23CSE111

OBJECT ORIENTED PROGRAMMING

LAB MANUAL



Department of Computer Science and Engineering

Amrita School of computing

Amrita Vishwa Vidyapeetham, Amaravati Campus

Name: Sumanth Deshpande

Sec: CSE-B

Verified By: Roll No: AV.SC.U4CSE24325

**TASK 1: JAVA INSTALLATION ON WINDOWS**

**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/

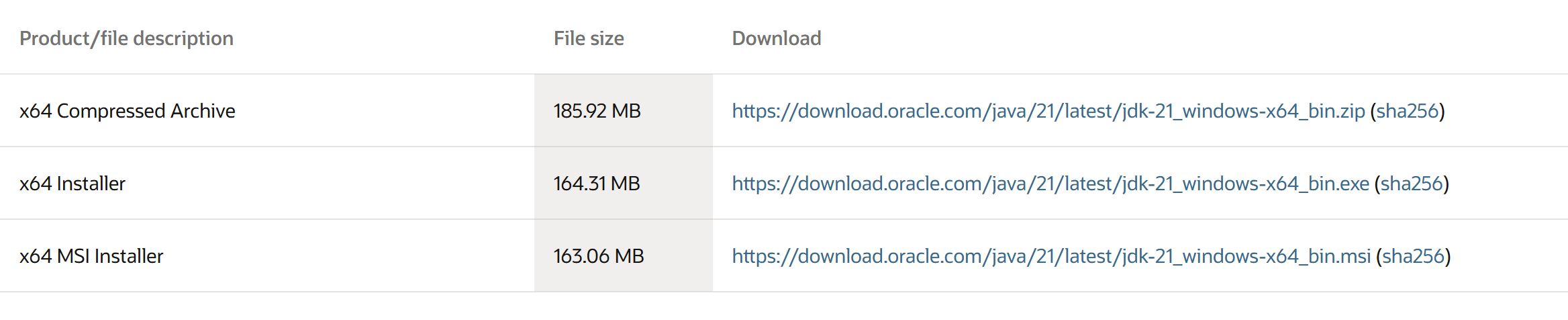
Step 3: And navigate to the java downloads. Then some earlier versions are available like

* [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.**

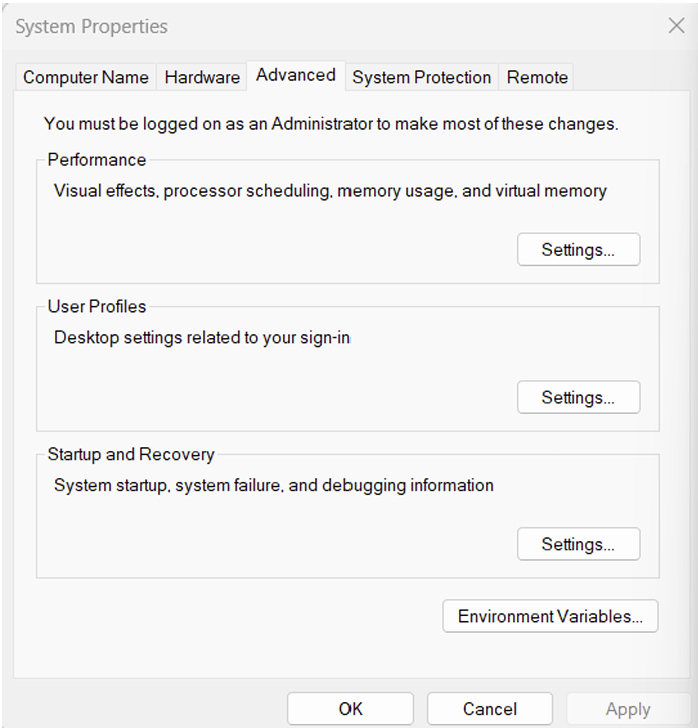
Step 1: Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on Environment Variables.

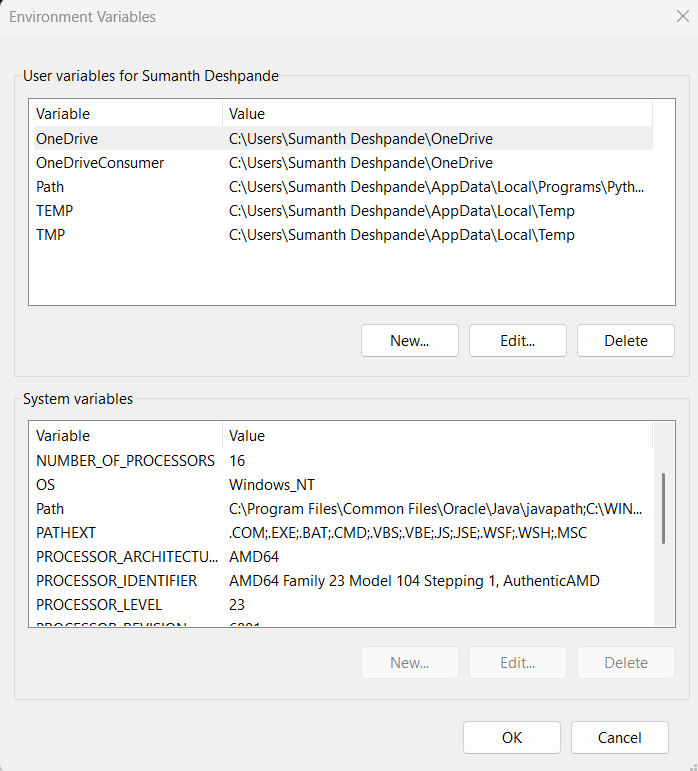
Step 2: Click New under System Variables:

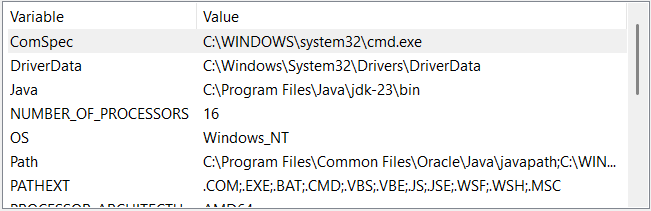
Set variable name as:java\_

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)

****

****

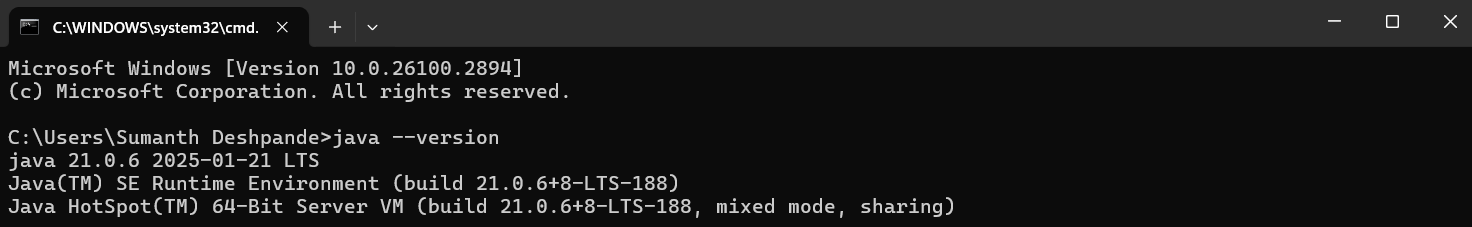
****

**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.

****

**TASK 2: First Java Program**

Write a simple Java code to print your Name, Roll.no and Section.

**Code:**

class demo

{

public static void main(String[] args)

{

System.out.println("NAME: Sumanth Deshpande");

System.out.println("SECTION: B");

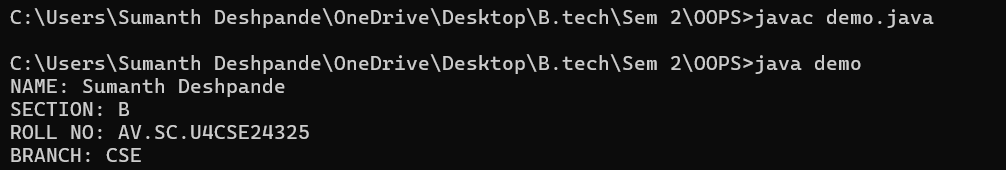
System.out.println("ROLL NO: AV.SC.U4CSE24325");

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

}

}

**Output:**

****

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **1** | Syntax error | Semicolon Added |
| **2** | Runtime Error | Copied correct path |
| **3** | Name Error | Rectified |

**WEEK-2**

**AIM: WRITE A JAVA PROGRAM TO CALCULATE THE SIMPLE INTEREST WITH INPUTS**

**PROGRAM:**

public class SimpleInterest {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in)

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

        double principal = scanner.nextDouble();

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

        double rate = scanner.nextDouble();

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

        double time = scanner.nextDouble();

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

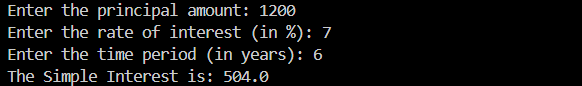
        System.out.println("The Simple Interest is: " + simpleInterest);

        scanner.close();

    }

}

**OUTPUT:**

****

**NEGATIVE CASE:**

**A screen shot of a computer code

AI-generated content may be incorrect.**

1.When we give input in decimal form for principal value it will show error.

2.It can’t compile the code and can’t show the output

**IMPORTANT POINTS:**

1.We have used scanner method to give input to the code

2.Formula of simple interest is (P\*T\*R)/100

3.We have assigned values for P,T,R

**Errors:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Reason for error** | **Rectification** |
| 1 | Runtime error | Incorrect path | Copied correct path |
| 2 | Syntax error | { missing | { added |

1. **AIM: WRITE A JAVA PROGRAM TO FIND THE FACTORIAL OF A NUMBER**

**PROGRAM:**

import java.util.Scanner;

public class FactorialCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

int result = factorial(number);

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

scanner.close();

}

public static int factorial(int n) {

if (n == 0) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

}

OUTPUT:

A black background with white text

AI-generated content may be incorrect.

**NEGATIVE CASE:**

A screen shot of a computer program

AI-generated content may be incorrect.

1.When we give input in decimal form for principal value it will show error.

2.It can’t compile the code and can’t show the output.

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
|  | Logical error | Incorrect input | Correcting input |
|  | Runtime error | Incorrect path | Using correct path |

**IMPORTANT POINTS:**

1. **Importing Scanner:**
   * The code imports the Scanner class from the java.util package to read user input.
2. **Main Method:**
   * The main method is the entry point of the program. It creates a Scanner object to read user input and prompts the user to enter a number**.**
3. **Factorial Calculation: The factorial method is defined as a static method that takes an integer n and returns the factorial of n.**
   * method uses recursion to calculate the factorial. If n is 0, it returns 1 (base case). Otherwise, it returns n \* factorial (n - 1).
4. **Closing Scanner:**
   * The scanner.close() method is called to close the Scanner object and release the resources associated with it.
5. **Recursion:**

The use of recursion in the factorial method is a key point. It repeatedly calls itself with decremented values of n until it reaches the base case

**3 . AIM:WRITE A JAVA PROGRAM TO CONVERT CELCIUS TO FAHRENHEIT**

**PROGRAM :**

import java.util.Scanner;

public class CelsiusToFahrenheit {

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 = celsiusToFahrenheit(celsius);

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

scanner.close();

}

public static double celsiusToFahrenheit(double celsius) {

return (celsius \* 9/5) + 32;

}

}

**OUTPUT:**

**A black background with white text

AI-generated content may be incorrect.**

**NEGATIVE CASE:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**IMPORTANT POINTS TO NOTE:**

**1. Importing Scanner Class:**

* The Scanner class is imported from java.util.Scanner to take user input.

**2. Class Declaration:**

* The class name CelsiusToFahrenheit follows Java naming conventions (PascalCase).

**3. Main Method:**

* The public static void main(String[] args) method is the entry point of the program

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect input** | **Correcting input** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |

**3.AIM: JAVA PROGRAM TO CONVERT FAHRENHEIT TO CELCIUS**

**PROGRAM:**

import java.util.Scanner;

public class FahrenheitToCelsius {

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 = fahrenheitToCelsius(fahrenheit);

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

scanner.close();

}

public static double fahrenheitToCelsius(double fahrenheit) {

return (fahrenheit - 32) \* 5/9;

}

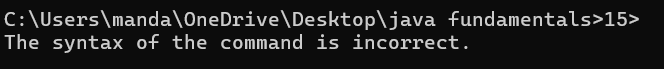
}

**OUTPUT:**

**A black background with white text

AI-generated content may be incorrect.**

**NEGATIVE CASE:**

****

**IMPORTANT POINTS:**

* import java.util.Scanner; is used to include the Scanner class to take user input.

**1.Class Declaration:**

* The class is named FahrenheitToCelsius which follows Java naming conventions.

**2. Main Method:**

* The program execution starts from the main() method:

public static void main(String[] args)

**3.Scanner Class:**

* The Scanner object is created to take input from the user:

Scanner scanner = new Scanner(System.in);

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1. | Logical error | Due to incorrect input | Corrected by giving correct input |
| 2. | Runtime error | Incorrect path | Using correct path |

**5 . AIM: WRITE A JAVA PROGRAM TO FIND THE AREA OF A TRIANGLE**

**PROGRAM:**

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 = calculateArea(base, height);

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

scanner.close();

}

public static double calculateArea(double base, double height) {

return (base \* height) / 2;

}

}

**OUTPUT:**

**A black background with white text

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**IMPORTANT POINTS:**

**User Input:**

* The user is prompted to enter:
  + Base of the triangle
  + Height of the triangle

**Method Call:**

* The method calculateArea(base, height) is called to calculate the area of the triangle.

**Return Value:**

* The method returns the area value to the main() method.

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **2.** | **Syntax error** | **No semicoln** | **Using semicoln** |

**6. AIM: WRITE A JAVA PROGRAM TO FIND THE AREA OF A RECTANGLE**

**PROGRAM:**

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 = calculateArea(length, width);

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

scanner.close();

}

public static double calculateArea(double length, double width) {

return length \* width;

}

}

**OUTPUT:**

**A black background with white text

AI-generated content may be incorrect.**

**NEGATIVE CASE:**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**IMPORTANT POINTS:**

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

**2. User Input:**

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

**3.Modularity:**

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

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **1** | **Syntax error** | **Semicolon added** |
| **2.** | **Name error** | **rectified** |

**7.AIM: WRITE A JAVA PROGRAM FOR FIBONACCI SEQUENCE**

**PROGRAM:**

import java.util.Scanner;

public class Fibonacci {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num;

int f3;

int f1 = 0;

int f2 = 1;

int i = 2;

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

num = sc.nextInt();

System.out.println(f1);

System.out.println(f2);

while(i < num) {

f3 = f1 + f2;

f1 = f2;

f2 = f3;

System.out.println(f3);

i = i + 1;

}

sc.close();

}

}

**OUTPUT:**

**A black screen with white text

AI-generated content may be incorrect.**

**NEGATIVE CASE:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

1.I have rearranged the starting two elements from 0 and 1 to 1 and 0.

2.Therefore, the order of Fibonacci series is changed.

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Name error | Incorrect usage of function | Correcting by using correct formula |
| 2 | Syntax error | No semicolon | Semicolon added |
| 3 | Runtime error | Incorrect path | Copied correct path |

**IMPORTANT POINTS:**

1. Importing Scanner:

* The code imports the Scanner class from the java.util package to read user input.

2. Class Declaration:

* The code declares a class named fibo.

3. Main Method:

* The main method is the entry point of the program. It creates a Scanner object to read user input and prompts the user to enter a number.

4. Variable Initialization:

* The code initializes variables:
  + num: The number of Fibonacci terms to be generated.
  + f3: Used to store the next Fibonacci term.
  + f1 and f2: The first two terms of the Fibonacci sequence, initialized to 0 and 1, respectively.
  + i: Counter variable initialized to 2 since the first two terms are already known.

**WEEK-3**

**Aim:**

**To create java program with following instructions**

**1.Create a class with name car**

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

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

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

**5.Create a constructor with parameter CarColour, CarBrand, FuelType and Mileage**

**PROGRAM:**

class car {

    String CarColor;

    String CarBrand;

    String fuelType;

    int Mileage;

    public void start(){

        System.out.println("Car is Starting...");

    }

    public void stop(){

        System.out.println("Car is Stopping....");

    }

    void car(String *CarColor* , String *CarBrand* , String *fuelType* , int *Mileage* ){

        this.CarColor=CarColor;

        this.CarBrand=CarBrand;

        this.fuelType=fuelType;

        this.Mileage=Mileage;

        System.out.println("Car color is :"+ CarColor);

        System.out.println("Car Brand is:"+CarBrand);

        System.out.println("Car fuel type is:"+fuelType);

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

    }

    public void service(int *a*){

        if (a>760) {

            System.out.println("Car requires service");

        }else{

            System.out.println("Car is ok");

        }

    }

}

public class Check {

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

        car c1 = new car();

        car c2 = new car();

        car c3 = new car();

        c1.start();

        c1.car("Red","Honda","Diesel",160);

        c1.service(830);

        c1.stop();

        c2.start();

        c2.car("Black","Ferrari","Petrol",100);

        c2.service(600);

        c2.stop();

        c3.start();

        c3.car("Blue","Suzuki","CNG",240);

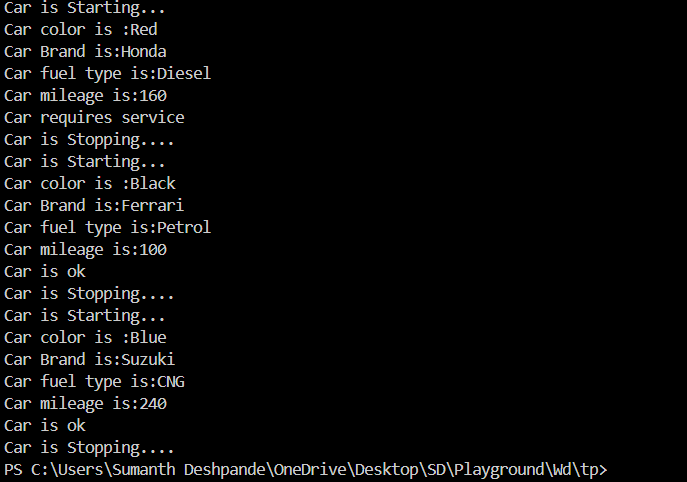
        c3.service(760);

        c3.stop();

    }

}

**OUTPUT:**

****

**CLASS DIAGRAM:**

|  |
| --- |
| car |
| + CarColor: String  + CarBrand: String  +fueltype: String  +mileage: int |
| + start(): void  + service(): void  + stop(): void |

**IMPORTANT POINTS**:

1. The car class has four attributes: CarColor, CarBrand, fueltype, and mileage.

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

3. It has 1 Constructor method

4. Constructor method prints the details of the class

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

6. Each car object is assigned specific values for CarColor, CarBrand, fueltype, and Mileage**.**

**AIM:**

**Create a class named Bank Account with methods Deposit and withdraw where the deposit method should accept parameter and when this method is called, the deposited amount should be added to current balance. In addition, when a withdraw method is called, it has to verify whether the withdraw is less than current balance. If not, display a message saying insufficient funds should be displayed. Use the constructor to display the details of the customer - Cname, Accnumber, IFSC and branch. Create 2 customer objects C1 and C2.**

**PROGRAM:**

class Bank {

    int depamount;

    int balance;

    int withamount;

    String Cname;

    int AccNo;

    int Ifsc;

    String Branch;

    void deposit(int depamount){

        balance = balance + depamount;

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

    }

    void withdraw(int withamount){

        if (withamount>balance) {

            System.out.println("Insufficiant funds to withdraw Fund");

        }else{

            balance = balance - withamount;

            System.out.println("Amount Withdrawn : " + withamount);

            System.out.println("Balance after Withdrawal : " + Integer.toString(balance));

        }

    }

    void Bank(String Cname , int AccNo , int Ifsc , String Branch){

        this.Cname=Cname;

        this.AccNo=AccNo;

        this.Ifsc=Ifsc;

        this.Branch=Branch;

        System.out.println("Customer name : " + Cname);

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

        System.out.println("IFSC Number : " + Ifsc);

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

    }

}

public class Check {

    public static void main(String[] args) {

        Bank C1 = new Bank();

        Bank C2 = new Bank();

        Bank C3 = new Bank();

        C1.Bank("Sumanth", 123456789, 987654230, "Dombivali");

        C1.balance=6000;

        C1.deposit(2000);

        C1.withdraw(3940);

        C2.Bank("Mahesh",2123323234, 910464202, "Thane");

        C2.balance=100000;

        C2.deposit(2000000);

        C2.withdraw(91930);

        C3.Bank("YAsh", 131626789, 897534230, "Mumbai");

        C3.balance=100000;

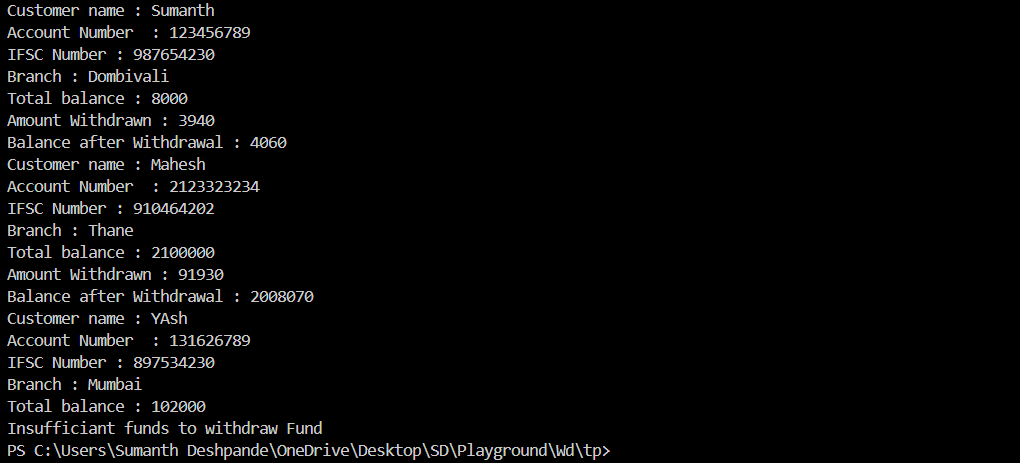
        C3.deposit(2000);

        C3.withdraw(183720);

    }

}

**OUTPUT:**

****

**CLASS DIAGRAM:**

|  |
| --- |
| **BankAccount** |
| +int depamount;      +int balance;      +int withamount;      +String Cname;     + int AccNo;      +int Ifsc;      +String Branch; |
| + void deposit(int *depamount*)  + void withdraw(int *withamount*)  + void Bank(String *Cname* , int *AccNo* , int *Ifsc* , String *Branch*) |

**WEEK-4**

**1.AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOUKD CONTAIN VARIOUS ATTRIBUTES SUCH AS TITLE, AUTHOR, YEAR OF PUBLICATION. IT SHOULD ALSO CONTAIN A CONSTRUCTOR WITH PARAMETERS WHICH INITIALIZES TITLE, AUTHOR, YEAR OF PUBLICATION AND CREATE A METHOD WHICH DISPLAYS THE DETAILS OF 2 BOOKS.**

**CLASS DIAGRAM:**

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

**PROGRAM:**

public class Book {

public String title;

public String author;

public int year;

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

this.title = title;

this.author = author;

this.year = year;

}

public void displayDetails() {

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

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

System.out.println("Year of Publication" +year);

}

public static void main(String[] args) {

Book b1 = new Book("To Kill a Monkingbird", "Harper Lee", 1960);

Book b2 = new Book("The Great Gatsby", "F. Scott Fitzgerald", 1925);

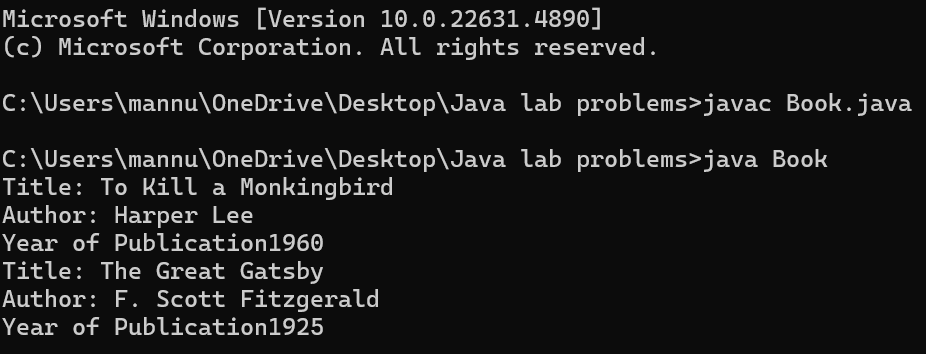
b1.displayDetails();

b2.displayDetails();

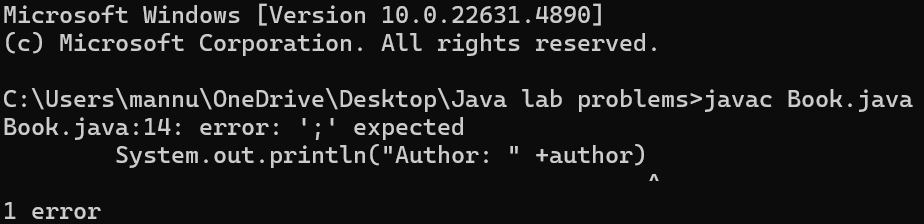
}

}

**OUTPUT:**



**NEGATIVE CASE:**



**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **ERROR TYPE** | **Reason for error** | **Rectification** |
| **1.** | Syntax error | No semicolon | Semicolon added |
| **2.** | Runtime error | Incorrect path | Copied correct path |

**IMPORTANT POINTS:**

1. **Constructor**:

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

2.**Method**:

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

3. **Object Creation**:

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

**2.AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “MyClass” WITH A STATIC VARIABLE COUNT OF INT TYPE. INTIALIZE IT TO ZERO AND A CONSTANT VARIABLE “Pi” OF TYPE DOUBLE INITIALIZED TO “3.14” AS ATTRIBUTES OF THAT CLASS. NOW DEFINE A CONSTRUCTOR FOR “MyClass”, THAT INCREMENTS THE COUNT VARIABLE EACH TIME AN OBJECT OF “MyClass” IS CREATED. FINALLY, PRINT THE FINAL VALUES**

**OF ‘COUNT’ AND ‘PI’ VARIABLES AND CREATE 3 OBJECTS.**

**CLASS DIAGRAM:**

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

**PROGRAM:**

public class MyClass {

static int count = 0;

static final double pi = 3.14;

MyClass() {

count++;

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

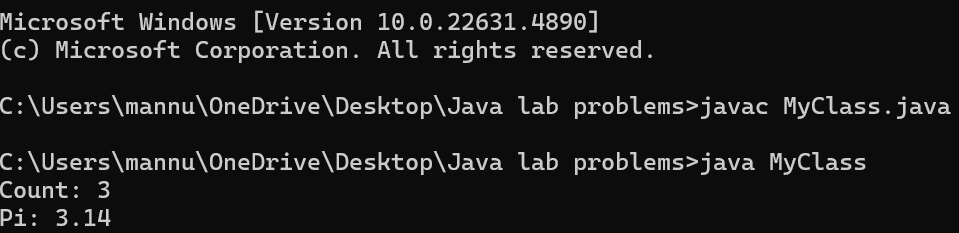
System.out.println("Count: " +count);

System.out.println("Pi: " +pi);

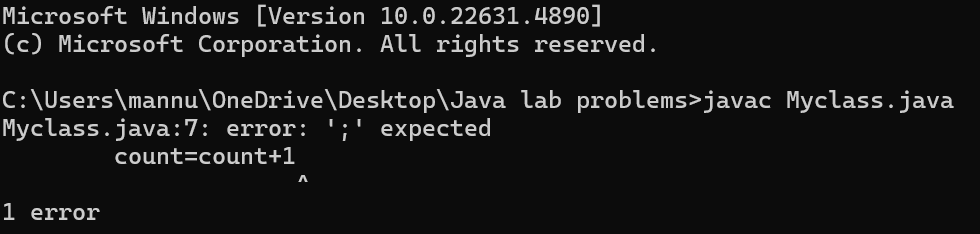
}

}

**OUTPUT:**

****

**sNEGATIVE CASE:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Reason for error** | **Rectification** |
| **1.** | No class | No class name declared | Created class named ‘MyClass’ |
| **2.** | Syntax error | Not added keyword | Added keyword named ‘new’ |

**IMPORTANT POINTS:**

**1.Static Keyword**

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

**2.Static Variable**

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

**3.Final Variable**

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

**WEEK-5**

**AIM: Create a calculator using the operations including addition, subtraction Multiplication and division using multilevel inheritance and display the desired Output**

**Class Diagram:**



**Code:**

class addition

{

public int add(int a, int b)

{

int addition = a+b;

return addition;

}

}

class subtraction extends addition

{

public int sub(int a, int b)

{

int subtraction = a-b;

return subtraction;

}

}

class multiplication extends subtraction

{

public int mult(int a, int b)

{

int multiplication = a\*b;

return multiplication;

}

}

class division extends multiplication

{

public int div(int a,int b)

{

int division = a/b;

return division;

}

}

class calculator

{

public static void main(String args[])

{

division obj = new division();

System.out.println("Addition is:"+ obj.add(10,2));

System.out.println ("Subtraction is:"+obj.sub(8,4));

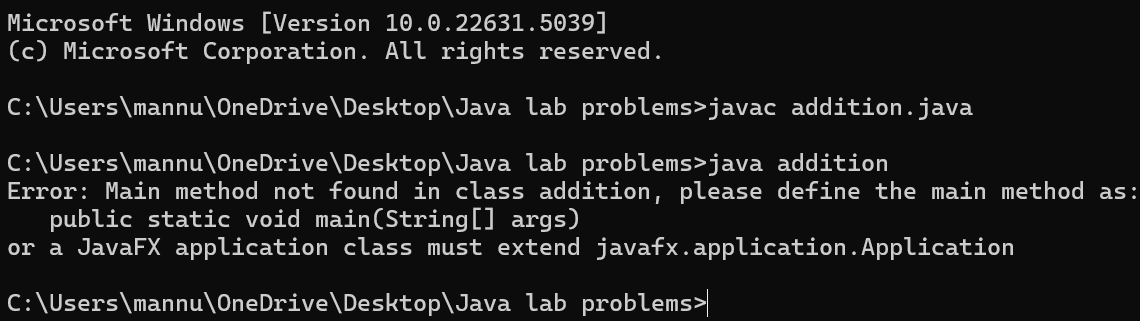
System.out.println("Multiplication is:"+obj.mult(12,4));

System.out.println("Division is:"+obj.div(8,4));

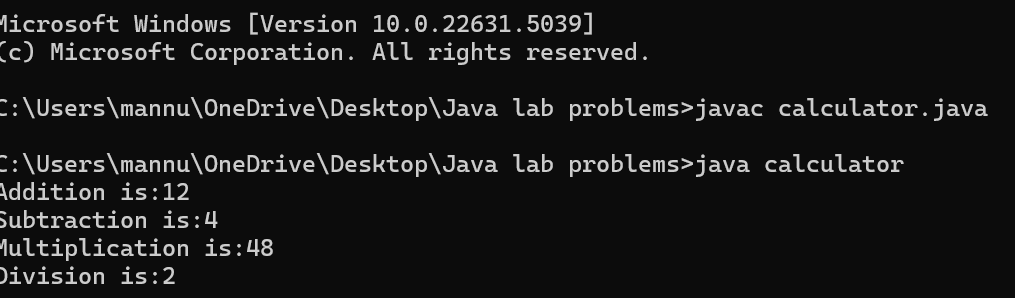
  }

}

**Negative Case:**



**OUTPUT:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error Type | Cause | Rectification |
| 1 | Constructor error | Invalid name to method | Defined class name |
| 2 | Syntax error | Expected ‘(‘ | Added parenthesis |
| 3 | Logical error | Incorrect arithmetic  operation | Correct operation  rectified |

**Important Points:**

**Inheritence:**

The concept of OOP where a class inherits the properties and behaviours from

Another class (parent class) which promotes code reusability and hieratchical relationships

**Multilevel Inheritence:**

This is a type of inheritance in which a class inherited from another class, and

That superclass, in turn, inherits from yet another class, creating a chain of

inheritence

**extends:**

The extends keyword defines the relation of child class with the parent class

**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, bikes and truck 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**

**Bikes 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**

**Class diagram**

**CLASS DIAGRAM:**



**Code:**

class vehicle{

String brand;

int speed;

public vehicle(String brand,int speed){

this.brand=brand;

this.speed=speed;

}

public static void main(String[] args) {

car obj1=new car("ford",34,4);

bike obj2=new bike("hero",100,true);

truck obj3=new truck("tata",60,40);

}

}

class car extends vehicle{

int noofdoors;

public car(String brand, int speed,int noofdoors) {

super(brand, speed);

this.noofdoors=noofdoors;

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

System.out.println("Speed of car is:"+speed);

System.out.println("no of doors of car:"+noofdoors);

}

}

class bike extends vehicle{

boolean gears;

public bike(String brand,int speed,boolean gears){

super(brand, speed);

this.gears=gears;

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

System.out.println("Speed of bike is:"+speed);

System.out.println("Gears of bike:"+gears);

}

}

class truck extends vehicle{

int weight;

public truck(String brand,int speed,int weight){

super(brand,speed);

this.weight=weight;

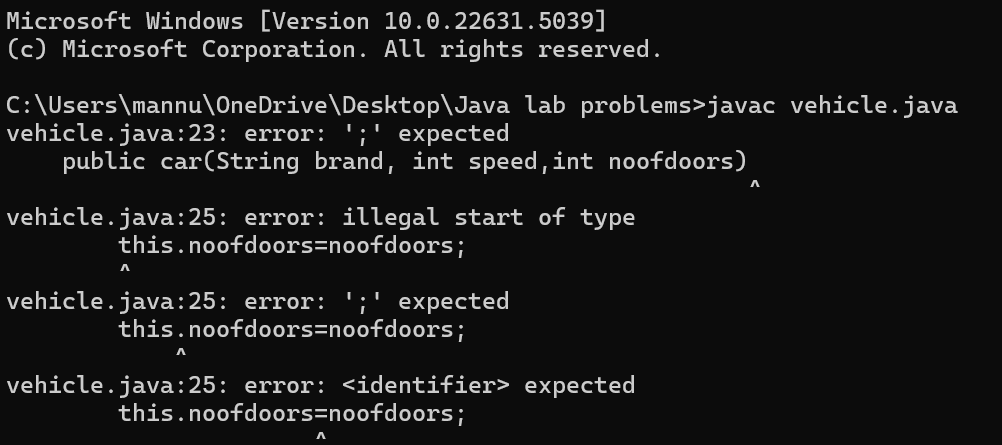
System.out.println("Brand name is:"+brand);

System.out.println("Speed of Truck is:"+speed);

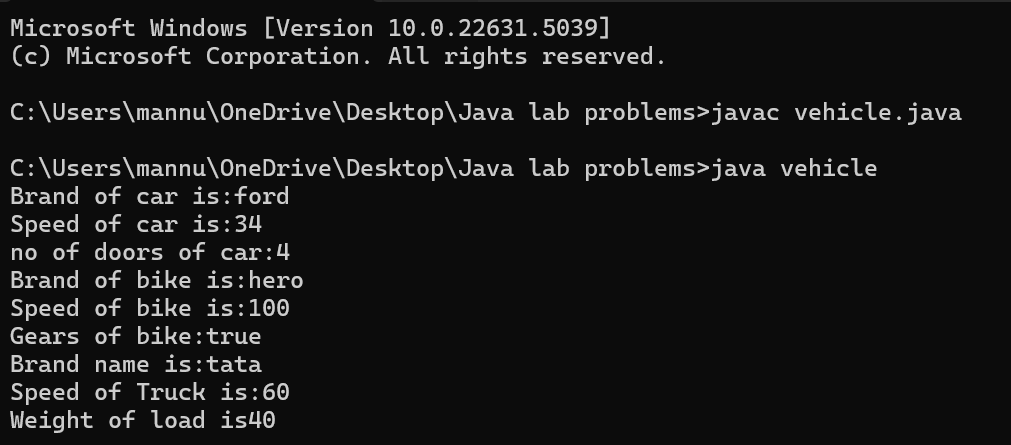
System.out.println("Weight of load is"+weight);

}

}

**NEGATIVE CASE:** ****

**OUTPUT:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 |  |  |  |

**Important Points:**

**Hierarchical Inheritence:**

This is a type of inheritance occurs when multiple subclasses inherit from a

Single parent class

**WEEK-6**

**1.Aim: Write a java program to create a vehicle class with a method displayInfo().**

**Override this method in the car subclass to provide specific information a About car.**

**CODE:**

class vehicle

{

public String car\_model;

public String car\_company;

public int car\_price;

void displayinfo()

{

System.out.println("Bhanu Teja Car Agency");

}

}

class car extends vehicle

{

void displayinfo()

{

System.out.println("Car Model is:"+car\_model);

System.out.println("Car Company is:"+car\_company);

System.out.println("Car price is:"+car\_price);

}

}

class Overriding

{

public static void main(String args[])

{

car obj1 = new car();

obj1.car\_model="Swift";

obj1.car\_company="Suzuki";

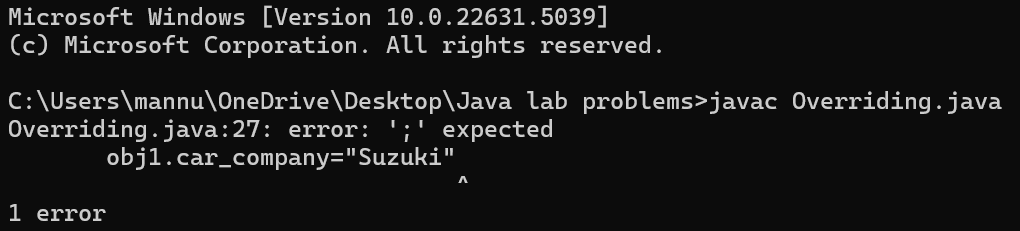
obj1.car\_price=1000000;

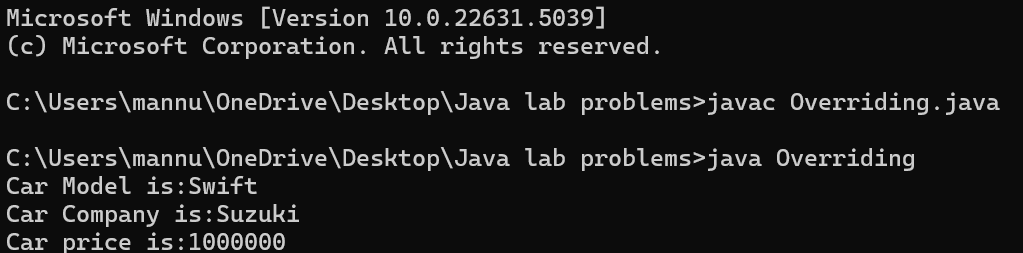
obj1.displayinfo();

}

}

**NEGATIVE CASE:**

****

**OUTPUT:** ****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

**2.A college is developing an automated admission system that verifies Student eligibility for UG and PG programs. Each program has different Eligibility criteria based on the student’s percentage in their previous**

**Qualification:**

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

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

**CODE:**

class Student

{

public String name;

public double percentage;

Student(String name, double percentage)

{

this.name = name;

this.percentage = percentage;

}

void eligibility()

{

System.out.println("Amrita Vishwa Vidyapeetham");

}

}

class UGStudent extends Student

{

UGStudent(String name, double percentage)

{

super(name,percentage);

}

void eligibility()

{

if (percentage>60)

{

System.out.println(name+" is selected to the college");

}

else

{

System.out.println(name+" is not matching the required criteria");

}

}

}

class PGStudent extends Student

{

PGStudent(String name, double percentage)

{

super(name,percentage);

}

void eligibility()

{

if(percentage>70)

{

System.out.println(name+" is selected to college");

}

else

{

System.out.println(name+"is not matching the required criteria");

}

}

}

class School

{

public static void main(String args[])

{

PGStudent obj1 = new PGStudent("Sumanth",90.0);

UGStudent obj2 = new UGStudent("Karunya",80.0);

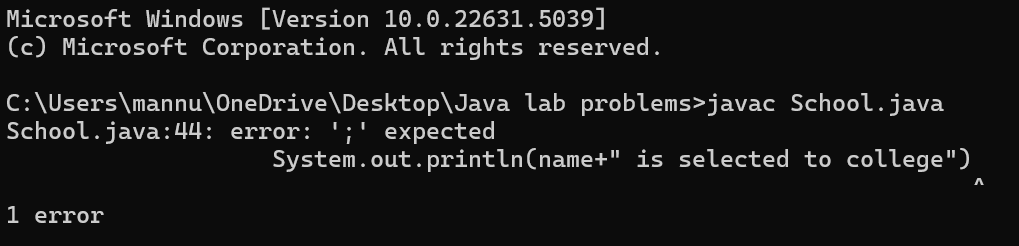
obj1.eligibility();

obj2.eligibility();

}

}

**NEGATIVE CASE:**



**OUTPUT:**

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

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

1. **Add two integers 2. Add two doubles 3. Add three integers**

**CODE:**

class 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;

}

}

class Overloading

{

public static void main(String args[]){

calculator c = new calculator();

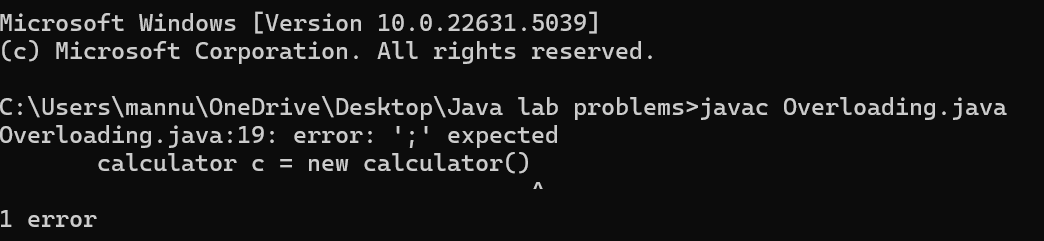
System.out.println("Addition of 3 and 4 is:"+c.add(3,4));

System.out.println("Addition of 2.2 and 4.4 is:"+c.add(2.2,4.4));

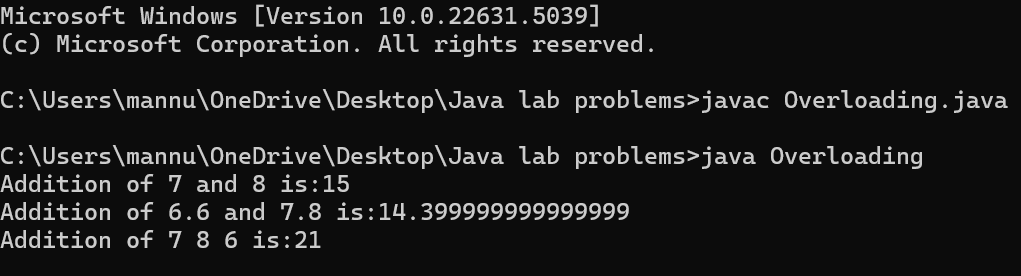
System.out.println("Addition of 2 4 6 is:"+c.add(2,4,6));

}}

**NEGATIVE CASE:**



**OUTPUT:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

**4. Create a Shape class with a method calculateArea() that is overloaded for Different shapes. Then, create a subclass circle that overrides the calculateArea() method for a circle**

**CODE:**

class shape

{

public double calculateArea(double side)

{

return side\*side;

}

public int calculateArea(int length, int width)

{

return length\*width;

}

}

class circle extends shape

{

public double calculateArea(double radius)

{

return 3.14\*radius\*radius;

}

}

class Example

{

public static void main(String args[])

{

circle obj1 = new circle();

shape obj2 = new shape();

System.out.println("The area of side 6 is:"+obj2.calculateArea(4.0));

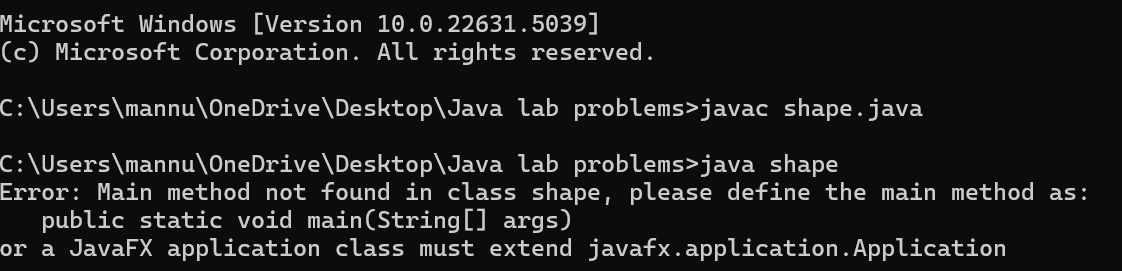
System.out.println("The area of circle with radius 3 :"+obj1.calculateArea(5.0));

System.out.println("The area of length 3 and width 4 is:"+obj2.calculateArea(5,4));

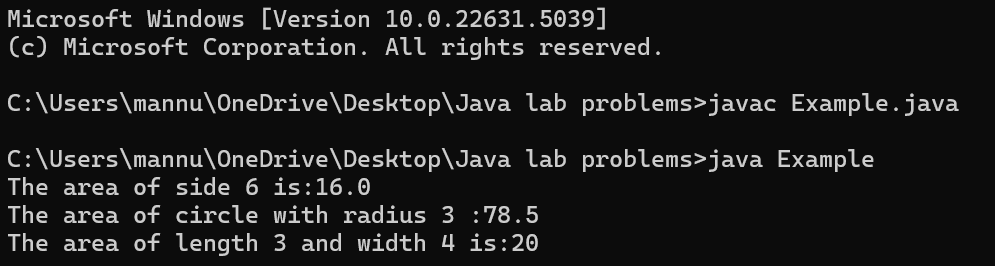
}

}

**NEGATIVE CASE:**

****

**OUTPUT:**



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |
| 2 | Writing class name |  |  |

**WEEK-7**

**Task-1**

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

**- Important Points:**

* + - 1. **Understand the calling of a Constructor**
      2. **Giving class name correctly**
      3. **Give the parameters Correctly**

**INPUT:**

public abstract class Animal {

     public abstract void sound();

}

 class Lion extends Animal{

    public  void sound(){

        System.out.println("Roars");

    }

}

class Tiger extends Animal{

    public void sound(){

        System.out.println("Growls");

    }

}

class Bag{

    public static void main(String[] args){

       System.out.println("Name: Sumanth D, Class: CSE-B, Roll no: AV.SC.U4CSE24325");

        Lion l=new Lion();

        l.sound();

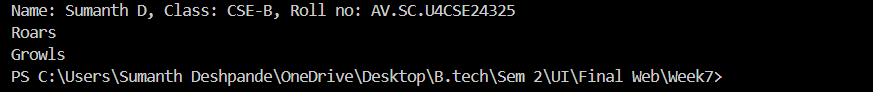
        Tiger t=new Tiger();

        t.sound();

    }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** | |
| 1 | Syntax error | Abstract key word is missed before method | Abstract keyword is added |
| 2 | Logical error | Incorrect logic in subclass method | Corrected logic in subclass method |

**Task-2**

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

**INPUT:**

public abstract class Shape3D {

    public abstract void CalculateVolume();

    public abstract void CalculateSurfaceArea();

}

class Sphere extends Shape3D {

    int radius;

    Sphere(int radius) {

        this.radius = radius;

    }

    public void CalculateVolume() {

        System.out.println("Volume of sphere is: " + (4.0 / 3.0) \* Math.PI \* radius \* radius \* radius);

    }

    public void CalculateSurfaceArea() {

        System.out.println("Surface area of sphere is: " + 4 \* Math.PI \* radius \* radius);

    }

}

class Cube extends Shape3D {

    int side;

    Cube(int side) {

        this.side = side;

    }

    public void CalculateVolume() {

        System.out.println("Volume of cube is: " + side \* side \* side);

    }

    public void CalculateSurfaceArea() {

        System.out.println("Surface area of cube is: " + 6 \* side \* side);

    }

}

class Normal{

    public static void main(String[] args){

        System.out.println("Name: Karthi Tanikonda,class: CSE-B,Rollno:AV.SC.U4CSE24332");

        Sphere sp = new Sphere(5);

        Cube c = new Cube(4);

        System.out.println("    ");

        sp.CalculateSurfaceArea();

        System.out.println("    ");

        sp.CalculateVolume();

        System.out.println("    ");

        c.CalculateSurfaceArea();

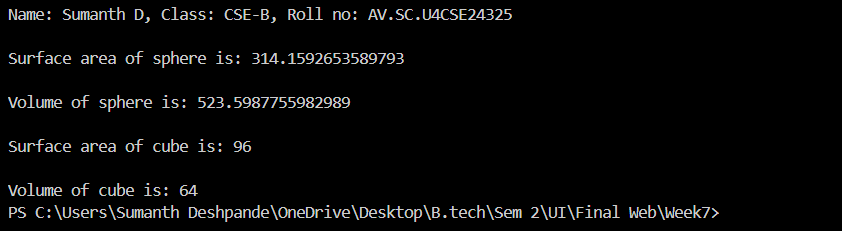
        System.out.println("    ");

        c.CalculateVolume();

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | package is missed before abstract class | Package is imported |
| 2 | Logical error | Incorrect logic in subclass method | Corrected logic in subclass method |

**Task-3**

**AIM: Write a java program using an abstract class to define a method for pattern printing Create an abstract class named pattern printer with an abstract method printpattern(int n) and a concrete method to display the pattern title.**

**Implement two subclasses:**

**1) Star pattern - Prints a right-angled triangle of stars(\*).**

**2) Number pattern - Prints a right- angled triangles of increasing numbers.**

**In the main() method, create Objects**

Star Pattern Number pattern

\* 1

\*\* 1 2

\*\*\* 1 2 3

\*\*\*\* 1 2 3 4

\*\*\*\*\* 1 2 3 4 5

**Importantpoints:**

1.understanding the loops

2.understanding the nested for

3.learning the pattern printing

4.understanding abstract classes and methods.

**INPUT:**

public abstract class pattrenprinter {

    abstract void printPattern(int n);

    void displayTitle(String title) {

        System.out.println(title);

    }

}

class StarPattern extends pattrenprinter {

    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 pattrenprinter {

    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();

        }

    }

}

 class PatternProgram {

    public static void main(String[] args) {

        System.out.println("Name: Karthi Tanikonda,class: CSE-B,Rollno:AV.SC.U4CSE24332");

        StarPattern sp = new StarPattern();

        NumberPattern np = new NumberPattern();

        System.out.println("    ");

        sp.displayTitle("Star Pattern");

        sp.printPattern(5);

        System.out.println("    ");

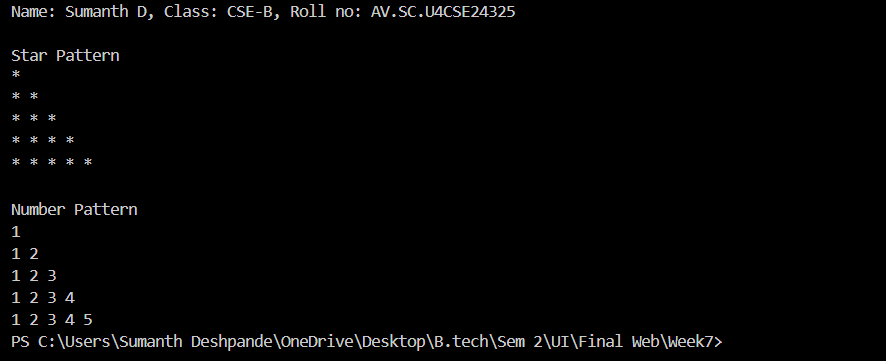
        np.displayTitle("Number Pattern");

        np.printPattern(5);

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error** | **Rectification** |
| 1 | Syntax error | Increment condition is added in subclass method |
| 2 | Logical error | Corrected logic in subclass method |
| 3 | Incorrect condition | Rectified the condition |