**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name:Y.SOWMYA**

**Verified By Roll No: 24352**

**WEEK-1**

PROGRAM-1:

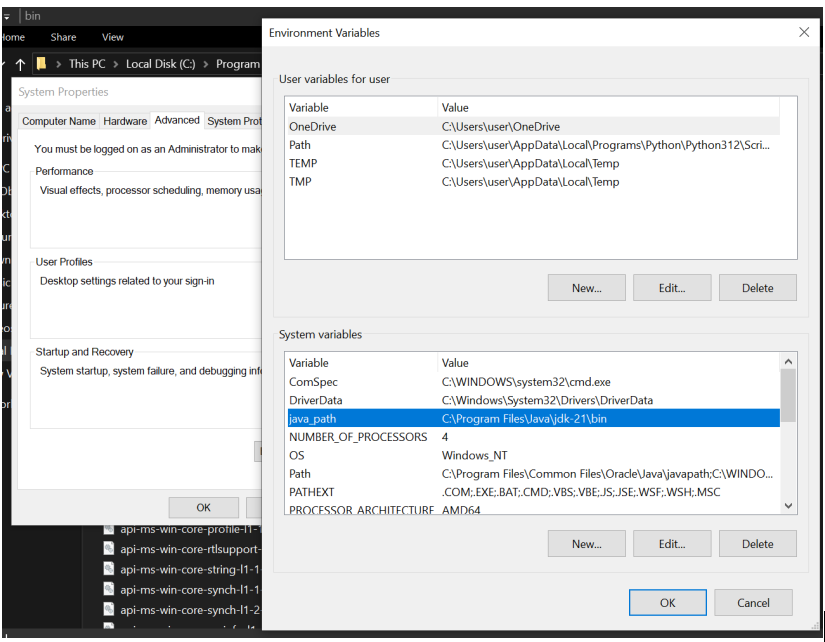
AIM: Download and Install Java Software

PROCEDURE:

1) Explain the process of Installing JDK (Java Development Kit)

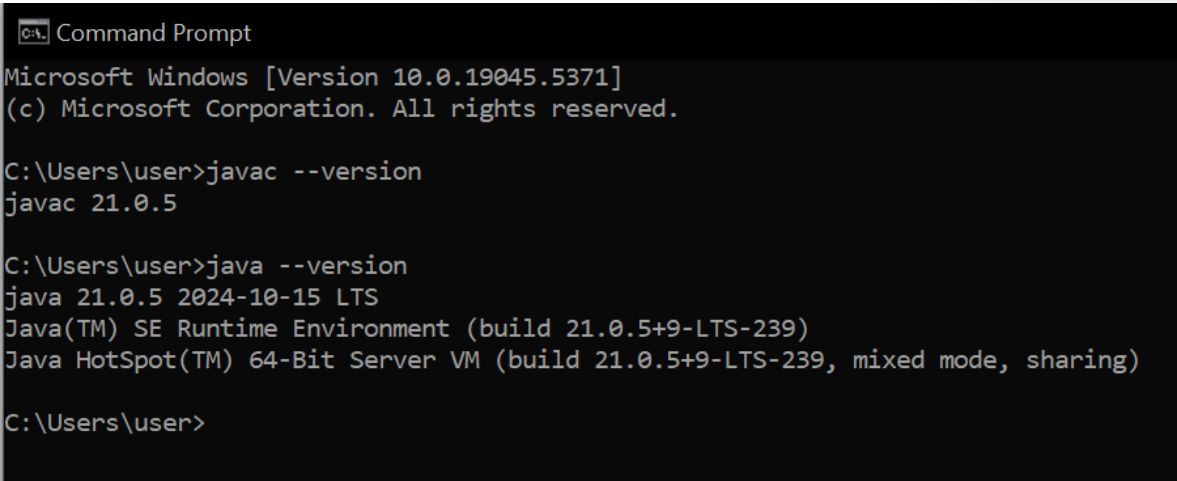
Installing of JDK (Java Development Kit):

1. Download JDK:
   * Go to the Oracle JDK download page in your web browser and click on JDK-21 version which is Long term support (LTS) version.
   * Click on the download link for your operating system (Windows, macOS, or Linux).
2. Install JDK:
   * Once downloaded, run the installer.
   * Follow the instructions and keep clicking "Next" until it's done.
3. Set Environment Variables (Windows):
   * 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.
   * Click New under System Variables:
     + Set Variable name as: java\_home
     + Variable value: The folder address where JDK is installed (like C:\Program Files\Java\jdk-21\bin)
   * Find Path under System Variables, click Edit, and add the path of the jdk21(C:\Program Files\Java\jdk-21\bin)

****

**4.Checking of JDK Version:**

1. **Open Command Prompt:**
   * Press win+R, type cmd, and press Enter.
2. **Check Version:**
   * Type java --version and press Enter.
   * Type javac --version and press Enter.



**PROGRAM-2:  
AIM : Write a java program to print the message “helloworld”.**

**CODE:**

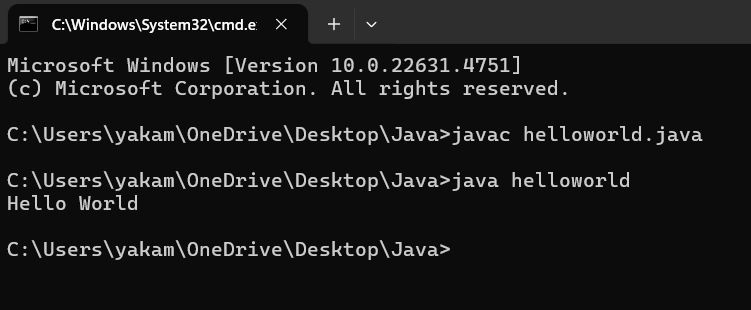
public class helloworld {

public static void main(String[] args) {

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

}

}  
OUTPUT:



ERRORS: No errors found

**PROGRAM-3:**

**AIM: Write a Java Program that prints Name, Roll No, Section of a student.**

**CODE:**

public class student details {

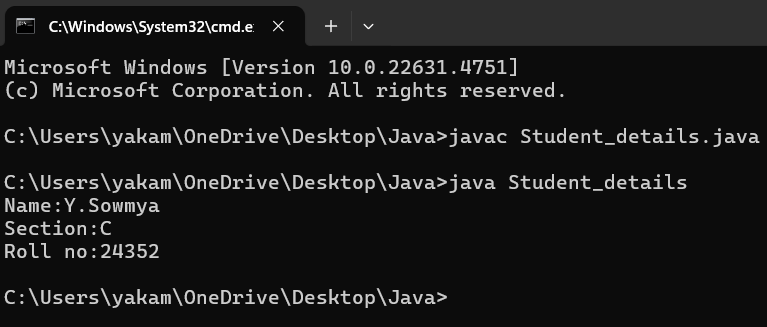
public static void main(String args[]) {

System.out.println("NAME:Y.Sowmya");

System.out.println("Section:C");

System.out.println("Roll no:AV.SC.U4CSE24352");

}

}  
OUTPUT:   
  
ERROR TABLE:

|  |  |
| --- | --- |
| **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. writing small “S” in place of ”S”   In system.out.println()  2)Giving class name with space Eg:public class student details. | 1. code is rectified by keeping capital “S”   2)code is rectified by giving class name with no space or underscore between two words Eg: public class studentdetails. public class student\_details. |

WEEK-2

**PROGRAM 1:**

**AIM : Write a Java program to Simple Interest where read values from users.**

**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) in percentage: ");

        double rate = scanner.nextDouble();

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

        double time = scanner.nextDouble();

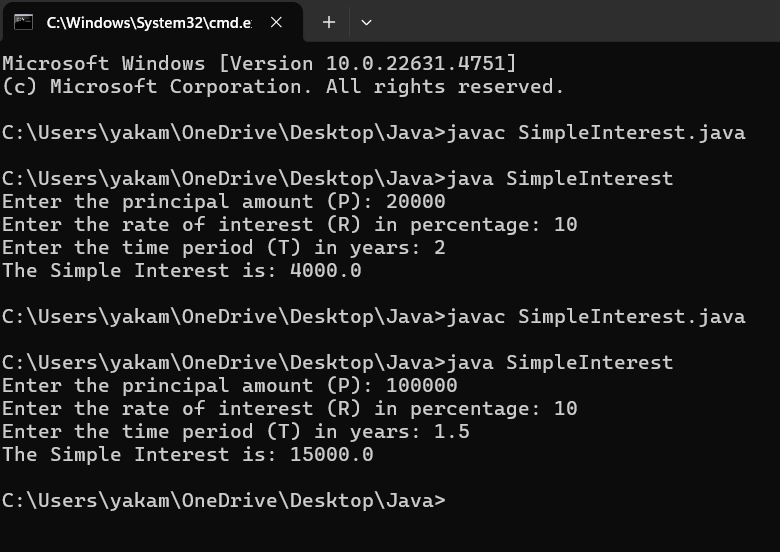
       double simpleInterest = (principal \* rate \* time) / 100;

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

       scanner.close();

    }

}

**OUTPUT:** 

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **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: ")**;** |
| **2.** | error: cannot find symbol          Scanner scanner = new scanner(System.in);  symbol:   class scanner   location: class SimpleInterest | Replace capital S in s:        Scanner scanner = new **S**canner(System.in); |

**IMPORTANT POINTS:**

1.used Scanner library to get input from user in run time.

2.”import java.util.Scanner;”-step to import library.

3.”Scanner input= new Scanner(System.in);”-step to use the scanner.[case sensitive]

4. Should give ‘;’ symbol at the end of System.out.print("Enter the rate of interest (R) in percentage:").  
**PROGRAM 2:  
AIM :Write a program on java on Factorial of a number with inputs from the user.  
CODE:**import java.util.Scanner;

  public class Factorialofanumber {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

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

        int number = scanner.nextInt();

        long factorial = 1;

        if (number < 0) {

            System.out.println("No Factorial For Negative Numbers.");

        } else {

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

                factorial \*=i;

            }

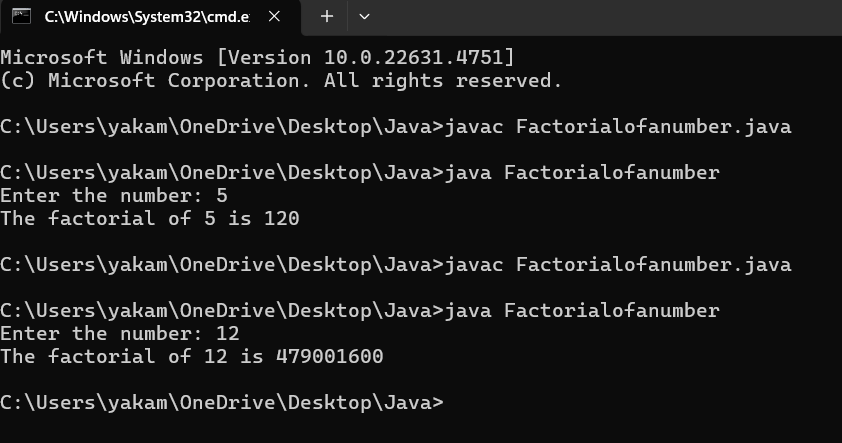
            System.out.println("The factorial of " + number + " is " + factorial);

        }

       scanner.close();

    }

}

**OUTPUT:** 

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **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; |
| **2.** | error: 'else' without 'if'          } else {  Reason:  if (number < 0); | Solution:  if (number < 0) { |

**Important points:**

1. In the loop, the expression factorial\*i; is incorrect because the result of the multiplication is not assigned back to factorial. It should be factorial\*i; for proper multiplication and assignment.

2. **Using a colon** after if will cause a syntax error.

**3. Curly braces** {} are used to group the code inside the if block.

**PROGRAM-3**

**AIM:Write a java program to convert the Temperature from Celsius to Fahrenheit & Fahrenheit to Celsius.**

**I.CODE FOR CELSIUS TO FAHRENHEIT:**

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: ");

        float celsius = scanner.nextFloat();

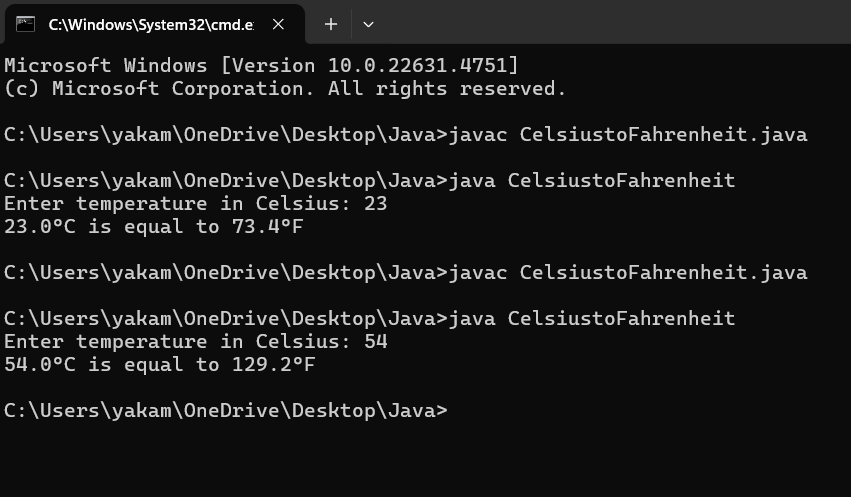
        float fahrenheit = (celsius \* 9 / 5) + 32;

        System.out.println(celsius + "°C is equal to " + fahrenheit + "°F");

        scanner.close();

    }

}

**OUTPUT: **

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| **1.** | Error: float fahrenheit = celsius \* 9 / 5 + 32;  Reason : Formula mistake. | It should be ((celsius \* 9) / 5) + 32 |
| **2.** | Error: System.out.println(celsius + "°C is equal to " + fahrenheit "°F") | Print Missing semicolon here,  Print statement is missing a semicolon |

**Important Points:**

1. In the line float fahrenheit=celsius \* 9 / 5 + 32;, the formula does not have proper parentheses for the multiplication and division. It should be float fahrenheit = (celsius \* 9 / 5) + 32; for correct order of operations.

2. In the System.out.println() statement, there is a missing semicolon at the end of the line.

**II.CODE FOR FAHRENHEIT TO CELSIUS:**

import java.util.Scanner;

  public class FahrenheittoCelsius {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter Fahrenheit temp: ");

        float f = scanner.nextFloat();

        float celsius = (f - 32) \* 5 / 9;

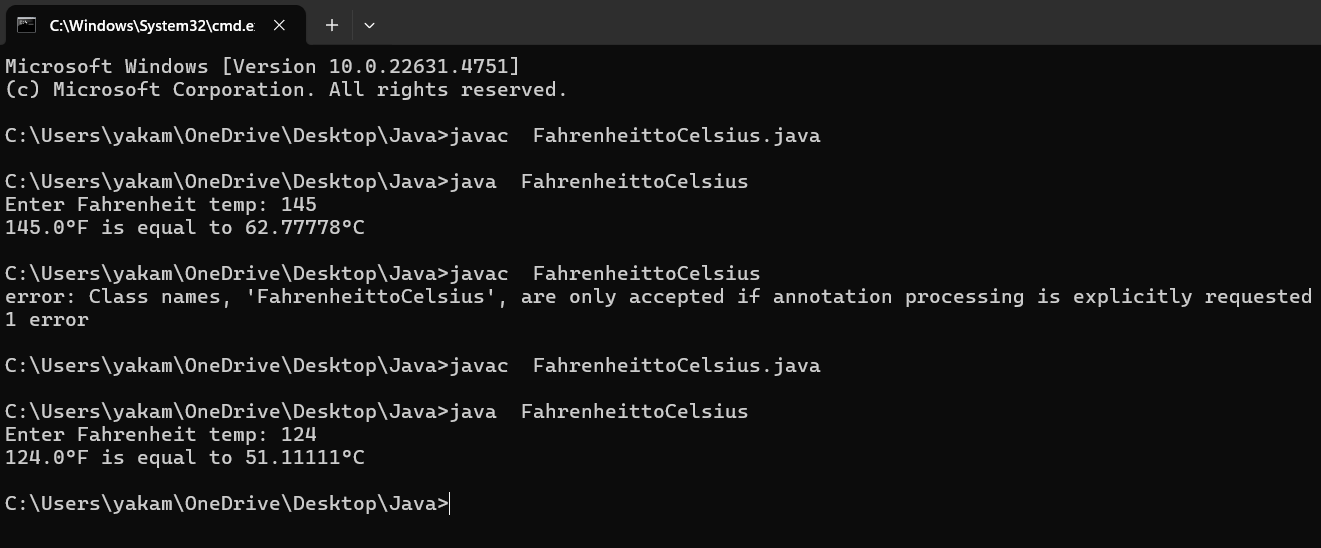
        System.out.println(f + "°F is equal to " + celsius + "°C");

        scanner.close();

    }

}

**OUTPUT:**

**  
ERRORS:**

|  |  |  |
| --- | --- | --- |
| **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’. |
| **2.** | Duplicate close of scanner- scanner.close(); scanner.close() | Reason:  Use only one  scanner.close(); |

**Important points:**

1. The System.out.println statement is using Celsius with a capital "C" which doesn't exist as a variable. Java is case-sensitive, so this will cause a compilation error. It should be lowercase celsius.

2. The scanner.close() method is called twice, which is unnecessary and could cause issues. It should only be called once.

**PROGRAM-4**

**AIM:Write a Program on Fibonacci Seqquence where input taken from user.**

**CODE:**

public class FibonacciSeries {

    public static void main(String[] args) {

        int n = 10;

        int firstTerm = 0, secondTerm = 1;

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

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

            System.out.println(firstTerm + ", ");

            int nextTerm = firstTerm + secondTerm;

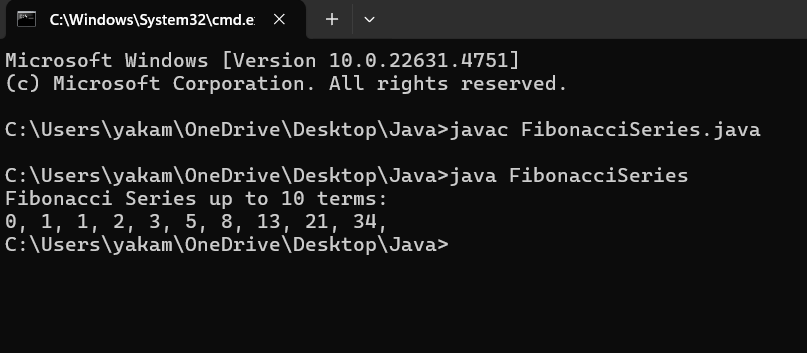
            firstTerm = secondTerm;

            secondTerm = nextTerm;

        }

    }

}

**OUTPUT:  
**

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR** | **ERROR RECTIFICATION** |
| **1.** | int firstTerm;  Error: variables not initialized prop | // should be initialized |
| **2.** | int[] fibonacci = new int[n];  System.out.println(fibonacci[n + 1]); | Invalid access |

**PROGRAM-5**

**AIM:Write a java code to find Area of rectangle where input is taken from user.  
CODE:**import java.util.Scanner;

public class rectangle {

public static void main(String[]args) {

Scanner input = new Scanner(System.in);

System.out.println("length:");

int length = input.nextInt();

System.out.println("breadth:");

int breadth = input.nextInt();

int area = length\*breadth;

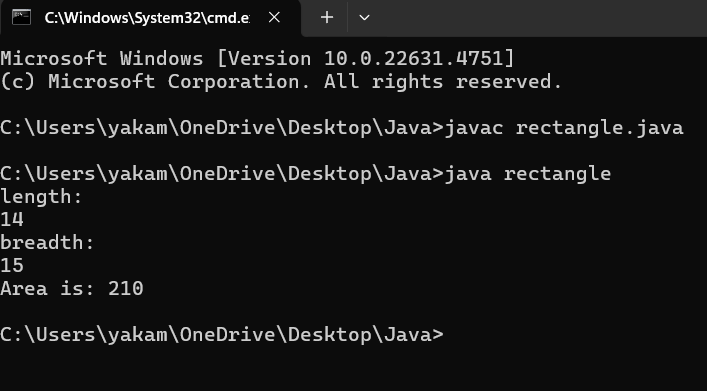
System.out.println("Area is: " +area);

input.close();

}

}

**OUTPUT:**



**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR** | **ERROR RECTIFICATION** |
| **1.** | While using for iteration, not giving the conditions correctly. | We should give iterative statements correctly. |
| **2.** | Declaring the data type as double instead of int. | We should give the data type as int for integers. |

**IMPORTANT POINTS:**

1.Area of a rectangle is area = l\*b, where

L = length of a side of the rectangle,

B= breadth of a side of the rectangle.

2.Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**PROGRAM-6**

**AIM:** **Write a Java Program for calculating area of triangle using herons formula.**

**CODE:**

import java.util.Scanner;

public class HeronsFormula {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

double a = scanner.nextDouble();

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

double b = scanner.nextDouble();

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

double c = scanner.nextDouble();

if (a + b > c && a + c > b && b + c > a) {

double s = (a + b + c) / 2;

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

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

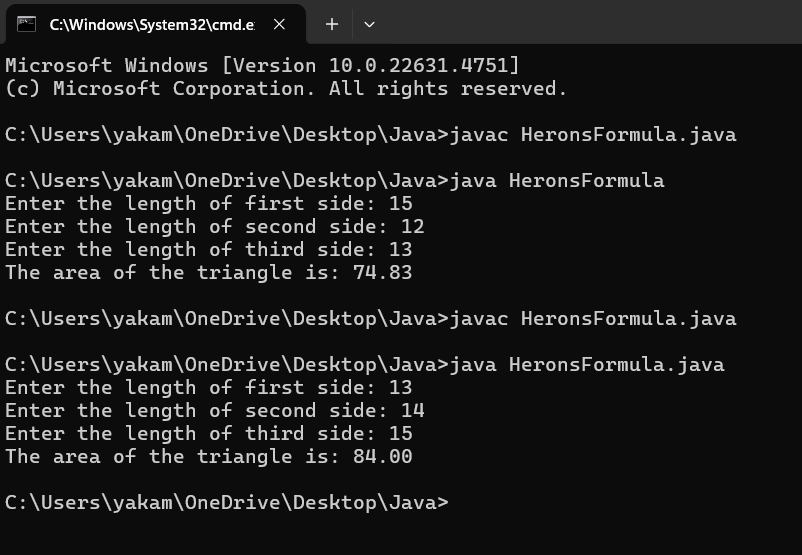
} else {

System.out.println("The given sides do not form a valid triangle.");

}

scanner.close();

}

}  
**OUTPUT:**

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | While printing the variable not giving + sign. | We should give correct indentation. |
| 2. | Not closing the scanner. | Closing the scanner is must. |

**IMPORTANT POINTS:**

1.Here, we’re finding the area of a triangle using heron’s formula.

2.Heron’s formula for finding a triangle is:

S = (a +b +c)/2

Where S is the semi-perimeter of the triangle.

Now the area formula is:

Area = sqrt(s\*(s-a)\*(s-b)\*(s-c)).

**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 these methods named start(),stop(),service()

4.Create the objects named car, car1,car3;

**CODE:**

class Car {

private String car\_color;

private String car\_brand;

private String fuel\_type;

private double mileage;

public Car(String car\_color,String car\_brand,String fuel\_type,double mileage){

this.car\_color = car\_color;

this.car\_brand = car\_brand;

this.fuel\_type = fuel\_type;

this.mileage = mileage;

System.out.println("the car is described as "+car\_color+","+car\_brand+","+fuel\_type+","+mileage+" miles");

}

public void start(String car\_brand){

System.out.println(car\_brand +" is starting");

}

public void stop(String car\_brand){

System.out.println(car\_brand +" is stoping");

}

public void service(String car\_brand){

System.out.println(car\_brand +" needs service");

}

public static void main(String[] args){

// object 1 is taken as car1

Car car1 = new Car("White","Benz","Diesel",47.9);

car1.start("Benz");

car1.stop("Benz");

car1.service("Benz");

// object 2 is taken as car 2

Car car2 = new Car("Red","Ferrari","Petrol",14.6);

car2.start("Ferrari");

car2.stop("Ferrari");

car2.service("Ferrari");

// object 3 is taken as car 3

Car car3 = new Car("Grey","Lamborghini","Gasoline",22);

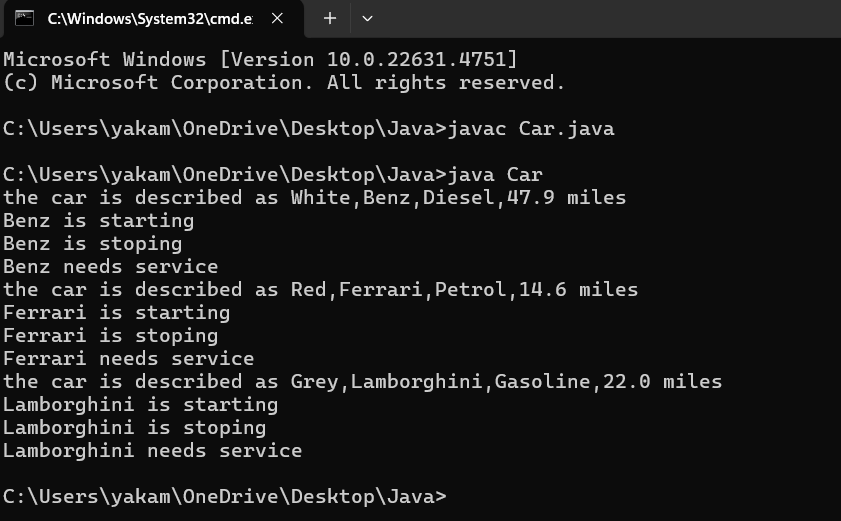
car3.start("Lamborghini");

car3.stop("Lamborghini");

car3.service("Lamborghini");

}

}

**OUTPUT:  
**

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR** | **ERROR RECTIFICATION** |
| **1.** | Not putting the semi-colon; after calling the function. | After Start, Stop, Service not giving the parenthesis ( ). |
| **2.** | After Start, Stop, Service not giving the parenthesis ( ). | After every method, put the parenthesis ( ). |

**IMPORTANT POINTS:**

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

CLASS DIAGRAM:

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

PROGRAM- 2:

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

CODE:

import java.util.Scanner;

public class bankaccount {

private double balance;

public bankaccount(double initialbalance) {

this.balance = initialbalance;

}

public void deposit(double amount) {

balance = +amount;

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

}

public void withdraw(double amount) {

if (balance < amount)

{

balance = amount;

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

} else

{

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

}

}

public double getbalance() {

return balance;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter initial balance: ");

double initialbalance = scanner.nextDouble();

bankaccount account = new bankaccount(initialbalance);

System.out.print("Enter deposit amount: ");

double depositAmount = scanner.nextDouble();

account.deposit(depositAmount);

System.out.print("Enter withdrawal amount: ");

double withdrawalAmount = scanner.nextDouble();

account.withdraw(withdrawalAmount);

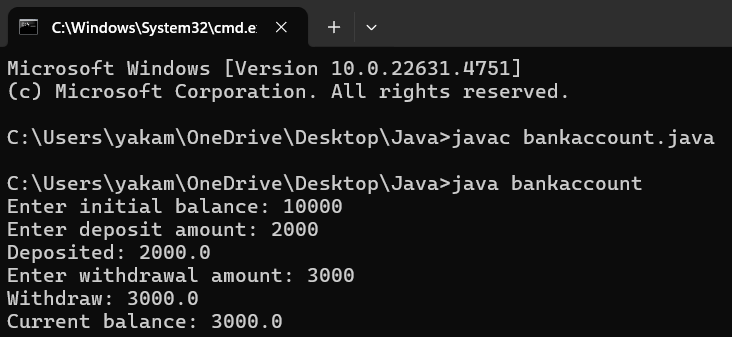
System.out.println("Current balance: " + account.getbalance());

scanner.close();

}

}

OUTPUT:



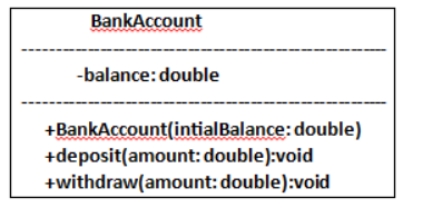
ERROR TABLE:

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

IMPORTANT POINTS:

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

CLASS DIAGRAM:



WEEK -4

PROGRAM – 1:

AIM: Write a java program with class named “book”, the class should contain various attributes .

CODE: class Book {

// Fields to hold book details

public String bookTitle;

public String bookAuthor;

public int bookYearOfPublication;

// Method to print book title

public void title() {

System.out.println("Book Title");

}

// Method to print book author

public void author() {

System.out.println("Book Year of Publishing");

}

// Main Method

public static void main(String[] args) {

// First book

Book book1 = new Book();

book1.bookTitle = "Atomic Habits";

book1.bookAuthor = "James Clear";

book1.bookYearOfPublication = 2018;

// Calling methods

book1.title();

book1.author();

// Printing book details

System.out.println("Book title is: " + book1.bookTitle);

System.out.println("Book author is: " + book1.bookAuthor);

System.out.println("Book year of publication is: " + book1.bookYearOfPublication);

// Second book

Book book2 = new Book();

book2.bookTitle = "South Pole Pig";

book2.bookAuthor = "James";

book2.bookYearOfPublication = 2014;

// Calling methods

book2.title();

book2.author();

// Printing book details

System.out.println("Book title is: " + book2.bookTitle);

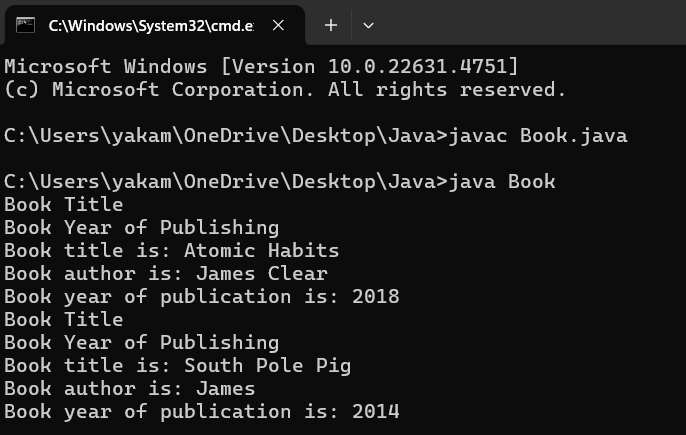
System.out.println("Book author is: " + book2.bookAuthor);

System.out.println("Book year of publication is: " + book2.bookYearOfPublication);

}

}

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 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.

CLASS DIAGRAM:

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

PROGRAM – 2:

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

CODE:

class MyClass {

// Static variable to hold count of objects created

static int count = 0;

// Constant variable for pi, initialized to 3.14

final double pi = 3.14;

// Constructor

public MyClass() {

// Increment count every time an object is created

count++;

}

// Main method to create objects and print values

public static void main(String[] args) {

// Creating three objects of MyClass

MyClass object1 = new MyClass();

MyClass object2 = new MyClass();

MyClass object3 = new MyClass();

// Printing the final values of count and pi

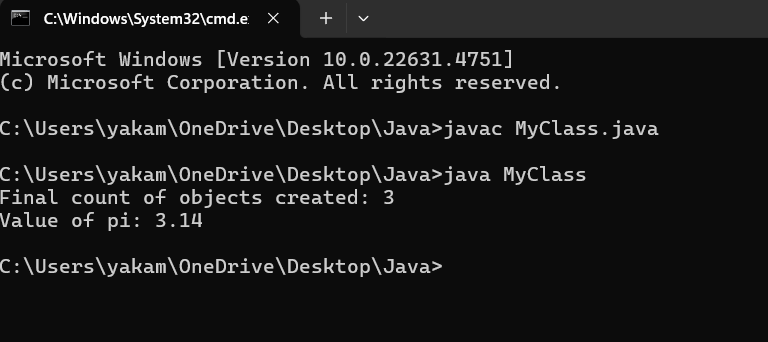
System.out.println("Final count of objects created: " + count);

System.out.println("Value of pi: " + object1.pi); // Pi is constant, and we can access it via any object

}

}

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

PROGRAM-1:

AIM: 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.

CODE:

class calculator {

protected double a, b;

public calculator(double a, double b) {

this.a = a;

this.b = b;

}

}

class Addition extends calculator {

public Addition(double a, double b) {

super(a, b);

}

public double add() {

return a + b;

}

}

class Subtraction extends Addition {

public Subtraction(double a, double b) {

super(a, b);

}

public double subtract() {

return a - b;

}

}

class Multiplication extends Subtraction {

public Multiplication(double a, double b) {

super(a, b);

}

public double multiply() {

return a \* b;

}

}

class Division extends Multiplication {

public Division(double a, double b) {

super(a, b);

}

public double divide() {

if (b != 0) {

return a / b;

} else {

System.out.println("Error");

return Double.NaN;

}

}

}

class Final extends Division {

public Final(double a, double b) {

super(a, b);

}

public void displayResults() {

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

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

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

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

}

}

import java.util.Scanner;

public class allcalculator {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

double a = input.nextDouble();

System.out.println("Enter b number: ");

double b = input.nextDouble();

Final calc = new Final( a, b);

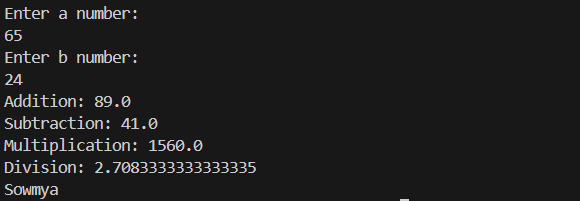
calc.displayResults();

System.out.println("Sowmya ");

input.close();

}

}  
OUTPUT:

  
CLASS DIAGRAM:

CLASS DIAGRAM:

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. not providing the return method correctly. 2. Not mentioning super to obtain the super class constructor. | 1. After declaring methods, we must provide the return method correctly. 2. To obtain the super class we need to mention super. |

IMPORTANT POINTS:

1. To get the inputs from the user we use import java.util.Scanner; this is a package.
2. Scanner class is used to get the user input.
3. in java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.
4. to import a whole package, end the sentence with an asterisk sign(\*).

PROGRAM-2

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

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

Question:

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

CODE:   
class Vehicle {

private String brand;

private 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 Car extends Vehicle {

private int doors;

private int capacity;

public Car(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);

}

@Override

void details() {

super.details();

carDetails();

} }

class Bike extends Vehicle {

private boolean gears;

Bike(String brand, int speed, boolean gears) {

super(brand, speed);

this.gears = gears;

}

void bikeDetails() {

System.out.println(gears ? "This bike has gears." : "This bike does not have gear system.");

}

@Override

void details() {

super.details();

bikeDetails();

} }

class Truck extends Vehicle {

private int tons;

Truck(String brand, int speed, int tons) {

super(brand, speed);

this.tons = tons;

}

void truckDetails() {

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

}

@Override

void details() {

super.details();

truckDetails();

} }

class Rent {

public static void main(String[] args) {

Car c = new Car("Toyota", 120, 5, 5);

c.details();

Bike b = new Bike("KTM", 80, true);

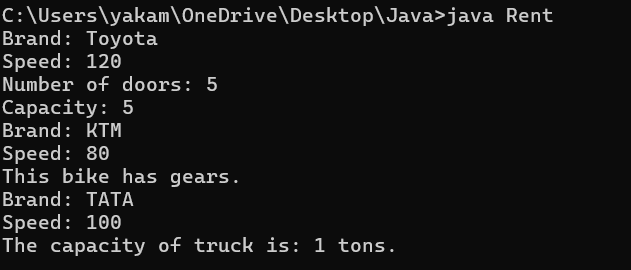
b.details();

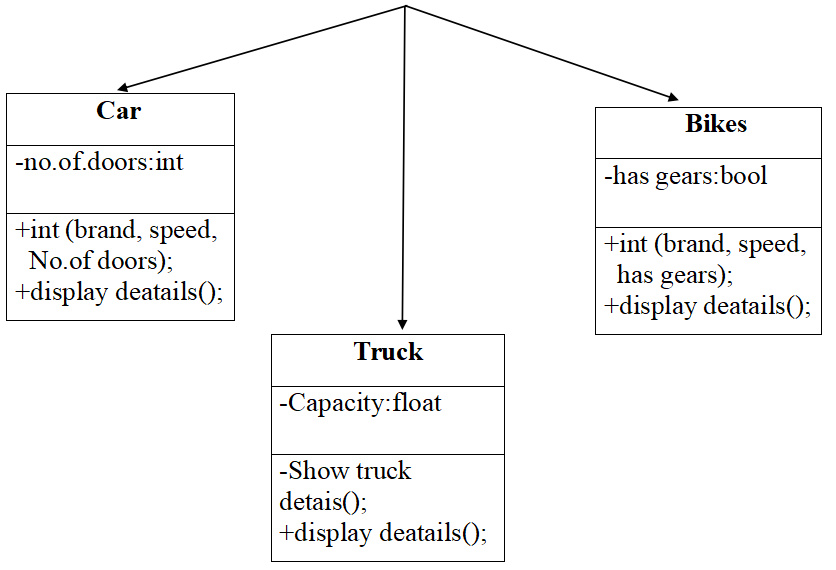
Truck t = new Truck("TATA", 100, 1);

t.details();

} }

OUTPUT:



CLASS DIAGRAM:  
  


ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Declaring two superclasses inside the same file. 2. Not declaring the variable using ‘this’ keyword inside the constructor. | 1. Make two separate files to save the two super classes. 2. Declare the variable using this keyword to run the program. |

IMPORTANT POINTS:

1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. a double method can represent more decimal point numbers than float method.
4. the void keyword in java is used to specify that a method does not return any value. it is a return type that indicates the method performs a function and doesn't produce a result.

Answer:

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

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

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

**WEEK-6**

**PROGRAM-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 about car (car company, seating capacity, petrol or not).

**CODE:**

class Vehicle {

String car\_company;

String car\_model;

long car\_price;

int seating\_capacity;

boolean petrol;

Vehicle(String car\_company, String car\_model, long car\_price, int seating\_capacity, boolean petrol) {

this.car\_company = car\_company;

this.car\_model = car\_model;

this.car\_price = car\_price;

this.seating\_capacity = seating\_capacity;

this.petrol = petrol;

}

void displayInfo() {

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

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

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

System.out.println("Car seating capacity: " + seating\_capacity);

System.out.println("Car uses petrol: " + petrol);

}

}

class Car extends Vehicle {

Car(String car\_company, String car\_model, long car\_price, int seating\_capacity, boolean petrol) {

super(car\_company, car\_model, car\_price, seating\_capacity, petrol)

}

}

public class Main {

public static void main(String[] args) {

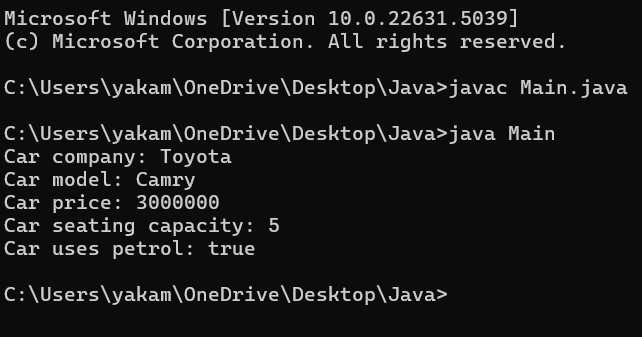
// Creating a Car object with correct arguments

Car c1 = ew Car("Toyota", "Camry", 3000000, 5, true);

c1.displayInfo(); // Display car details

}

}  
**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Incorrect class name for main method(Truck).  2. Inconsistent car model output in displayinfo(). | 1.Rename Truck to Main or place main inside car or vehicle.  2. Ensure Car correctly passes Toyota” to super(car\_model,color,fueltype) |

**IMPORTANT POINTS:**

**1.Inheritance:** The Car class extends the Vehicle class, demonstrating inheritance in Java.

**2.Constructor Chaining:**The Car class calls the parent constructor using super(car\_model, color, fuel\_type); to initialize inherited attributes.

**3.Method Overriding:**The Car class overrides the displayInfo() method from Vehicle and calls super.displayInfo() to reuse the parent method before adding its own output.

**4.Incorrect** main **Class Name:**The main method is inside Truck, which is unrelated to Vehicle and Car. The class should be renamed for clarity.

**CLASS DIAGRAM:**

|  |
| --- |
| **Vehicle** |
| * Brand: String * Speed: int |
| + vehicle(brand: string  Speed: int)  +start vehicle(): void  +displaydetails():void |

**PROGRAM-2:**

**AIM:** A college is developing an automated admission system that verifies students eligibility(UG) and postgraduation(PG) programs. Each program has different eligibility criteria based on the students percentage in their previous qualification.

1. UG admission recquire a minimum of 60%.

2. PG admission recquire a minimum of 70%.

**CODE:**

class Student {

String name;

double percentage

Student(String name, double percentage) {

this.name = name;

this.percentage = percentage;

}

void studentsinfo() {

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

System.out.println("Percentage: " + percentage);

}

}

class UG extends Student {

UG(String name, double percentage) {

super(name, percentage);

}

void checkEligibility() {

if (percentage >= 60) {

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

}

else {

System.out.println(name + " is not eligible for admission in UG.");

}

}

}

class PG extends Student {

PG(String name, double percentage) {

super(name, percentage);

}

void checkEligibility() {

if (percentage >= 70) {

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

}

else {

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

}

}

}

public class AutomatedAdmission {

public static void main(String[] args) {

UG ug = new UG("Mokshitha", 80);

ug.studentsinfo();

ug.checkEligibility();

PG pg = new PG("Charan", 75);

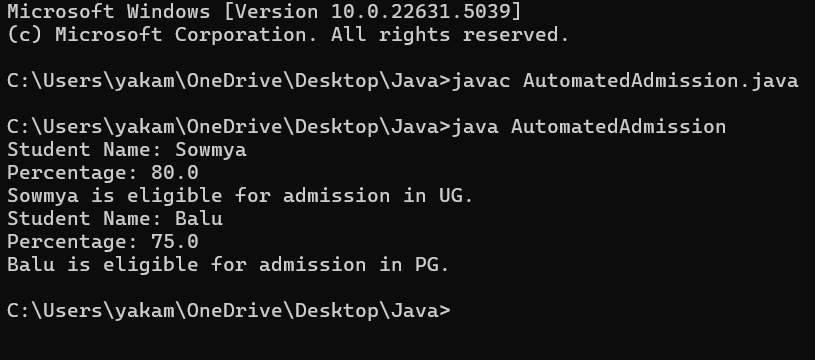
pg.studentsinfo();

pg.checkEligibility();

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| **1.Scanner nextLine() issue after nextDouble():** After scanner.nextDouble(), the newline character remains in the buffer, causing nextLine() to be skipped.  **2.Program type input case sensitivity issue**: If the user enters ug or pg in lowercase, it may cause incorrect comparisons. | **1**.Add scanner.nextLine(); after nextDouble(); to consume the leftover newline.  **2.**Use program.toUpperCase() to ensure case-insensitive comparison. |

**IMPORTANT POINTS:**

**1.User Input Handling:** Uses Scanner to take user input for name, percentage, and program type.

**2.Decision Making with Conditions:** Uses if-else statements to check eligibility criteria.

**3.String Handling:** Converts program input to uppercase (toUpperCase()) to handle case variations.

**4.Closing Scanner:** Properly closes scanner using scanner.close(); to prevent resource leaks.

**CLASS DIAGRAM:**

|  |
| --- |
| **AutomatedAdmission** |
| * Scanner: scanner * Name: String * Percentage : double * Program: stirng |
| + main(args:String[]): void  +takeInput(): void  +checkEligibility(): void  +closeScanner(); void |

**PROGRAM-3:**

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

1. Add two integers

2. Add two doubles

3. Add three integers

**CODE:**

class Calculator\_6 {

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 Main\_6 {

public static void main(String[] args) {

Calculator\_6 calculator = new Calculator\_6();

System.out.println("Addition of two integers: " + calculator.add(5, 10));

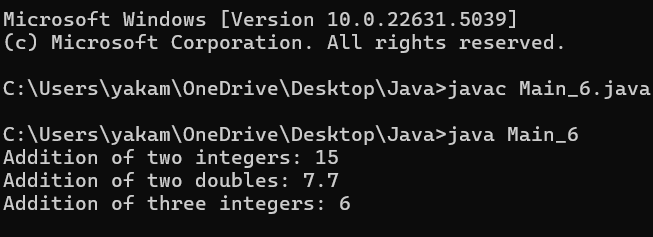
System.out.println("Addition of two doubles: " + calculator.add(5.5, 2.2));

System.out.println("Addition of three integers: " + calculator.add(1, 2, 3));

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1.Method parameters missing spaces. E.g.,”inta, intb”should be “int a, int b”  2.Inconsistent indentation in method bodies | 1**.** Add proper spacing between parameters: (int a, int b)  2.Fix indentation:  Consistent 4 space o indentation. |

**IMPORTANT POINTS:**

**1.Method Overloading:** The add method is overloaded with different parameter types and counts, demonstrating compile-time polymorphism.

**2.Automatic Method Selection:** Java selects the appropriate add method based on the argument types during compilation.

**CLASS DIAGRAM:**

|  |
| --- |
| **Calculator** |
| + add(int, int): int  +add(double, double): double  +add(int,int,int): int |

**PROGRAM-4:  
AIM:** Create a shape class with a method to calculate area i.e., overloaded for different shapes eg: Squares, Recatangle. 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 double calculateArea(double length, double width) {

return length \* width;

}

}

class Circle extends Shape {

private double radius

public Circle(double radius) {

this.radius = radius;

}

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

class Tools {

public static void main(String[] args) {

Shape shape = new Shape();

Circle circle = new Circle(5);

System.out.println("Area of square: " + shape.calculateArea(4));

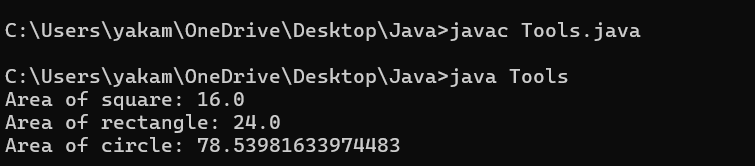
System.out.println("Area of rectangle: " + shape.calculateArea(4,

6));

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

}

}

**OUTPUT:  
**

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Method calls in main are missing an object reference (e.g., calculateArea(4) instead of s.calculateArea(4)).  2. Circle class method does not override theparent class method properly. | 1.Use s.calculateArea(4) and c.calculateArea(2) to call the method correctly.  2. Ensure @Override is used, and the method signature should match correctly. |

**CLASS DIAGRAM:**

|  |
| --- |
| **SHAPE** |
| + CalculateArea(side:double): double  +CalculateArea(width: double, length: double): double |

|  |
| --- |
| **CIRCLE** |
| + CalculateArea(radius: double): double |

|  |
| --- |
| **Tools** |
| +main(args:String[]): Void |

**IMPORTANT POINTS:**

**1.Inheritance**: Circle class extends Shape, inheriting its methods.

**2.Method Overloading**: Shape has multiple calculateArea methods with different parameters.

**3.Method Overriding**: Circle overrides calculateArea from Shape to implement its own formula.

**4.Polymorphism**: The overridden method in Circle demonstrates runtime polymorphism.

**5.Proper Object Reference**: Methods should be called using an object (s.calculateArea(4), c.calculateArea(2)).