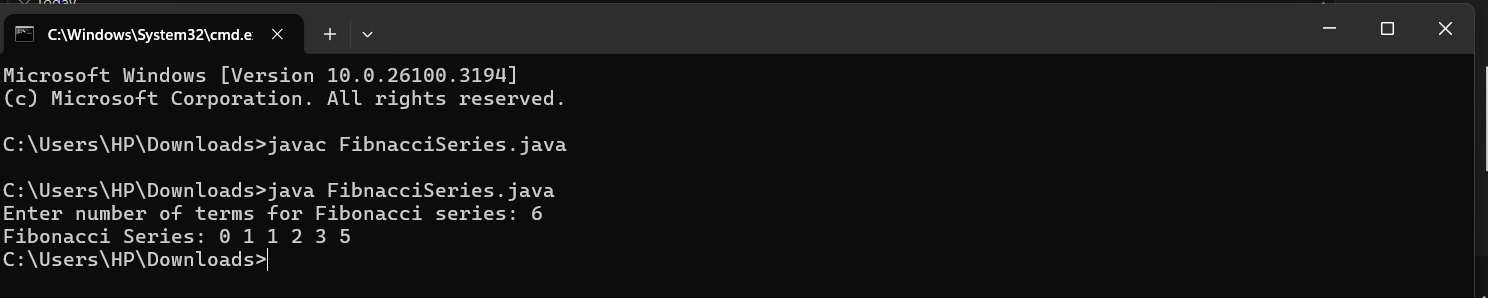
**}**

**input.close();**

**}**

**}**

**Output:**

****

**Errors table and rectification**

|  |  |
| --- | --- |
| **1.Giving space between next and Double.**  **2.Not giving parenthesis after closing the input.** | **1.Should not give space between next and Double.**  **2.We must put parenthesis after closing the input.** |

**4.Area of Triangle:**

**Code:**

**import java.util.Scanner;**

**public class HeronsFormula {**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

**System.out.print("Enter side a: ");**

**double a = sc.nextDouble();**

**System.out.print("Enter side b: ");**

**double b = sc.nextDouble();**

**System.out.print("Enter side c: ");**

**double c = sc.nextDouble();**

**double s = (a + b + c) / 2; // Semi-perimeter**

**double area = Math.sqrt(s \* (s - a) \* (s - b) \* (s - c));**

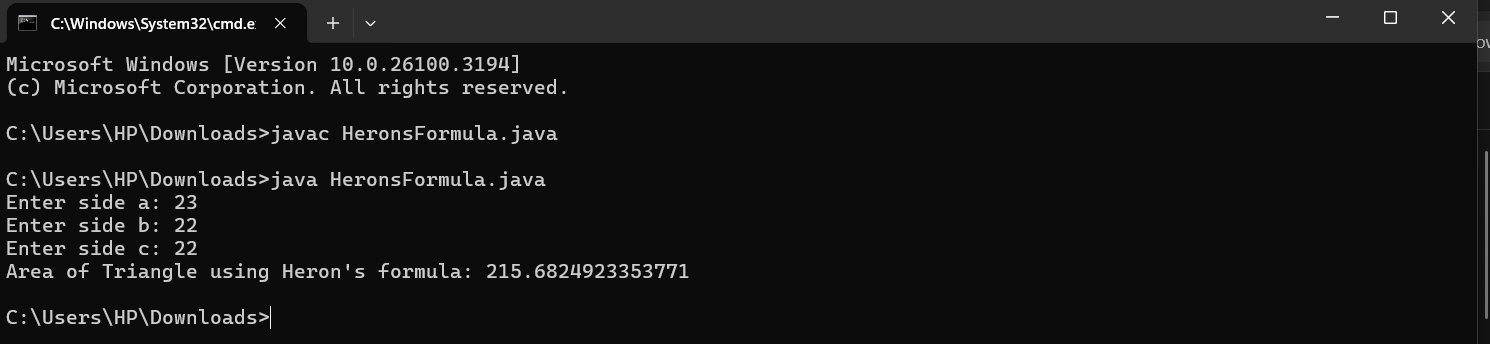
**System.out.println("Area of Triangle using Heron's formula: " + area);**

**sc.close();**

**}**

**}**

**Output:**

****

(No errors occurred in this code )

**Area of Rectangle:**

**Code:**

**import java.util.Scanner;**

**public class RectangleArea {**

**public static void main(String[] args) {**

**Scanner sc = new Scanner(System.in);**

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

**double length = sc.nextDouble();**

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

**double breadth = sc.nextDouble();**

**double area = length \* breadth;**

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

**sc.close();**

**}**

**}**

**Output:**

****

(No errors occurred in this code )

**5.Convert of temperature:**

**Code:**

**import java.util.Scanner;**

**class TemperatureConverter {**

**public static void main(String[] args) {**

**Scanner input = new Scanner(System.in);**

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

**double celsius = input.nextDouble();**

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

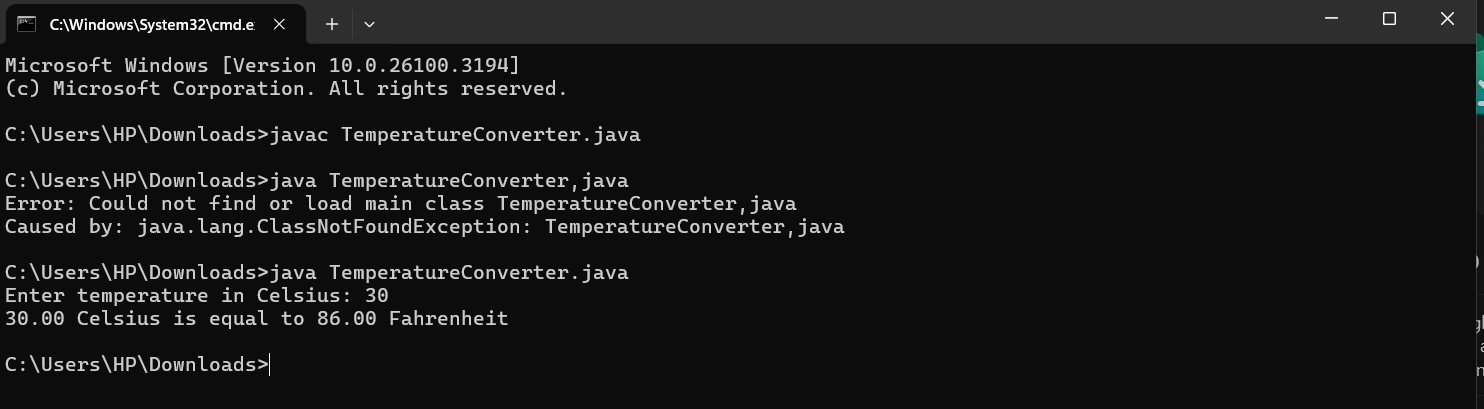
**System.out.printf("%.2f Celsius is equal to %.2f Fahrenheit%n", celsius, fahrenheit);**

**input.close();**

**}**

**}**

**Output:**



(No errors occurred in this code )

**Week-3**

Write a Java program with the following instructions:

1. Create a class with name Car

2. Create four attributes

car colour

car brand

mileage

injector type

3. Create three methods

start

stop

service

4. Create three objects C1, C2, C3

5. Create a constructor with parameters

.car color

.car brand

.milage

.fuel type

Code:

class Car{

String car\_color;

String car\_brand;

String fuel\_type;

String milage;

public void start(){

System.out.println("the car started with a roaring dragon sound....");

}

public void stop(){

System.out.println("the car as stopped with a great skid....");

}

public void service(){

System.out.println("the went to garage with a huge accident at our road side pole.....");

}

public static void main(String[] args){

Car C1 = new Car();

C1.car\_color = "navy blue";

C1.car\_brand = "roles roys";

C1.fuel\_type = "CNG gas,Deseil";

C1.milage = "15km/hr";

System.out.println("the car color is" + C1.car\_color);

System.out.println( "the brand of car is" + C1.car\_brand);

System.out.println("the fuel type is" + C1.fuel\_type);

System.out.println("the milage give by the car is" + C1.milage);

Car C2 = new Car();

C2.car\_color = " blue";

C2.car\_brand = "BMW";

C2.fuel\_type = "Deseil";

C2.milage = "17km/hr";

System.out.println("the car color is" + C2.car\_color);

System.out.println( "the brand of car is" + C2.car\_brand);

System.out.println("the fuel type is" + C2.fuel\_type);

System.out.println("the milage give by the car is" + C2.milage);

Car C3 = new Car();

C3.car\_color = " red";

C3.car\_brand = "Audi";

C3.fuel\_type = "Deseil";

C3.milage = "20km/hr";

System.out.println("the car color is" + C3.car\_color);

System.out.println( "the brand of car is" + C3.car\_brand);

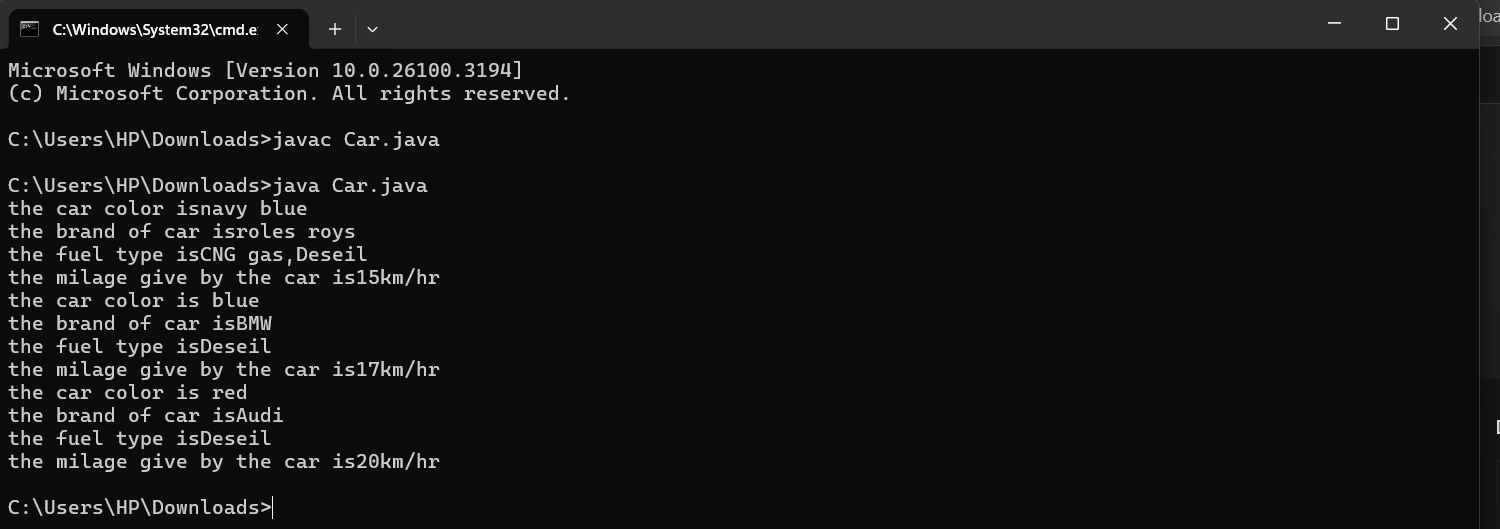
System.out.println("the fuel type is" + C3.fuel\_type);

System.out.println("the milage give by the car is" + C3.milage);

}

}

Output:



**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Error name | Cause of error | Rectification |
| 1 | Syntax Error | Missing ‘{‘ | ‘{‘ added |
| 2 | Compile time Error | Mispelled Variable call | Rectified with  Correct variable name |
| 3 | Case sensitive error | Uppercase and lowercase | rectified |

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.

Class diagram:

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

**2.:**

**Create a class named BankAccount with methods deposit and withdraw, where:**

**The deposit method should accept a parameter amt. When this method is called, the deposit amount should be added to the current balance.**

**The withdraw method should accept a parameter amt. In addition to that, when it is called, it has to verify whether the amount is less than the current balance. If yes, then the current balance is deducted. If not, display a message saying that there is a risk of insufficient funds.**

**Use the constructor to display the details of the customer: (customer name, account number, IFSC, branch).**

**Code:**

**public class BankAccount {**

**protected String accountHolder;**

**protected double balance;**

**protected 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 = balance - amount;**

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

**} else {**

**System.out.println("Insufficient funds");**

**}**

**}**

**public void deposit(double amount) {**

**balance = balance + amount;**

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

**}**

**public static void main(String[] args) {**

**BankAccount BA = new BankAccount("Abdul", 24248, 1000);**

**BA.withdrawal(500);**

**BA.deposit(1500);**

**}**

**}**

**Output:**

****

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Error name** | **Error name** | **Rectification** |
| **1** | **Name Error** | **Undefined name** | **Correct variable**  **Name replaced** |
| **2** | **Syntax Error** | **Missing Parenthesis** | **Parenthesis Added** |

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** |

**WEEK-4**

**1.** **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 {**

**String title;**

**String author;**

**int year\_of\_publication;**

**public Book(String title, String author, int year\_of\_publication) {**

**this.title = title;**

**this.author = author;**

**this.year\_of\_publication = year\_of\_publication;**

**}**

**public void displayDetails() {**

**System.out.println(this.title);**

**System.out.println(this.author);**

**System.out.println(this.year\_of\_publication);**

**System.out.println();**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**Book book\_one = new Book("The Magic of the lost Story", "Sudha Murthy", 2022);**

**Book book\_two = new Book("Three thousand stitches", "Sudha Murthy", 2021);**

**System.out.println("Book one details: ");**

**book\_one.displayDetails();**

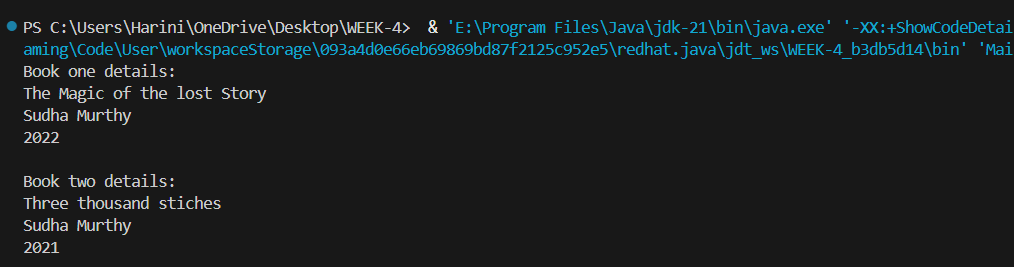
**System.out.println("Book two details: ");**

**book\_two.displayDetails();**

**}**

**}**

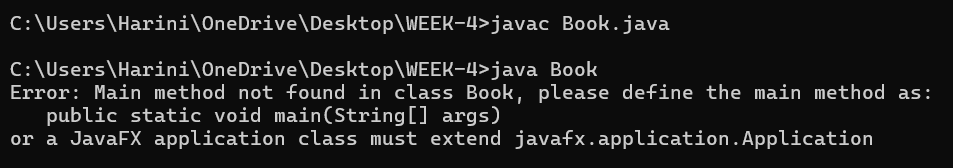
**OUTPUT:**



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 |

**2.** 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 obj1 = new myclass();**

**myclass obj2 = new myclass();**

**myclass obj3 = new myclass();**

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

**System.out.println("Value of pi: " + obj1.pi);**

**System.out.println("Value of pi: " + obj2.pi);**

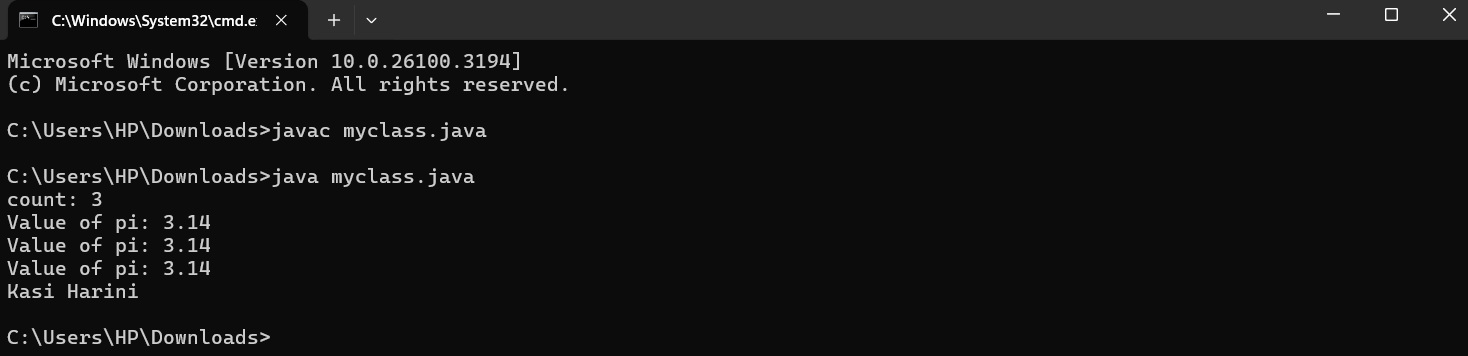
**System.out.println("Value of pi: " + obj3.pi);**

**System.out.println("Kasi Harini");**

**}**

**}**

**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. |

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**

**1.** **Create a calculator using the operations including**

**addition, subtraction, multiplication and division using**

**multilevel in heritance and display the desired output**

**Code:**

**import java.util.Scanner;**

**class Calculator {**

**public void add(double a, double b) {**

**System.out.println("Sum of two numbers is: " + (a + b));**

**}**

**public void subs(double a, double b) {**

**System.out.println("Difference of two numbers is: " + (a - b));**

**}**

**}**

**class Calculator1 extends Calculator {**

**public void mul(double a, double b) {**

**System.out.println("Multiplication of two numbers is: " + (a \* b));**

**}**

**}**

**class Calculator2 extends Calculator1 {**

**public void div(double a, double b) {**

**if (b == 0) {**

**System.out.println("Division is not allowed when b = 0");**

**} else {**

**System.out.println("Division of two numbers is: " + (a / b));**

**}**

**}**

**}**

**public class Cals {**

**public static void main(String[] args) {**

**Calculator2 C = new Calculator2();**

**Scanner input = new Scanner(System.in);**

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

**double a = input.nextDouble();**

**double b = input.nextDouble();**

**C.add(a, b);**

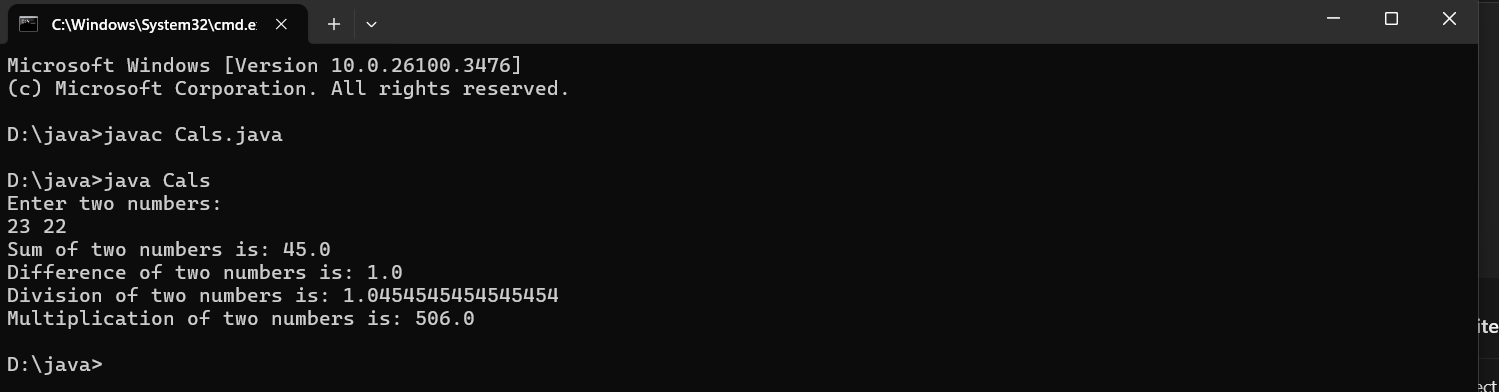
**C.subs(a, b);**

**C.div(a, b);**

**C.mul(a, b);**

**}**

**}**



**Important points:**

**In order to do this, we have to use inheritance concept. Here**

**we used the multi-inheritance concept.**

**Class Diagram:**

**+-----------------------------+**

**| Calculator |**

**+-----------------------------+**

**| + add(double, double): void |**

**| + subs(double, double): void|**

**+-----------------------------+**

**▲**

**|**

**+-----------------------------+**

**| Calculator1 |**

**+-----------------------------+**

**| + mul(double, double): void |**

**+-----------------------------+**

**▲**

**|**

**+-----------------------------+**

**| Calculator2 |**

**+-----------------------------+**

**| + div(double, double): void |**

**+-----------------------------+**

**+-----------------------------+**

**| Cals |**

**+-----------------------------+**

**| + main(String[] args): void |**

**+-----------------------------+**

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Error name** | **Error name** | **Rectification** |
| **1** | **Name Error** | **Undefined name** | **Correct variable**  **Name replaced** |
| **2** | **Syntax Error** | **Missing Parenthesis** | **Parenthesis Added** |

**2.** **A vehicle rental company wants to develop a system**

**that maintains information about diterent 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.**

**i. Cars should have an additional property: number of doors,**

**Seating capacity.**

**ii. Bikes should have a property indicating whether they have**

**gears or not.**

**iii. The system should also include a function to display**

**details about each vehicle and indicate when a vehicle is**

**starting.**

**iv. Each class should have a constructor.**

**Questions:**

**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?**

**a. Truck should include and additional property capacity**

**(in tons).**

**36**

**b. Create a showTruck() 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 subclasses. Finally display the details**

**Code:**

**class Vehicle {**

**String brand;**

**int speed;**

**Vehicle(String brand, int speed) {**

**this.brand = brand;**

**this.speed = speed;**

**}**

**void Details() {**

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

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

**}**

**}**

**class Cars extends Vehicle {**

**int doors;**

**int capacity;**

**public Cars(String brand, int speed, int doors, int capacity) {**

**super(brand, speed);**

**this.doors = doors;**

**this.capacity = capacity;**

**}**

**void cardetails() {**

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

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

**}**

**}**

**class Bikes extends Vehicle {**

**boolean gears;**

**Bikes(String brand, int speed, boolean gears) {**

**super(brand, speed);**

**this.gears = gears;**

**}**

**void bikedetails() {**

**if (gears)**

**System.out.println("This bike has gears.");**

**else**

**System.out.println("This bike does not have a gear system.");**

**}**

**}**

**class Trucks extends Vehicle {**

**int tons;**

**Trucks(String brand, int speed, int tons) {**

**super(brand, speed);**

**this.tons = tons;**

**}**

**void truckdetails() {**

**System.out.println("The capacity of the truck is: " + tons + " tons");**

**}**

**}**

**public class Rent {**

**public static void main(String[] args) {**

**Cars c = new Cars("Toyota", 120, 5, 5);**

**c.cardetails();**

**c.Details();**

**Bikes b = new Bikes("KTM", 80, true);**

**b.bikedetails();**

**b.Details();**

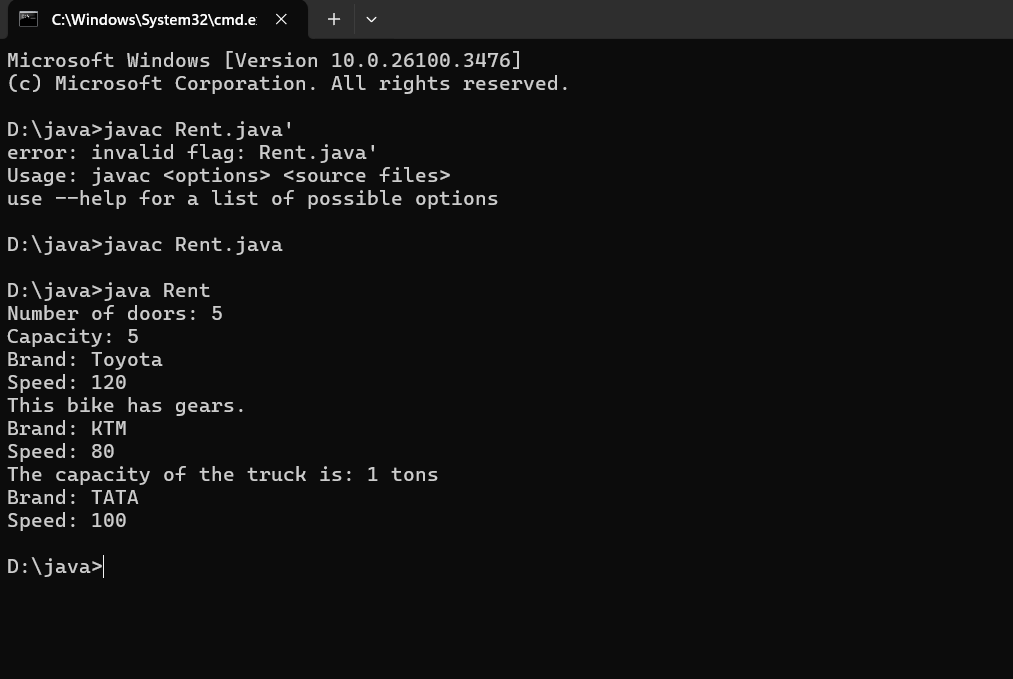
**Trucks t = new Trucks("TATA", 100, 1);**

**t.truckdetails();**

**t.Details();**

**}**

**}**



**Class Diagram:**

**+-----------------------------+**

**| Vehicle |**

**+-----------------------------+**

**| - brand: String |**

**| - speed: int |**

**+-----------------------------+**

**| + Details(): void |**

**+-----------------------------+**

**▲ ▲ ▲**

**| | |**

**+------------------+ +------------------+ +------------------+**

**| Cars | | Bikes | | Trucks |**

**+------------------+ +------------------+ +------------------+**

**| - doors: int | | - gears: boolean | | - tons: int |**

**| - capacity: int | +------------------+ +------------------+**

**| + cardetails() | | + bikedetails() | | + truckdetails() |**

**+------------------+ +------------------+ +------------------+**

**+-----------------------------+**

**| Rent |**

**+-----------------------------+**

**| + main(String[]): void |**

**+-----------------------------+**

**Important points:**

**Multi-inheritance: It is one of the types of the inheritance**

**where subclass 2 inherits subclass1 and subclass1 inherits**

**superclass.**

**Here Vehicle is the super class or parent class and**

**remaining cars, bikes, trucks are the subclasses or child**

**classes.**

**Errors :**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Error name | Cause of error | Rectification |
| 1 | Syntax Error | Missing ‘{‘ | ‘{‘ added |
| 2 | Compile time Error | Mispelled Variable call | Rectified with  Correct variable name |
| 3 | Case sensitive error | Uppercase and lowercase | rectified |

**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 about a car, model,**

**fuel type, and colour using the constructor**

**Code:**

**class Vehicle {**

**String brand;**

**int speed;**

**public Vehicle() {**

**this.brand = "Unknown";**

**this.speed = 0;**

**}**

**void displayInfo() {**

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

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

**}**

**}**

**class Car extends Vehicle {**

**String model;**

**String company;**

**int mileage;**

**String fuelType;**

**public Car(String brand, int speed, String model, String company, int mileage, String fuelType) {**

**this.brand = brand;**

**this.speed = speed;**

**this.model = model;**

**this.company = company;**

**this.mileage = mileage;**

**this.fuelType = fuelType;**

**}**

**@Override**

**void displayInfo() {**

**super.displayInfo();**

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

**System.out.println("Company: " + company);**

**System.out.println("Mileage: " + mileage);**

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

**}**

**}**

**public class Help {**

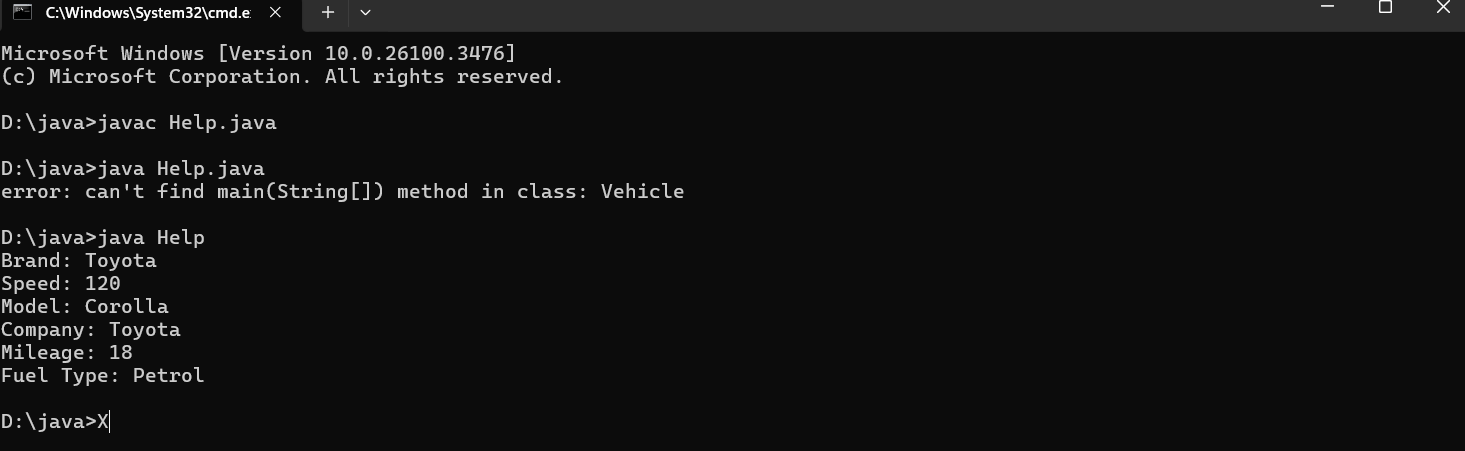
**public static void main(String[] args) {**

**Car car = new Car("Toyota", 120, "Corolla", "Toyota", 18, "Petrol");**

**car.displayInfo();**

**}**

**}**



**Class Diagram:**

**+------------------+**

**| Vehicle |**

**+------------------+**

**| - brand: String |**

**| - speed: int |**

**+------------------+**

**| + Vehicle() |**

**| + displayInfo() |**

**+------------------+**

**▲**

**|**

**|**

**+------------------------------+**

**| Car |**

**+------------------------------+**

**| - model: String |**

**| - company: String |**

**| - mileage: int |**

**| - fuelType: String |**

**+------------------------------+**

**| + Car(brand, speed, ...) |**

**| + displayInfo() |**

**+------------------------------+**

**Main Method:**

**+----------------------------+**

**| Help |**

**+----------------------------+**

**| + main(String[] args) |**

**+----------------------------+**

**Important points:**

**In order to do this, we have to use inheritance concept. Here**

**we used the multi-inheritance concept.**

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. |

**2.** **Create a Java program for the scenario.**

**A college is developing an automated admission system that**

**verifies student eligibility for undergraduate (UG) and**

**postgraduate(PG) programs. Each program has diterent**

**eligibility criteria based on the student's percentage in their**

**previous qualification.**

**i) UG admissions require a minimum of 60%**

**ii) PG admissions require a minimum of 70%**

**Code:**

**class AdmissionSystem {**

**String studentName;**

**double percentage;**

**public AdmissionSystem(String studentName, double percentage) {**

**this.studentName = studentName;**

**this.percentage = percentage;**

**}**

**void checkEligibility() {**

**System.out.println("Checking eligibility for: " + studentName);**

**}**

**}**

**class UGAdmission extends AdmissionSystem {**

**public UGAdmission(String studentName, double percentage) {**

**super(studentName, percentage);**

**}**

**@Override**

**void checkEligibility() {**

**super.checkEligibility();**

**if (percentage >= 60) {**

**System.out.println(studentName + " is eligible for UG admission.");**

**} else {**

**System.out.println(studentName + " is not eligible for UG admission.");**

**}**

**}**

**}**

**class PGAdmission extends AdmissionSystem {**

**public PGAdmission(String studentName, double percentage) {**

**super(studentName, percentage);**

**}**

**@Override**

**void checkEligibility() {**

**super.checkEligibility();**

**if (percentage >= 70) {**

**System.out.println(studentName + " is eligible for PG admission.");**

**} else {**

**System.out.println(studentName + " is not eligible for PG admission.");**

**}**

**}**

**}**

**public class Main {**

**public static void main(String[] args) {**

**UGAdmission ugStudent = new UGAdmission("John", 65);**

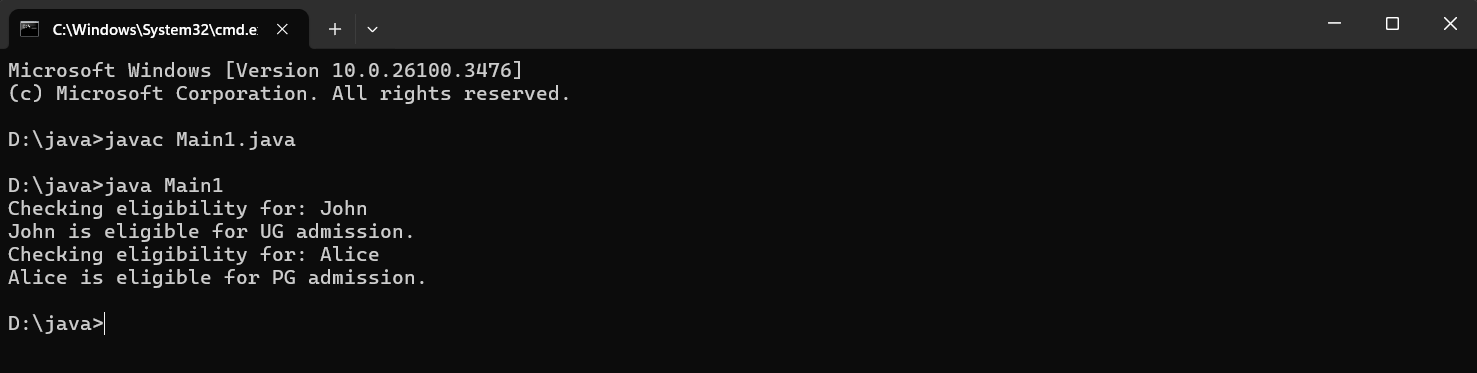
**ugStudent.checkEligibility();**

**PGAdmission pgStudent = new PGAdmission("Alice", 72);**

**pgStudent.checkEligibility();**

**}**

**}**



**+-------------------------------+**

**| AdmissionSystem |**

**+-------------------------------+**

**| - studentName: String |**

**| - percentage: double |**

**+-------------------------------+**

**| + AdmissionSystem(name, %) |**

**| + checkEligibility() |**

**+-------------------------------+**

**▲ ▲**

**| |**

**+------------------+ +------------------+**

**| UGAdmission | | PGAdmission |**

**+------------------+ +------------------+**

**| + UGAdmission(...)| | + PGAdmission(...)|**

**| + checkEligibility()| | + checkEligibility()|**

**+------------------+ +------------------+**

**Main Method:**

**+------------------------------+**

**| Main |**

**+------------------------------+**

**| + main(String[] args) |**

**+------------------------------+**

**Important points:**

**Super keyword is used take the method,variable,constructor**

**from the super class.**

**Errors:**

|  |  |
| --- | --- |
| **1.Giving space between next and Double.**  **2.Not giving parenthesis after closing the input.** | **1.Should not give space between next and Double.**  **2.We must put parenthesis after closing the input.** |

**`**

**3.** **Write a Java Program to create a Calculator class with**

**overloaded methods to perform addition: Take the integer**

**values a and b from the user.**

**i) Add two integers**

**ii) Add two doubles**

**iii) Add three integer**

**Code:** **class Calculatoroverloading {**

**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 loading {**

**public static void main(String[] args) {**

**Calculatoroverloading c = new Calculatoroverloading();**

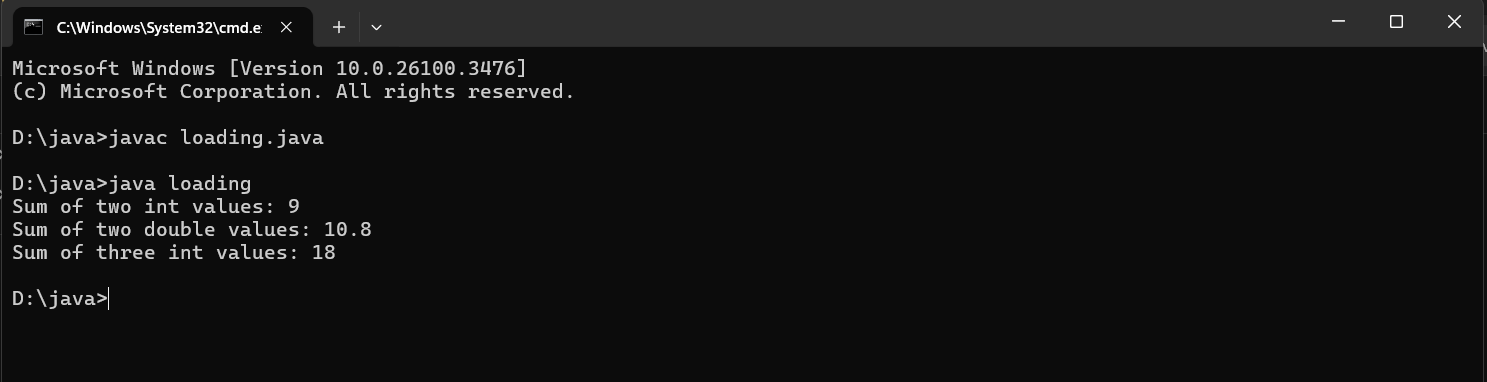
**System.out.println("Sum of two int values: " + c.add(5, 4));**

**System.out.println("Sum of two double values: " + c.add(5.9, 4.9));**

**System.out.println("Sum of three int values: " + c.add(5, 4, 9));**

**}**

**}**



**Class Diagram:**

**+------------------------------------+**

**| Calculatoroverloading |**

**+------------------------------------+**

**| + add(int a, int b): int |**

**| + add(double a, double b): double |**

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

**+------------------------------------+**

**▲**

**|**

**+------------------------------+**

**| loading |**

**+------------------------------+**

**| + main(String[] args): void |**

**+------------------------------+**

**Important points:**

**We should carefully pass the double and integer and**

**diterent types of input to an constructor when creating an**

**object to access the diterent constructors based on the**

**parameter.**

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.no** | **Error name** | **Error name** | **Rectification** |
| **1** | **Name Error** | **Undefined name** | **Correct variable**  **Name replaced** |
| **2** | **Syntax Error** | **Missing Parenthesis** | **Parenthesis Added** |

**4.** **Write a Java Program to create a shape class with a**

**method calculateArea() that is overloaded for diterent**

**shapes(e.g., Square, Rectangle ). Then create a subclass**

**Circle that overrides the**

**calculateArea() method for a circle.**

**Code:**

**class Shape {**

**void calculatearea(int l, int b) {**

**int area = l \* b;**

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

**}**

**void calculatearea(int x) {**

**int area = x \* x;**

**System.out.println("Area of Square is: " + area);**

**}**

**}**

**class Circle {**

**public void calculatearea(double pi, double r) {**

**double area = pi \* (r \* r);**

**System.out.println("Area of circle is: " + area);**

**}**

**}**

**public class Areas2 {**

**public static void main(String[] args) {**

**Shape s = new Shape();**

**Circle c = new Circle();**

**s.calculatearea(4, 5);**

**s.calculatearea(4);**

**c.calculatearea(3.1415, 1);**

**}**

**}**



**Class Diagram:**

**+---------------------------+**

**| Shape |**

**+---------------------------+**

**| + calculatearea(int, int): void | ← Rectangle**

**| + calculatearea(int): void | ← Square**

**+---------------------------+**

**+---------------------------+**

**| Circle |**

**+---------------------------+**

**| + calculatearea(double, double): void | ← Circle**

**+---------------------------+**

**+---------------------------+**

**| Areas2 |**

**+---------------------------+**

**| + main(String[] args): void |**

**+---------------------------+**

**Important points:**

**In this program we use both method overloading and**

**overriding to calculate area of diterent shapes.**

**Errors:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Error name | Cause of error | Rectification |
| 1 | Syntax Error | Missing ‘{‘ | ‘{‘ added |
| 2 | Compile time Error | Mispelled Variable call | Rectified with  Correct variable name |
| 3 | Case sensitive error | Uppercase and lowercase | rectified |

**Week 7**

**1.** **abstract class Animal {**

**abstract void sound();**

**}**

**class Lion extends Animal {**

**public void sound() {**

**System.out.println("Lion Roars Wildly");**

**}**

**}**

**class Tiger extends Animal {**

**public void sound() {**

**System.out.println("Tiger is making sound but less than Lion");**

**}**

**}**

**public class AnimalSounds {**

**public static void main(String[] args) {**

**System.out.println("name Chaitanya; roll no 24112; Sec CSE-B ");**

**Lion lion = new Lion();**

**lion.sound();**

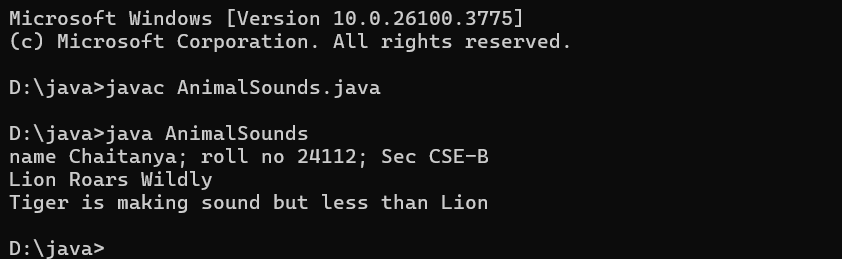
**Tiger tiger = new Tiger();**

**tiger.sound();**

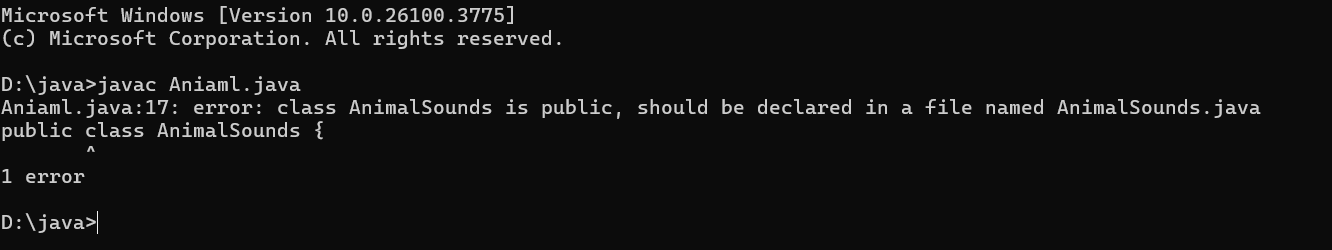
**}**

**}**

**Output:**

****

**Errors:**

****

**Rectification:**

**Saving name with wrong name.**

**Important Points:**

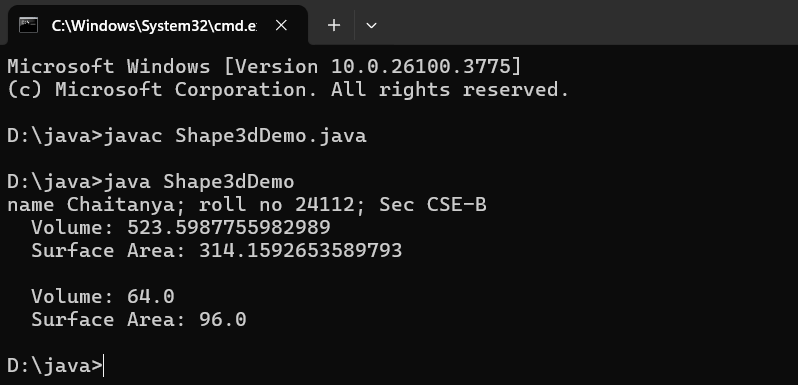
1. **Abstract Class Usage: The Animal class is declared as abstract, meaning it cannot be instantiated directly and must be extended. It defines a common structure (method sound()) that all animal types must implement.**
2. **Polymorphism: The use of the abstract class as a reference type (Animal lion = new Lion();) showcases polymorphism, allowing us to call the overridden sound() method dynamically at runtime.**

**2.** **write a java program to create an abstract class Shape3d with abstract method calculateVolume() and Calculate SurfaceArea() Create subclasses Sphere and Cube that extends the shape3d class and implements the respective method to calculate the volume and surface area of type shape.**

**Code:**

**abstract class Shape3d {  
     
    public abstract double calculateVolume();  
  
     
    public abstract double calculateSurfaceArea();  
}  
  
  
class Sphere extends Shape3d {  
    private double radius;  
  
    public Sphere(double radius) {  
        this.radius = radius;  
    }  
  
     
    public double calculateVolume() {  
        return (4.0/3.0) \* Math.PI \* Math.pow(radius, 3);  
    }  
  
     
    public double calculateSurfaceArea() {  
        return 4 \* Math.PI \* Math.pow(radius, 2);  
    }  
}  
  
  
class Cube extends Shape3d {  
    private double side;  
  
    public Cube(double side) {  
        this.side = side;  
    }  
  
     
    public double calculateVolume() {  
        return Math.pow(side, 3);  
    }  
  
     
    public double calculateSurfaceArea() {  
        return 6 \* Math.pow(side, 2);  
    }  
}  
  
public class Shape3dDemo {  
    public static void main(String[] args) {  
         
        Sphere sphere = new Sphere(5.0);  
         
         
        System.out.println("  Volume: " + sphere.calculateVolume());  
        System.out.println("  Surface Area: " + sphere.calculateSurfaceArea());  
        System.out.println();  
  
         
        Cube cube = new Cube(4.0);  
         
         
        System.out.println("  Volume: " + cube.calculateVolume());  
        System.out.println("  Surface Area: " + cube.calculateSurfaceArea());  
    }  
}**

**Ouput:**

****

**Errors:**

**No errors occurred in this code.**

**Important Points:**

**1. Abstract Method Implementation: The Shape3D class declares two abstract methods: calculateVolume() and calculateSurfaceArea(). All subclasses must implement these methods, ensuring a consistent interface for 3D shapes.**

**2. Encapsulation of Properties: Sphere and Cube each manage their own properties (radius and side), which supports encapsulation and makes the program more modular and scalable.**

**3.** **Write a Java program using an abstract class to define a method for printing patterns.**

**1. Create an abstract class named PatternPrinter with:**

**An abstract method printPattern(int n)**

**A concrete method to display the pattern title**

**2. Implement 2 subclasses:**

**Subclass 1 (Star Pattern): Prints a right-angled triangle of stars (\*)**

**Subclass 2 (Number Pattern): Prints a right-angled triangle of numbers**

**3. In the main method, create objects of both subclasses and print the pattern for a given number of rows.**

**Code:**

**abstract class PatternPrinter {**

**abstract void printPattern(int n);**

**void displayTitle(String title) {**

**System.out.println("Pattern: " + title);**

**}**

**}**

**class StarPattern extends PatternPrinter {**

**@Override**

**void printPattern(int n) {**

**displayTitle("Star Pattern");**

**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) {**

**displayTitle("Number Pattern");**

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

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

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

**}**

**System.out.println();**

**}**

**}**

**}**

**public class PatternDemo {**

**public static void main(String[] args) {**

**System.out.println("name Chaitanya; roll no 24112; Sec CSE-B ");**

**int rows = 5; // You can take input if needed**

**PatternPrinter star = new StarPattern();**

**PatternPrinter number = new NumberPattern();**

**star.printPattern(rows);**

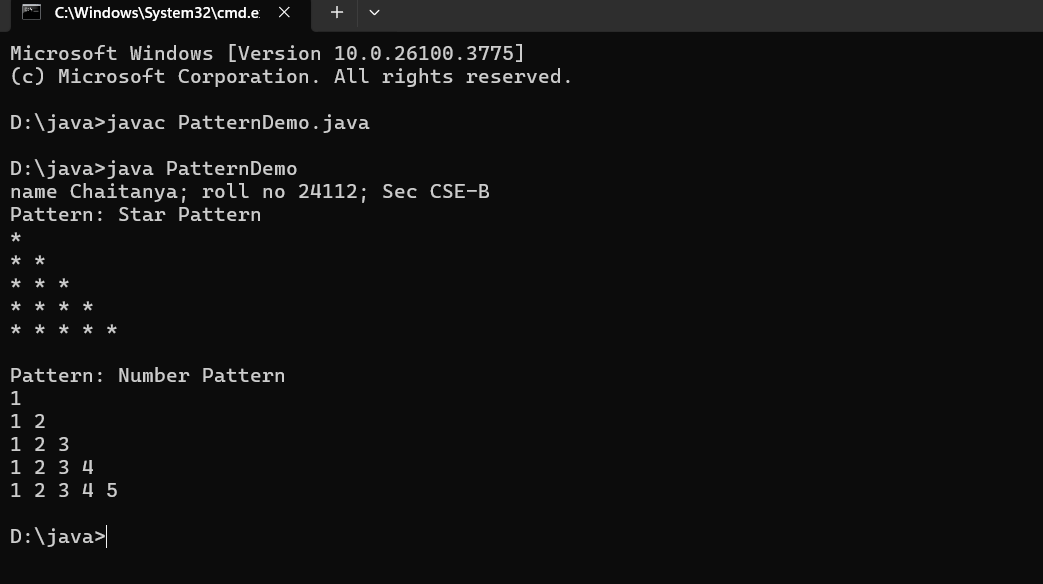
**System.out.println();**

**number.printPattern(rows);**

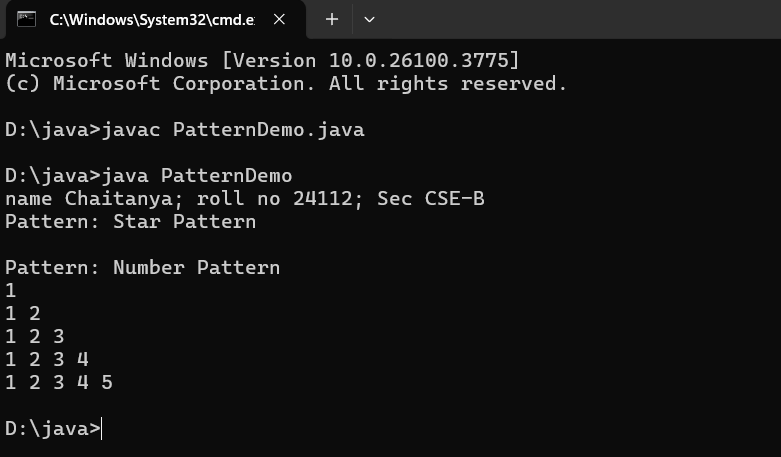
**}**

**}**

**Output:**

****

**Error:**

****

**Rectification:**

**Sign Correction**

**Importat Points:**

1. **Abstraction with Polymorphism: The PatternPrinter abstract class enforces a contract (printPattern(int n)) that all subclasses must implement. This ensures a consistent interface while allowing different behaviors (e.g., star vs number pattern).**
2. **Code Reuse with Concrete Method: The displayTitle() method is a concrete method defined in the abstract class, which all subclasses can use without redefining. This promotes code reuse and avoids duplication.**

**Week-8**

1. **Aim : 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. Implement the getPerimeter() method for each of the three classes.**

**Code:**

**interface Shape {**

**double getPerimeter();**

**}**

**class Rectangle implements Shape {**

**private double length;**

**private double width;public Rectangle(double length, double width) {**

**this.length = length;**

**this.width = width;**

**}**

**@Override**

**public double getPerimeter() {**

**return 2 \* (length + width);**

**}**

**}**

**class Circle implements Shape {**

**private double radius;**

**public Circle(double radius) {**

**this.radius = radius;**

**}**

**@Override**

**public double getPerimeter() {**

**return 2 \* Math.PI \* radius;**

**}**

**}**

**class Triangle implements Shape {**

**private double side1;**

**private double side2;**

**private double side3;**

**public Triangle(double side1, double side2, double side3) {**

**this.side1 = side1;**

**this.side2 = side2;**

**this.side3 = side3;**

**}**

**@Override**

**public double getPerimeter() {**

**return side1 + side2 + side3;**

**}**

**}**

**public class q20 {**

**public static void main(String[] args) {**

**System.out.println("Name Ajay;Roll No 24124;Sec CSe-B");**

**Rectangle rectangle = new Rectangle(10, 5);Circle circle = new Circle(7);**

**Triangle triangle = new Triangle(6, 8, 10);**

**System.out.println("Rectangle Perimeter: " +**

**rectangle.getPerimeter());**

**System.out.println("Circle Perimeter: " +**

**circle.getPerimeter());**

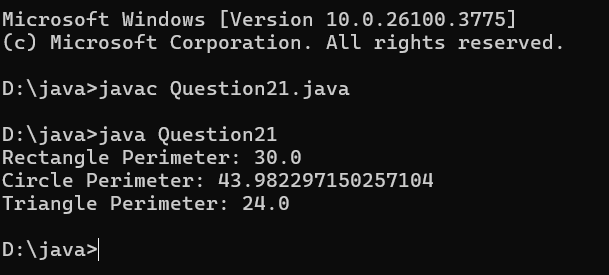
**System.out.println("Triangle Perimeter: " +**

**triangle.getPerimeter());**

**}**

**}**

**Output:**



**Errors:**

Compile type errors occurred.

**Important Points:**

○ Interface: Defines a contract for classes that implement it.

○ Implementing an Interface: Classes must provide implementations for all methods

declared in the interface.

○ Polymorphism: Objects of different classes can be treated uniformly through the

Shape interface.

**2)Aim:** 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.

Code:

interface Playable {

void play();

}

class Football implements Playable {

@Override

public void play() {

System.out.println("Playing football with a team of 11

players.");

}

}

class Volleyball implements Playable {

@Override

public void play() {

System.out.println("Playing volleyball with a team of 6

players.");

}

}

class Basketball implements Playable {

@Override

public void play() {

System.out.println("Playing basketball with a team of 5

players.");

}

}

public class Question25 {

public static void main(String[] args) {

System.out.println("Name Chaitanya; Roll No 24112; Sec CSE-B");

Football football = new Football();

Volleyball volleyball = new Volleyball();

Basketball basketball = new Basketball();

football.play();

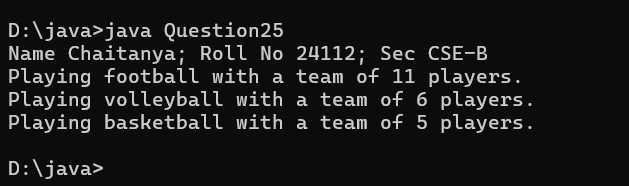
volleyball.play();

basketball.play();

}

}

Output:



**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | Error in calling play method in football class because object is not created for football class | Object created for football class |

**Important Points:**

○ Interface: Defines a common action (play()) for different sports.

○ Implementing Interface: Classes provide specific implementations of how to play

each sport.

○ Polymorphism: Different sports can be played through a common Playable

interface

3) **Aim:** write a java program to implement a login system using interfaces.

Code:

interface LoginSystem {

    boolean Login(String ID, int pass);

}

class CollegePortal implements LoginSystem {

    public boolean Login(String ID, int pass) {

        if ((ID=="TEJA") && (pass==24138)){

            System.out.println("Login Successful..!");

            return true;

        }else {

            System.out.println("Invalid ID or Password");

            return false;

        }

    }

}

class LoginPortal {

    public static void main(String[] args) {

        CollegePortal CP = new CollegePortal();

        System.out.println("Name Chaitanya;Roll No 24112;Sec CSe-B");

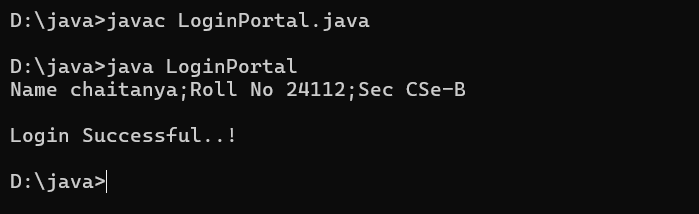
        System.out.println("    ");

        CP.Login("Raja", 24112);

    }

}

Output:



**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | Error in If statement condition | If statement condition is corrected |

Important Points:

* Interface-Based Design: The Login interface abstracts the authentication mechanism.
* Loose Coupling: The LoginSystem depends on the Login interface, not the implementation (UserLogin), making it easily replaceable or extendable.
* Security Practice: Simplified hardcoded check, but illustrates real-world login logic via interface.
* User Interaction: Accepts input via Scanner and validates it against fixed credentials.