**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



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

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: Mohan Sai**

**Roll No: 24219**

**Verified By :**

# INDEX

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S no** | **Title** | **Page no** | **Date** | **Signature** |
|  | **WEEK -1** |  |  |  |
| **1** | **Explain the process of download & Installation of JDK** | **4** |  |  |
| **2** | **Write a program to print Student’s name, Roll no, Section** | **7** |  |  |
|  | **WEEK-2** |  |  |  |
| **1** | **Write a simple java program to calculate factorial of a number** | **9** |  |  |
| **2** | **Write a simple java program to find the simple interest by taking iinputs from the user** | **10** |  |  |
| **3** | **Write a program to calculate the Fibonacci sequence and take the input from the user** | **13** |  |  |
| **4** | **Write a program to find the area of triangle using hereon’s formula**  **.** | **15** |  |  |
| **5** | **Write a program to convert temperature from celssius to fahrenheit** | **17** |  |  |
|  | **WEEK-3** |  |  |  |
| **1** | **Write a java program with following instructions** | **19** |  |  |
| **2** | **Write a java program with following instructions** | **26** |  |  |
|  | **WEEK-4** |  |  |  |
| **1** | **Write a java program with following instructions** | **34** |  |  |
| **2** | **Write a java program with following instructions** | **38** |  |  |
|  | **WEEK-5** |  |  |  |
| **1** | **Write a java program with following instructions** |  |  |  |
| **2** | **Write a java program with following instructions** |  |  |  |
|  | **WEEK -6** |  |  |  |
| **1** | **Write a java program with following instructions** |  |  |  |
| **2** | **Write a java program with following instructions** |  |  |  |
| **3** | **Write a java program with following instructions** |  |  |  |
| **4** | **Write a java program with following instructions** |  |  |  |

# WEEK-1

1. **AIM:**

**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

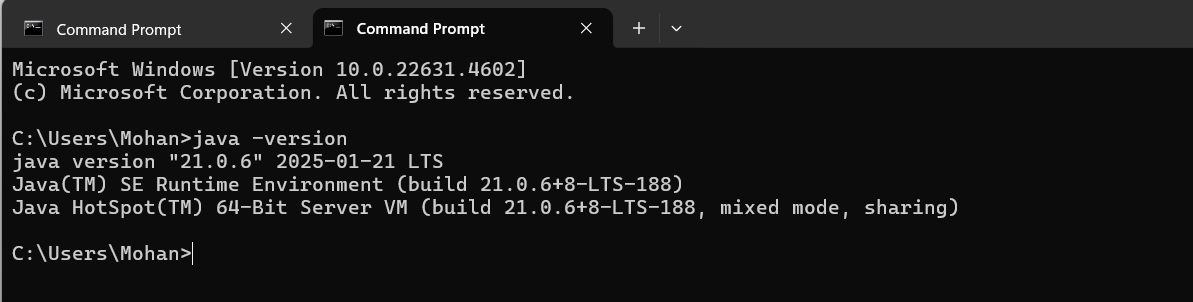
C:\Program Files\Java\jdk-21\bin)

* 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: -**

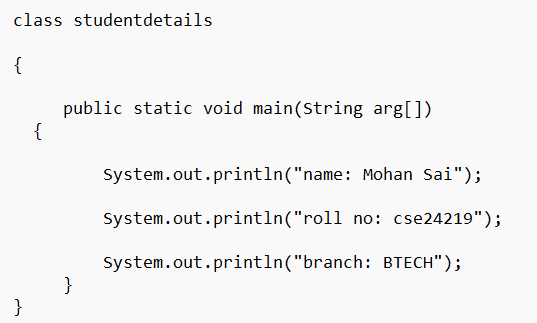
* 1. **Open Command Prompt:**
* Presswin+R, typecmd, and press Enter.
  1. **Check Version:**
* Type java -version and press Enter.
* Type javac --version and press Enter.



**2)AIM:**

**Simple Java Program for printing Name, Class, Roll No, of a Student**

**CODE: -**



**Output:**



**ERROR TABLE:**

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

**WEEK-2**

**1)**

**AIM ;**

**Simple Java Program for finding simple interest by taking input from**

**User**

**Code:**



**Output:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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** | **Formula rectified** |

**2)**

**AIM:**

**Write a simple program to calculate factorial of a number and read the**

**input from user.**

**code:**



**Output:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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** |

**3)**

**AIM:**

**Write a program to to calculate the fibonacii sequence and take the**

**input 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;

}

}

}

**Output:**

**\***

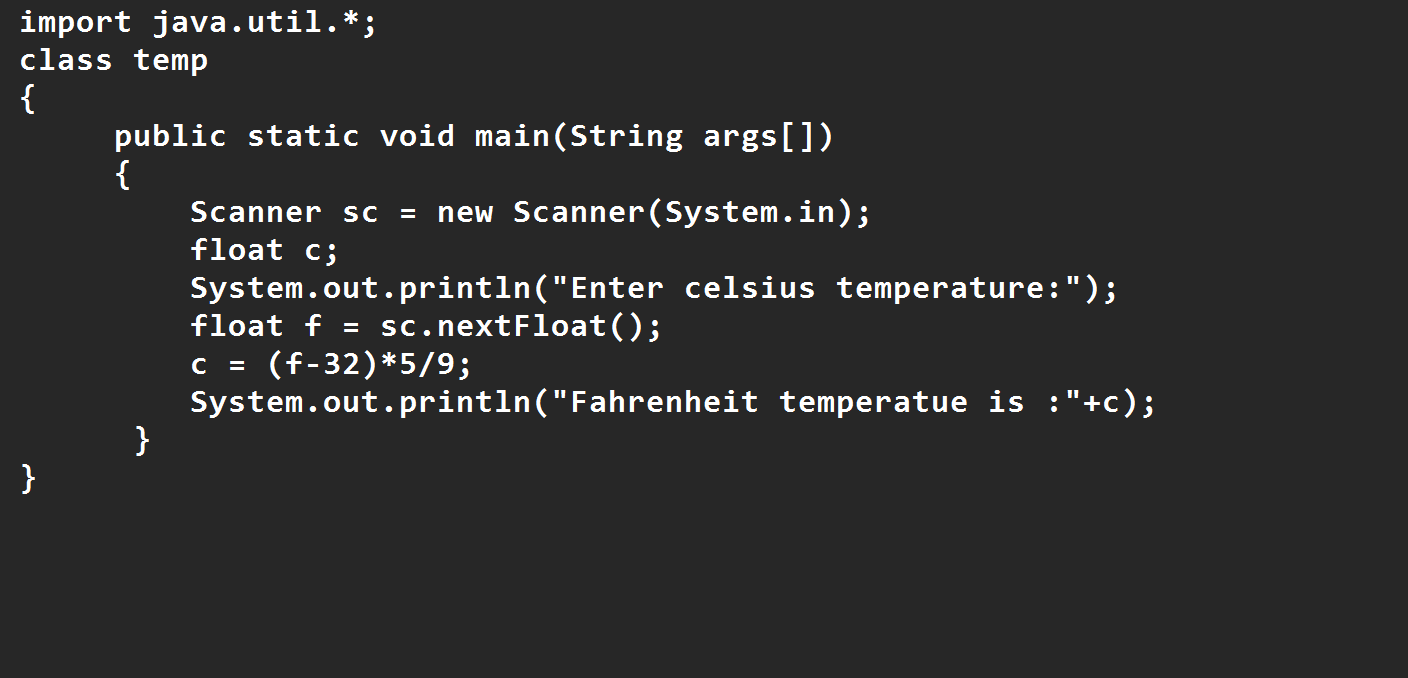
**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **Incorrect formula** | **Formula rectified** |
| **2** | **Run-time error** | **Incorrect path** | **Added correct path** |
|  |  |  |  |

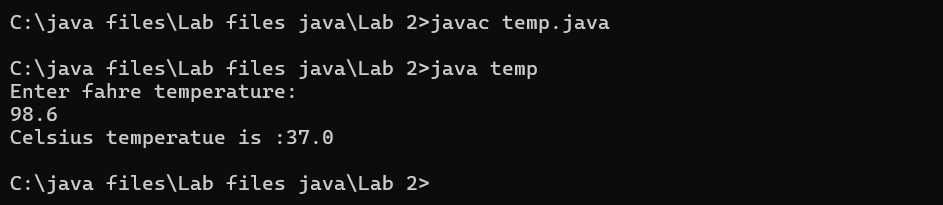
**4)AIM;**

**Write a java program to convert temperature from Fahrenheit to celsius**

**Code:**



**Output:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Syntax error** | **Missing ”** | **“ is added** |
| **2** | **Missing import error** | **Util package missing** | **Util package added** |

**5)AIM:**

**Write a java program to convert temperature from Celsius to Fahrenheit**

**Code:**



**Output:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Runtime error** | **Incorrect path selection** | **Correct path added** |
| **2** | **Logical error** | **Incorrect logic** | **Correct logic** |
|  |  |  |  |

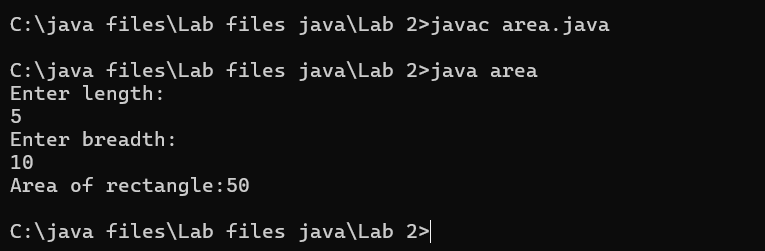
**6)AIM:**

**Write a simple program to find the area of rectangle:**

**Code:**



**Output:**



**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Semi colon missing** | **Semi colon added** |
| **2** | **Missing import error** | **Import package missing** | **Import package added** |

**7)AIM:**

**Write a program to find the area of triangle by using heron’s formula**

**take the input from the user**

**Code:**



**Output:**

****

**ERROR TABLE:**

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

**WEEK 3**

**1.Aim:**

**To create java program with following instructions**

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

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

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

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

**Class Diagram:**

|  |
| --- |
| Car |
| + car\_color: String  + car\_brand: String  + fuel\_type: String  + mileage: int |
| + Car(): void  + start(): void  + service(): void  + stop(): void |

**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("\n mohan\n\n");

car car1 = new car();

car1.Car\_color = "Blue";

car1.Car\_brand = "BMW";

car1.fuel\_type = "Deisel";

car1.mileage = 10;

car1.start();

car car2 = new car();

car2.Car\_color = "Red";

car2.Car\_brand = "Tesla";

car2.fuel\_type = "EV";

car2.mileage = 300;

car2.stop();

car car3 = new car();

car3.Car\_color = "Yellow";

car3.Car\_brand = "MAHINDRA";

car3.fuel\_type = "Petrol";

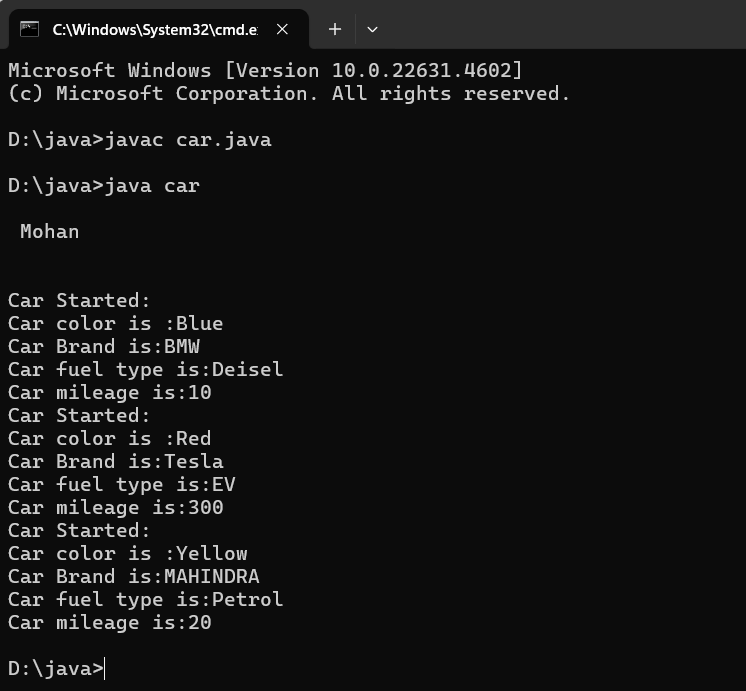
car3.mileage = 20;

car3.service();

}

}

**Output:**

****

**Concepts to be known:**

1. public String car\_color; - Used to declare a variable named car\_color, with data type as String with public accessibility.
2. Car(String car\_color,String car\_brand,String fuel\_type,int mileage){ } – It is a constructor (method with name same as class), which requires parameters such as car\_color (String data-type) and so on.
3. this.car\_color=car\_color; - “this” is a default method, which is used to point to the instance variables.
4. public void start(){} – used to declare a method, which will return nothing(void) in public accessibility.
5. Car car1=new Car("Red","Maruti","Diesel",20); - used to create a object in class Car, with object name as car1.

car1.start(); - Calling a method, under object car1.

**2.)**

**AIM:**

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

**Class diagram :**

|  |
| --- |
| **BankAccount** |
| **- balance: double** |
| **+ BankAccount(initialBalance: double)**  **+ deposit(amount: double): void**  **+ getBalance(): double** |

|  |
| --- |
| **Main1** |
| **+ main(args: String[]): void** |

**Code:**

class BankAccount

{

private double balance;

public BankAccount(double initialBalance)

{

if(initialBalance > 0)

{

this.balance = initialBalance;

}

else

{

this.balance = 0;

}

}

public void deposit(double amount)

{

if(amount>0)

{

balance = balance+amount;

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

}

else

{

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

}

}

public double getBalance()

{

return balance;

}

}

public class Main1

{

public static void main(String args[])

{

BankAccount account = new BankAccount(1000);

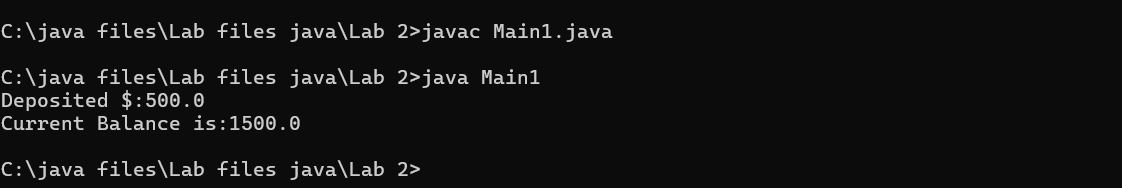
account.deposit(500);

System.out.println("Current Balance is:"+account.getBalance());

}

}

**Output:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  cust1.withdraw(3050) | Add a “;”    cust1.withdraw(3050); |
| 2. | error: cannot find  symbol  thisCurrBal=CurrBal; | Add a “.”    this.CurrBal=CurrBal; |

**Concepts to be known:**

1. private String name; - Used to declare a variable named name,

with data type as String with private accessibility.

1. BankAccount(String name,int Accno,int CurrBal){ } – It is a

constructor (method with name same as class), which requires

parameters such as name (String data-type) and so on.

1. this.CurrBal=CurrBal; - “this” is a default method, which is used to

point to the instance variables.

1. public void withdraw(int WAmt){ } – used to declare a method,

which will return nothing(void) in public accessibility, which requires

a parameter WAmt(integer data type).

1. public int deposit(int DAmt){} - used to declare a method, which will return integer data type in public accessibility, which requires a

parameter DAmt(integer data type).

1. BankAccount cust1=new BankAccount("Ram",5587,20000); - used to create a object in class BankAccount, with object name as cust1.
2. cust1.withdraw(50000); - Calling a method, under object cust1, by passing a parameter.

System.out.println("Your current balance after depositing money is:"+cust1.deposit(25000)); - Deposit method will return the value, which will be directly printed.

**WEEK-4**

**1.AIM:**

**WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOUKD CONTAIN VARIOUS ATTRIBUTES SUCH AS TITLE, AUTHOR, YEAR OF**

**PUBLICATION. IT SHOULD ALSO CONTAIN A CONSTRUCTOR WITH**

**PARAMETERS WHICH INITIALIZES TITLE, AUTHOR, YEAR OF PUBLICATION**

**AND CREATE A METHOD WHICH DISPLAYS THE DETAILS OF 2 BOOKS.**

**CLASS DIAGRAM:**

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

**PROGRAM:**

public class Book {

public String title;

public String author;

public int year;

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

this.title = title;

this.author = author;

this.year = year;

}

public void displayDetails() {

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

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

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

}

public static void main(String[] args) {

Book b1 = new Book("Math", "Ramanujan", 1950);

Book b2 = new Book("Physics", "CV Raman", 1960);

b1.displayDetails();

b2.displayDetails();

}

}

**Output:**



**NEGATIVE CASE:**

A black screen with white text

AI-generated content may be incorrect.

**cc**

**2.AIM:**

**WRITE A JAVA PROGRAM WITH CLASS NAMED “MyClass” WITH A STATIC VARIABLE COUNT OF INT TYPE. INTIALIZE IT TO ZERO AND A CONSTANT**

**VARIABLE “Pi” OF TYPE DOUBLE INITIALIZED TO “3.14” AS ATTRIBUTES OF**

**THAT CLASS. NOW DEFINE A CONSTRUCTOR FOR “MyClass”, THAT**

**INCREMENTS THE COUNT VARIABLE EACH TIME AN OBJECT OF “MyClass”**

**IS CREATED. FINALLY, PRINT THE FINAL VALUES OF ‘COUNT’ AND ‘PI’**

**VARIABLES AND CREATE 3 OBJECTS.**

**CLASS DIAGRAM:**

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

**PROGRAM:**

public class MyClass {

static int count = 0;

static final double pi = 3.14;

MyClass() {

count++;

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

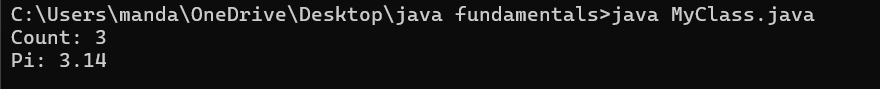
MyClass obj3 = new MyClass();

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

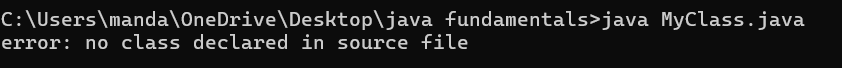
System.out.println("Pi: " +pi);

} }

**OUTPUT:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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**

**1.AIM:**

**Create a calculator using the operations including addition,**

**Subtraction Multiplication and division using multilevel inheritance**

**and display the desired Output.**

**Class diagram:**



**Code:**

class addition

{

public int add(int a, int b)

{

int addition = a+b;

return addition;

}

}

class subtraction extends addition

{

public int sub(int a, int b)

{

int subtraction = a-b;

return subtraction;

}

}

class multiplication extends subtraction

{

public int mult(int a, int b)

{

int multiplication = a\*b;

return multiplication;

}

}

class division extends multiplication

{

public int div(int a,int b)

{

int division = a/b;

return division;

}

}

class calculator

{

public static void main(String args[])

{

division obj = new division();

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

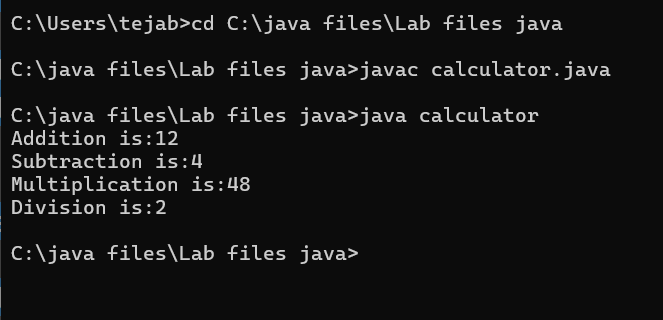
System.out.println ("Subtraction is:"+obj.sub(8,4));

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

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

  }}

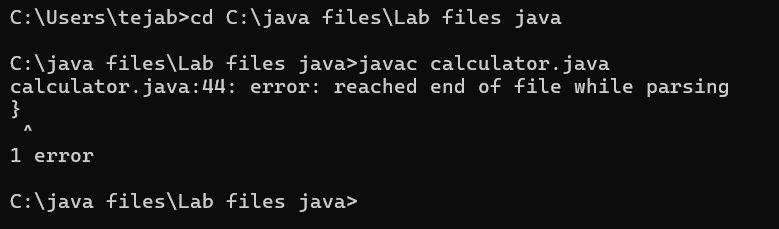
**Output:**



Error Table

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

**Negative Case:**



**Important Points:**

Inheritence:

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

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

Multilevel Inheritence:

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

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

inheritence

extends:

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

**2.**

**Aim**:

**A vehicle rental company wants to develop a system that maintains**

**Information about different types of vehicles available for rent The Company rents out cars, bikes and truck and they need a program to Store details about each vehicle, such as brand and speedCars 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("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);

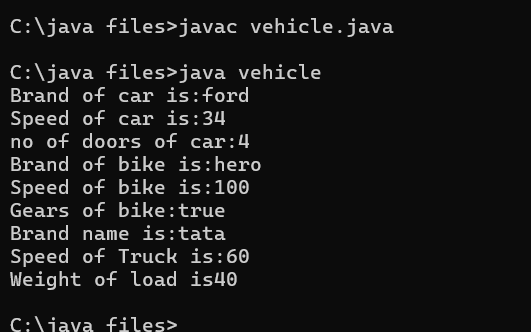
System.out.println("Speed of Truck is:"+speed);

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

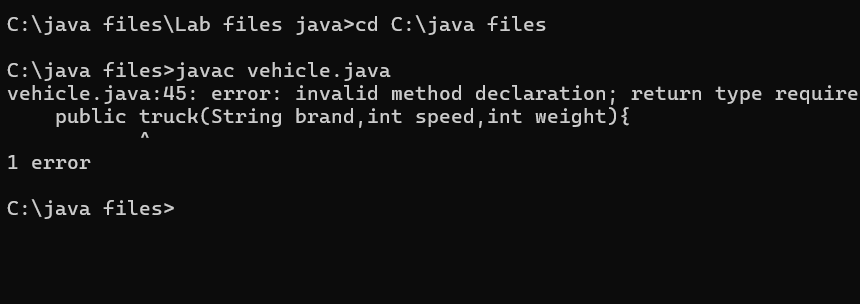
}

}

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

**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 about a car.**

**CLASS DIAGRAM:**

|  |
| --- |
| CLASS VEHICLE |
| ATTRIBUTES :  + COMPANY  + MODEL  +FUEL  + CAPACITY |
| +DISPLAYINFO():void  + VEHICLE (STRING COMPANY,STRING MODEL, STRING FUEL,INT CAPACITY) |

|  |
| --- |
| class car |
| car(STRING COMPANY ,STRING MODEL,string FUEL, INT CAPACITY)  displayinfo() : void |

**CODE:**

class vehicle{

String company;

String model;

String fuel;

int capacity;

void displayInfo(String company,String model,String fuel,int capacity){

System.out.println("The details of vehicle: ");

this.company=company;

this.model=model;

this.fuel=fuel;

this.capacity=capacity;

}}

class car extends vehicle{

void displayInfo(String company,String model,String fuel,int capacity){

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

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

System.out.println("Fuel: "+fuel);

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

}

}

class poly1{

public static void main(String[] args){

System.out.println(“Name: Mohan, roll no: 24219”);

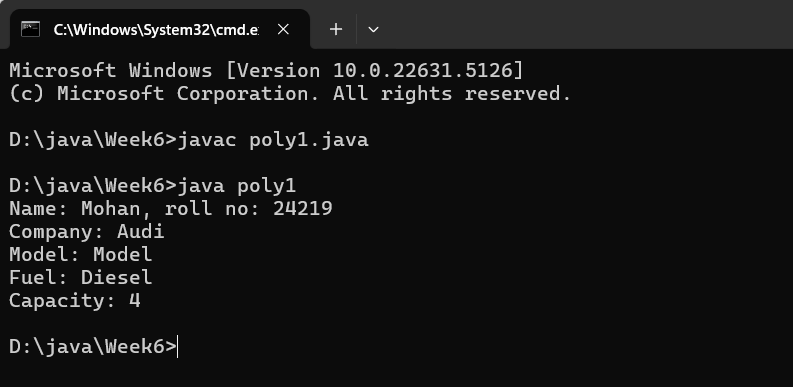
car car1=new car();

car1.displayInfo("Audi","Model","Diesel",4);

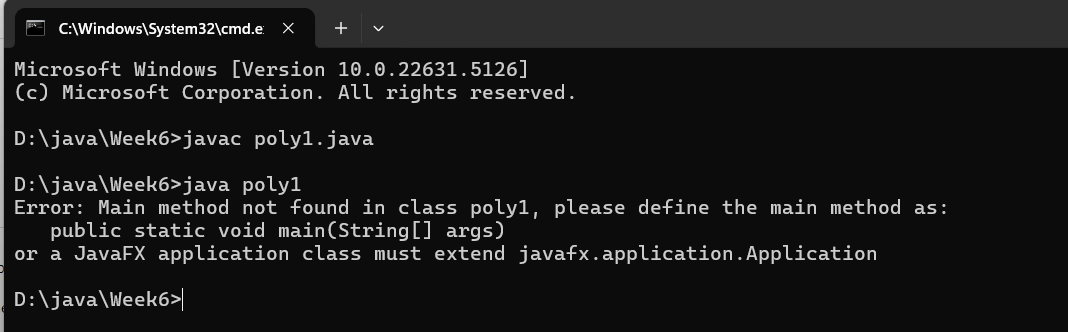
}}

**OUTPUT:**

**Positive case:**



**Negative case:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | [] is missed | [] is added |
| 2 | Logical error | Incorrect logic | Correct logic |

**Important points:**

1. **Inheritance** allows the Car class to reuse the Vehicle class

functionality without repeating code.

2. **Method Overriding** enables the Car class to provide its own implementation of the displayInfo() method.

3. **Polymorphism** makes the code flexible, allowing different classes

(e.g., Car, Bike) to provide customized behavior for displayInfo().

1. **Aim :**

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

**verifies students' eligibility for undergraduate (UG) and**

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

**eligibility criteria based on the students' percentage in their**

**previous qualifications.**

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

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

**Class diagram :**

|  |
| --- |
| University |
| - name: String  - percentile: int |
| + University(String, int)  + office(): void |

|  |
| --- |
| PG |
| + PG(String, int)  + office(): void |

|  |
| --- |
| UG |
| + UG(String, int)  + office(): void |

|  |
| --- |
| admissions |
| + main(String[]): void |

**CODE:**

class College{

    String name;

    int percentage;

    void geteligibility(String name,int percentage){

        this.name=name;

        this.percentage=percentage;

    }

}

class UG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=60){

            System.out.println(name+" is eligible");

        }

        else{

            System.out.println(name+" is not eligible");

        }

    }

}

class PG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=70){

            System.out.println(name+" is eligible");

        }

        else{

            System.out.println(name+" is not eligible");

        }}}

class poly2{

    public static void main(String[] args){

System.out.println(“Name: Mohan, roll no: 24219”);

        UG ug=new UG();

        ug.geteligibility("Person-1",40);

        PG pg=new PG();

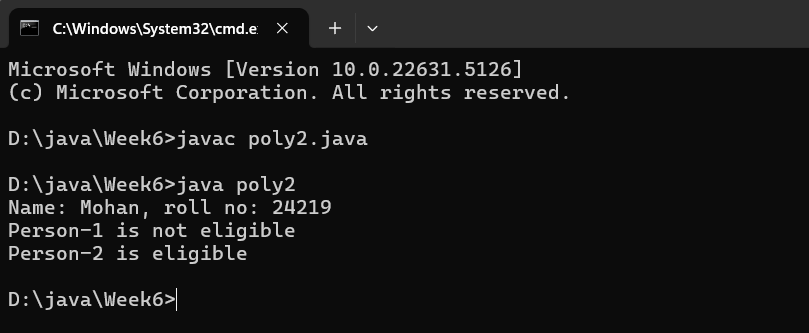
        pg.geteligibility("Person-2",80);

    }

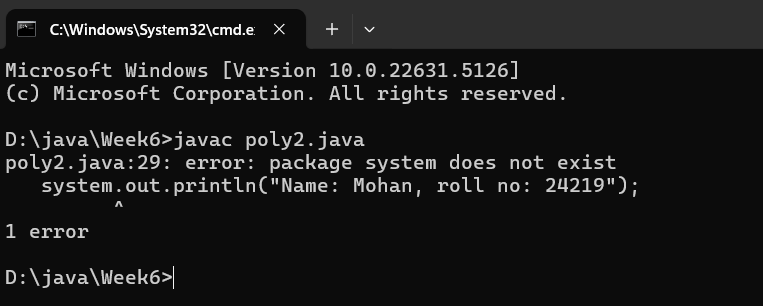
}

**OUPUT :**

**Positive case:**

****

**Negative case:**

****

**ERROR :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | syntax error | Capital ‘S’ forgot at print statement | Is added |
| 2 | Logical error | Incorrect logic | Correct logic |

IMPORTANT POINTS:

1.**Inheritance** allows PG and UG classes to reuse the University class’s attributes and constructor.

2. **Method Overriding** customizes the office() method in PG and UG classes to implement specific admission criteria.

**3.Aim :**

**Create a Calculator class with overloaded methods to perform addition:**

**(i) Add two integers.**

**(ii) Add two doubles.**

**(iii) Add three integers**.

**CLASS DIAGRAM:**

|  |
| --- |
| CLASS CALCULATORMAN |
| +add(int, int): int  +add(double, double): double + +add(int, int, int): int |
|  |

|  |
| --- |
| HOSTEL |
| +main(String[]): void |

**Program :**

**CODE:**

class Calc{

    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 poly3{

    public static void main(String[] args){

System.out.println(“Name: Mohan, roll no: 24219”);

        CalcC1=new Calc();

        System.out.println("Sum of 2 and 5 is: "+C1.add(2,5));

        System.out.println("Sum of 3.2 and 6.5 is: "+C1.add(3.2,6.5));

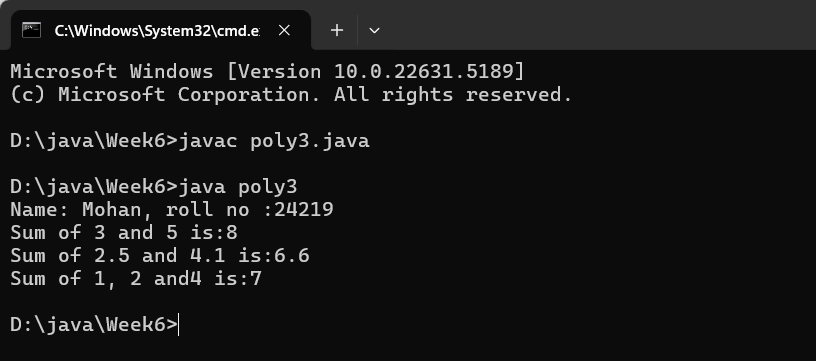
        System.out.println("Sum of 2,5 and 3 is: "+C1.add(2,5,3));

    }

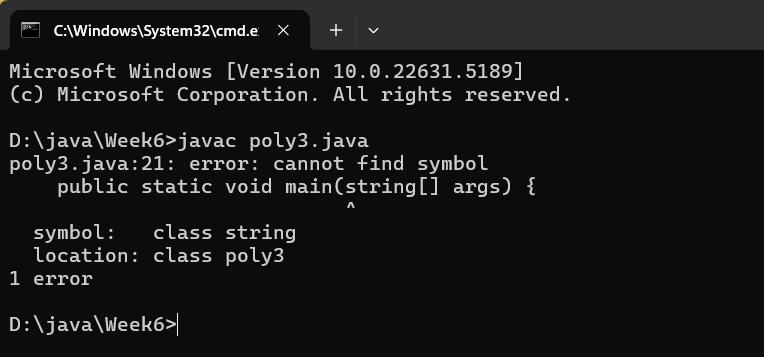
}

**OUTPUT :**

**Positive case:**

****

**Negative case:**

****

**ERROR :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | syntax error | String forgot in main function | String is added |
| 2 | Logical error | Incorrect logic | Correct logic |

**IMPORTANT POINTS :**

1. This program demonstrates **method overloading**, where multiple

add() methods have the same name but different parameter types or counts.

1. Each method performs addition and prints the type of addition

being done (int, double, or three integers).

1. In the main() method, the correct overloaded method is called

based on the arguments passed, showcasing polymorphism.

**4)Aim :**

**Create a Shape class with a method calculateArea() that is overloaded for different shapes (e.g., square, rectangle). Then, create a subclass Circle that overrides the calculateArea() method for a circle.**

**Claas diagram :**

|  |
| --- |
| **Class shapes** |
| **+ calculateArea(int, int): void**  **+ calculateArea(double, double): void**  **+ calculateArea(int): void** |

|  |
| --- |
| Class circle |
| + calculateArea(double): void |

|  |
| --- |
| Area |
| + main(String[]): void |

**Program :**

class Shape {

    void calculateArea( int a) {

        System.out.println("The area of Square is :" + (a\*a) );

    }

    void calculateArea(int a , int b) {

        System.out.println("The area of rectangle is :" + (a\*b));

    }

}

class circle extends Shape {

    void calculateArea(double a){

        System.out.println("The area of circle is :" + (3.14\*a\*a));

  } }

class main {

    public static void main(String[] args) {

        Shape s = new Shape();

        circle c = new circle();

        s.calculateArea(4);

        System.out.println("    ");

        s.calculateArea(4, 5);

        System.out.println("    ");

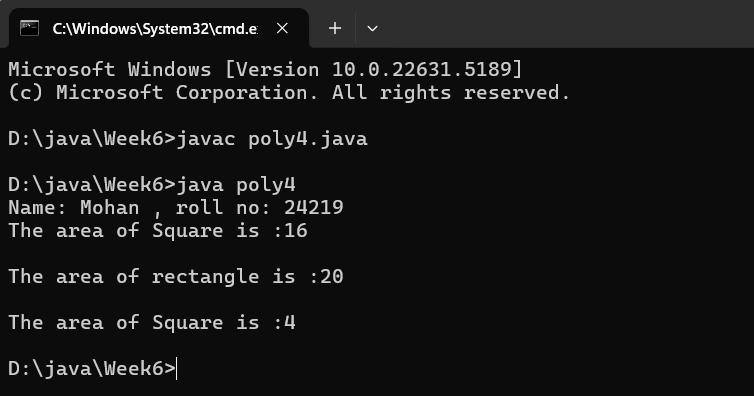
        c.calculateArea(2);

    }

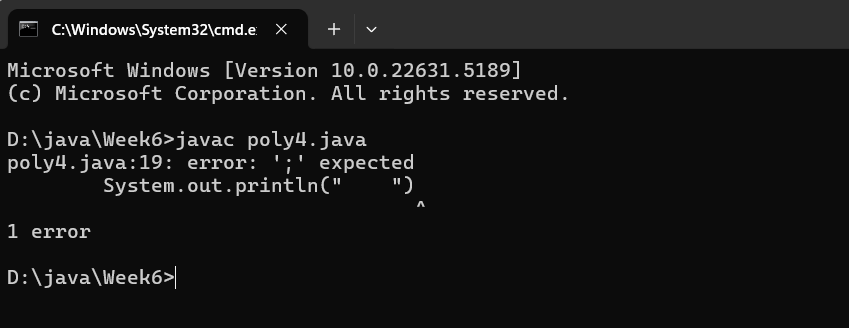
}

**OUTPUT :**

**Positive case:**

****

**Negative case:**

****

**ERRORS :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | ; is missed | ; is added |
| 2 | Logical error | Incorrect logic | Correct logic |

IMPORTANT POINTS :

1. **Method Overloading** allows the Shape class to calculate areas

for different shapes (rectangle, triangle, square) using multiple calculateArea() methods with different parameters.

1. **Method Overriding** in the Circle class provides a custom implementation of calculateArea() for calculating the area of a

circle.

1. **Polymorphism** enables objects of different types (e.g., Shape,

Circle) to call the appropriate version of calculateArea() based on the object type and input parameters.

**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 extend the Animal class and implement the sound() method to make a specific sound for each animal.**

**PROGRAM:**

abstract class Animal {

    abstract void sound();

}

class Lion extends Animal {

    void sound() {

        System.out.println("Lion Roar...!");

    }

}

class Tiger extends Animal {

    void sound() {

        System.out.println("Tiger Roar...!");

    }

}

class Sound {

    public static void main(String[] args) {

        Lion l = new Lion();

        Tiger t = new Tiger();

System.out.println(“Name: Mohan, roll no: 24219”);

        System.out.println("    ");

        l.sound();

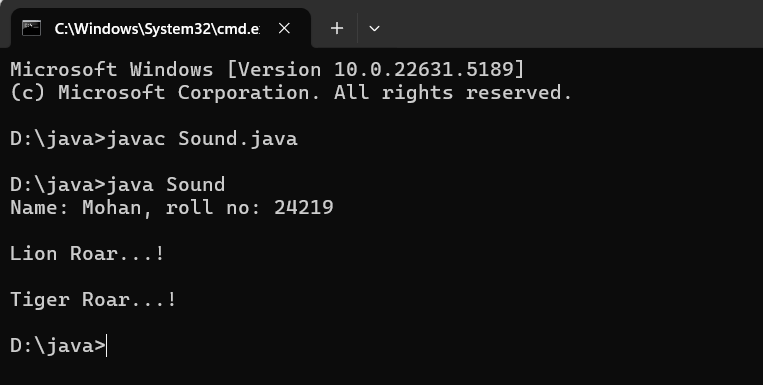
        System.out.println("    ");

        t.sound();

    }

}

**OUTPUT:**

****

**ERRORS:**

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

**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.**

**PROGRAM :**

import java.math.\*;

abstract class Shapes3D {

    abstract void CalculateVolume();

    abstract void CalculateSurfaceArea();

}

class Sphere extends Shapes3D {

    int radius;

    Sphere(int radius) {

        this.radius = radius;

    }

    void CalculateVolume() {

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

    }

    void CalculateSurfaceArea() {

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

radius \* radius);

    }

}

class Cube extends Shapes3D {

    int side;

    Cube(int side) {

        this.side = side;

    }

    void CalculateVolume() {

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

    }

    void CalculateSurfaceArea() {

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

    }

}

public class Shapes18 {

    public static void main(String[] args) {

        Sphere sp = new Sphere(5);

        Cube c = new Cube(4);

        System.out.println(“Name: Mohan, roll no: 24219”);

        System.out.println("    ");

        sp.CalculateSurfaceArea();

        System.out.println("    ");

        sp.CalculateVolume();

        System.out.println("    ");

        c.CalculateSurfaceArea();

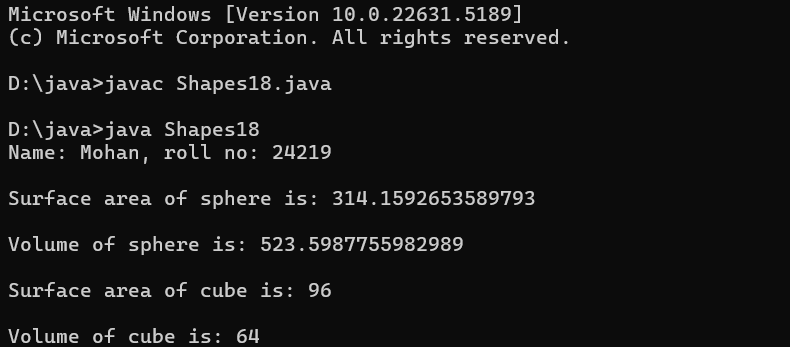
        System.out.println("    ");

        c.CalculateVolume();

    }

}

**OUTPUT:**

****

**ERROR:**

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

**3)Aim :**

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

**Implement two subclasses:**

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

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

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

**Star Pattern Number pattern**

**\* 1**

**\*\* 1 2**

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

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

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

**PROGRAM :**

abstract class PatternPrinter {

    abstract void printPattern(int n);

    void displayTitle(String title) {

        System.out.println(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) {

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

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

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

            }

            System.out.println();

        }

    }

}

 class PatternProgram {

    public static void main(String[] args) {

        StarPattern sp = new StarPattern();

        NumberPattern np = new NumberPattern();

System.out.println(“Name: Mohan, roll no: 24219”);

        System.out.println("    ");

        sp.displayTitle("Star Pattern");

        sp.printPattern(5);

        System.out.println("    ");

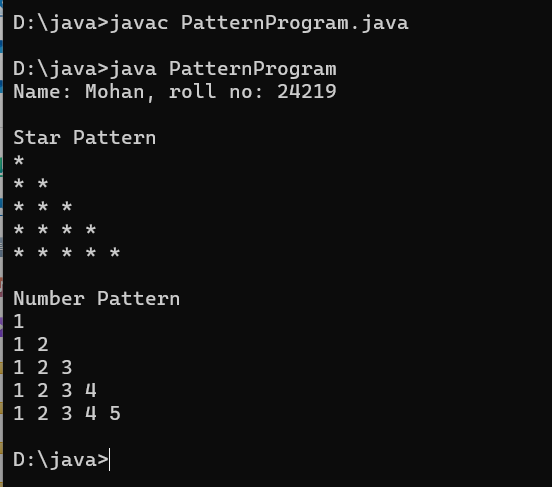
        np.displayTitle("Number Pattern");

        np.printPattern(5);

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **For loop increment condition is missed in subclass method** | **Increment condition is added in subclass method** |
| **2** | **Logical error** | **Incorrect logic in subclass method** | **Corrected logic in subclass method** |

**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.**

**PROGRAM :**

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: Mohan, roll no: 24219”);

        System.out.println("    ");

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

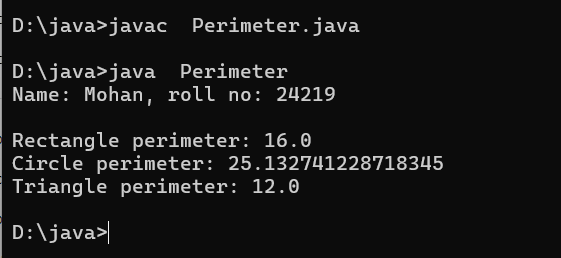
        System.out.println("Circle perimeter: " + circle.getPerimeter());

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

    }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Used extends keyword instead of implements keyword in inheritance** | **Implements key word is added in inheritance** |

1. **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.**

**PROGRAM:**

interface Playable {

    void play();

    public static void main(String[] args) {

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

        football.play();

        volleyball.play();

        basketball.play();

        System.out.println("uday CSE24207 CSE-C");

    }

}

class Football implements Playable {

    public void play() {

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

    }

}

class Volleyball implements Playable {

    public void play() {

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

    }

}

class Basketball implements Playable {

    public void play() {

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

    }

}

public class PlayableTest {

    public static void main(String[] args) {

        Playable football = new Football();

        Playable volleyball = new Volleyball();

        Playable basketball = new Basketball();

System.out.println(“Name: Mohan, roll no: 24219”);

      System.out.println("    ");

        football.play();

        System.out.println("    ");

        volleyball.play();

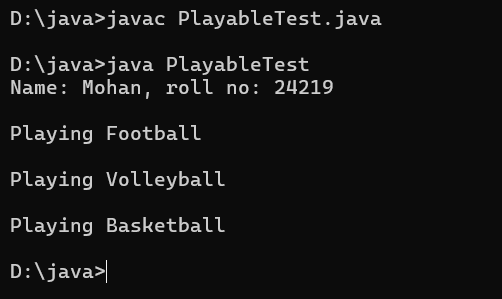
        System.out.println("    ");

        basketball.play();

    }

}

**OUTPUT:**

****

**ERROR:**

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

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

**PROGRAM:**

interface LoginSystem {

    boolean Login(String ID, int pass);

}

class CollegePortal implements LoginSystem {

    public boolean Login(String ID, int pass) {

        if ((ID=="mohan") && (pass==24219)){

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

            return true;

        }else {

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

            return false;

        }

    }

}

class LoginPortal {

    public static void main(String[] args) {

        CollegePortal CP = new CollegePortal();

System.out.println(“Name: Mohan, roll no: 24219”);

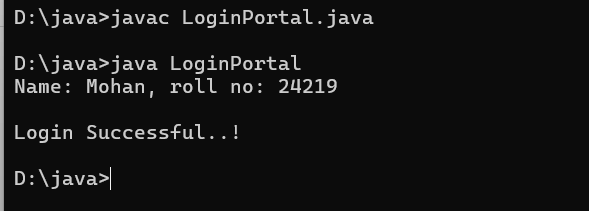
        System.out.println("    ");

        CP.Login("mohan", 24219);

    }

}

**OUTPUT:**

****

**ERROR:**

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