**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**   **Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: M.** **Mahamad**

**Roll No: 24220**

**Verified By**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **WEEK** | **SNO.** | **TITLE** | **DATE** | **PAGE** | **SIGNATURE** |
| **WEEK-1** |  |  |  |  |  |
|  | **1** | **How to install jdk (java development kit) from Oracle.** |  | **9-11** |  |
|  | **2** | **Write a java program of student details.** |  | **11-12** |  |
| **WEEK-2** |  |  |  |  |  |
|  | **1** | **Write a java program to calculate the area of rectangle.** |  | **13-14** |  |
|  | **2** | **Write a java program to find simple interest where all inputs are taken from user.** |  | **14-16** |  |
|  | **3** | **Write a java program to calculate the FibonacciSequence of a input taken from user.** |  | **16-19** |  |
|  | **4** | **Write a java program to convert temperature from Celsius to Fahrenheit.** |  | **19-20** |  |
|  | **5** | **Write a java program to convert temperature from Fahrenheit to Celsius:** |  | **21-22** |  |
|  | **6** | **Write a java program to calculate factorial of a number, read the input from user.** |  | **22-24** |  |
|  | **7** | **Write a java program to calculate the area of triangle by using heron’s formula.** |  | **24-26** |  |
| **WEEK-3** |  |  |  |  |  |
|  | **1** | **To create a java program with the following instructions:**   1. **Create a class with name “Car”** 2. **Create 4 attributes, named: car\_color, car\_brand, fuel\_type, mileage.** 3. **Create 3 methods, named: start(), service(), stop()** 4. **Create 3 objects, named: car1, car2, car3** 5. **Create a constructor, which should print, “Welcome to car garage”.** |  | **27-31** |  |
|  | **2** | **To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().**   1. **deposit(): Whenever an amount is deposited, it has to be update the current amount.** 2. **withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”) .** |  | **31-34** |  |
| **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.** |  | **35-37** |  |
|  | **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.** |  | **38-40** |  |
| **WEEK-5** |  |  |  |  |  |
|  | **1** | **Create a calculator using the operations including addition, subtraction**  **Multiplication and division using multilevel inheritance and display the desired output.** |  | **41-44** |  |
|  | **2** | **A vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed.**   1. **Cars should have an additional property: number of doors, seating capacity.** 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.**   **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?**    * **Truck should include an additional property: capacity (in tons).**    * **Create a showTruckDetails() method to display the truck's capacity.**    * **Write a constructor for truck that initializes all properties.** 3. **Implement the Truck class and update the main method to create a Truck object and also create an object for car and bike subclasses. Properly display its details.** |  | **45-49** |  |
| **WEEK-6** |  |  |  |  |  |
|  | **1** | **Write a java program to create a vehicle class with a method displayInfo().Override this method in the car subclass to provide specific information a About car.** |  | **50-53** |  |
|  | **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%.** |  | **53-57** |  |
|  | **3** | **Create a calculator class with overloaded methods to perform addition.**  **1.Add two integers. 2. Add two doubles.**  **3. Add three integers.** |  | **57-60** |  |
|  | **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.** |  | **61-63** |  |
| **WEEK-7** |  |  |  |  |  |
|  | **1** | **Write a java program to create an abstract class Animal with an abstract Method called sound. Create subclasses lion and tiger that extends the animal Class and implements the sound() method to make a specific sound for each animal.** |  | **64-66** |  |
|  | **2** | **Write a java program to create an abstract class shape3D with abstract methods calculateVolume() and calculateSurfaceArea().Create subclasses sphere and cube that extend the shape3D class and implement the respective methods to calculate the volume and surface area of each shape.** |  | **66-69** |  |
|  | **3** | **Write a java program using an abstract class to define a method for pattern printing. Create an abstract class named PatternPrinter with an Abstract method printpattern(int n) and a concrete method to display the pattern title.Implement two subclasses .**  **1.star pattern- prints a right angled triangle of stars(\*)**  **2.Number pattern- prints a right angled triangle of increasing numbers.**  **In the main() method create objects of both subclasses and print the patterns For a given number of rows.** |  | **69-73** |  |
| **WEEK-8** |  |  |  |  |  |
|  | **1** | **Write a Java program to create an interface Shape with the getPerimeter method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.** |  | **74-77** |  |
|  | **2** | **Write a Java program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball, and Basketball that implement the Playable interface and override the play() method to play the respective sports.** |  | **77-81** |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 

# 

**WEEK-1**

1. **Process of Installing JDK (Java Development Kit)**

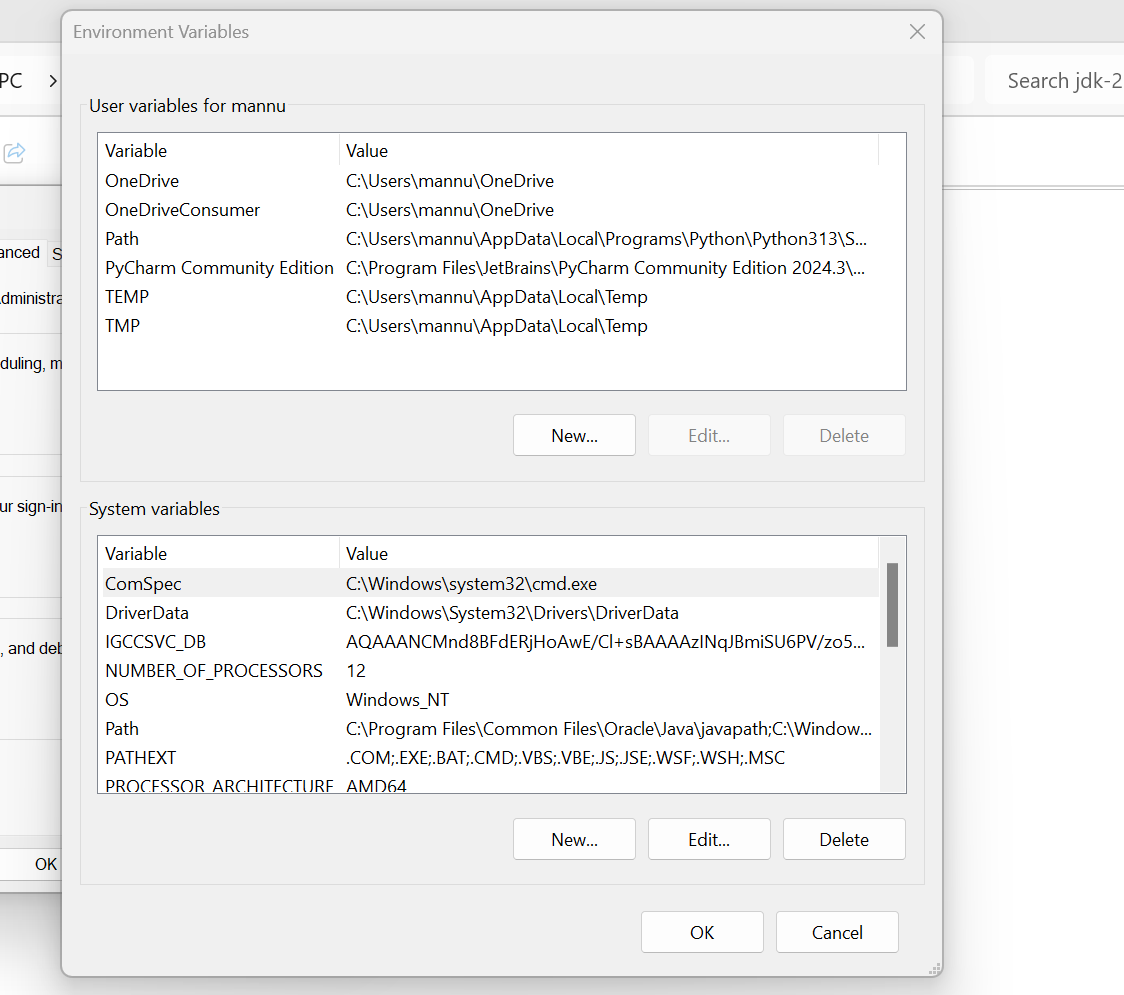
**Installing JDK (Java Development Kit):**

* 1. **Download JDK:**
* Go to the Oracle JDK download page in google and click on JDK-21 version which is Long term support (LTS) version.
* Click the download link as your operating system (Windows, macOS, or Linux).
  1. **Install JDK:**
* Once downloaded, run the installer.
* Follow the given instructions and keep clicking "Next" until it is done.
  1. **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 on path and new under **System Variables**:

**Variable value:** The folder address where JDK is installed (like

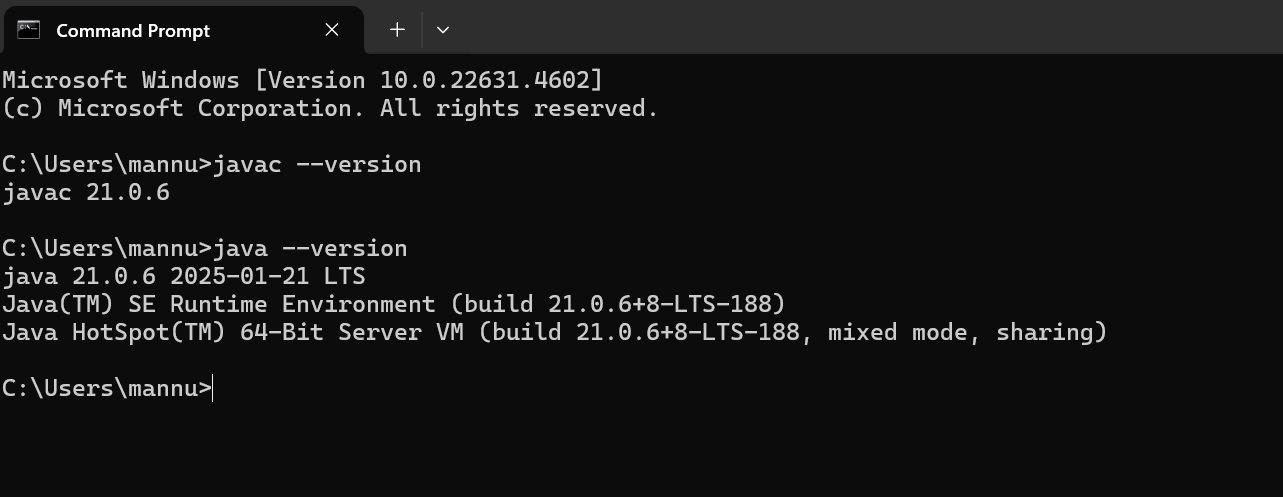
C:\Program Files\Java\jdk-21\bin).

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



**Checking JDK Version: -**

* 1. **Open Command Prompt:**
* Presswin+R, typecmd, and press Enter.
  1. **Check Version:**
* Type java -version and press Enter.
* Type javac --version and press Enter.



1. **Simple Java Program for printing Name, Class, Roll No, of a Student. Write your code in Notepad and execute it in cmd prompt .**

**CODE: -**

class Main

{

public static void main(String[] args)

{

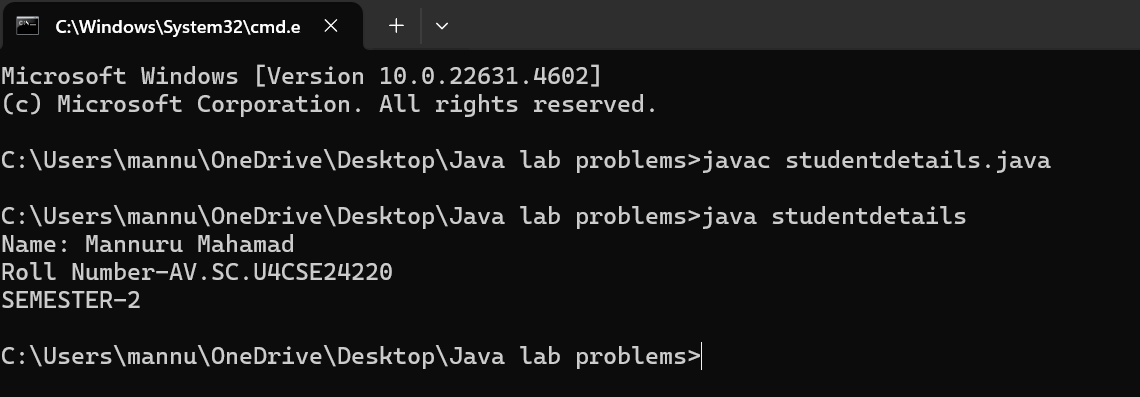
System.out.println("Name: Mannuru Mahamad");

System.out.println("Roll Number-AV.SC.U4CSE24220");

System.out.println("SEMESTER-2");

}

}

**Output: -**

**Errors:**

|  |  |  |
| --- | --- | --- |
| 1 | Syntax error | Semicolon added |
| 2 | Runtime error | Copied correct path |
| 3 | Name error | rectified |

**WEEK -2**

**1.Write a java program to calculate the area of rectangle:**

**CODE:**

import java.util.\*;

classdemo

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println(“enter a number:”);

int l = sc.nextInt();

System.out.println(“enter a number;”);

int b = sc.nextInt();

int a = l\*b;

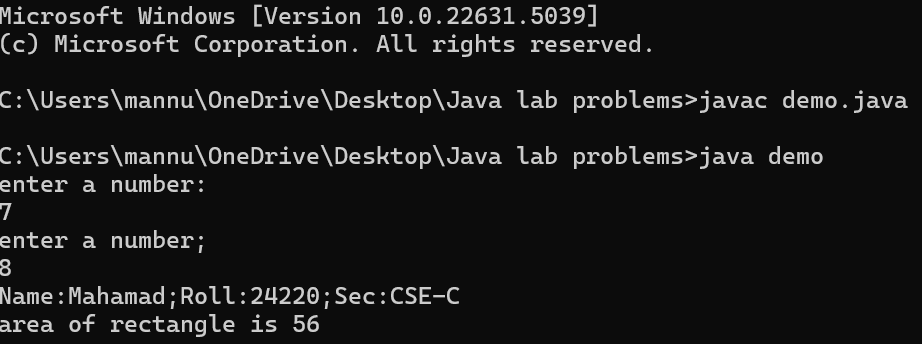
System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

System.out.println(“area of rectangle is “+a);

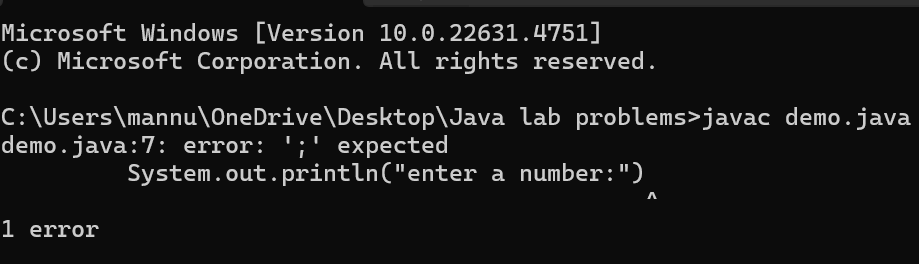
}

}

**OUTPUT:**



Negative Case:



**Errors:**

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

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

**CODE:**

import java.util.\*;

class test

{

public static void main(String[] args)

{

System.out.println(" taking input");

Scanner sc = new Scanner(System.in);

System.out.println("enter INTa number:");

float p = sc.nextFloat();

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

float t = sc.nextFloat();

System.out.println("enter a num:");

float r = sc.nextFloat();

float s = (p\*t\*r)/100;

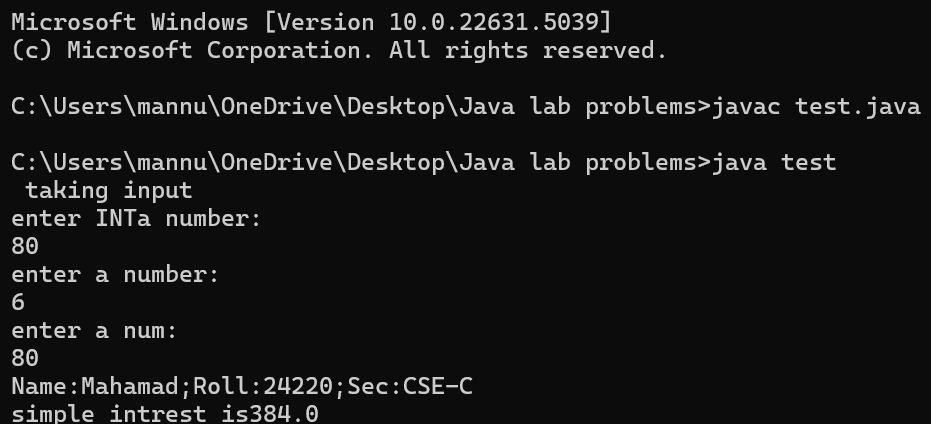
System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

System.out.println("simple intrest is"+s);

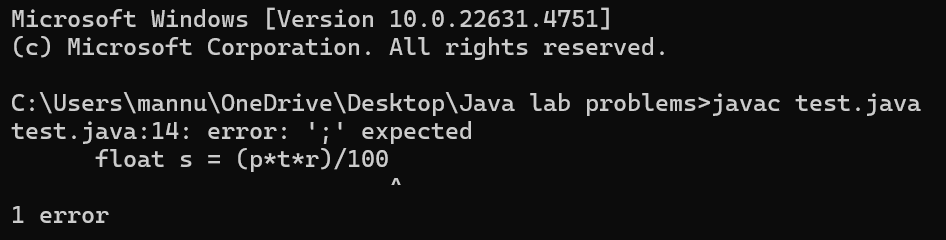
}

}

**OUTPUT:**



Negative Case:



**Errors:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | rectification |
| 1 | Runtime error | Incorrect path | Copied correct path |
| 2 | Syntax error | { missing | { added |
| 3 | Logical error | Wrong formula | Formularectified |

**3.Write a java program to calculate the FibonacciSequence of a input taken from user:**

**CODE:**

import java.util.\*;

class fibo

{

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;

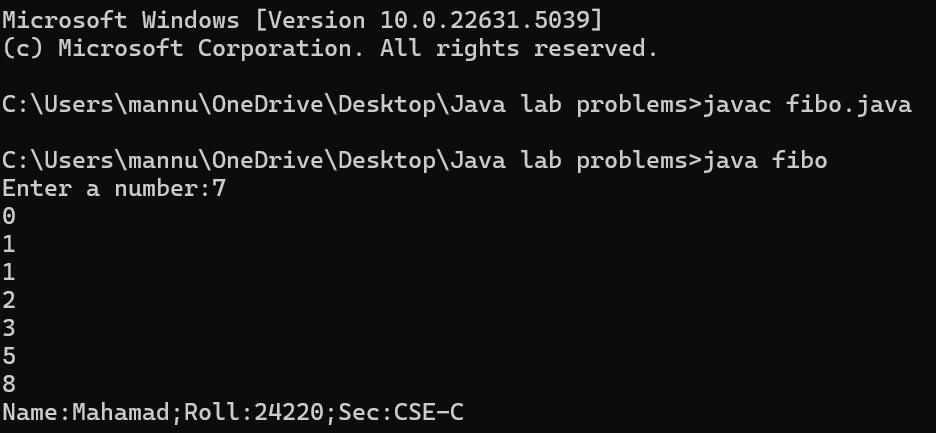
System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

}

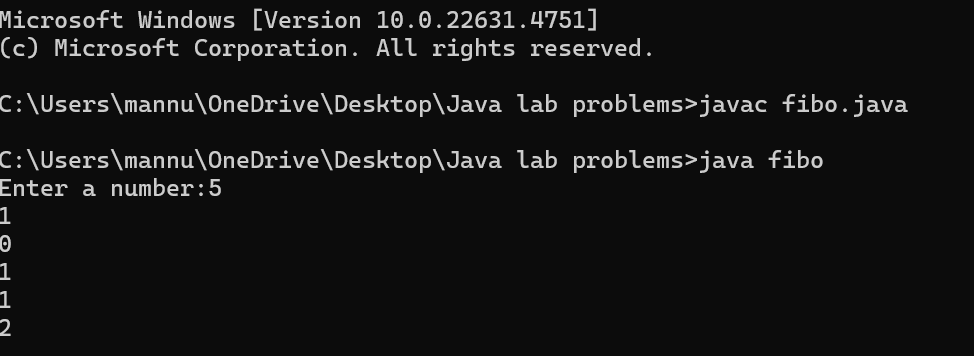
}

}

**OUTPUT:**



**Negative Case:**

****

**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 | Acolnidded sem |
| 3 | Runtime error | Incorrect path | Copied correct path |

**4.Write a java program to convert temperature from Celsius to Fahrenheit:**

**CODE:**

import java.util.\*;

class heat

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

float f;

System.out.println("Enter celsius temperature:");

float c = sc.nextFloat();

f = (c\*9/5)+32;

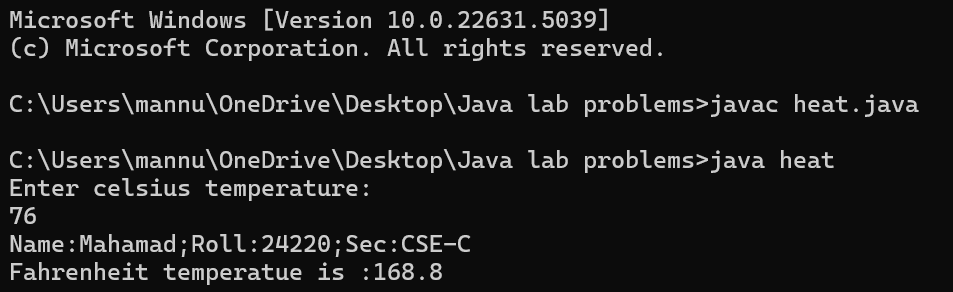
System.out.println("Fahrenheit temperatue is : "+f);

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

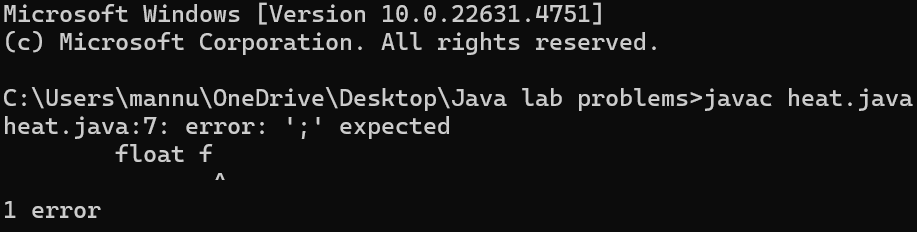
}

}

**OUTPUT:**



**Negative case:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Runtime error | Incorrect path selection | Correct path added |
| 2 | Logical error | Incorrect formula | Correct formula  rectified |
| 3 | Import package error | Incorrect importing of Packages | Imported util.\*;  Package |

**5.Write a java program to convert temperature from Fahrenheit to Celsius:**

import java.util.\*;

class far

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

float c;

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C")

System.out.println(" Enter temperature in farienheit :");

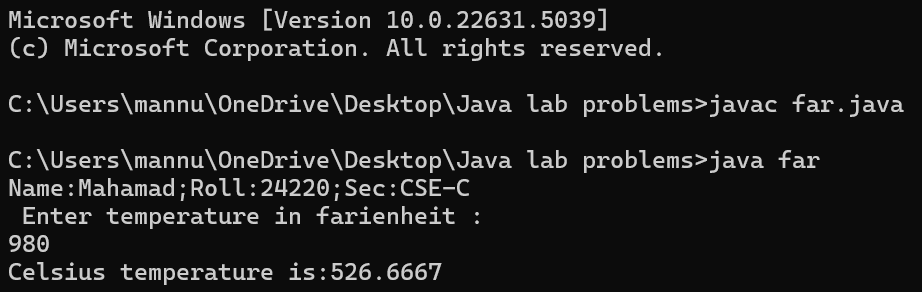
float f = sc.nextFloat();

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

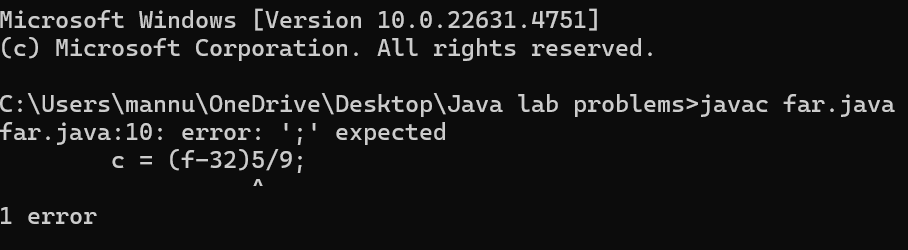
System.out.println("Celsius temperature is:"+c);

}

}

**OUTPUT:** ****

**Negative Case:**

****

**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | rectification |
| 1. | Syntax | No semicoln | Added semicoln |
| 2. | Logical error | Due to incorrect input | Corrected by giving correct input |
| 3. | Runtime error | Incorrect path | Using correct path |

**6. Write a java program to calculate factorial of a number , read the input from user:**

Syntax:

import java.util.\*;

class factorial

{

public static void main(String[] args)

{

int number;

Scanner sc = new Scanner(System.in);

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

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

number= sc.nextInt();

int answer= factorial(number);

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

}

static int factorial(int n){

{

if (n == 1)

return 1;

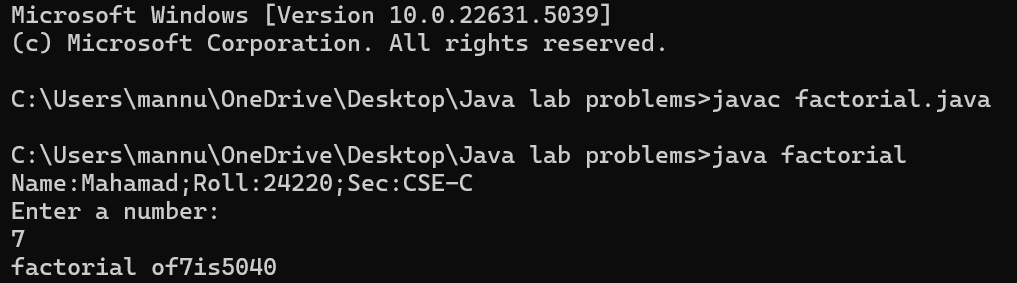
}

return n \* factorial(n - 1);

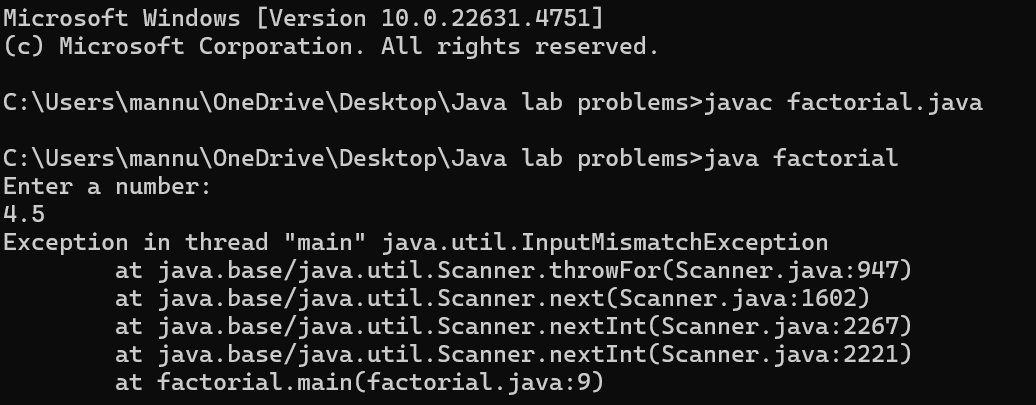
}

}

**OUTPUT:**

****

**Negative Case:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Undeclared variable error | Missing variable | Variable declared |
| 2 | Missing import statement | Not importing packages | Packages imported |
| 3 | Logical error | Wrong formula | Formula rectified |

**7.Write a java program to calculate the area of triangle by using heron’s formula:**

import java.util.\*;

import java.lang.Math;

class heron

{

public static void main(String args[])

{

double s, c,a,b,p;

Scanner sc = new Scanner(System.in);

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

System.out.println("Enter the values of a , b and c:");

a = sc.nextDouble();

b = sc.nextDouble();

c = sc.nextDouble();

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

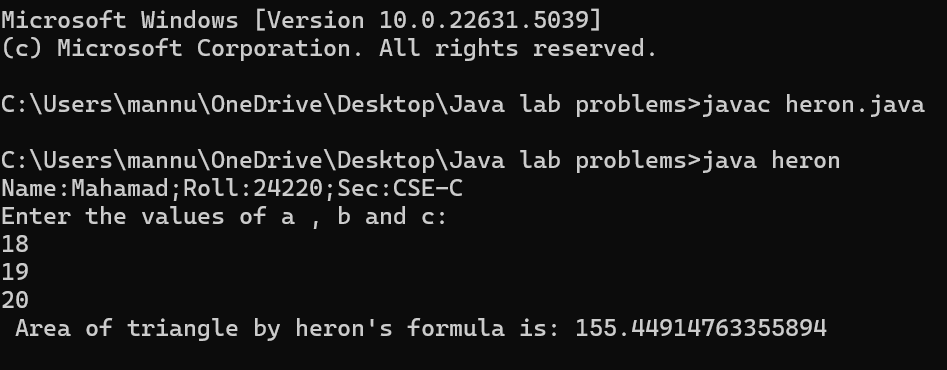
p = Math.sqrt(s\*(s-a)\*(s-b)\*(s-c));

System.out.println(" Area of triangle by heron's formula is: "+p);

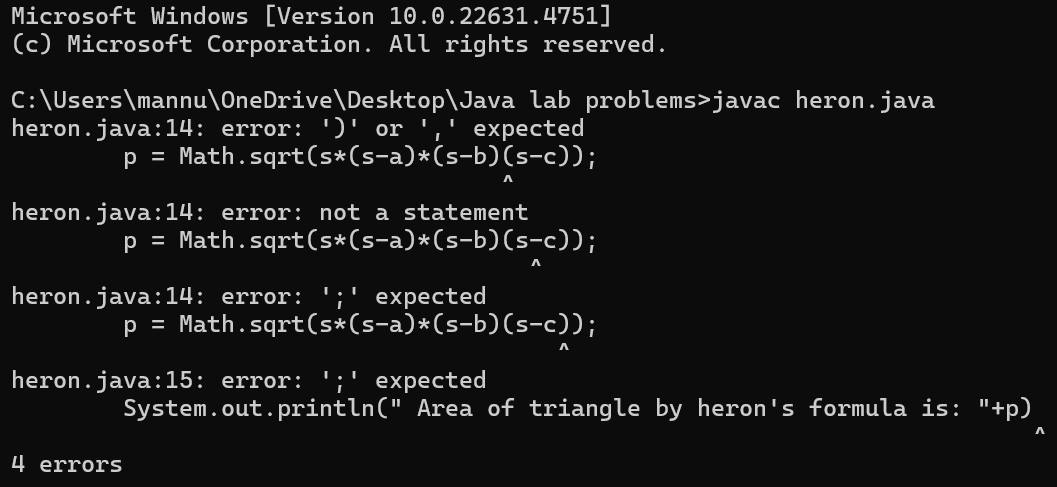
}

}

**OUTPUT:**

****

**Negative Case:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Error type | Reason for error | Rectification |
| 1 | Logical error | Incorrect formula | Formula rectified |
| 2 | Name error | Undeclared variable | Variable declared |

**WEEK -3**

**Aim:**

**To create java program with following instructions**

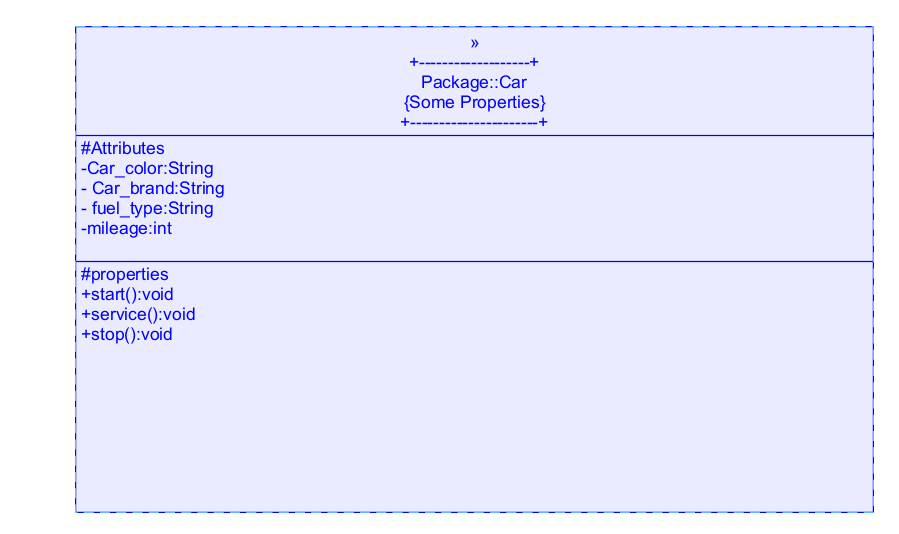
**1.Create a class with name car.**

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

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

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

**Class Diagram:**



**CODE:**

import java.util.\*;

class car

{

public String Car\_color;

public String Car\_brand;

public String fuel\_type;

public int mileage;

public void start()

{

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

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

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

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

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

}

public void service()

{

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

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

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

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

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

}

public void stop()

{

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

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

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

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

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

}

public static void main(String args[])

{ System.out.println("\nMahamad Mannuru\n\n");

car car1 = new car();

car1.Car\_color = "Blue";

car1.Car\_brand = "Audi";

car1.fuel\_type = "Deisel";

car1.mileage = 100;

car1.start();

car car2 = new car();

car2.Car\_color = "Red";

car2.Car\_brand = "Tesla";

car2.fuel\_type = "EV";

car2.mileage = 200;

car2.stop();

car car3 = new car();

car3.Car\_color = "Yellow";

car3.Car\_brand = "BMW";

car3.fuel\_type = "Petrol";

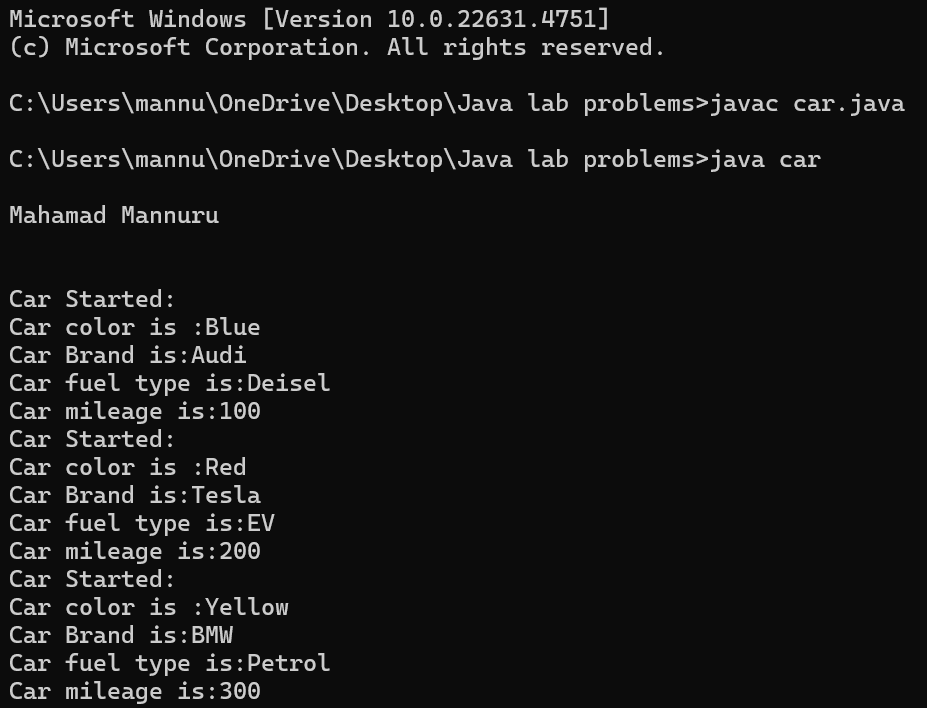
car3.mileage = 300;

car3.service();

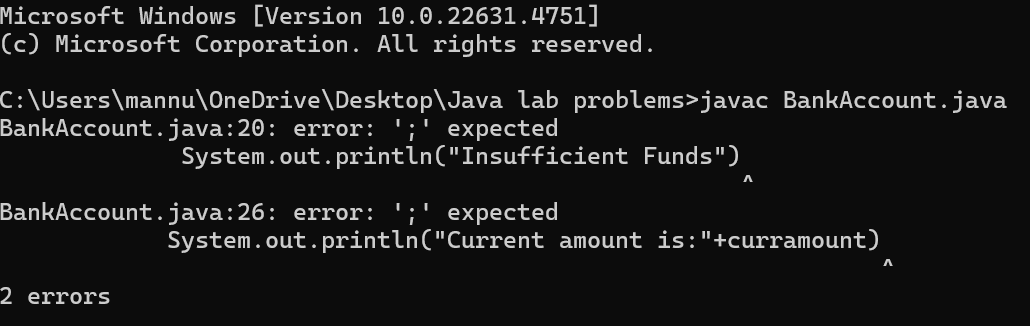
}

}

**OUTPUT:**

****

**Negative Case:**

****

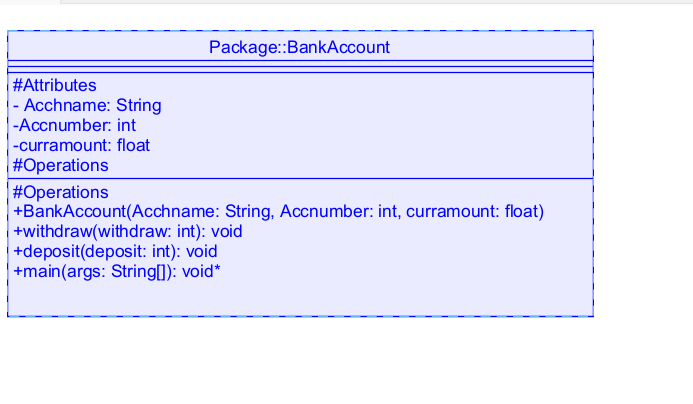
**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause of error | Rectification |
| 1 | Syntax Error | Missing ‘{‘ | ‘{‘ added |
| 2 | Compile time Error | Mispelled Variable call | Rectified with  Correct variable name |

**2.**

**Aim: To create a class bankAccount with methods deposit() and withdrawl.**

**Class Diagram:**



**Code:**

class BankAccount

{

public String Acchname;

public int Accnumber;

public float curramount;

BankAccount(String Acchname, int Accnum,float curramount)

{

this.Acchname = Acchname;

this.Accnumber = Accnumber;

this.curramount = curramount;

System.out.println("Enter Account holder name:"+Acchname);

System.out.println("Enter Account number:"+Accnum);

System.out.println("Enter current amount:"+curramount);

}

public void withdraw(int withdraw)

{

if(withdraw>curramount)

{

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

}

else

{

curramount = curramount-withdraw;

System.out.println("withdraw amount is:"+withdraw);

System.out.println("Current amount is:"+curramount);

}

}

public void deposit(int deposit)

{

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

curramount = curramount+deposit;

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

System.out.println("Total current amount is:"+curramount);

}

public static void main(String args[])

{

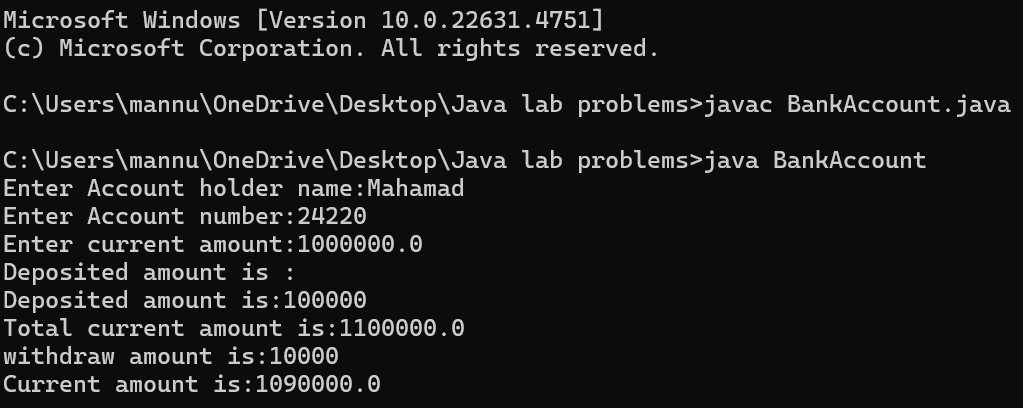
BankAccount b = new BankAccount("Hari",24210,100000);

b.deposit(10000);

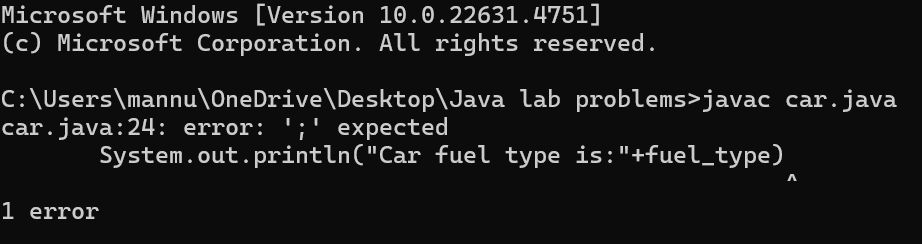
b.withdraw(500);

}}

**OUTPUT:**

****

**Negative Case:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| s.no | Error name | Cause of error | Rectification |
| 1 | Name Error | Undefined name | Correct variable  Name replaced |
| 2 | Syntax Error | Missing Parenthesis | Parenthesis Added |
| 3 | Logical Error | Incorrect Condition | Condition Rectified |

**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) {

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

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

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

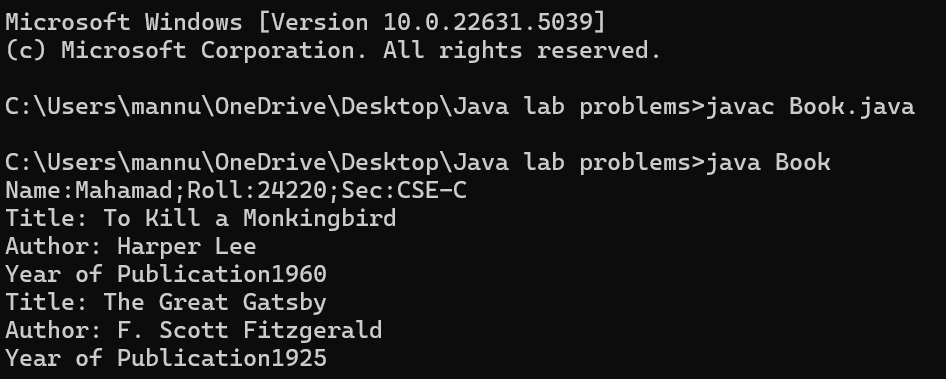
b1.displayDetails();

b2.displayDetails();

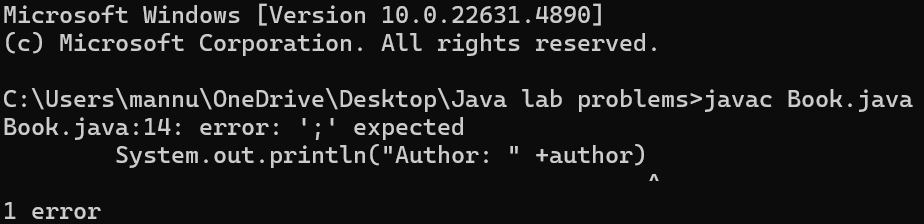
}

}

**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) {

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

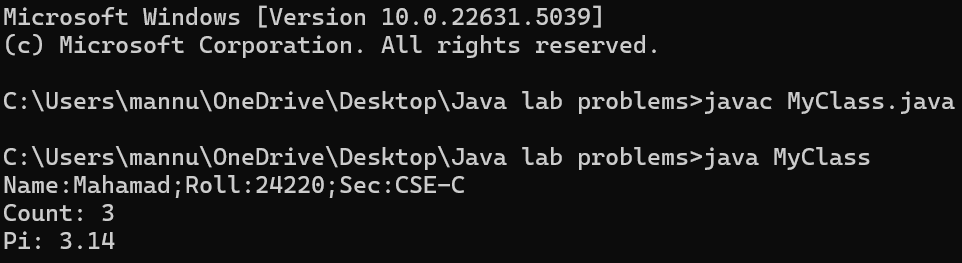
System.out.println("Count: " +count);

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

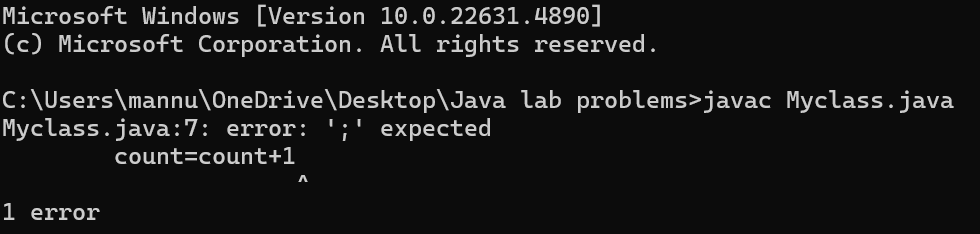
}

}

**OUTPUT:**

****

**NEGATIVE 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[])

{

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

division obj = new division();

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

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

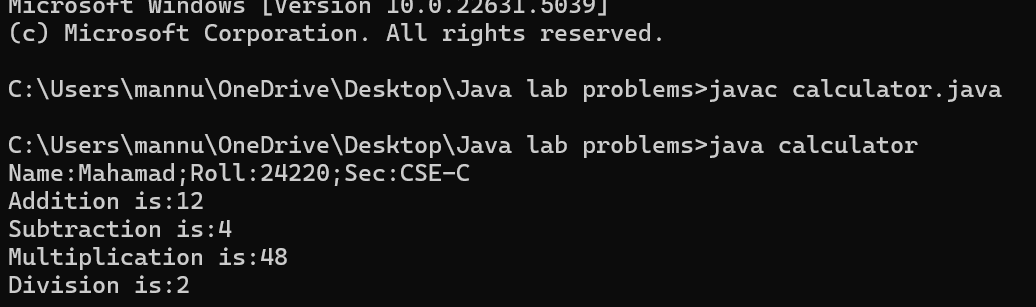
System.out.println("Multiplication is:"+obj.mult(12,4));

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

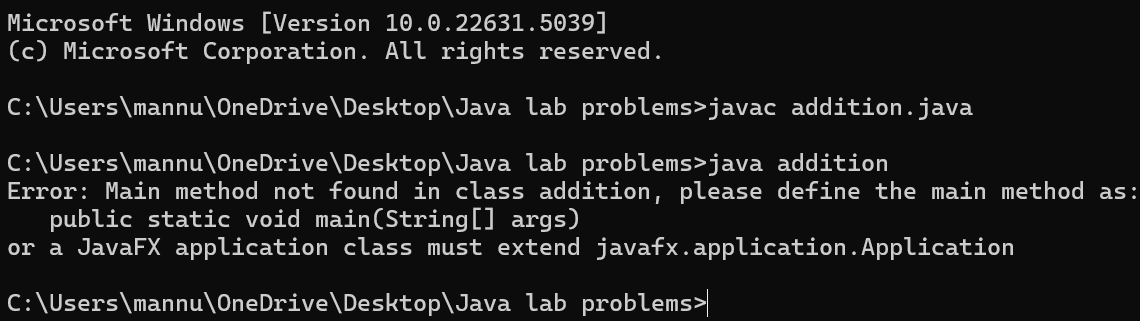
  }

}

**OUTPUT:**



**NEGATIVE CASE:**



**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 Inheritance.

**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:**



**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("Name:Mahamad;Roll:24220;Sec:CSE-C");

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);

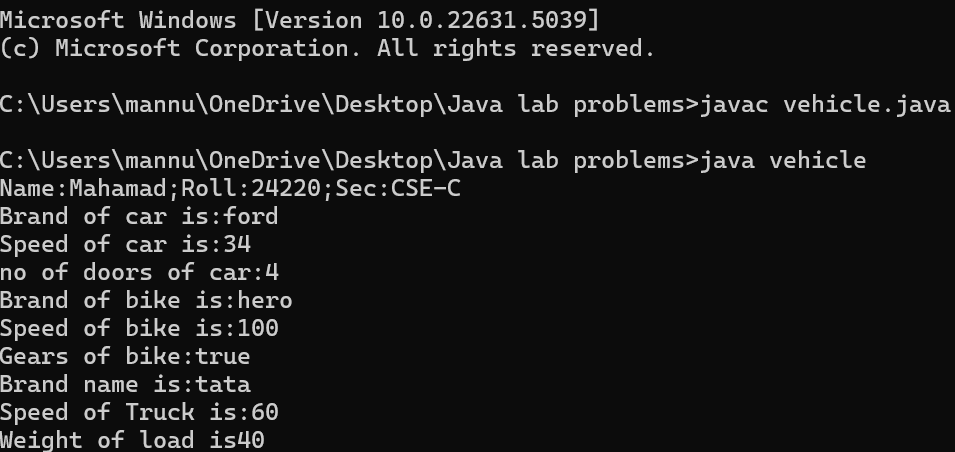
System.out.println("Speed of Truck is:"+speed);

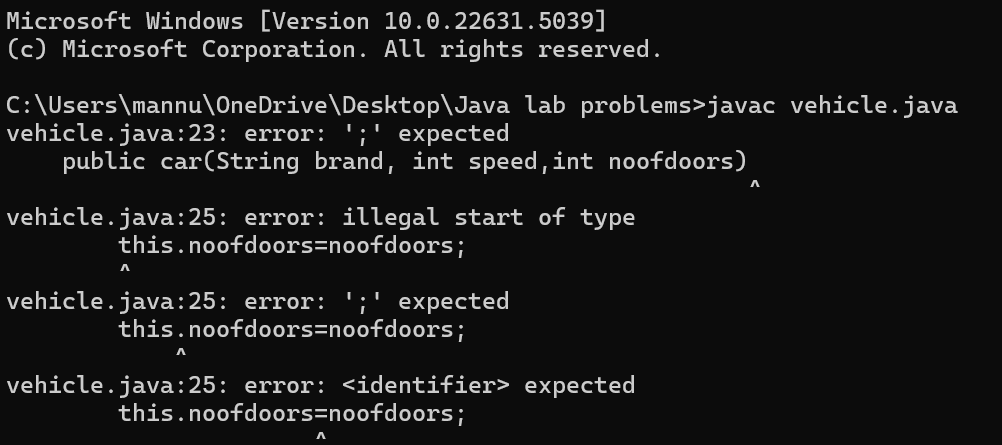
System.out.println("Weight of load is"+weight);

}

}

**OUTPUT:**

****

**NEGATIVE CASE:** ****

**Error Table:**

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

**Important Points:**

**Hierarchical Inheritence:**

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

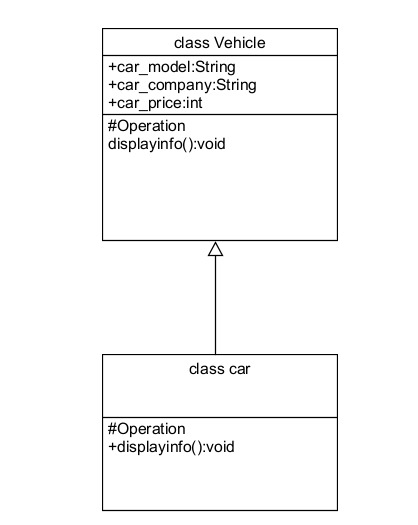
Single parent class.

**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.**

**CLASS DIAGRAM:**



**CODE:**

class vehicle

{

public String car\_model;

public String car\_company;

public int car\_price;

void displayinfo()

{

System.out.println("Mahamad 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;

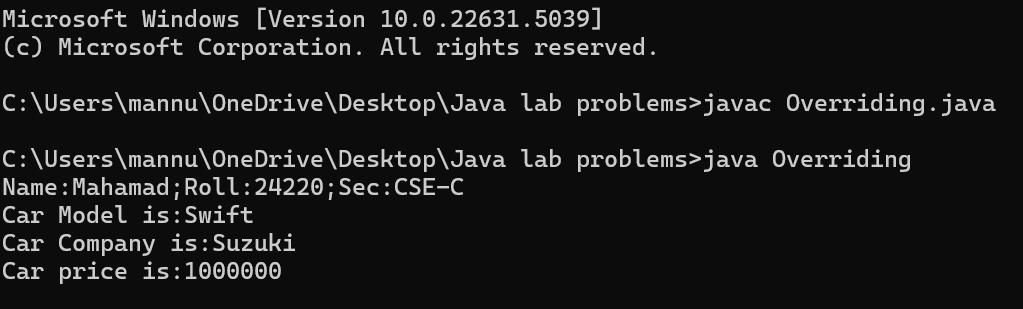
System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

obj1.displayinfo();

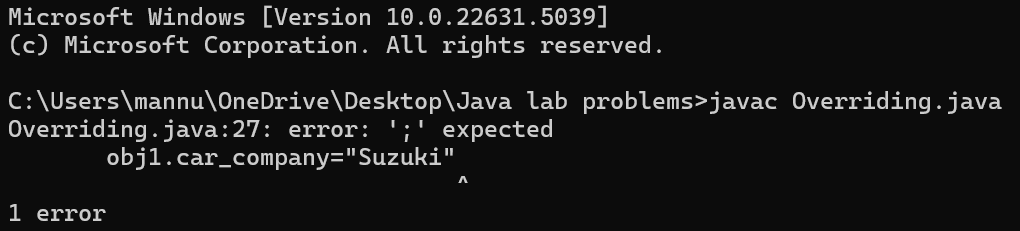
}

}

**OUTPUT:**

****

**NEGATIVE CASE:**

**:**

**Error Table:**

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

**Important Points:**

**Method Overriding:**

Method Overriding allows a subclass to provide a specific implementation of a

Method that is already defined in its superclass , enabling runtime Polymorphism.

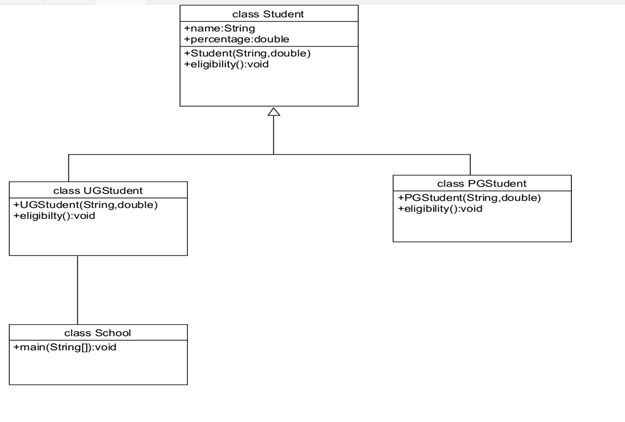
**2.A college is developing an automated admission system that verifies Student eligibility for UG and PG programs. Each program has different Eligibility criteria based on the student’s percentage in their previous**

**Qualification:**

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

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

**CLASS DIAGRAM:**



**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("Mahamad",90.0);

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

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

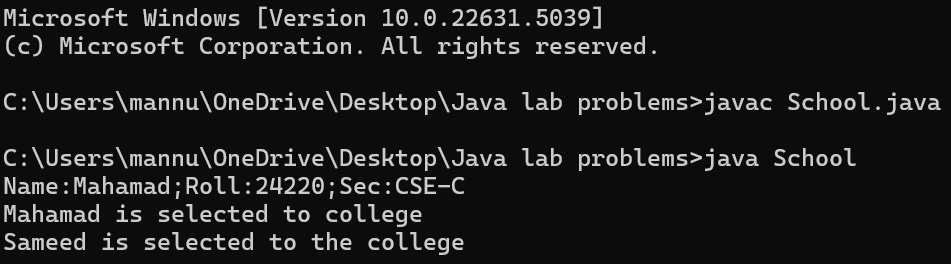
obj1.eligibility();

obj2.eligibility();

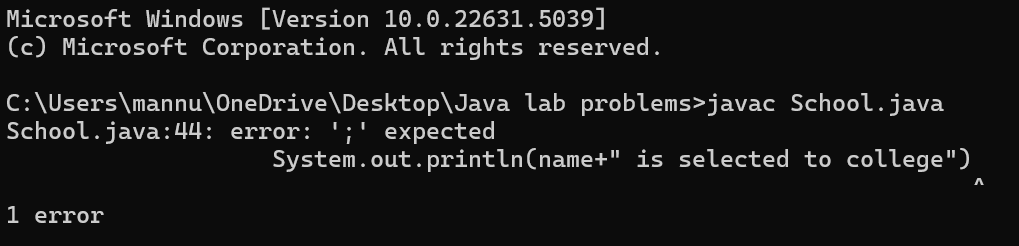
}

}

**OUTPUT:**



**NEGATIVE CASE:**



**Error Table:**

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

**Important points:**

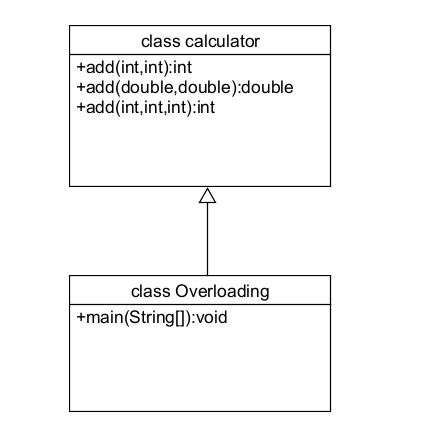
**Super() keyword:**

We use super class method to call a superclass method from within a subclass.

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

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

**CLASS DIAGRAM:**



**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[]){

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

calculator c = new calculator();

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

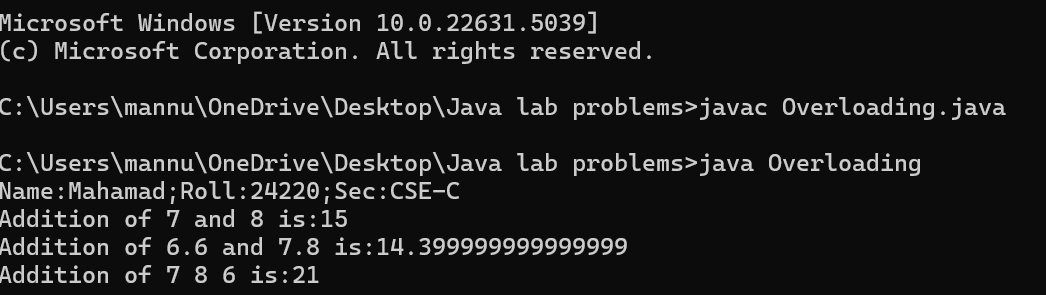
System.out.println("Addition of 2.2 and 4.4 is:"+c.add(2.2,4.4));

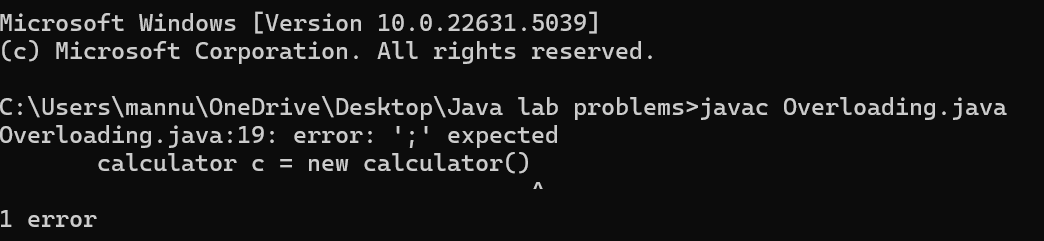
System.out.println("Addition of 2 4 6 is:"+c.add(2,4,6));

}

}

**OUTPUT:**



**NEGATIVE CASE:** 

**Error Table:**

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

**Important Points:**

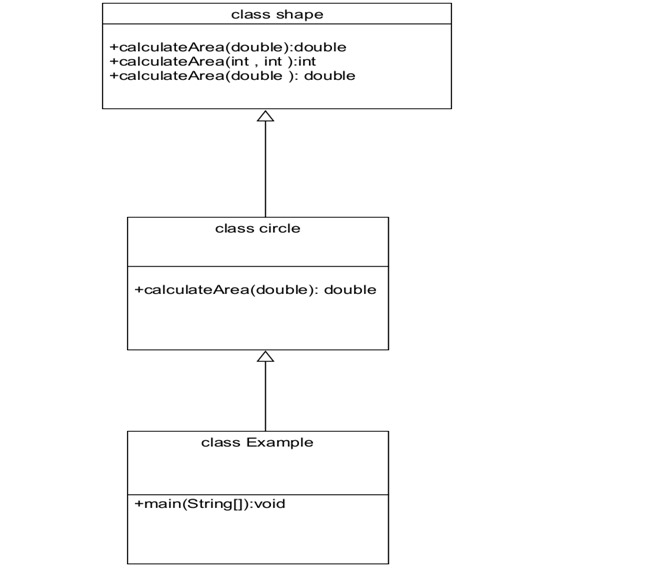
**Method Overloading:**

Method overloading allows defining multiple methods within the same classthat share the same name but have different parameter lists.

The datatypes in the parameters of method may be of same type or different.

**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.**

**CLASS DIAGRAM:**



**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[])

{

System.out.println("Name:Mahamad;Roll:24220;Sec:CSE-C");

circle obj1 = new circle();

shape obj2 = new shape();

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

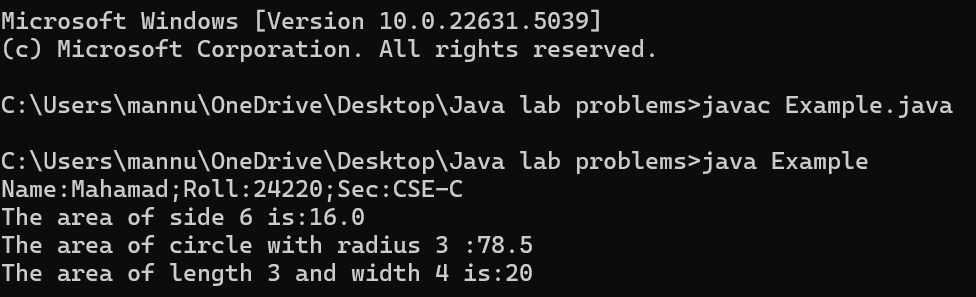
System.out.println("The area of circle with radius 3 :"+obj1.calculateArea(5.0));

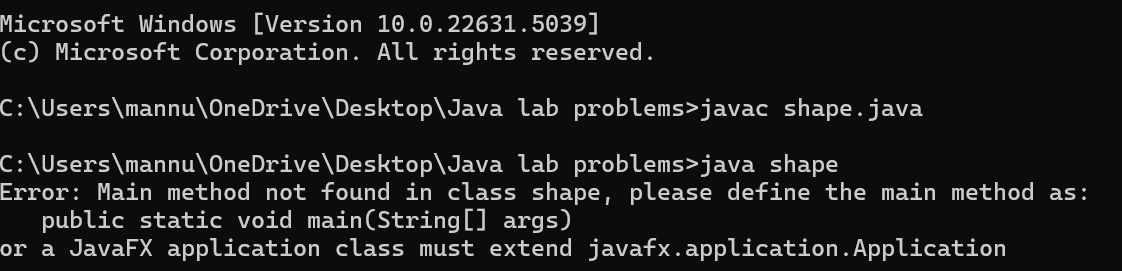
System.out.println("The area of length 3 and width 4 is:"+obj2.calculateArea(5,4));

}

}

**OUTPUT:**

****

**NEGATIVE CASE:** ****

**Error Table:**

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

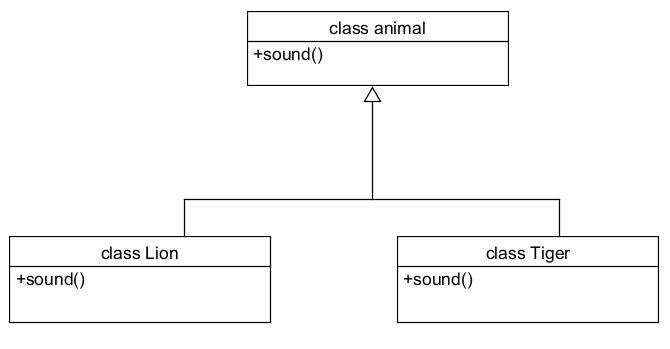
**Important points:**

The return keyword is used to return the value during the method call with the same data type Expected.

**WEEK-7**

**1.Aim: Write a java program to create an abstract class Animal with an abstract Method called sound. Create subclasses lion and tiger that extends the animal Class and implements the sound() method to make a specific sound for each animal.**

**Class Diagram:**



**Code:**  
abstract class animal

{

abstract void sound();

}

class lion extends animal

{

public void sound()

{

System.out.println("Lion Roaring");

}

}

class tiger extends animal

{

public void sound()

{

System.out.println("Tiger Roaring");

}

}

class animalsound

{

public static void main(String[] args)

{

System.out.println("Name:M.Mahamad;Roll:24220");

lion L = new lion();

tiger T = new tiger();

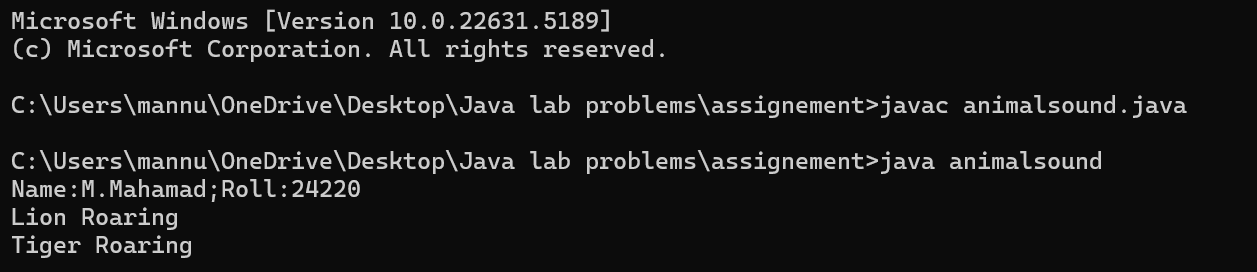
L.sound();

T.sound();

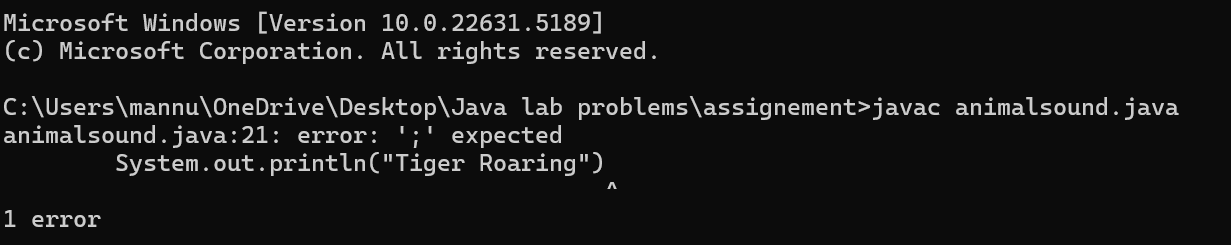
}

}

**Output:**

****

**Negative case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Cause** | **Rectification** |
| 1 | Static reference to non-static method | Calling non-static method from static context | Create object correctly |
| 2 | Instantiation of abstract class(OOP error) | Trying to create an object of an abstract class | Use subclass or interface implementation |
| 3 | Syntax error | Missing parenthesis | Adding parenthesis |

**Important points:**

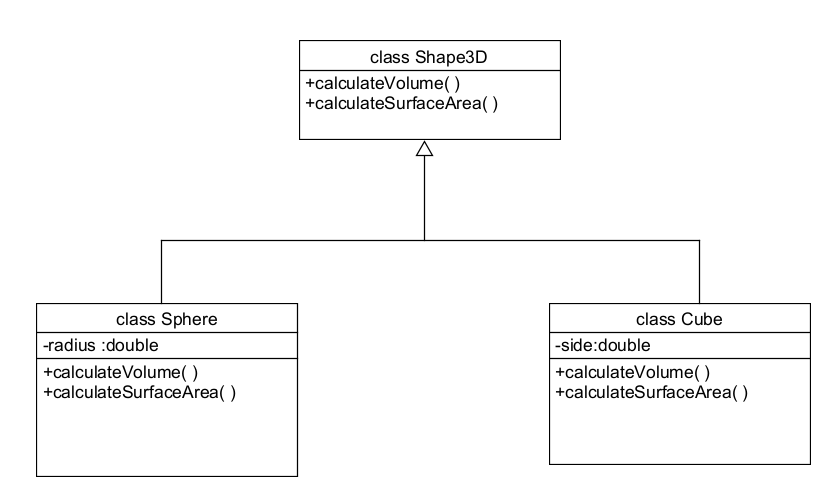
1.The abstract class animal defines the sound() method that must be implemented by subclasses.

2.Lion and Tiger extend animal and provide their own versions of the sound() method.

3.The main() method demonstrates runtime polymorphism by calling sound() on each object.

**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.**

**Class Diagram:**



**Code:**

abstract class Shape3D {

abstract double calculateVolume();

abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

double radius;

Sphere(double radius) {

this.radius = radius;

}

@Override

double calculateVolume() {

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

}

@Override

double calculateSurfaceArea() {

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

}

}

class Cube extends Shape3D {

double side;

Cube(double side) {

this.side = side;

}

@Override

double calculateVolume() {

return Math.pow(side, 3);

}

@Override

double calculateSurfaceArea() {

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

}

}

class ShapeTest {

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

Shape3D sphere = new Sphere(5);

Shape3D cube = new Cube(4);

System.out.println("Name:M.Mahamad;Roll:24220");

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

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

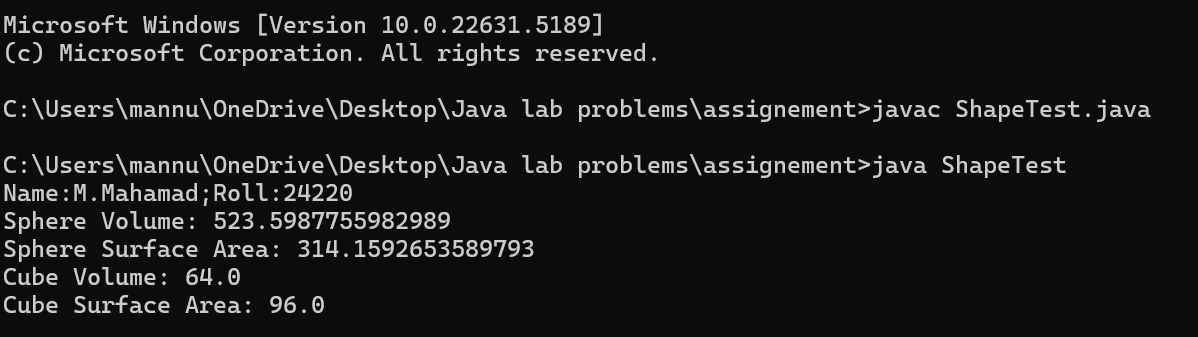
System.out.println("Cube Volume: " + cube.calculateVolume());

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

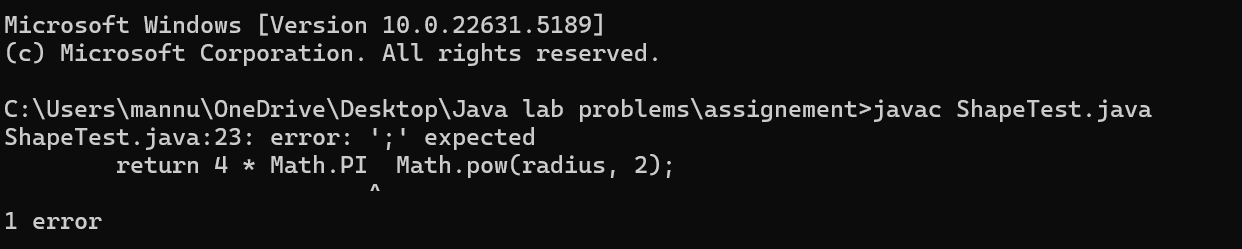
}

}

**Output:**

****

**Negative case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO | Error Type | Cause | Rectification |
| 1 | Return type missing | Method doesn’t specify return type | Adding the correct return type |
| 2 | Instantiation of abstract class | Trying to create an object of abstract class without any subclass or interface | Use subclass or interface implementation |
| 3 | Static reference to Non-static Method | Calling non-static method from static context | Create object or make method static |

**Important points:**

1.Shape3D is an abstract class with abstract methods to calculate volume and surface area.

2.Sphere and Cube extend Shape3D and provide specific implementations for volume and surface area.

3.The main method creates objects and prints the calculated values — demonstrating polymorphism and abstraction**.**

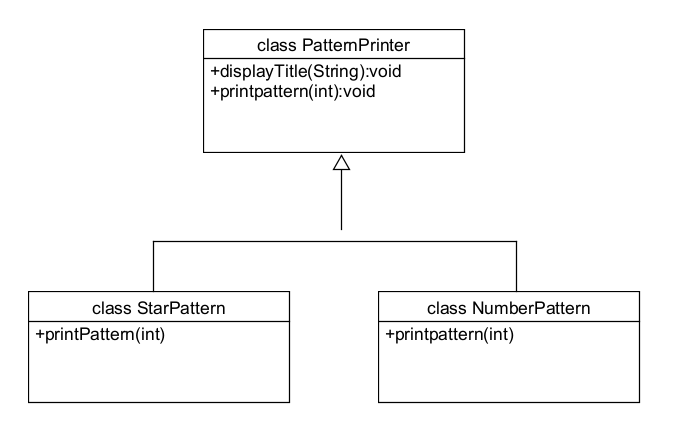
**3.Aim: Write a java program using an abstract class to define a method for pattern printing. Create an abstract class named PatternPrinter with an Abstract method printpattern(int n) and a concrete method to display the pattern title.Implement two subclasses .**

**1.star pattern- prints a right angled triangle of stars(\*)**

**2.Number pattern- prints a right angled triangle of increasing numbers.**

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

**Class Diagram:**



**Code:**

abstract class PatternPrinter

{

abstract void printpattern(int n);

public void display(String title)

{

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

}

}

class starpattern extends PatternPrinter

{

void printpattern(int n)

{

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

{

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

{

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

}

System.out.println();

}

}

}

class numberpattern extends PatternPrinter

{

void printpattern(int n)

{

int num=1;

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

{

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

{

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

num++;

}

System.out.println();

}

}

}

class Main2

{

public static void main(String args[])

{

System.out.println("Name:M.Mahamad;Roll:24220");

System.out.println("---------");

int rows=5;

PatternPrinter star = new starpattern();

star.display("Number pattern");

star.printpattern(rows);

PatternPrinter number = new numberpattern();

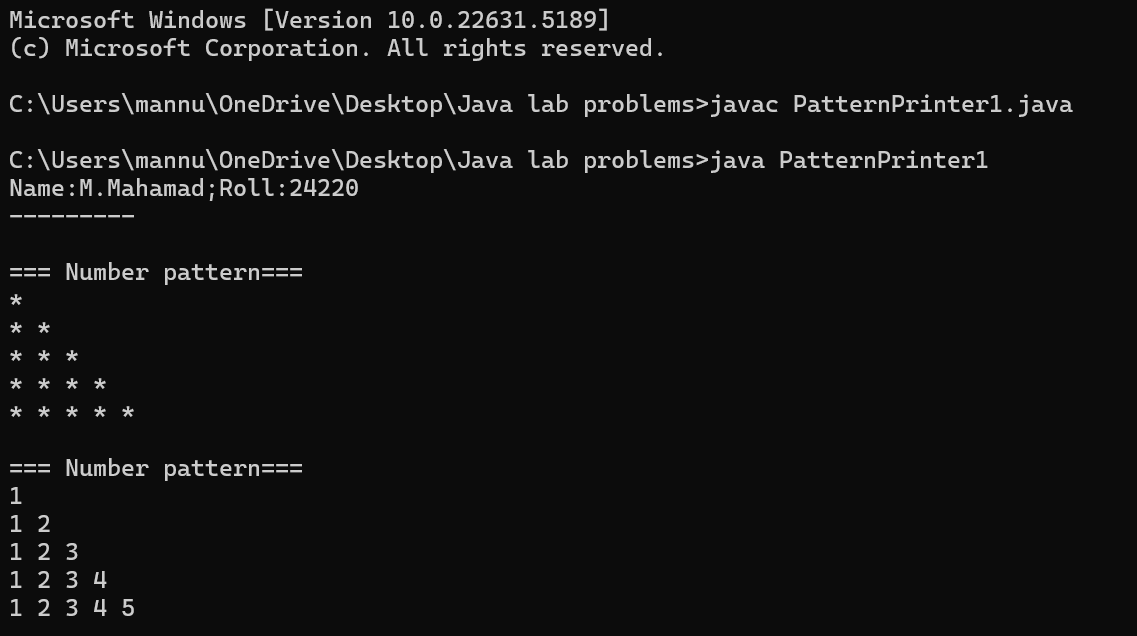
number.display("Number pattern");

number.printpattern(rows);

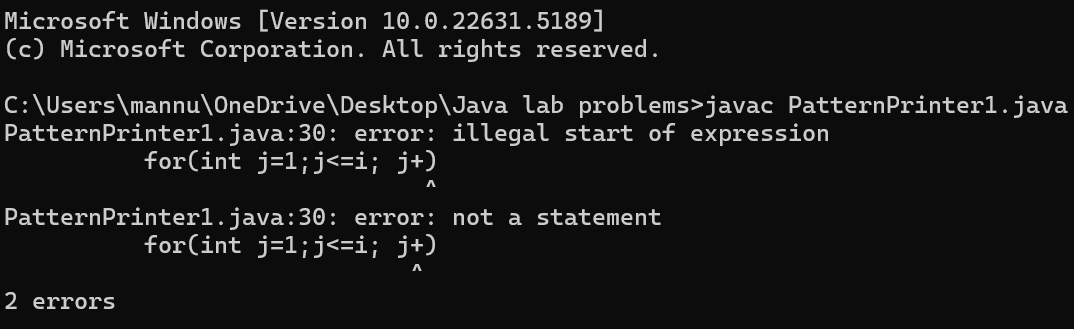
}

}

**Output:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| 1 | Logical error | Incorrect use of operator | Using correct operator |
| 2 | Run time error | Incorrect path | Selected correct path |
| 3 | Syntax error | Missing semicolon | Semicolon added |

**Important points:**

1.Abstraction is used to define a common method printpattern() in the abstract class PatternPrinter, allowing different patterns to be printed

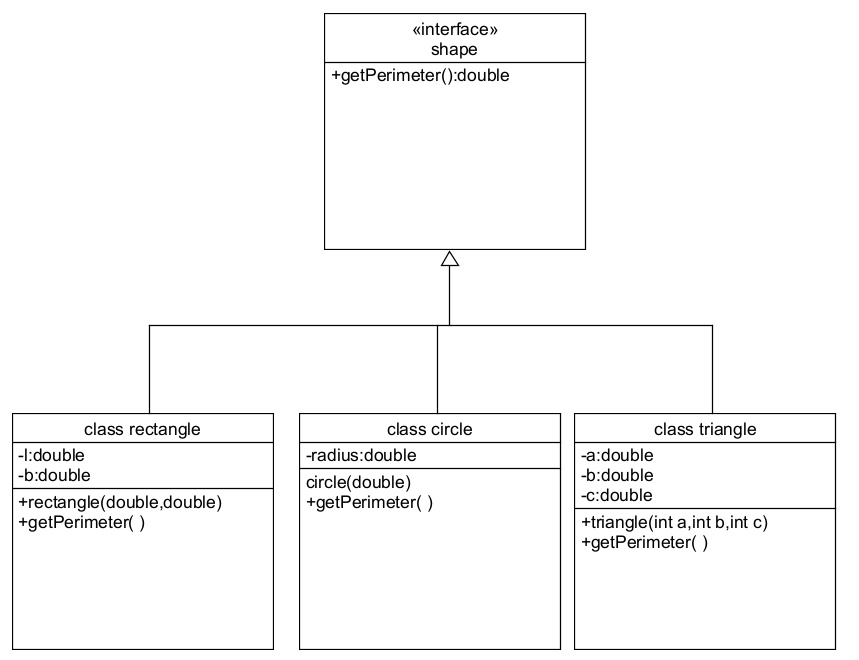
2.Through subclasses.The starpattern and numberpattern classes implement their own version of printpattern(), demonstrating polymorphism.

3.The display() method in the abstract class allows for a consistent header before printing any pattern, making the output structured and easy to manage.

**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.**

**Class diagram:**



**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;

}

public double getPerimeter() {

return 2 \* (length + width);

}

}

class Circle implements Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

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;

}

public double getPerimeter() {

return side1 + side2 + side3;

}

}

public class perimeter {

public static void main(String[] args) {

Shape rectangle = new Rectangle(5, 3);

Shape circle = new Circle(4);

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

System.out.println("Name:M.Mahamad;Roll:24220");

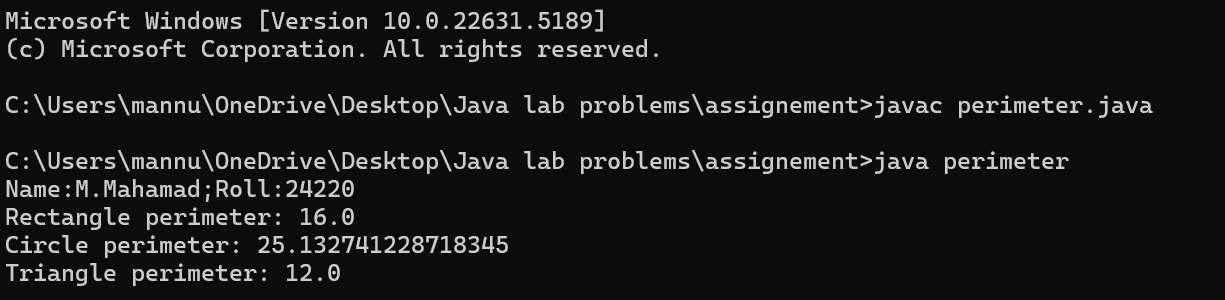
System.out.println("Rectangle perimeter: " + rectangle.getPerimeter());

System.out.println("Circle perimeter: " + circle.getPerimeter());

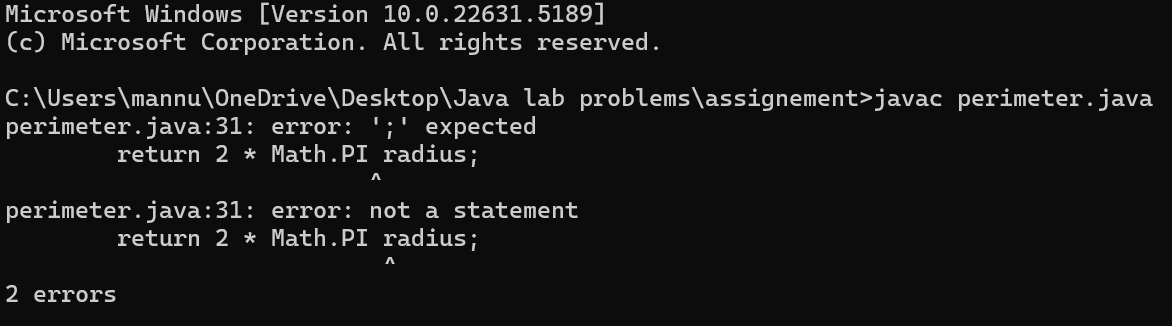
System.out.println("Triangle perimeter: " + triangle.getPerimeter());

}

}

**Output:  
**

**Negative Case:**

****

**Error Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| 1 | Compilation error | Return type missing | Add correct return type |
| 2 | Access modifier issues(OOP error) | Trying to access private members | Change the modifier or add getter/setter |
| 3 | Syntax Error | Mismatched brackets | Close properly all brackets |

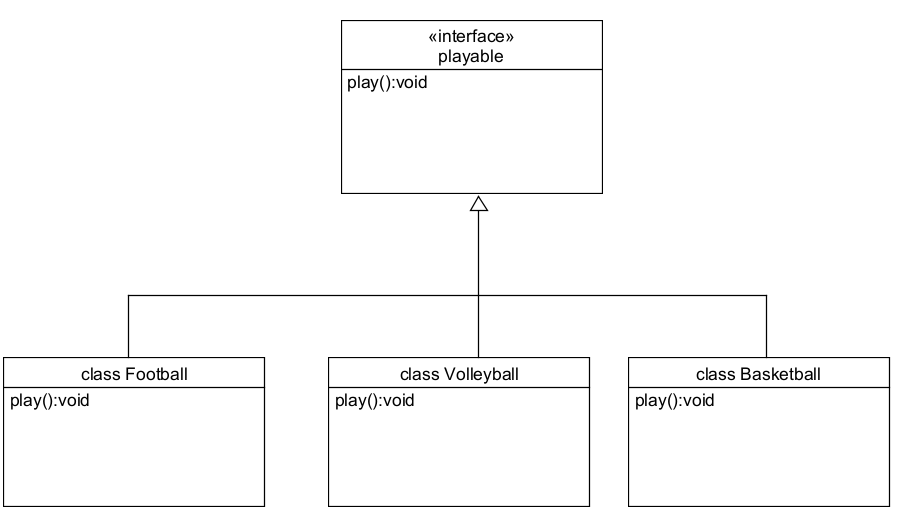
**Important points:**

**Flexibility and Extensibility:** This design makes it easy to add more shapes (like Square, Ellipse) in the future by simply implementing the Shape interface without modifying existing code.

**Interface Implementation:** The Shape interface is implemented by three classes: Rectangle, Triangle, and Circle, each defining the getPerimeter() method for their respective shapes.

**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.**

**Class diagram:**



**Code:**

interface playable

{

abstract void play();

}

class football implements playable

{

public void play()

{

System.out.println("Football");

}

}

class volleyball implements playable

{

public void play()

{

System.out.println("VolleyBall");

}

}

class Basketball implements playable

{

public void play()

{

System.out.println("BasketBall");

}

}

class sports

{

public static void main(String args[])

{

System.out.println("Name:M.Mahamad;Roll:24220");

playable obj = new football();

playable obj2 = new volleyball();

playable obj3 = new Basketball();

obj.play();

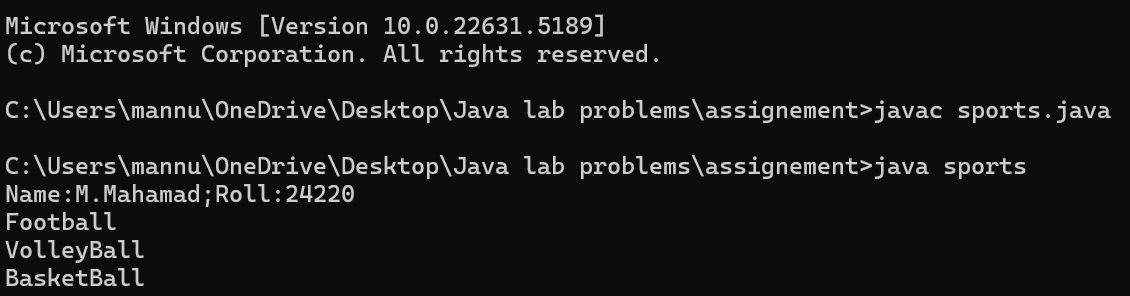
obj2.play();

obj3.play();

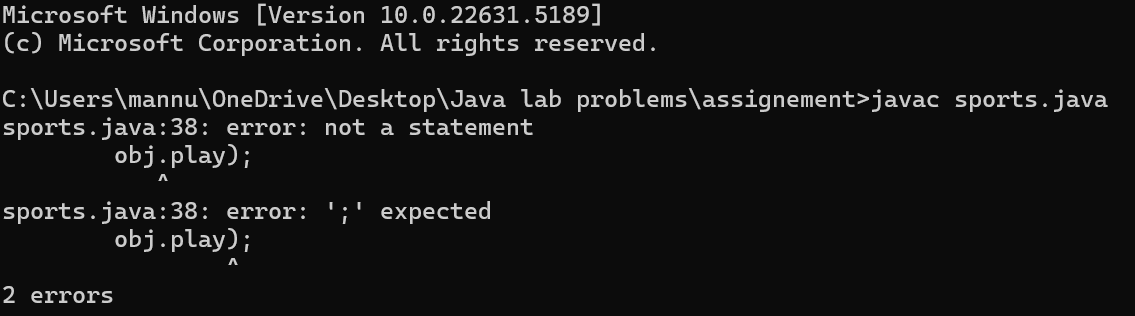
}

}

**Output:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Cause** | **Rectification** |
| 1 | File not found exception | Wrong selection of path | Select path correctly |
| 2 | Instantiation of abstract class | Trying to create an object of abstract class without any subclass or interface | Use subclass or interface implementation |
| 3 | Static reference to Non-static Method | Calling non-static method from static context | Create object or make method static |

**Important points:**

**Interface Implementation:** The playable interface defines the play() method, which is implemented by football, volleyball, and basketball classes, each representing a different sport.

**Polymorphism in Action:**The objects of football, volleyball, and basketball are all treated as playable types allowing the play() method to be called polymorphically.