**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Verified By Name: A.Krishna Hasini**

**ROLL NO:24013**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Title** | **Date** | **Page No.** | **Signature** |
| **Week 1** |  | **29-01-2025** |  |  |
| **1.** | **How to download and install Java Software.** |  | **3** |  |
| **2.** | **Write a Java Program to print the message “Welcome to Java Programming”.** |  | **4** |  |
| **3.** | **Write a Java program that prints: Name, Roll.no. , section of a student.** |  | **5** |  |
| **Week 2** |  | **12-02-2025** |  |  |
| **1.** | **Write a java program to calculate the area of a rectangle.** |  | **6** |  |
| **2.** | **Write a java program to temperature from Celsius to Fahrenheit and vica-versa.** |  | **7** |  |
| **3.** | **Write a java program to calculate the simple interest.** |  | **8** |  |
| **4.** | **Write a java program to find the largest of three numbers, using ternary operator.** |  | **9** |  |
| **5.** | **Write a java program to find the factorial of a number.** |  | **10** |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

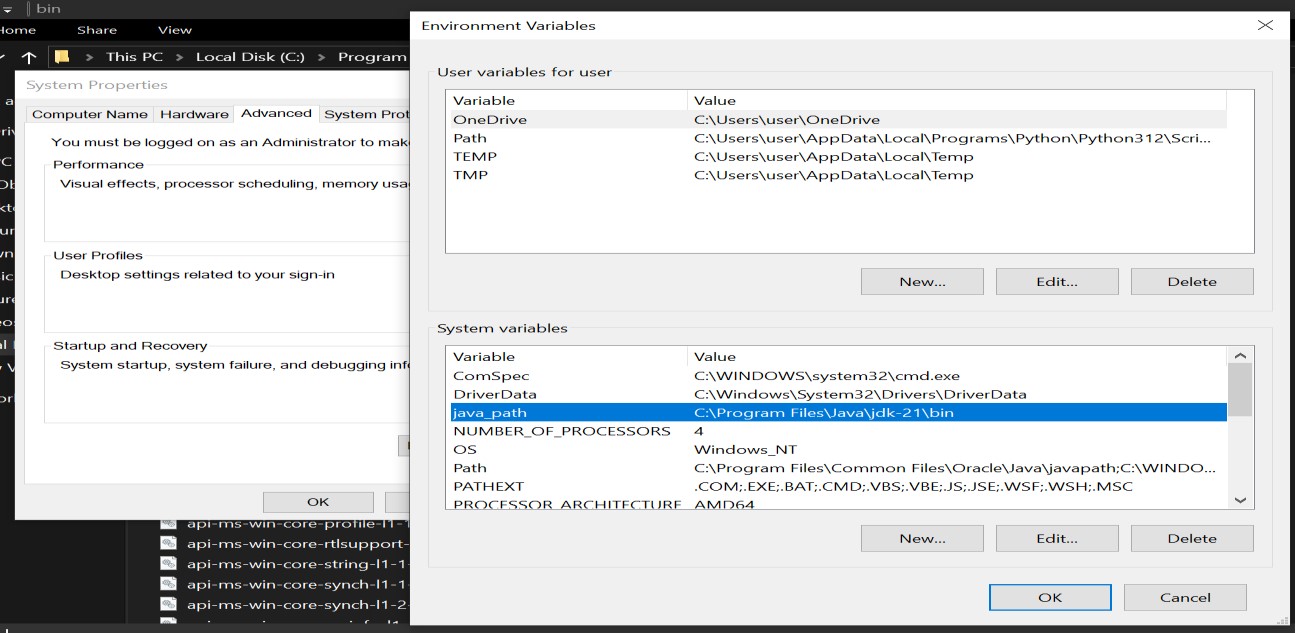
**WEEK-1**

**TASK-1**

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

**PROCEDURE:**

1. **Download JDK:**
   * Go to the Oracle JDK download page in your web browser and click on JDK21 version which is Long term support (LTS) version. o Click on the download link for your operating system (Windows, macOS, or Linux).
2. **Install JDK:**
   * Once downloaded, run the installer.
   * Follow the instructions and keep clicking "Next" until it's done.
3. **Set Environment Variables (Windows):**
   * Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on **Environment Variables**.
   * Click **New** under **System Variables**:
     + **Set Variable name as:** java\_home
     + **Variable value:** The folder address where JDK is installed (like C:\Program Files\Java\jdk-21\bin)
     + Find Path under **System Variables**, click **Edit**, and add the path of the jdk-21(C:\Program Files\Java\jdk-21\bin)



**Checking of JDK Version:**

1. **Open Command Prompt:** o Presswin+R, typecmd, and press Enter.
2. **Check Version:**

o Type java --version and press Enter. o Type javac --version and press Enter.

**TASK-2**

**AIM: Simple Java Program for printing a message “Welcome to Java Programming”.**

Write your code in Notepad and execute in cmd prompt

**CODE:**

class Main

{

public static void main(String[] args)

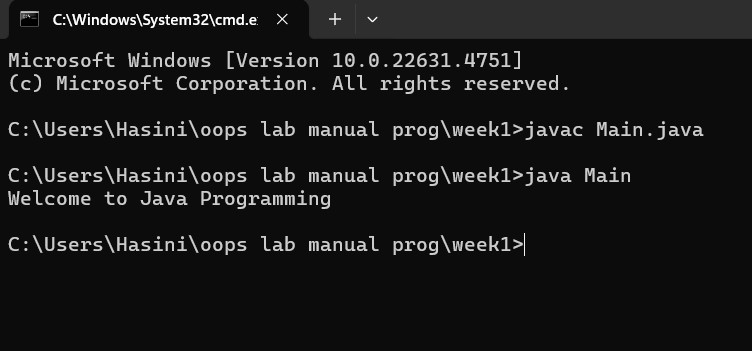
{

System.out.println("Welcome to Java Programming");

}

}

**Output:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: ';' expected** | **Adding a semi-colon at the last** |
|  | **System.out.println("Welcome to**  **Java Programming")** | **System.out.println("Welcome to Java Programming");** |
| **2.** | **error: reached end of file while parsing**  **}** | **Placing a curly bracket at the end of file, to close the class** |

**IMP POINTS:**

* **Every Java statement must end with a semicolon (;).**
* **If using a filename, it should match the class name (Main.java for class Main).** **Java is case-sensitive (Main ≠ main)**

**TASK-3**

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

Write your code in Notepad and execute in cmd prompt

**CODE:**

class Main

{

public static void main(String[] args)

{

System.out.println("Name: A.Krishna Hasini");

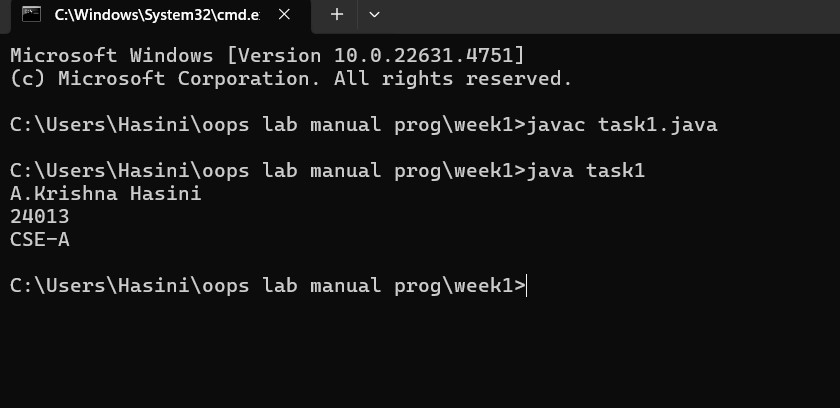
System.out.println("Section: CSE-A");

System.out.println("Roll No : 24013")

}

}

**Output:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: reached end of file while parsing**  **}** | **Placing a curly bracket at the end of file, to close the class** |

**IMP POINTS:**

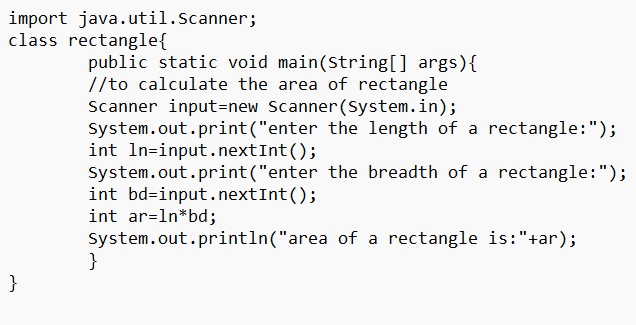
* Every Java statement must end with a semicolon (;).
* If using a filename, it should match the class name (Main.java for class Main). Java is case-sensitive (Main ≠ main)
* Every Java program must have at least one class

**WEEK 2**

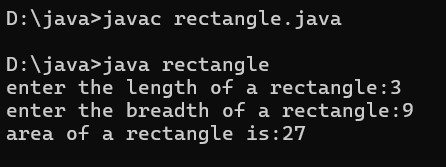
**Program 1**

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

**Code:**



**Output:**



**Error:**

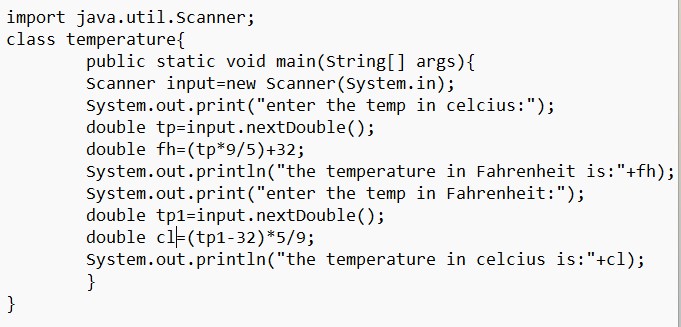
|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: incompatible types: Scanner cannot be converted to System**  **System input= new**  **Scanner(System.in);** | **Change System into Scanner** |

**IMP POINTS:**

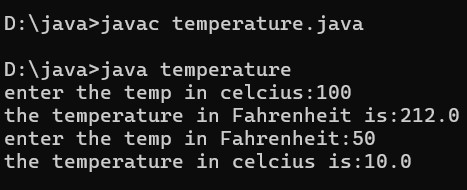
1. **import java.util.Scanner; -** To accept input from user, Scanner class under util package has to be imported.
2. **Scanner input=new Scanner(System.in); -** Used to create a Scanner object
3. **int ln=input.nextInt(); -** Used to read the integer data type stored under the object create
4. **System.out.println(“ “);** - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line**.**

**Program 2**

**Write a java program to convert temperature from Celsius to Fahrenheit and vicaversa. Code:**



**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: cannot find symbol**  **fh=(tp\*9/5)+32;** | **Declare the variable: double fh=(tp\*9/5)+32;** |
| **2.** | **error: ';' expected**  **System.out.print("Enter the temp in Farenheit:")** | **Add a semicolon at the end of the statement**  **System.out.print("Enter the temp in Farenheit:");** |

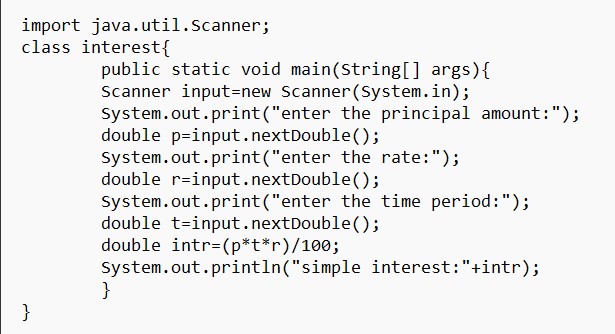
**IMP POINTS:**

1. **import java.util.Scanner;** - To accept input from user, Scanner class under util package has to be imported.
2. **Scanner input=new Scanner(System.in); -** Used to create a Scanner object
3. **double fh=input.nextDouble(); -** Used to read double data type stored under the object created
4. **System.out.println(“ “);** - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

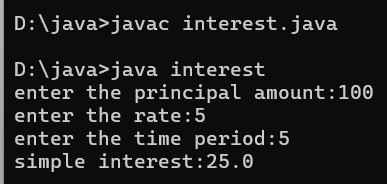
**Program 3:**

**Write a java program to calculate the simple interest.**

**Code:**



**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  double intr=(p\*r\*t)/100 | Add a semicolon at the end of the statement  double intr=(p\*r\*t)/100; |
| 2. | error: cannot find symbol double intr=(p\*r\*t)/100; symbol: variable p location: class interest | Create a reader object double p=input.nextDouble(); |

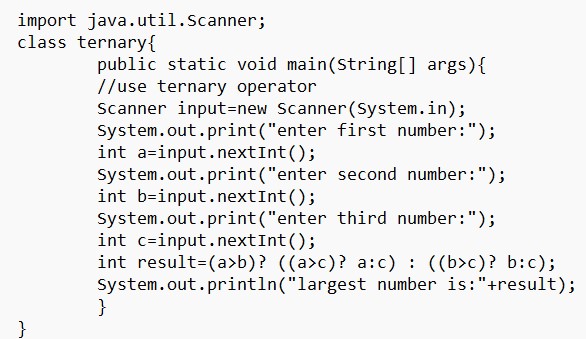
**IMP POINTS:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. double p=input.nextDouble(); - Used to read double data type stored under the object created
4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

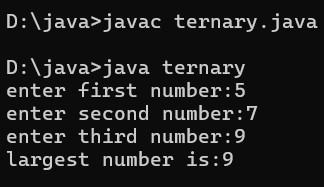
**Program 4**

**Write a java program to find the largest of three numbers, using ternary operator.**

**Code:**



**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected int result=(a>b) ((a>c)? a:c) :  ((b>c)? b:c); error: not a statement int result=(a>b) ((a>c)? a:c) :  ((b>c)? b:c); | Add a ‘?’ int result=(a>b)? ((a>c)? a:c)  : ((b>c)? b:c); |
| 2. | error: ';' expected int result=(a>b)? ((a>c)? a:c) :  ((b>c)? b:c) | Add a ‘;’ int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); |

**IMP POINTS:**

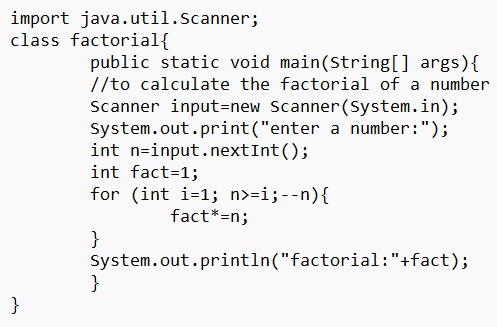
1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. int a=input.nextInt (); - Used to read integer data type stored under the object created
4. int result=(a>b)? ((a>c)? a:c) : ((b>c)? b:c); - Nested Ternary operator is used here.

Syntax for ternary operator is- condition? expression 1: expression 2; , whose answer is stored in a variable and then used.

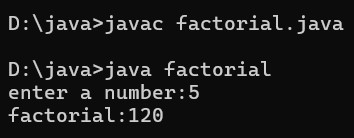
**Program 5**

**Write a java program to find the factorial of a number.**

**Code:**



**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected | Add a “;” |
|  | fact\*=n | fact\*=n; |

**IMP POINTS:**

1. for (int i=1; n>=i;--n){ } - For loop syntax: for(initial expression; test expression; update expression){} The loop is executed, until the test expression evaluates to be false.

**WEEK- 3**

**Program 1**

**Create a java program with the following instructions.**

1. Create a class with name car
2. Create four attributes named car color,car brand,fuel type and milage.
3. Create three methods named start(),stop(),service(). IV. Create three objects named car1,car2,car3.

**Code:**

class car

{

public String car\_color; public String car\_brand; public String fuel\_type; public float mileage;

public void start()

{

System.out.println("Car starts");

}

public void stop()

{

System.out.println("Car stops");

}

public void service()

{

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

}

public static void main(String [] args){

// object 1 creation car car1= new car(); car1.car\_color="White"; car1.car\_brand="Toyota"; car1.fuel\_type="Diesel";

car1.mileage=13.8F;

//calling methods for object 1 car1.start(); car1.stop(); car1.service();

System.out.println("Color of the car1 is "+car1.car\_color);

System.out.println("Brand of the car1 is "+car1.car\_brand);

System.out.println("Fuel type of the car1 is "+car1.fuel\_type);

System.out.println("Mileage of the car1 is"+car1.mileage);

// object 2 creation car car2= new car(); car2.car\_color="Black"; car2.car\_brand="Mahindra"; car2.fuel\_type="Diesel";

car2.mileage=17.3F;

// calling methods for object 2 car2.start();

car2.stop(); car2.service();

System.out.println("Color of the car2 is "+car2.car\_color);

System.out.println("Brand of the car2 is "+car2.car\_brand);

System.out.println("Fuel type of the car2 is "+car2.fuel\_type);

System.out.println("Mileage of the car2 is"+car2.mileage);

//object 3 creation car car3= new car(); car3.car\_color="Brown"; car3.car\_brand="Ford"; car3.fuel\_type="Diesel";

car3.mileage=15.5F;

// calling methods for object 3 car3.start(); car3.stop(); car3.service();

System.out.println("Color of the car3 is "+car3.car\_color);

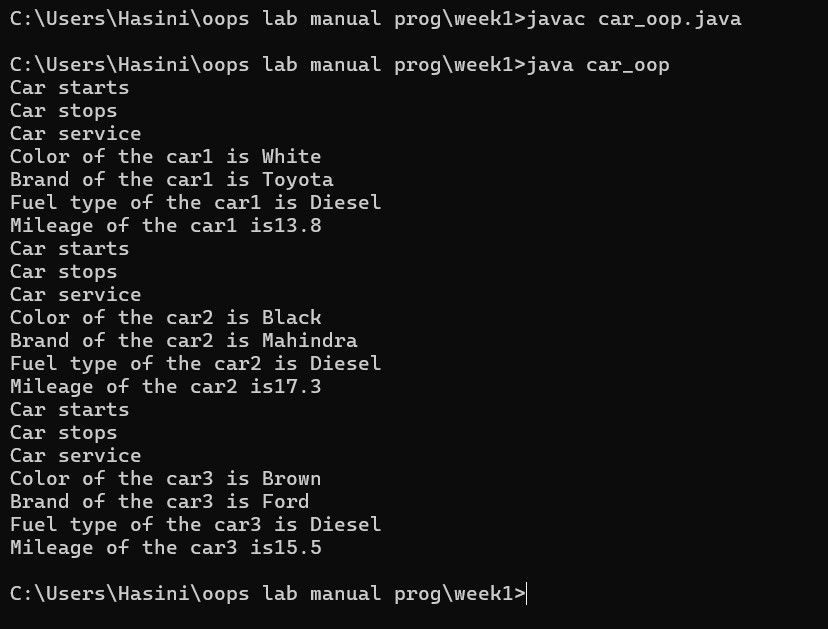
System.out.println("Brand of the car3 is "+car3.car\_brand);

System.out.println("Fuel type of the car3 is "+car3.fuel\_type);

System.out.println("Mileage of the car3 is"+car3.mileage) }

}

**OUTPUT:**



**ERROR:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: incompatible types: Scanner cannot be converted to System**  **System input= new**  **Scanner(System.in);** | **Change System into Scanner** |

**IMP POINTS:**

* A **class** (Car) acts as a blueprint.
* An **object** is an instance of a class (new Car(...)).
* Use **private** variables and provide **public** methods to access or modify them.
* A **constructor** initializes an object with specific values when it is created.

**Program 2**

**Create a java program with the following instructions.**

1. Create a class with named bank account
2. Create two methods named withdraw(),deposit().

**Code:**

import java.util.Scanner;

class BankAccount { String name; int AccountNum; float currentAmount;

public BankAccount(String name, int AccountNum, float currentAmount) { this.name = name; this.AccountNum = AccountNum; this.currentAmount = currentAmount;

}

public void deposit() {

Scanner input = new Scanner(System.in);

System.out.println("Enter amount to be Deposited: "); float deposit = input.nextFloat(); currentAmount += deposit;

System.out.println("Existing Balance now is: " + currentAmount);

}

public void withdrawal() {

Scanner input = new Scanner(System.in);

System.out.println("Enter amount to be withdrawn: "); float withdrawal = input.nextFloat();

if (currentAmount < withdrawal) {

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

} else {

currentAmount -= withdrawal;

System.out.println("Remaining Balance: " + currentAmount);

}

}

public static void main(String[] args) {

BankAccount BA = new BankAccount("Veena", 123456, 100000);

BA.withdrawal();

BankAccount BA1=new BankAccount(“Hasini",9876543,200000);

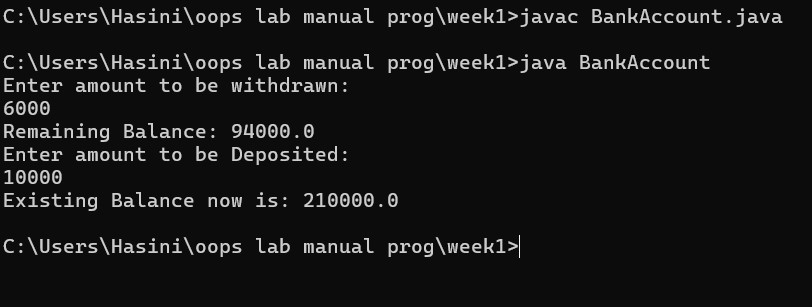
BA1.deposit();

}

}

**OUTPUT:**

**ERROR:**



|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: incompatible types: Scanner cannot be converted to System**  **System input= new Scanner(System.in);** | **Change System into Scanner** |

**IMP POINTS:**

* A **class** (BankAccount) acts as a blueprint.
* An **object** is an instance of a class (new BankAccount(...)).
* Use **private** variables and provide **public** methods to access or modify them.
* A **constructor** initializes an object with specific values when it is created.

**WEEK-4**

**Program 1:**

**1) Write a java program with class named “book” the class should contain various attributes such as “title”,”author”,”year\_publication”.It should also contain a constructor with parameters which initializes “title”,”author” and “year\_publication”.**

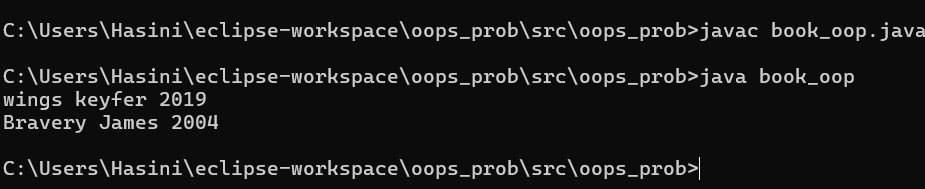
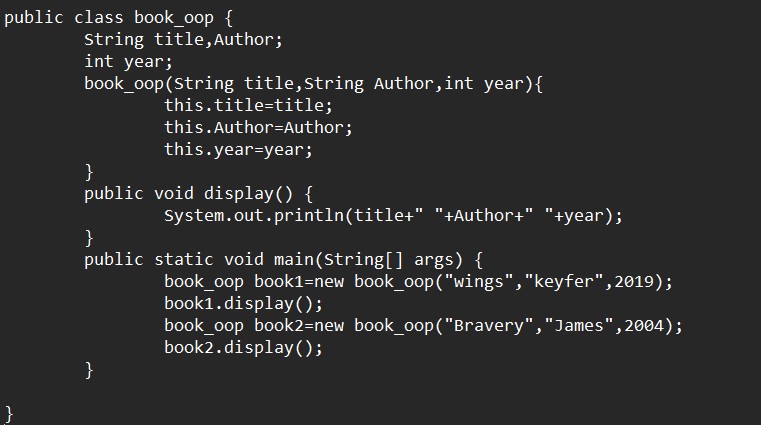
* **Create a method which displays the details of the book i.e.”title”,”author” and**

**“year\_Publication”.**

* **Display the details of two books by creating two objects.**

**Code:**

**OUTPUT:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **error: could not load main class;** | **Keep class name and file name same.** |

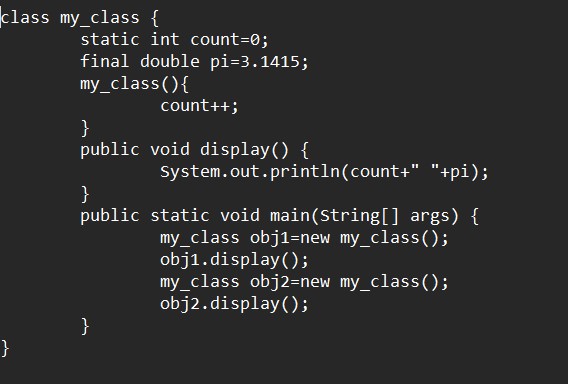
**IMP POINTS:**

* **Keep same name for class and file.**
* **Keep class name for constructor.**
* **To print various variables we should keep ‘+’ between them, not ‘,’.**

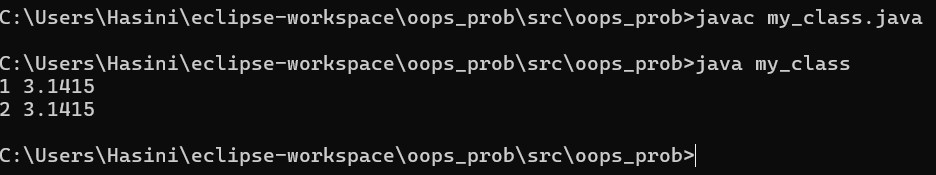
**--------------------------------------------------------------------------------------------------**

**Program-2:To create a java program with class named “My\_class” with a static variable “count” of “int” type,initialized to 0 and a constant variable “pi” of type**

**“double” initialized to 3.1415 as attributes of that class.Define a constructor for “My\_class” that increments the count variable each time an object of “My\_class” is created finally print the final values of “count” and “pi” values.**



**OUTPUT:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| **1.** | **Error: Static;** | **Static** |
| **2.** | **Error:float pi** | **double pi** |

**IMP POINTS:**

* **Use ‘static’ keyword to access class level variables.**
* **Use ‘final’ keyword for constant variables.**

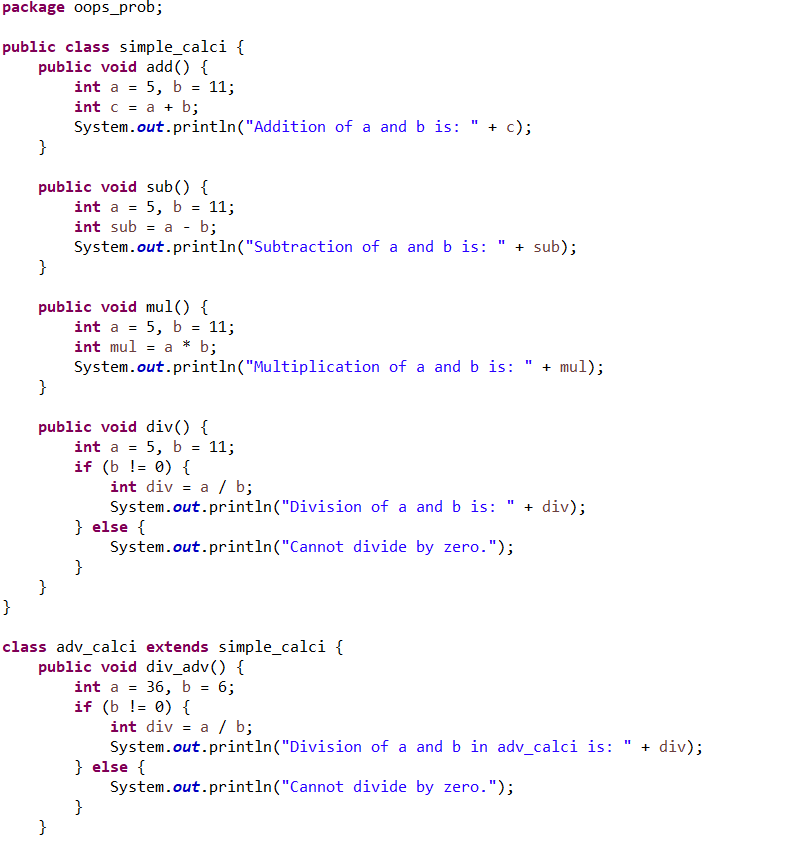
**WEEK-5**

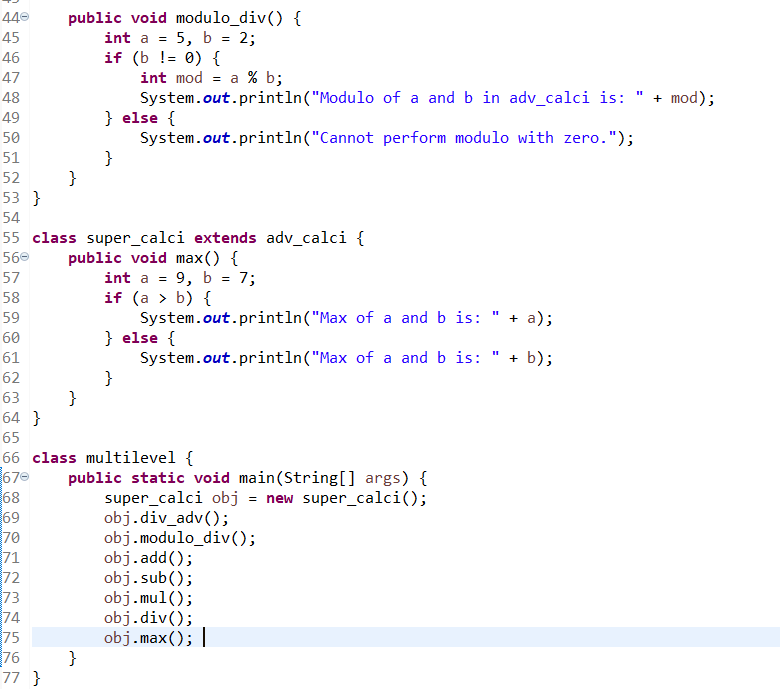
**Task-1:**

**AIM:** Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.

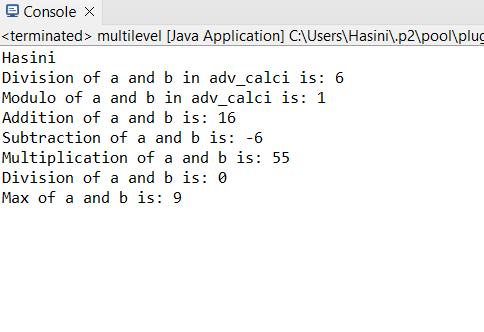
Hint: collect required variables using super class, create each class for a parameter and each class must contain a method.

Program 1:





**OUTPUT:**

****

**IMPORTANT POINTS:**

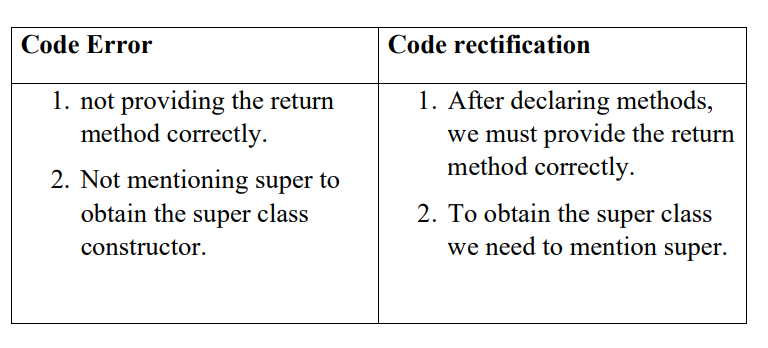
1. To get the inputs from the user we use import java.util.Scanner; this is a package.

2. Scanner class is used to get the user input.

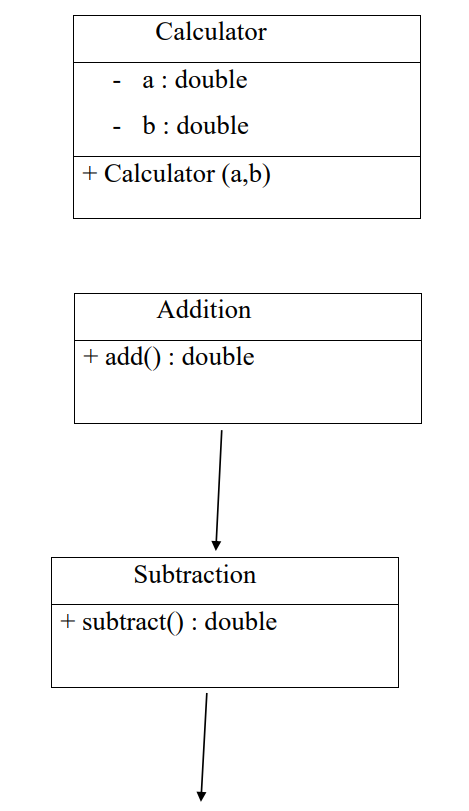
3. In java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.

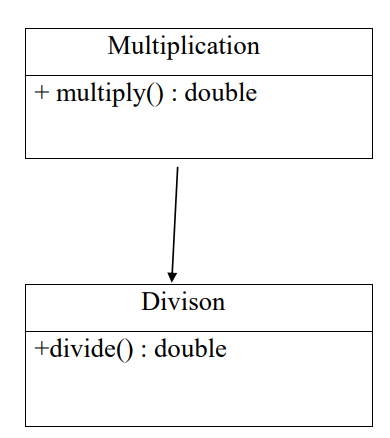
4. To import a whole package, end the sentence with an asterisk sign(\*).

**ERRORS:**

****

**CLASS DIAGRAM:**





**PROGRAM-2:**

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

1. cars should have an additional property: no.of doors

2. Bikes should have a property indicating whether they have gears or not.

3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.

4. Every class should have a constructor

**Question:** 1. Which oops concept is used in the above program

2. If the company decides to add a new type of vehicle, Truck, how would you modify the program? a. Truck should include an additional property capacity (in tons) Multiplication + multiply() : double Divison +divide() : double

b. Create a showTruckdetails() method to display the truck’s capacity.

c. Write a constructor for Truck that initializes all properties

3. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details.

**IMPORTANT** **POINTS:** 1. a constructor helps in initializing an object that doesn't exist.

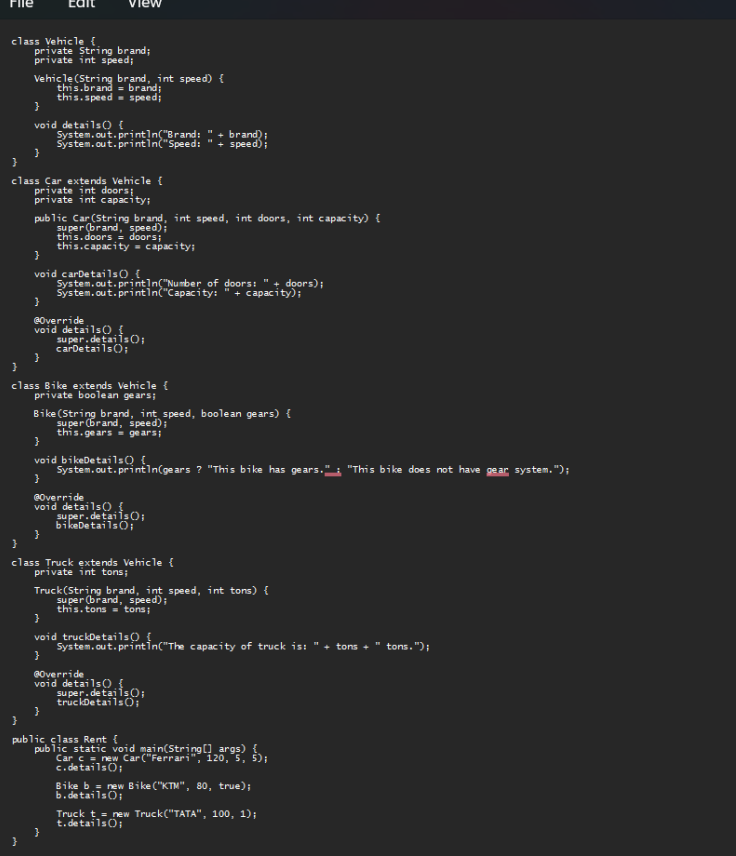
2. a method performs functions on pre-constructed or already developed objects.

3. a double method can represent more decimal point numbers than float method.

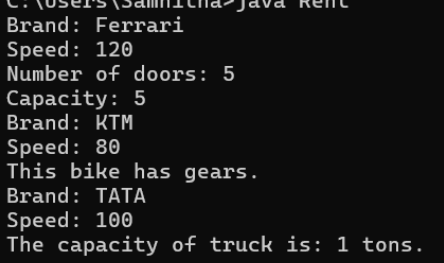
4. the void keyword in java is used to specify that a method does not return any value. it is a return type that indicates the method performs a function and doesn't produce a result.

**Answer for Q1:** The oops concepts used in the above program are: Inheritance, encapsulation, polymorphism, abstraction.

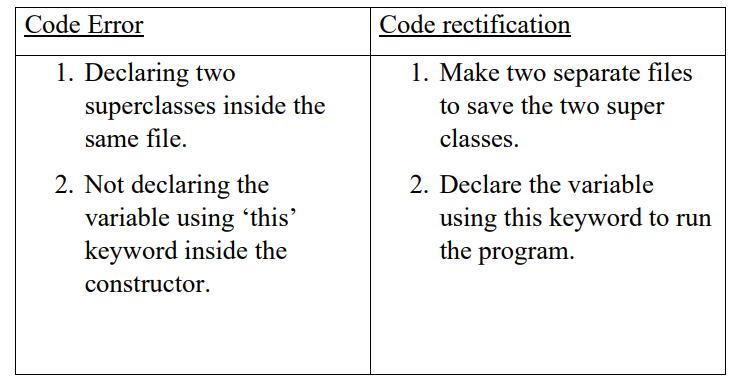
**CODE:**

****

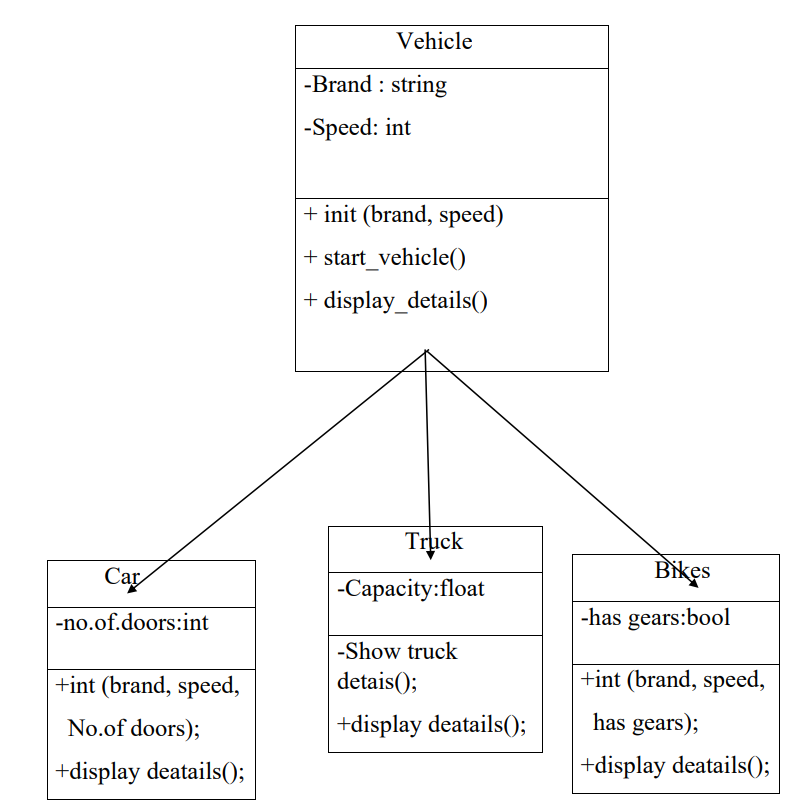
**OUTPUT:**

****

**ERROR TABLE:**

****

**CLASS DIAGRAM:**



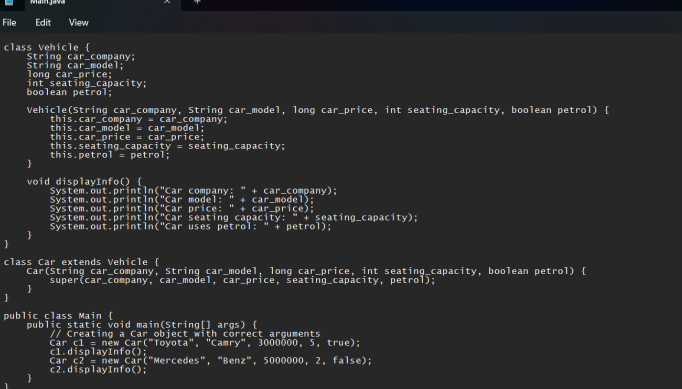
**WEEK-06**

**PROGRAM-1:**

**AIM:**

Write a java program to create a vehicle class with a method displayinfo(). Override this method in the car subclass to provide specific information about car (car company, seating capacity, petrol or not).

**CODE:**

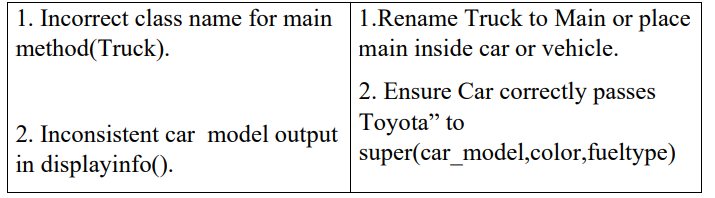
****

**OUTPUT:**

****

**ERRORS:**





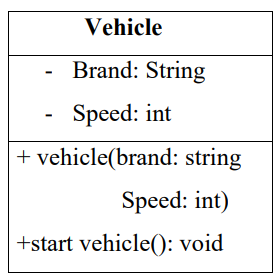
**IMPORTANT POINTS:**

1.**Inheritance:** The Car class extends the Vehicle class, demonstrating inheritance in Java. 2.**Constructor Chaining:**The Car class calls the parent constructor using super(car\_model, color, fuel\_type); to initialize inherited attributes.

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

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

**CLASS DIAGRAM:**

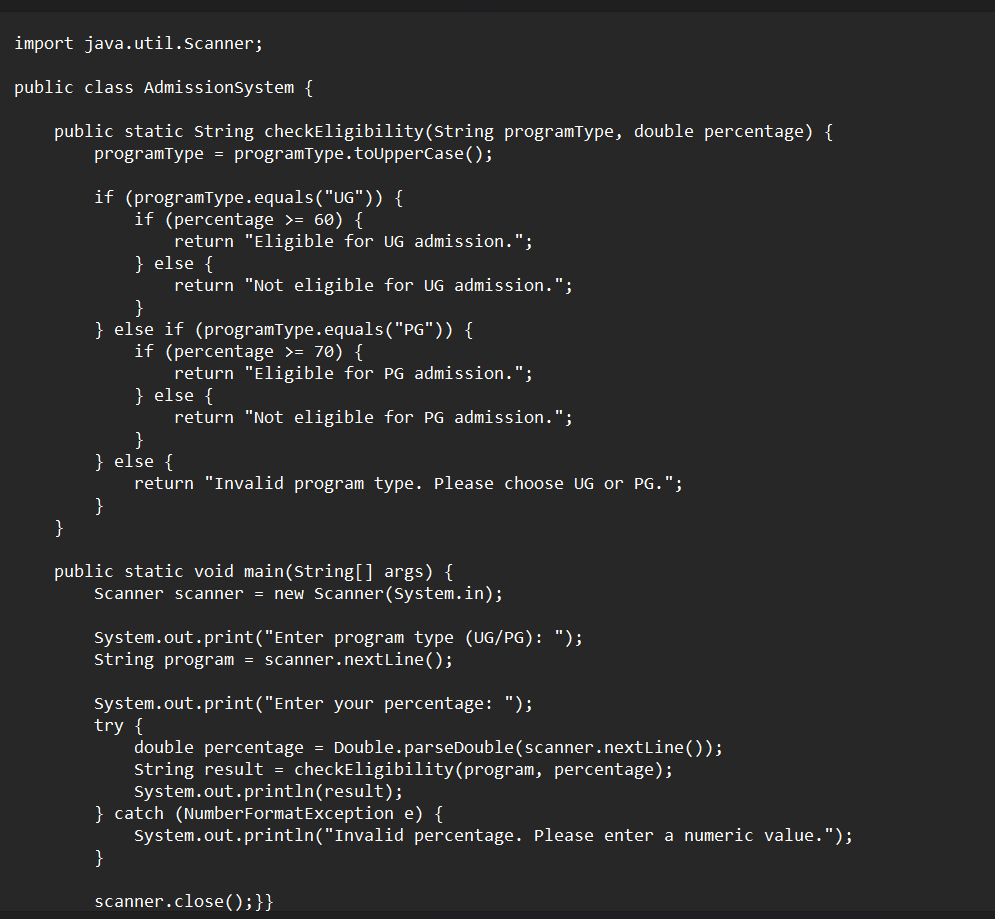
****

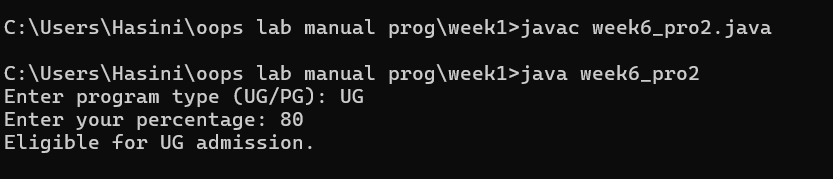
**PROGRAM-2:**

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

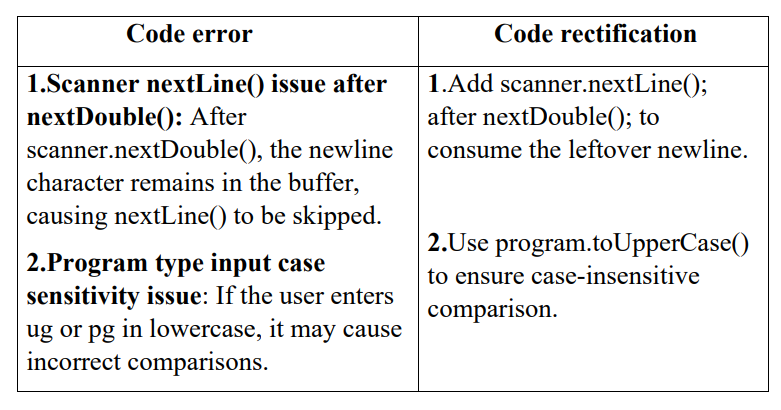
1. UG admission recquire a minimum of 60%.

2. PG admission recquire a minimum of 70%.

**CODE: **

**OUTPUT: **

**ERRORS:**

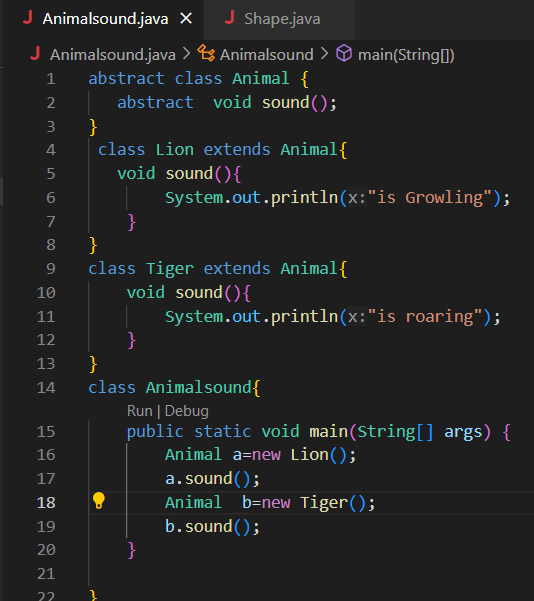
****

**WEEK-7**

**PROGRAM-1:**

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

**CODE:**



**OUTPUT:**



**IMPORTANT POINTS:**

* 1. abstract class Animal: Can't be directly used to create objects.
  2. abstract void sound(): Forces subclasses to implement this method.
  3. Lion and Tiger both override sound().
  4. Animal a = new Lion(); uses runtime polymorphism.

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Forgetting to use abstract keyword for the sound() method.  2 Not overriding the sound() method in subclasses. | 1.Rectified as abstract void sound();  2. Added void sound() { ... } in each subclass. |

|  |
| --- |
| + sound(): void |

**CLASS DIAGRAM:**

|  |
| --- |
| **TIGER** |
| + sound(): void |

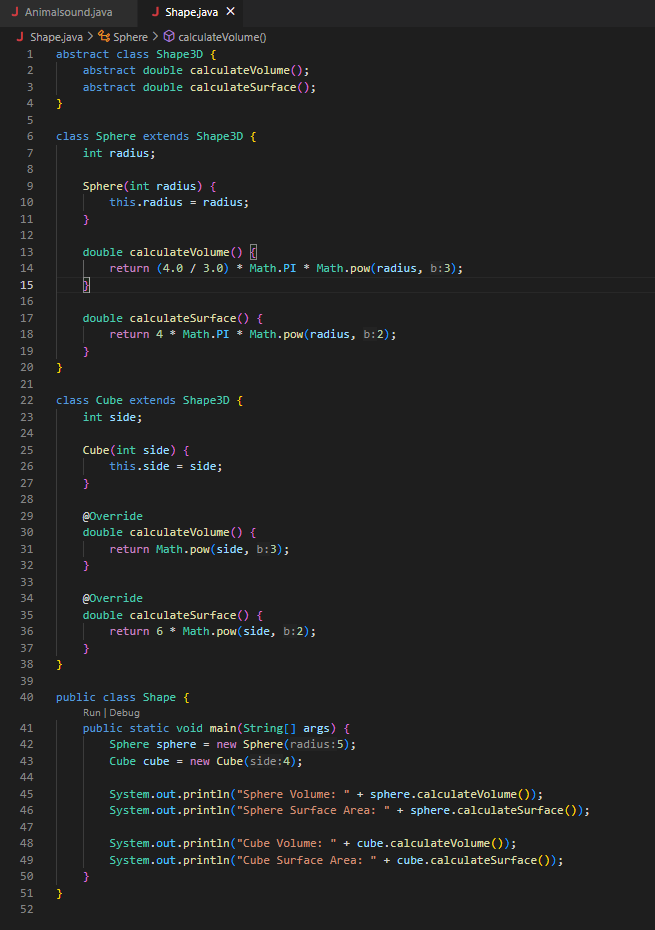
|  |
| --- |
| **LION** |
| + sound(): void |

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

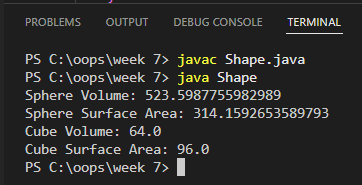
**PROGRAM-2:**

**AIM :** Write a Java program to create an abstract class shape 3D with abstract methods calculate volume ()and calculate surface Area ()create subclasses Sphere and cube that extend the Spape 3D clas and implement the respective methods to calculate ine volume and surface area of each shape.

**CODE:**



**OUTPUT:**



**IMPORTANT POINTS:**

**1.Abstract Class Used**:Shape3D is an abstract class with abstract methods – it can't be directly used to create objects.

**2.Method Overriding**:Sphere and Cube both override calculateVolume() and calculateSurface() with their own formulas.

**3.Return Type: double** :Volume and surface area can be decimal, so methods return double, not int.

**4.Use of Math.PI and Math.pow()**: More accurate than hardcoding 3.14 and r\*r\*r. It's a good practice for real calculations.

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| |  | | --- | |  |  |  |  | | --- | --- | | 1. int used instead of double for Volume surface | | |  |  |  |  |  | | --- | --- | | 1. (4 / 3) used instead of (4.0 / 3.0) | | |  |  |  |  | | --- | | 1. 3.14 used as approximation for π |  |  | | --- | |  |  |  | | --- | |  | | 1. Changed return types of calculateVolume() and calculateSurface() to double 2. Used floating-point division to avoid integer division loss. 3. 3.Used Math.PI for more accurate calculations. |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<abstract>>**  **SHAPE 3D** |
| +calculateVolume():double  +calculateSurface():double |

|  |
| --- |
| **CUBE** |
| - side: int |
| +calculateVolume():double  +calculateSurface():double |

|  |
| --- |
| **SPHERE** |
| - radius: int |
| +calculateVolume(): double  +calculateSurface():double |

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

**PROGRAM-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)
* A concrete method to display the pattern title

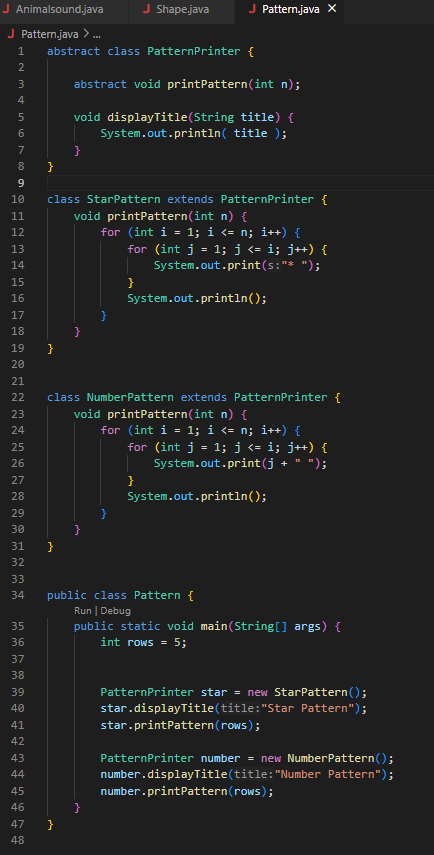
Create two subclasses:

1.StarPattern: Prints a right-angled triangle of stars (\*)

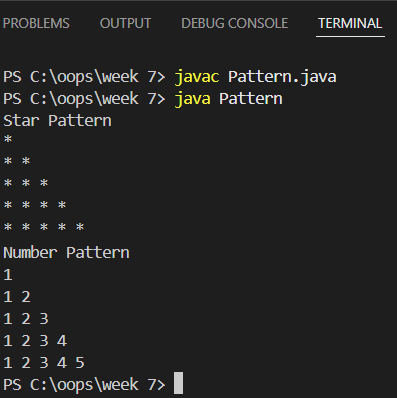
2.NumberPattern: Prints a right-angled triangle of increasing numbers

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

**CODE:**



**OUTPUT:**



**IMPORTANT POINTS:**

1. Abstract class PatternPrinter cannot be instantiated directly.
2. Abstract method printPattern(int n) must be implemented in all subclasses.
3. Concrete method displayTitle(String title) is reusable by both subclasses.
4. Use of inheritance: StarPattern and NumberPattern extend the abstract class.
5. In main(), objects are created from subclasses, not the abstract class.

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | 1. **Code rectification** |
| |  | | --- | |  |  |  |  | | --- | --- | | 1.Wrong loop logic ( printing \* without loop). | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | |  | | --- | | 2.displayTitle method not used before pattern printing |  |  | | --- | | 3.Forgot to implement printPattern(int n) in subclass | | | | |  |  |  |  | | --- | |  |  |  | | --- | |  | | 1.Use nested loops: outer loop for rows, inner loop for printing symbols or numbers.  2.Call displayTitle() before printing the pattern for proper formatting  3.Implemented the method in both subclasses |