**23CSE101**

**OBJECT ORIENTED PROGRAMMING**

**LAB MANUAL**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Verified By Name: K .BHAVAGNYA**

**Roll No: AV.SC.U4CSE2420**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **TITLE** | **DATE** | **PAGE.NO.** | **SIGNATURE** |
| **1.** | **WEEK-1** |  |  |  |
| i) | Download and install java software |  | 6 |  |
| ii) | Write a Java program to print the message “Welcome to Java Programming.” |  | 9 |  |
| iii) | Write a Java Program that prints Name, Roll No, Section of a student |  | 10 |  |
| **2.** | **WEEK-2** |  | 11 |  |
| i) | Write a java program to find the simple interest where all the inputs are given by the user. |  | 11 |  |
| ii) | Write a java program to find the factorial of a number where all the inputs are given by the user. |  | 12 |  |
| iii) | Write a java program to convert the temperature from Celsius to Fahrenheit and Celsius to Fahrenheit. |  | 13 |  |
| iv) | Write a java program to find the Fibonacci series of a given number where all the inputs are taken from the user. |  | 15 |  |
| v) | Write a java program to find the area of a rectangle where the inputs are given by the user. |  | 16 |  |
| vi) | Write a java program to find the area of a triangle where the inputs are given by the user |  | 17 |  |
| **3.** | **WEEK-3** |  | 18 |  |
| i) | Write a java program with  1.Create a class named Bankaccount.  2. Create a constructor.  3. Create 2 methods which are withdrawl() and deposit () |  | 18 |  |
| ii) | Create a class car. Create four attributes named car\_color, car\_brand, fuel\_type, mileage. Create three methods start() stop() service().Create 3 objects named car1 car2 car3. |  | 19 |  |
| **4.** | **WEEK-4** |  | 22 |  |
| i) | 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, Year of publication. Create a method which displays the details of book. Display the details of two books. |  | 22 |  |
| ii) | Create a java program with class name Myclass with a starting variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of that class. 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. Create three objects |  | 23 |  |
| **5.** | **WEEK-5** |  | 26 |  |
| i) | Create a calculator using the operations including addition, subtraction, multiplication and division using Multilevel Inheritance and display the desired output**.** |  | 26 |  |
| ii) | 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   * Cars should have an additional property: number of doors * Bike 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 * Every class should have a constructor  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, also create an object for car and bike subclassed. Finally display its details**.** |  | 29 |  |
| **6.** | **WEEK-6** |  | 35 |  |
| i) | 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 |  | 35 |  |
| ii) | 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 student's percentage in their previous qualification |  | 36 |  |
| iii) | Create a calculator with overloaded methods to perform addition:  i) Add two integers  ii) Add two doubles  iii) Add three integers |  | 38 |  |
| Iv) | 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. |  | 40 |  |
| **7.** | **WEEK-7** | 11-04-2025 | 42 |  |
| i) | Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion & Tiger that extend Animal class and implement the sound() method to make a specific sound for each animal. |  | 42 |  |
| ii) | 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 volume and surface area of each shape. |  |  |  |
| iii) | 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).  Create concrete subclasses:  1. StarPattern - Prints a right-angled triangle of stars (\*).  2. NumberPattern - Prints a right-angled triangle of increasing numbers. |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**WEEK-01**

**PROGRAM-1:**

**AIM:-** Download and install java software.

**PROCEDURE:-**

**1.Download JDK21:**

i. Open the web browser and go to Oracle JDK Downloads page.

ii. Scroll down to the Java SE Development Kit 21 section.

iii.Select` the Windows x64 Installer version.

iv. Click on Download, then Wait for the download to complete**.**



**Step 2:** **Installation of JDK 21:-**

1. Locate the downloaded jdk-21\_windows-x64\_bin.exe file.
2. Double-click to launch the installer.
3. Click Next on the setup wizard.
4. Choose the installation path (default is C:\Program Files\Java\jdk-21).
5. Click Next, then click Install.
6. Wait for the installation to complete.

vii. Click Close once the installation is finished.



**Step 3: Setting up the path:-**

i. Go to “Windows C” Drive in This PC.

ii. Choose Program Files, select Java, then JDK 21, then select Bin.

iii. Select and copy the path at the address bar.



**Step 4: Open System Properties:-**

i.Press Windows + R, type sysdm.cpl , and click Ok-

ii.The System Properties window will open.

iii.Navigate to the Advanced tab.

iv.Click on Environment Variables at the bottom.



**Step 5: Setting up of JAVA\_HOME:-**

i.Under System Variables, click New.

ii. Set the Variable name as JAVA\_HOME.

iii. Set Variable value as C:\Program Files\Java\jdk-21 (or your installation path).

iv. Click on OK.



**Step 6: Update the PATH Variable:-**

i.In System Variables, find Path and click Edit.

ii.Click New and add: C:\Program Files\Java\jdk-21\bin

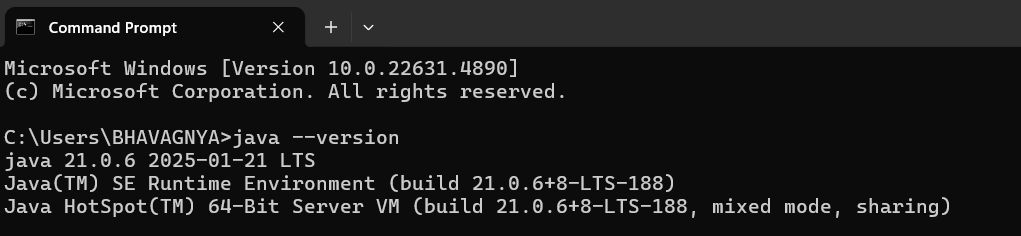
iii.Click OK to save.



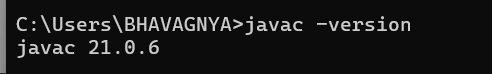
**Step 7:Verify Installation1:-**

i.Open Command Prompt.

ii.Type the following command: **java --version** and press Enter.



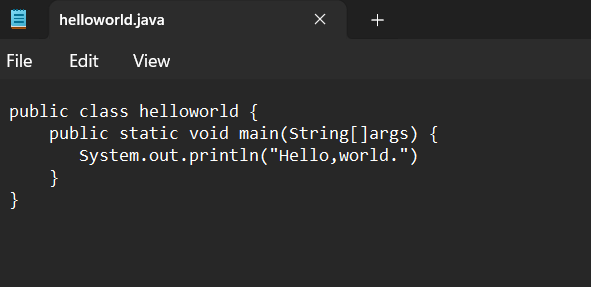
iii.To check the java compiler type: **javac –version.**

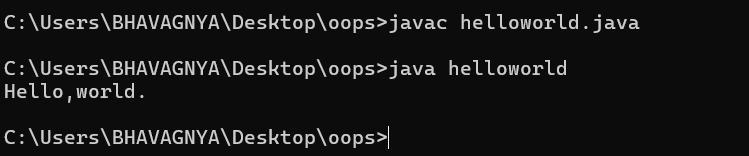
****

**PROGRAM-2:**

**AIM:** Write a Java program to print the message “Welcome to Java Programming.”

**CODE:**

**OUTPUT:-**

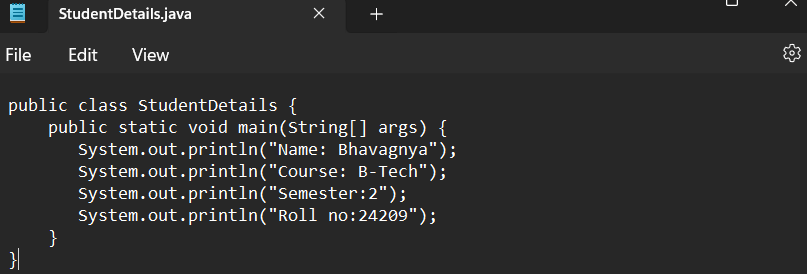
****

**ERROR TABLE:**

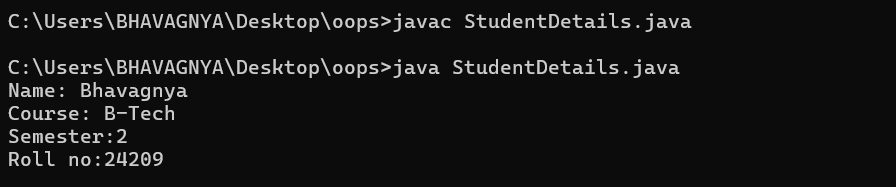
No Errors found.

**PROGRAM-3:**

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

****

**OUTPUT:-**



**ERROR TABLE:**

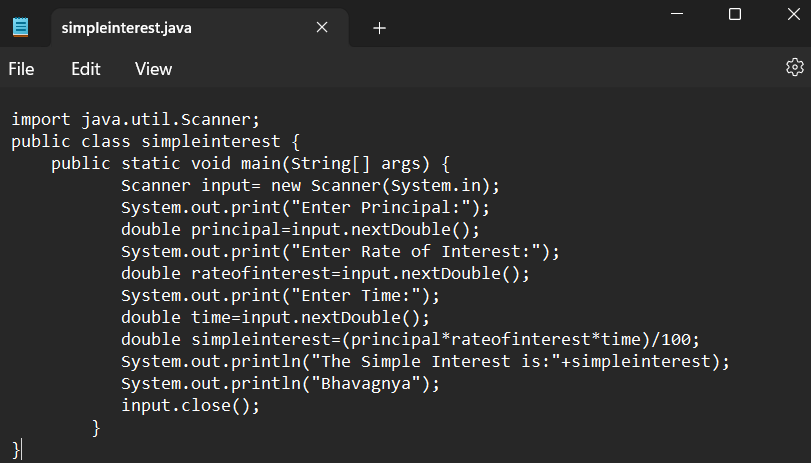
|  |  |
| --- | --- |
| **ERROR IN CODE** | **ERROR RECTIFICATION** |
| 1. ; is missing | 1. code is rectified by keeping a (;) in every place where it is missing. |

**WEEK-2:**

**PROGRAM-1:**

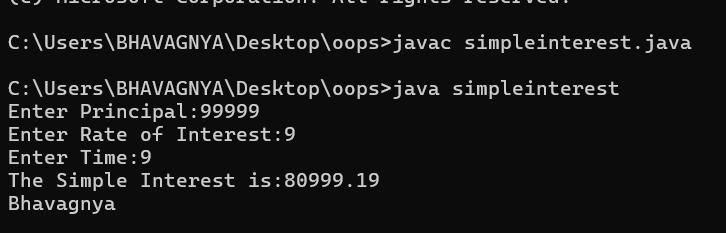
**AIM**: Write a java program to find the simple interest where all the inputs are given by the user.

**CODE:**

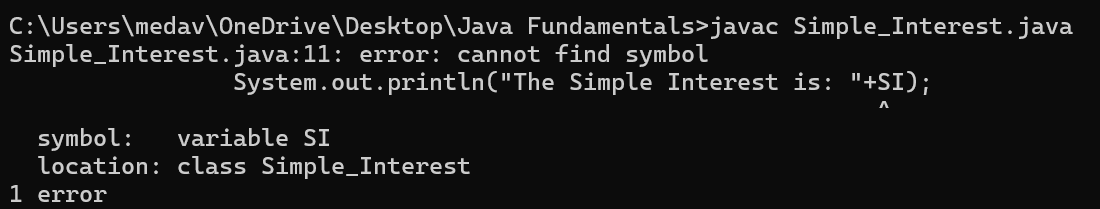


**OUTPUT:-**

**POSITIVE CASE:**



**NEGATIVE:**



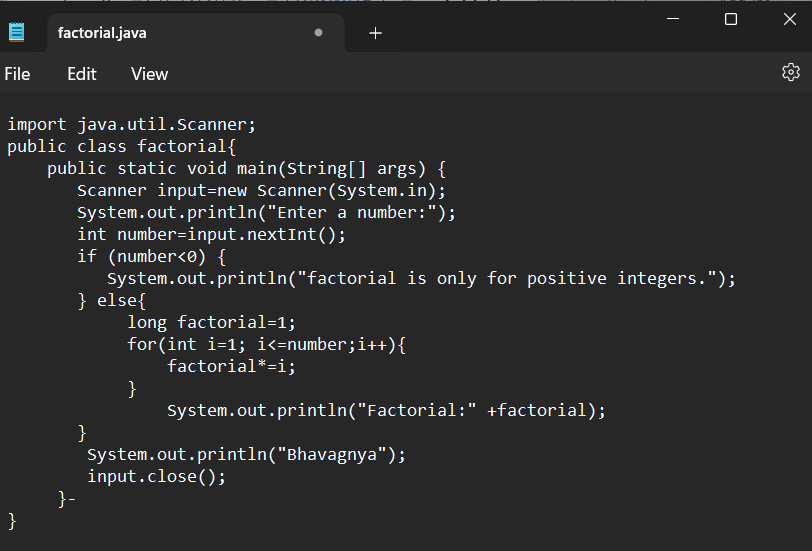
**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1) | ; is expected after input.close() | Corrected by writing a ; after input.close() |
| 2) | Error in the line double principal=input.next.double(); | Corrected by removing “. “after the next. |

**PROGRAM-2:**

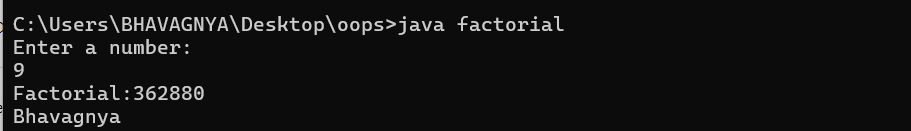
**AIM**: Write a java program to find the factorial of a number where all the inputs are given by the user.

**CODE:**

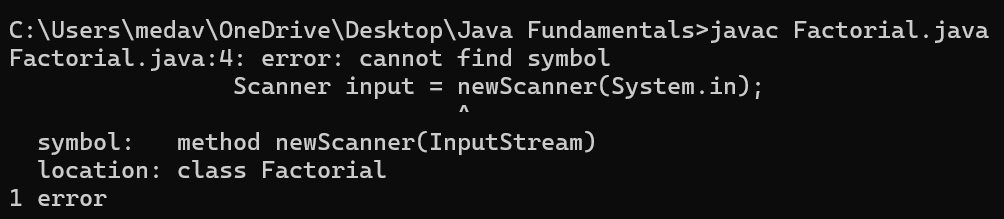


**OUTPUT:**

**POSITIVE CASE:**



NEGATIVE CASE:

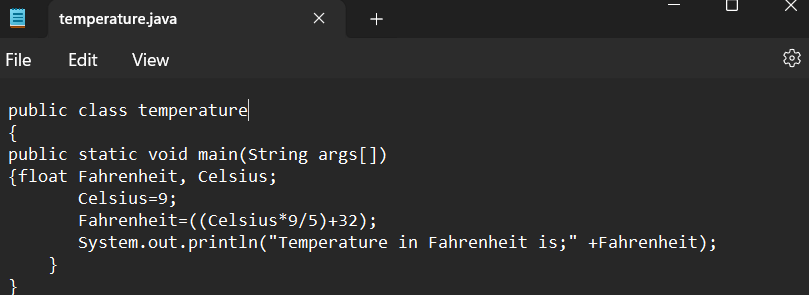


**ERROR TABLE:**

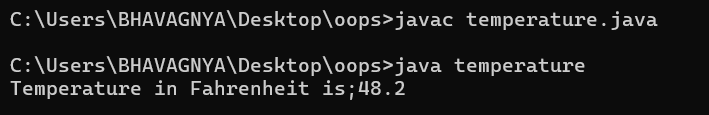
|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1) | writing small “S” in place of ”S” in system.out.println() | code is rectified by keeping capital “S” |
| 2) | “ is missed in the print statement | Corrected by keeping “ in the end of print statement. |

**PROGRAM-3:**

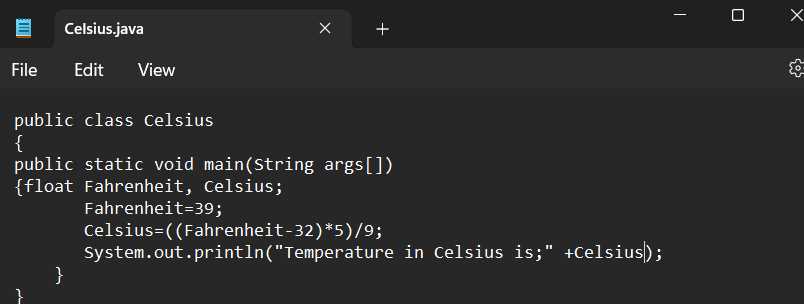
**AIM**: Write a java program to convert the temperature from Celsius to Fahrenheit and Celsius to Fahrenheit.



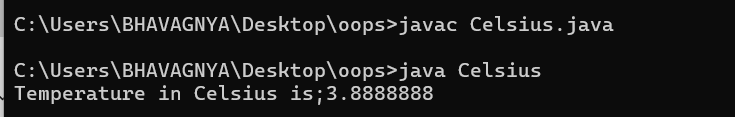
**OUTPUT:-**



**CODE2:**



**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | ERROR RECTIFICATION |
| 1. | Logical error  Missed \* after Celsius in code 1 | Corrected by keeping a \* operator in the appropriate place |

**IMPORTANT POINTS:**

1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

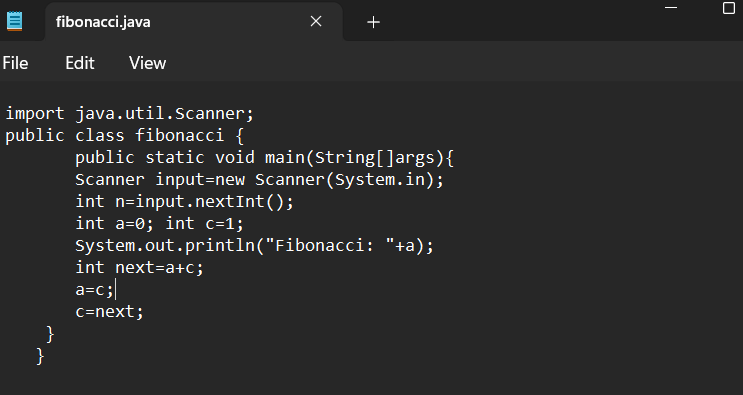
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object

3. double fh=input.nextDouble(); - Used to read double data type stored under the object created

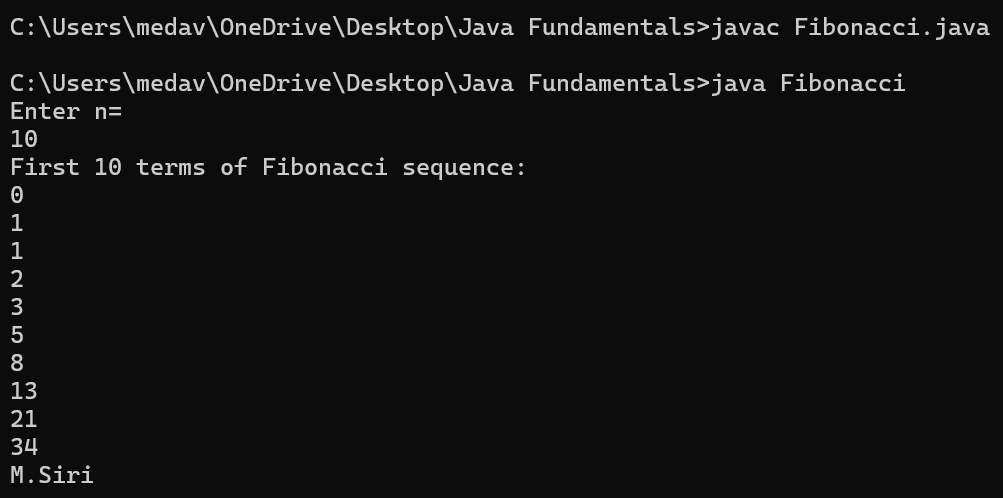
4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

**PROGRAM – 4:**

**AIM**: Write a java program to find the Fibonacci series of a given number where all the inputs are taken from the user.



**OUTPUT:**

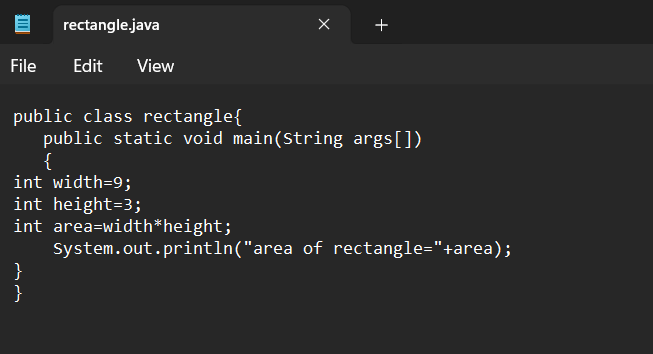




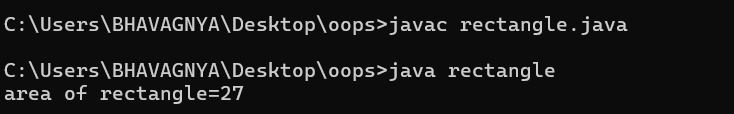
|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | ERROR RECTIFICATION |
| 1. | Expected an ; after line 4 | Corrected by keeping an ; after the line |

**PROGRAM-5**:

AIM: Write a java program to find the area of a rectangle where the inputs are given by the user.



**OUTPUT:**



ERROR:

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1. | “ “ expected in the print statement | Corrected by closing the print statement with “ |

**IMPORTANT POINTS:**

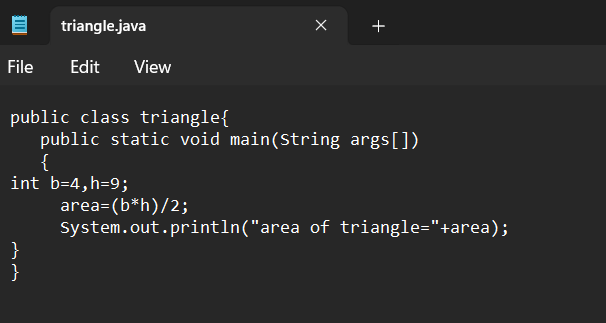
1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.

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 created

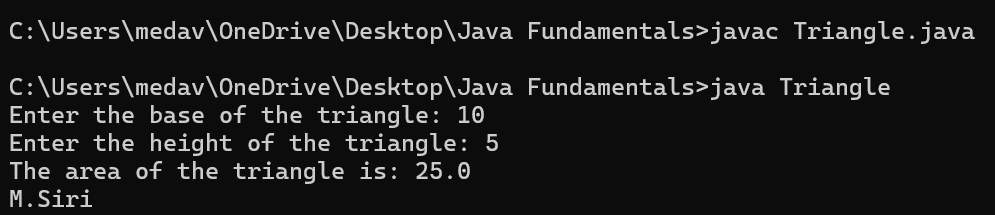
4. System.out.println(“ “); - It is used to print string inside the quotes. After printing, the cursor moves to the beginning of the next line.

**PROGRAM-6**

**AIM:** Write a java program to find the area of a triangle where the inputs are given by the user.



**OUTPUT:**





**ERROR TABLE:**

NO ERRORS

**IMPORTANT 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 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.

**WEEK-03**

**PROGRAM-1:**

**AIM:**  Write a java program with

1.Create a class named Bankaccount.

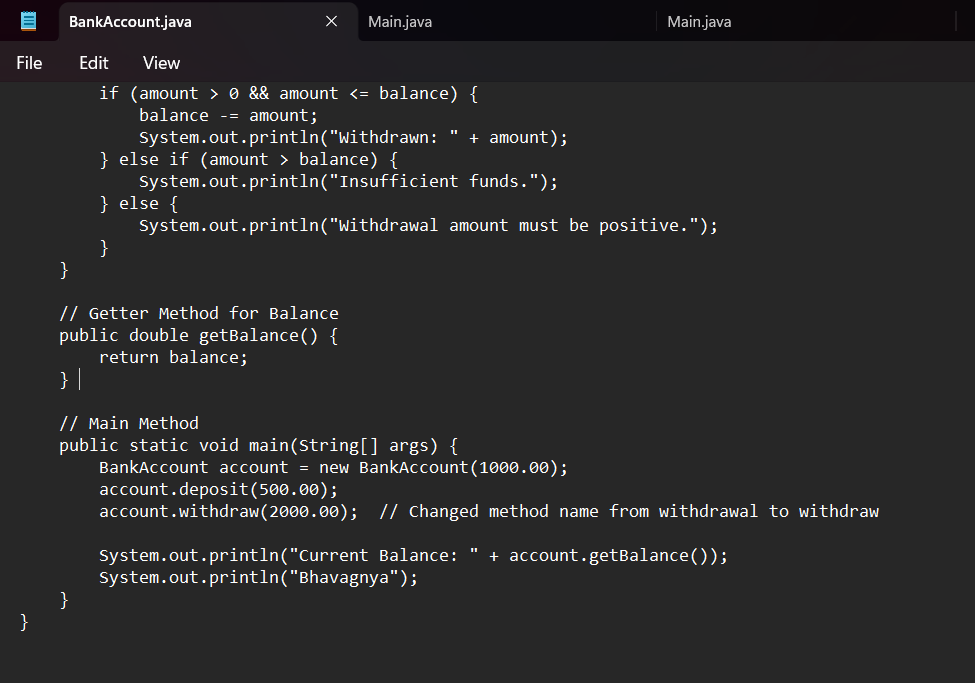
2. Create a constructor.

3. Create 2 methods which are withdrawl() and deposit ()

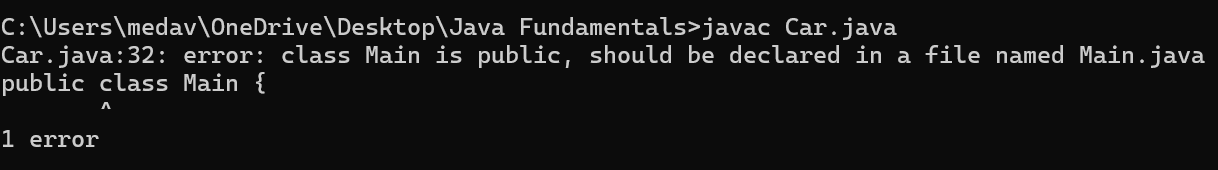
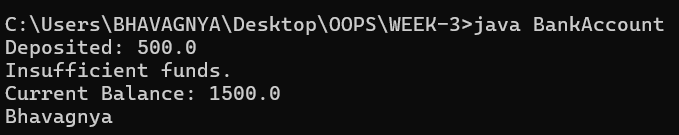
CLASS DIAGRAM:

|  |
| --- |
| Bank account |
| -balance:double |
| +bank account initial balance:double  +deposit amount:double  +withdrawal:double  +getbalance():double |

**CODE:**

****

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1. | writing small “S” in place of ”S” in system.out.println() | code is rectified by keeping capital “S” |

**IMPORTANT POINTS:**

The condition inside the if statement must be correct.

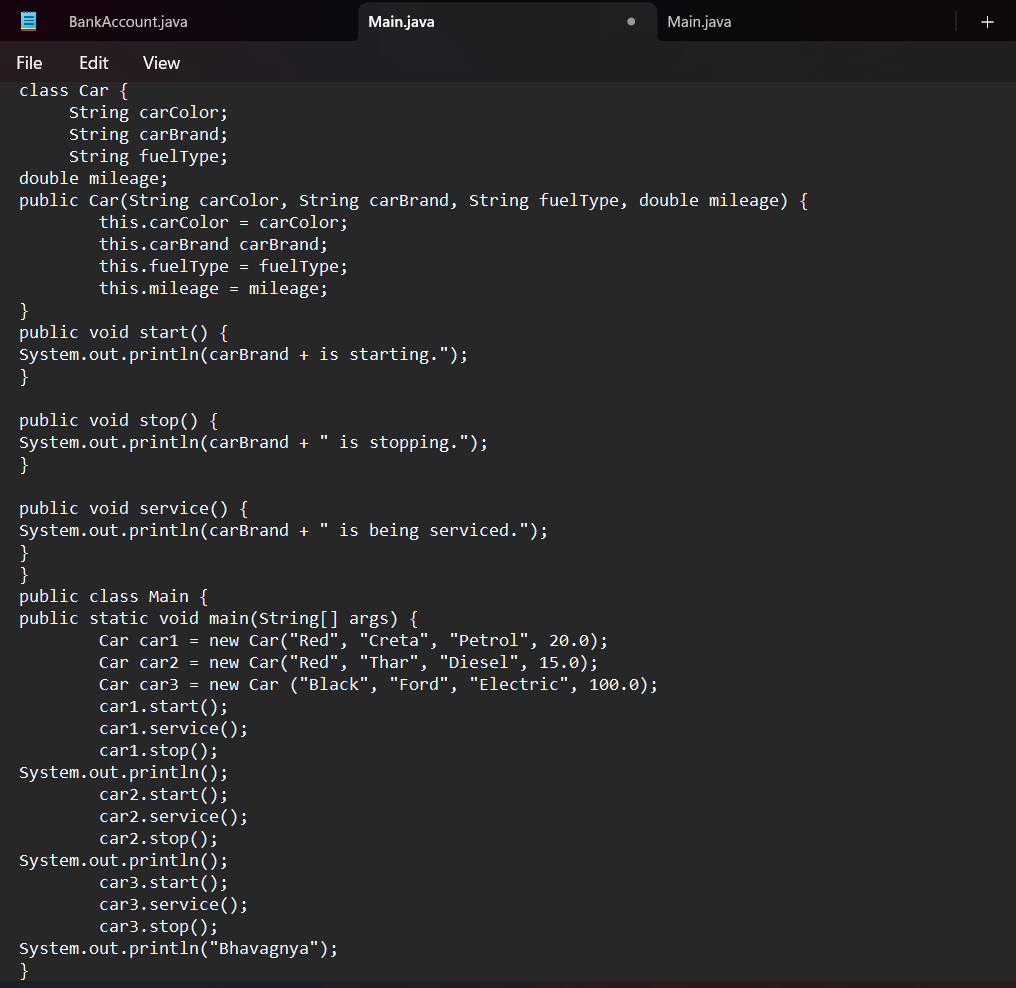
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount

**PROGRAM-2:**

**AIM:** Create a class car. Create four attributes named car\_color, car\_brand, fuel\_type, mileage. Create three methods start() stop() service().Create 3 objects named car1 car2 car3.

CLASS DIAGRAM:

|  |
| --- |
| car |
| +car color:string  +car brand:sstring  -fuel type:string  -mileage:double |
| +start():void  +stop():void  -service():void  +car1():void  +car2():void  +car3():void |

****

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1. | writing small “S” in place of ”S” in system.out.println() | code is rectified by keeping capital “S” |

**IMPORTANT POINTS:**

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

**WEEK-04**

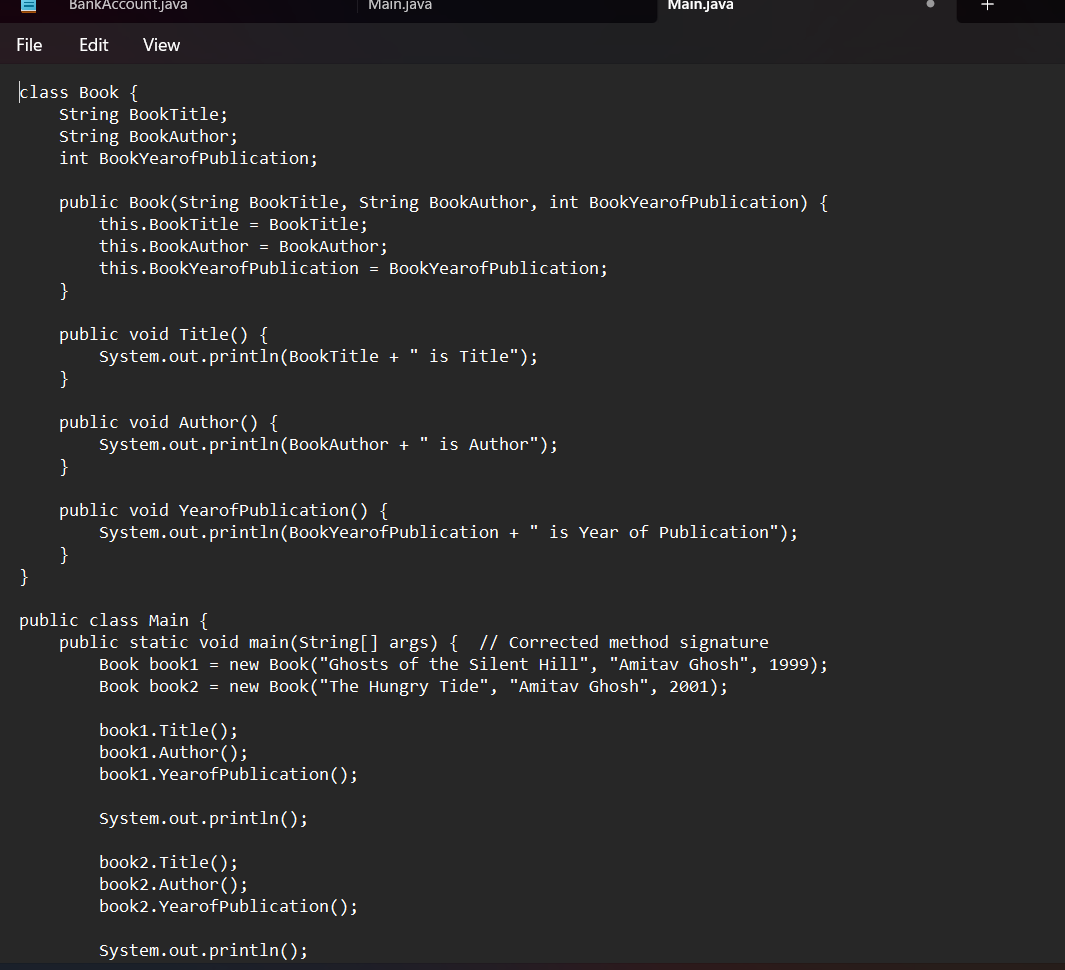
**PROGRAM-1:**

**AIM:** 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, Year of publication. Create a method which displays the details of book. Display the details of two books.

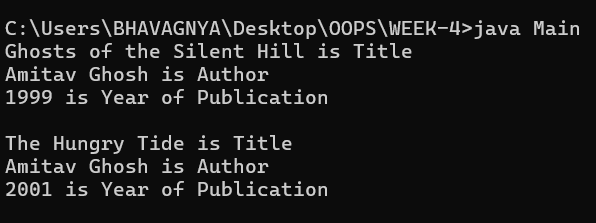
**CLASS DIAGRAM:**

|  |
| --- |
| **Book** |
| - title: String  - author: String  - year: int |
| + Book.display() : void |

**CODE:**

****

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| **1.** | “ is missed in the print statement | Corrected by keeping “ in the end of print statement. |

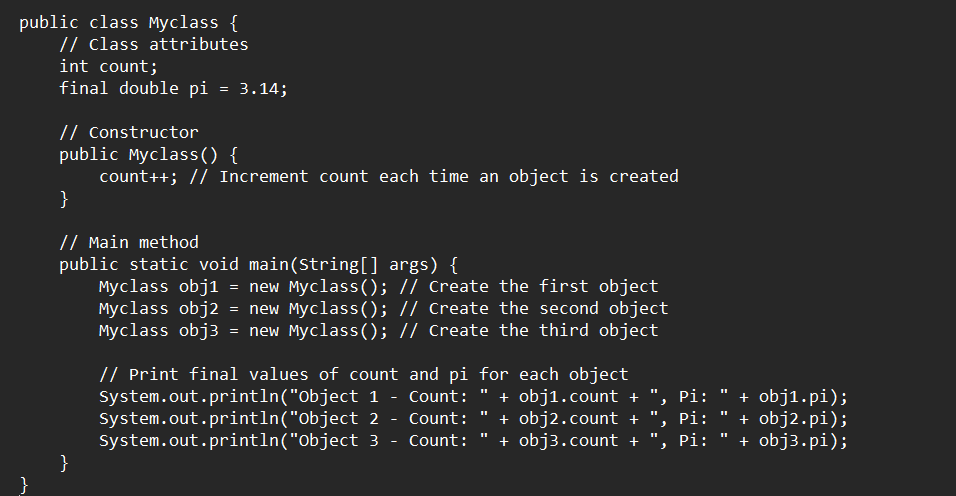
**PROGRAM-2:**

**AIM:** Create a java program with class name Myclass with a starting variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of that class. 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. Create three objects.

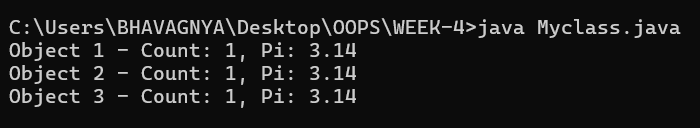
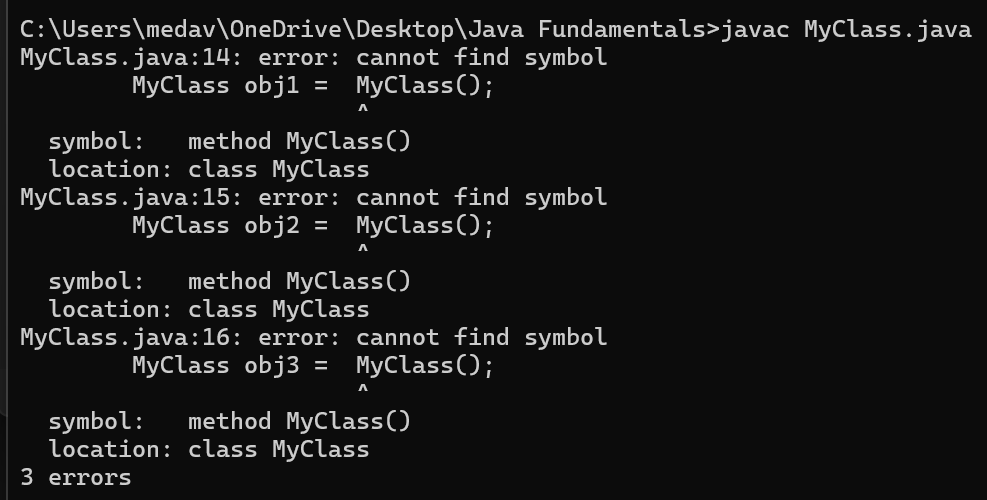
**CLASS DIAGRAM:**

|  |
| --- |
| Book |
| - count: int  - pi: double |
| + displayCount() : void |

**CODE:**

**/**

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1. | Missed; after creating object | Corrected by placing; after every object |
| 2. | } closing is missing | Corrected by closing the code with } |

**IMPORTANT POINTS:**

1. We must declare the initial value of the variable before declaring the final one.

2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

**WEEK-05**

**PROGRAM-1:**

**AIM**: Create a calculator using the operations including addition, subtraction, multiplication and division using Multilevel Inheritance and display the desired output**.**

**CLASS DIAGRAM:**

**MULTILEVEL INHERITANNCE**

|  |
| --- |
| Calculator |
| - num1: int  - num2: int |
| + add(): int  + subtract(): int |

|  |
| --- |
| Calculator1 |
| - num1: int  - num2: int |
| + multiply(): int |

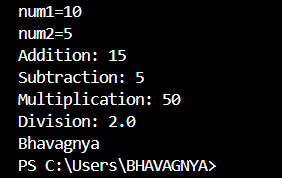
|  |
| --- |
| Calculator2 |
| - num1: int  - num2: int |
| + divide(): int |

**CODE:**

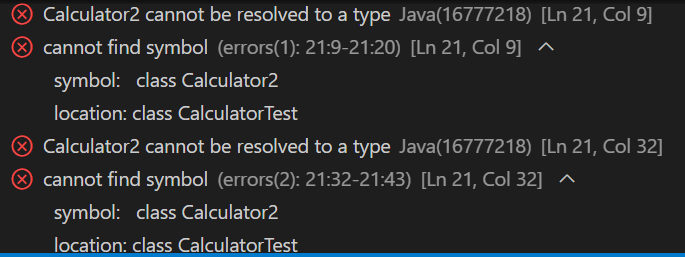
****

**OUTPUT:**

**POSITIVE CASE:**



**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.No | Errors | Rectification |
| 1) | Given wrong class name while object is created | Rectifying the given wrong class name |

**IMPORTANT POINTS**

**1. Multilevel Inheritance:**

* Multilevel inheritance occurs when a class is derived from another class, and then a third class is derived from the second class, forming a chain.

**2. Code Reusability:**

* The add() and subtract() methods are defined once in the Calculator class and can be reused by both the Calculator1 and Calculator2 classes without rewriting the code.
* The multiply() method is defined in Calculator1 and inherited by Calculator2
* The divide() method is added to the Calculator2 class.
* This shows how inheritance helps to reduce redundant code and increases reusability.

**3. Polymorphism (Use of Parent Class Reference):**

* The code demonstrates polymorphism (though not explicitly in the form of method overriding). Even though the object calc is of the class Calculator2, it can call methods from the base class (Calculator) and intermediate class (Calculator1) due to inheritance.

For example, calling calc.add(num1, num2) and calc.subtract(num1, num2) works because Calculator2 inherits from Calculator

**PROGRAM-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 and bikes 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
* Bike 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
* Every class should have a constructor

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, also create an object for car and bike subclassed. Finally display its details**.**

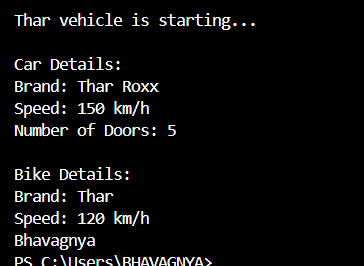
**CLASS DIAGRAM**:

|  |
| --- |
| Vehicle |
| + brand: String  + speed: int |
| + Vehicle(brand: String, speed: int)  + start(): void  + displayDetails(): void |

|  |
| --- |
| Car |
| +numberOfDoors: int |
| +Car(brand:String, speed: int, numberOfDoors: int)  + displayDetails(): void |

|  |
| --- |
| Bike |
| + hasGears: boolean |
| + Bike(brand: String, speed: int, hasGears: boolean) +displayDetails(): void |

## 



* 1. The OOP (Object-Oriented Programming) concepts used in the program are:
* **Inheritance**:
  1. Inheritance is a key concept here, as the Car, Bike, and Truck classes will all extend a common base class, Vehicle
  2. **Usefulness**: Inheritance allows common properties (like brand and speed) to be defined in the Vehicle class and inherited by the Car, Bike, and Truck classes. This avoids code duplication and allows new vehicle types to easily be added in the future without having to repeat common code.
* **Encapsulation**:
  1. Encapsulation is achieved by bundling the data (attributes such as brand, speed, etc.) and the behavior (methods like start(), displayDetails()) together into a single unit (the Vehicle class and its subclasses).
  2. **Usefulness**: It helps to hide internal details and protect the integrity of data by only exposing the necessary methods for interaction.
* **Polymorphism**:
  1. Polymorphism allows us to treat objects of different classes (Car, Bike, Truck) uniformly as objects of type Vehicle. Each subclass can override or extend methods like displayDetails() to provide its own behavior.
  2. **Usefulness**: This enables the program to handle different vehicle types generically while allowing specific behavior for each vehicle

**2 & 3)**

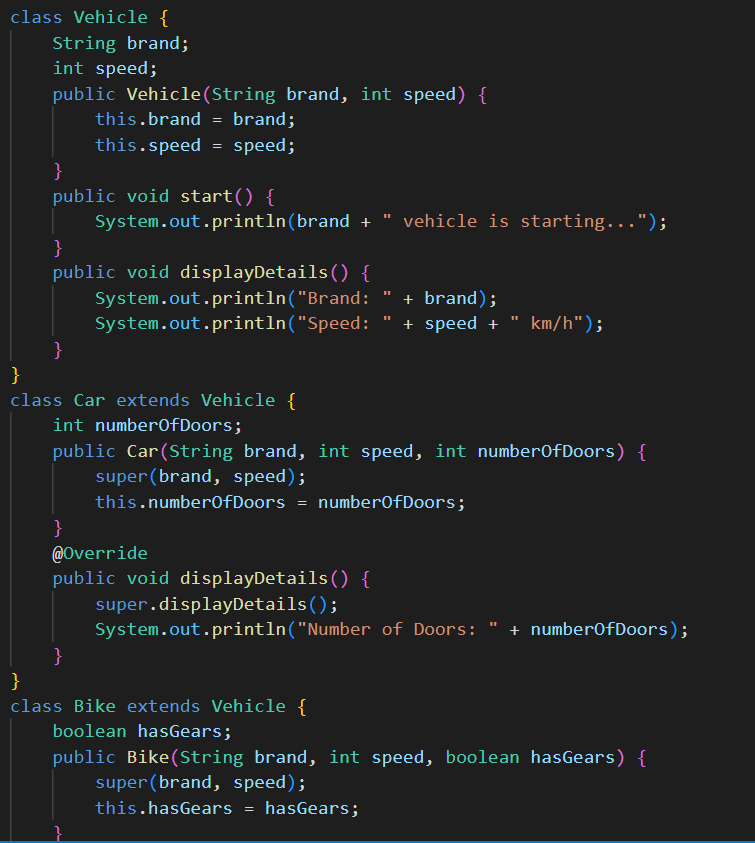
**CLASS DIAGRAM**:

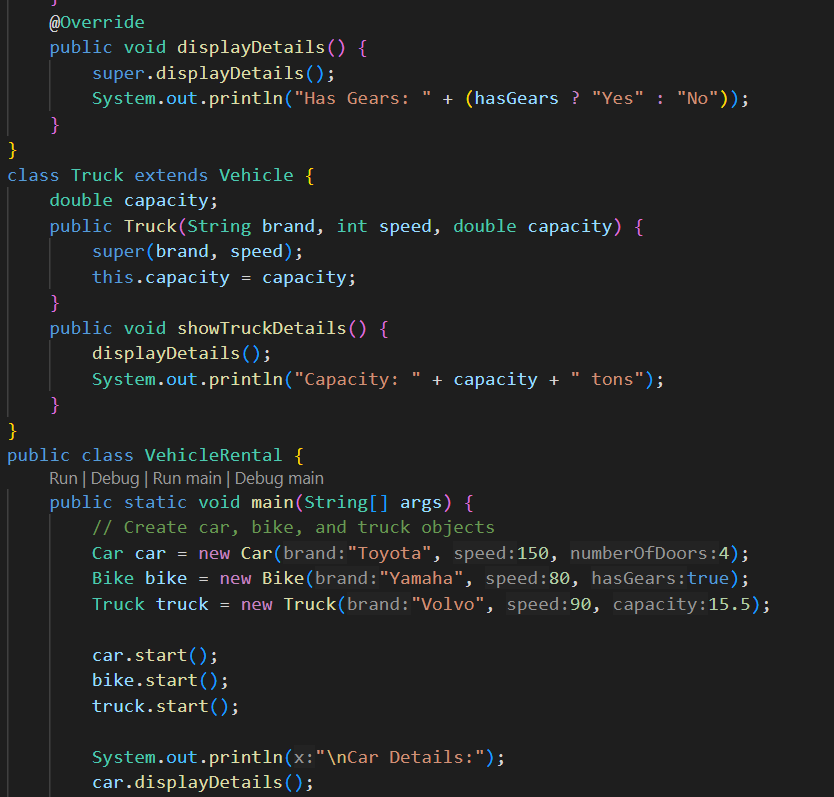
|  |
| --- |
| z Vehicle |
| + brand: String  + speed: int |
| + Vehicle(brand: String, speed: int)  + start(): void  + displayDetails(): void |

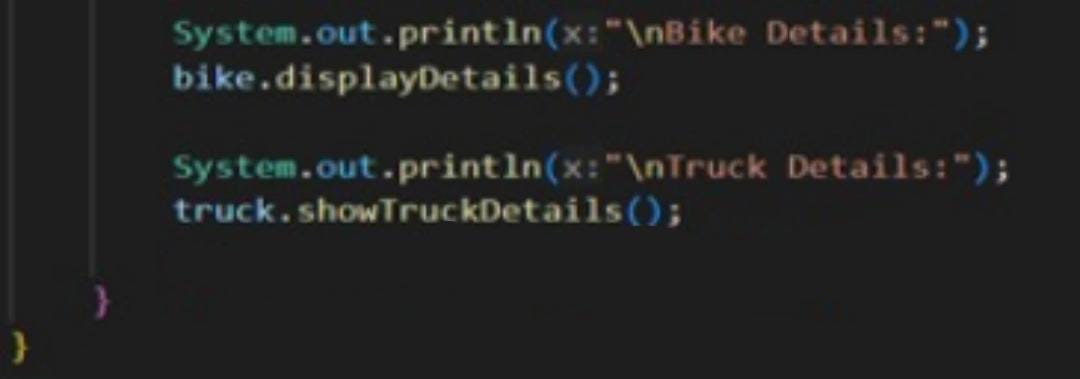
|  |
| --- |
| Bike |
| + hasGears: boolean |
| + Bike(brand: String, speed: int, hasGears: boolean) +displayDetails(): void |

|  |
| --- |
| Truck |
| + Capacity : double |
| +Truck(brand: String, speed: int, Capacity: double)  +showTruckDetails(): void |

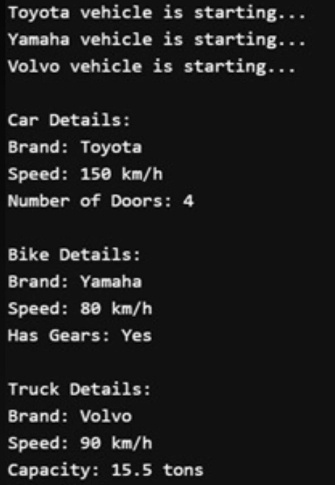
|  |
| --- |
| Car |
| +numberOfDoors: int |
| +Car(brand:String, speed: int, numberOfDoors: int)  + displayDetails(): void |







**OUTPUT:**



**WEEK-6**

**PROGRAM-1:-**

**AIM:** Write a java program to create a Vehicle class with a method displayInfo(). Override this method in the Car subclass to provide specific information about a car.

**CLASS DIAGRAM:**

|  |
| --- |
| **vehicle** |
| **+displayInfo()** |

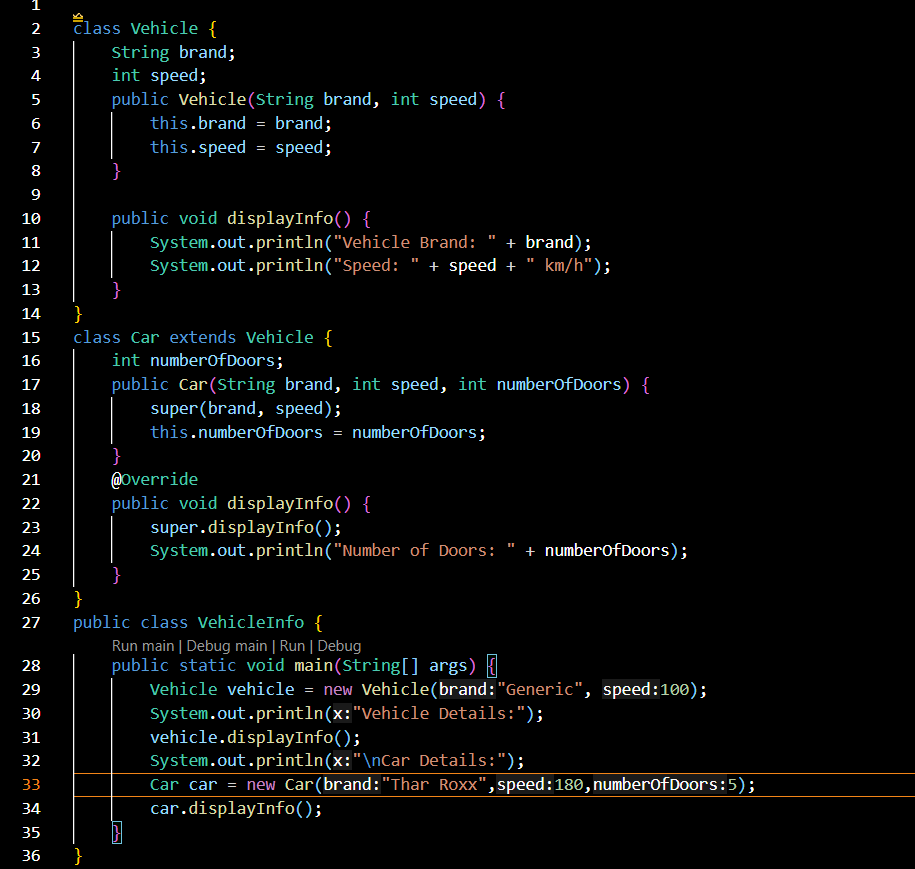
**▲**

**│ (inherits)**

**│**

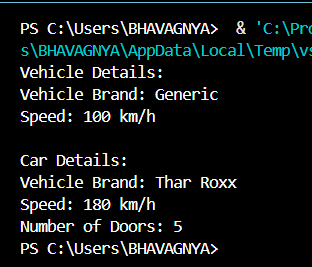
|  |
| --- |
| **car** |
| **+displayInfo()** |

**CODE:**

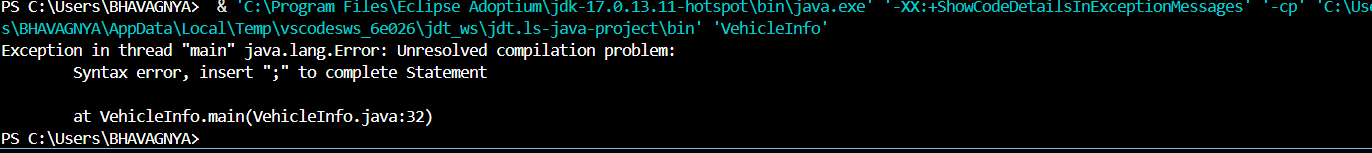
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Error rectification** |
| 1. | error: class Main is public, should be declared in a file named Main.java  public class Main { | Save the file name as the name of the main class |

**IMPORTANT POINTS:**

. We use the concept of method overriding where the names of the methods in the different classes. The method of the parent class is overridden by the method of the child class

**PROGRAM-2:**

**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 student's percentage in their previous qualification.

UG Admissions require a minimum of 60%.

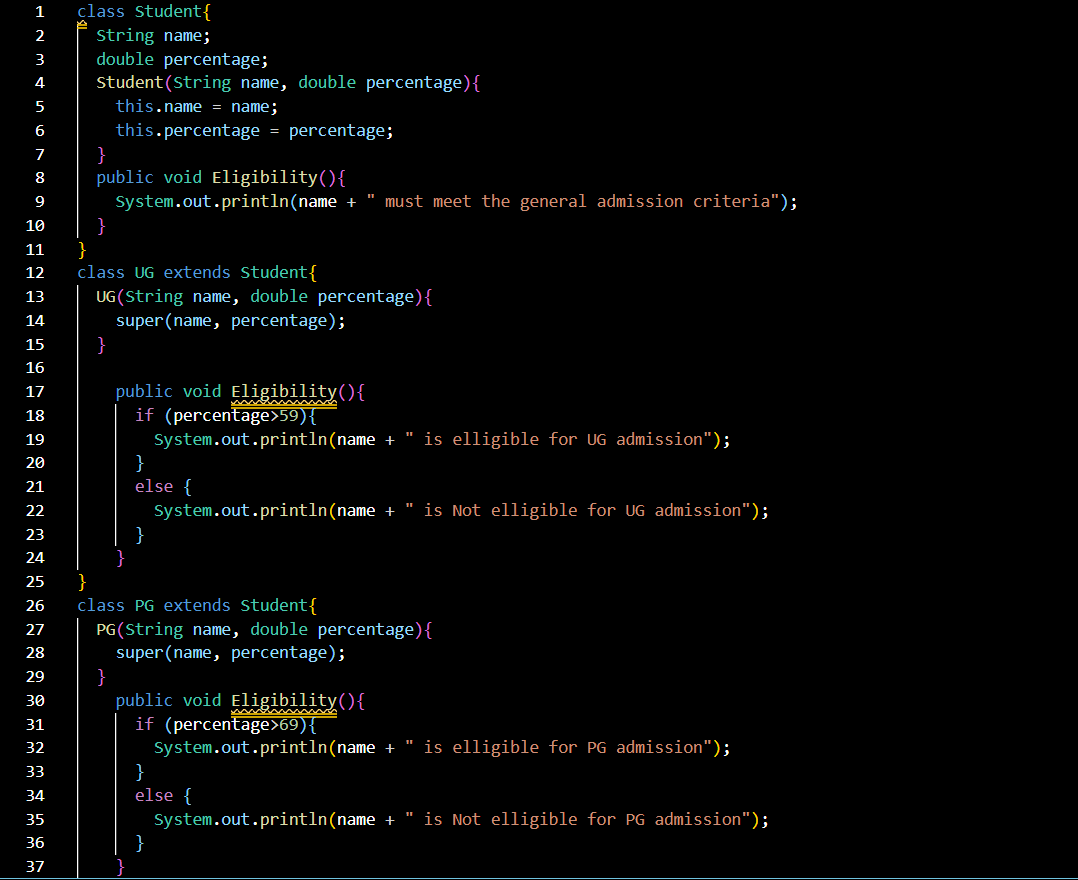
PG Admissions require a minimum of 70%.

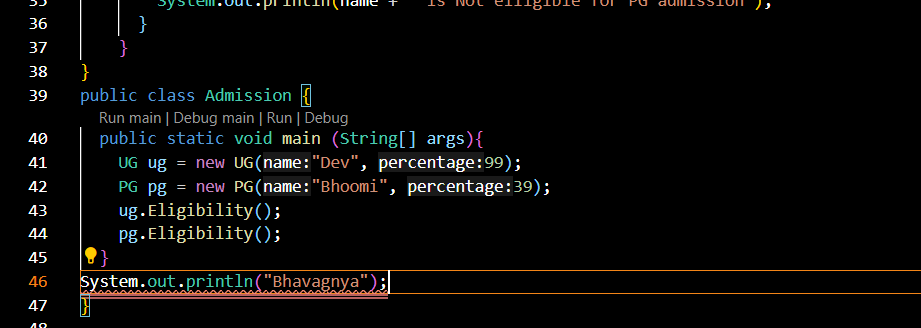
**CLASS DIAGRAM:**

|  |
| --- |
| **UG** |
| UG(String name, double percentage) |

|  |
| --- |
| **Student** |
| + name : String  + percentage : double |
| + Student(String name, double percentage): void  + Eligibility(): void |

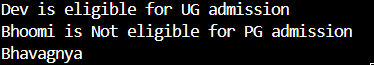
**CODE:**

****

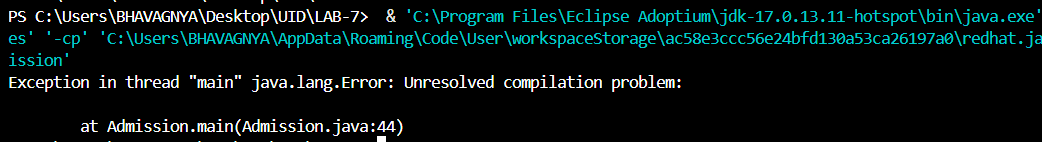
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Error** | **Error rectification** |
| **1.** | error: ';' expected System.out.println(name + " is elligible for UG admission") | Add a ‘;’ after the print statement**.** |

**IMPORTANT POINTS:**

1. The variables once declared in the super class need not be declared twice in any of the sub classes.

2. super keyword is used in sub classes to access the methods of super classes, they are basically the reverse of overriding.

**PROGRAM-3:  
AIM:** Create a calculator with overloaded methods to perform addition:

i) Add two integers

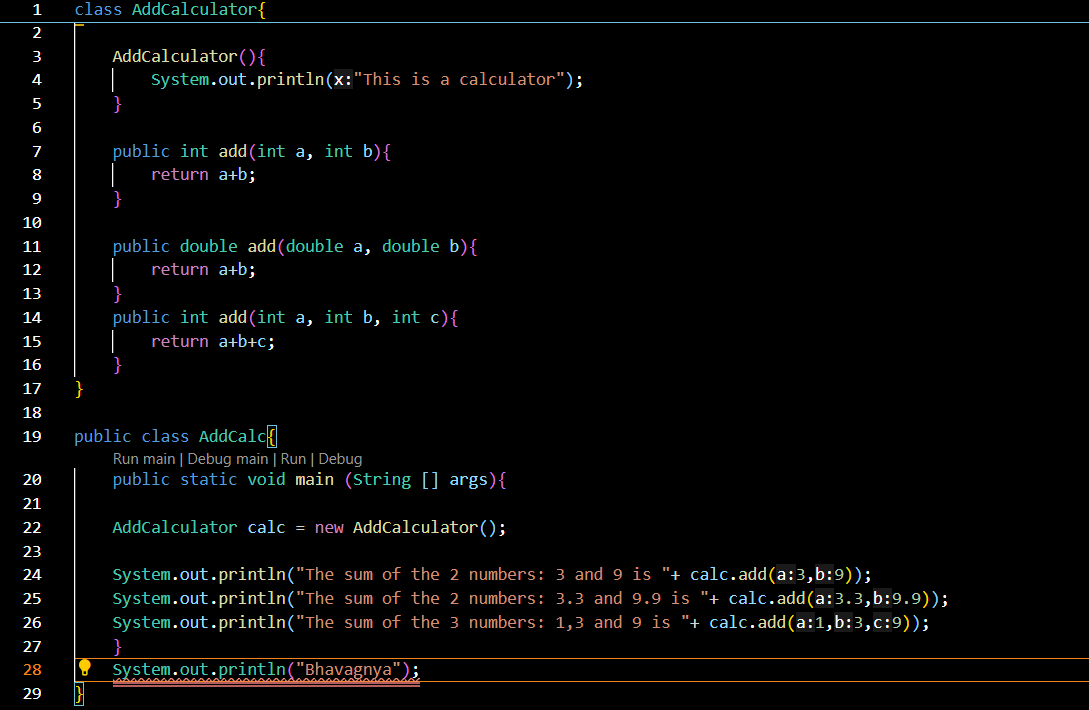
ii) Add two doubles

iii) Add three integers

**CLASS DIAGRAM:**

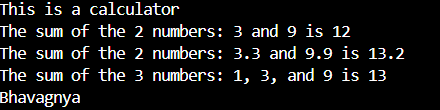
|  |
| --- |
| **AddCalculator** |
| + AddCalculator ()  + add(int a, int b) : int  + add(double a, double b) : double  + add(int a, int b, int c) : int |

**CODE:**

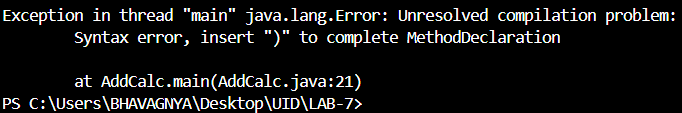
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Error rectification** |
| 1. | error: Main method not defined | Add public static void main (String [] args){ |
| 2. | error: ';' return type is not expected | Change return type from int to double in its case. |

**IMPORTANT POINTS:**

1. We use the concept of method overloading where the names of the methods in the same class are same but the parameters are given different.

**PROGRAM-4:**

**AIM:** Create a Shape class with a method calculateArea() that is overloaded for different shapes (e.g., square, rectangle).

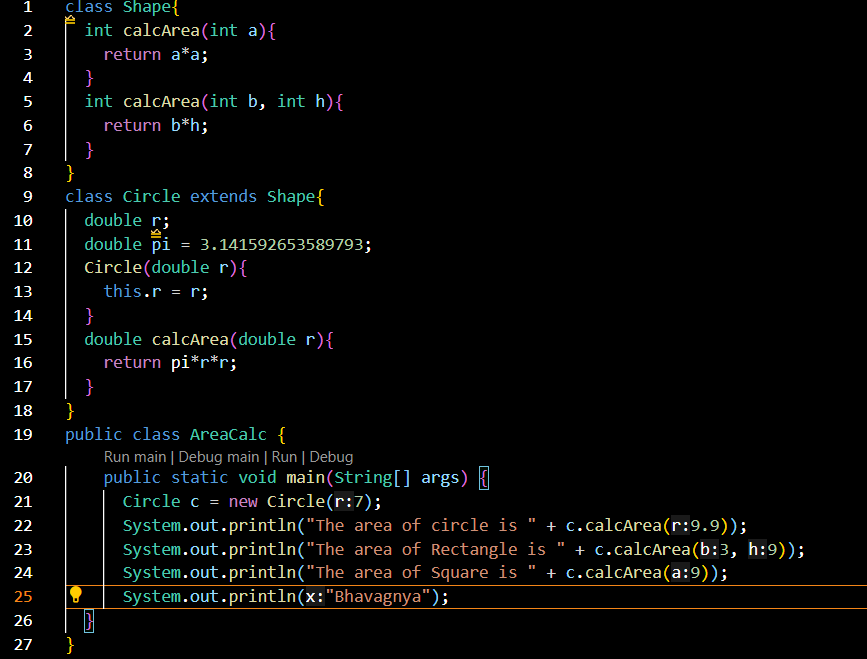
Then create a subclass Circle that overrides the calculateArea() method for a circle.

**CLASS DIAGRAM:**

|  |
| --- |
| **Circle** |
| + r : double  + pi : double |
| + calcArea(int r) : double |

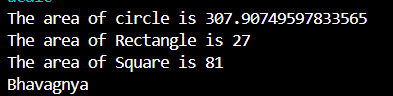
|  |
| --- |
| **Shape** |
| + calcArea(int a) : int  + calcArea(int b, int h) : int |

**CODE:**

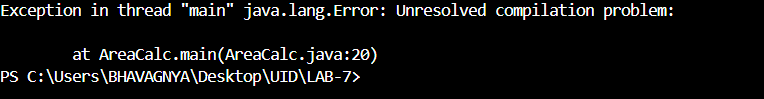
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

**  
ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error** | **Error rectification** |
| 1. | error: invalid method declaration; return type required  calcArea(int a){ | Enter the return type as per required. Here it is int |
| 32. | error: incompatible types: possible lossy conversion from double to int  return pi\*r\*r; | For calculating area of circle, we need to give return type double. |

**IMPORTANT POINTS:**

1. We use the concept of method overloading to calculate the area of square and rectangle in the parent class Shape.

2 we use method overriding in the child class Circle to calculate it’s area.

**WEEK-07**

**PROGRAM-1:**

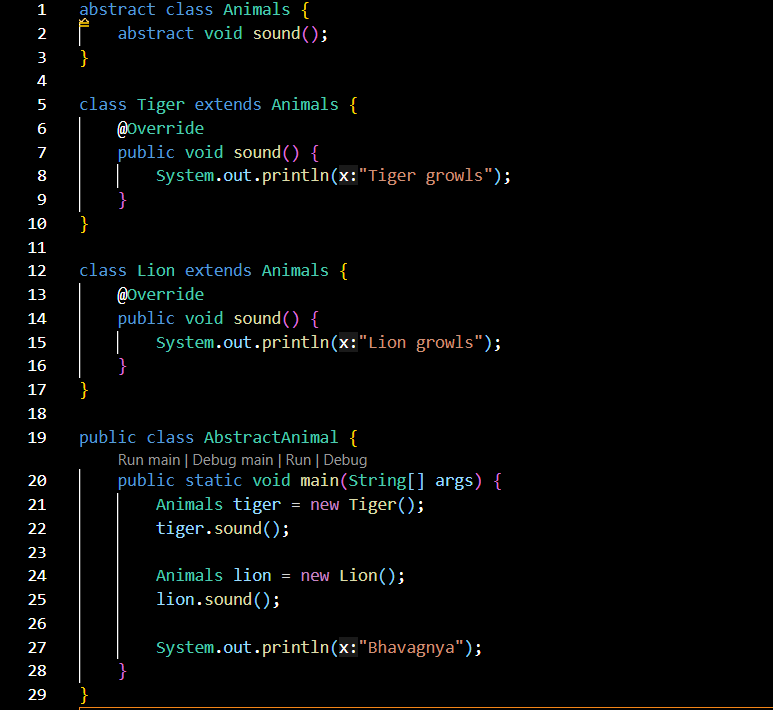
**AIM:** Write a Java program to create an abstract class Animal with an abstract method called sound(). Create subclasses Lion & Tiger that extend Animal class and implement the sound() method to make a specific sound for each animal.

CLASS DIAGRAM:

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

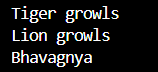
|  |  |
| --- | --- |
| **Lion** | **Tiger** |
| **+sound():void** | **+sound():void** |

**CODE:**

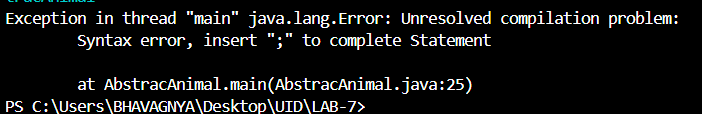
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1. | Missed; after creating object | Corrected by placing; after every object |
| 2. | } closing is missing | Corrected by closing the code with } |

**IMPORTANT POINTS:**

Marked with <<abstract>> to indicate it's an abstract class.

Contains an abstract method sound()

Both extend Animal and implement the sound() method.

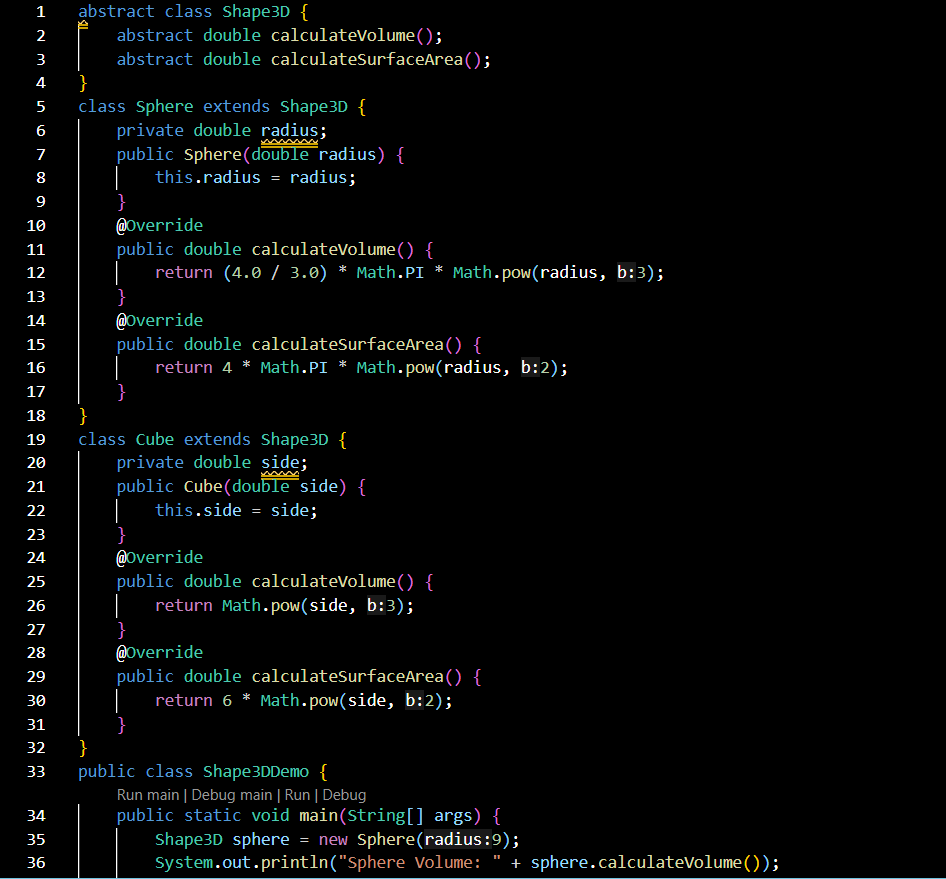
**PROGRAM-2:  
AIM:** Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate volume and surface area of each shape. **CLASS DIAGRAM:**

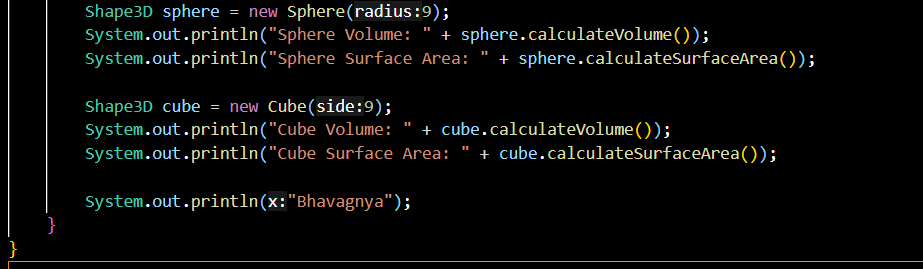
Shape3D

+calculatevolume(): double

+calculatesurfacearea(): double

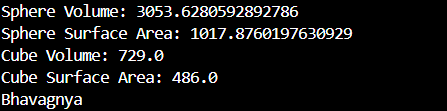
**CODE:**

****

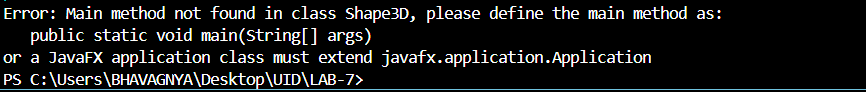
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
|  | ERROR | ERROR RECTIFICATION |
| 1. | Missed; after creating object | Corrected by placing; after every object |
| 2. | } closing is missing | Corrected by closing the code with } |

**IMPORTANT POINTS:**

Declares abstract methods calculateVolume() and calculateSurfaceArea().

Concrete Classes Sphere and Cube

Implement calculateVolume() and calculateSurfaceArea().

Use appropriate mathematical formulas for volume and surface area.

Polymorphism in main Method

Uses Shape3D references to create Sphere and Cube objects

Formulas Used:

Sphere:

Volume = (4/3) \* π \* r³

Surface Area = 4 \* π \* r²

Cube:

Volume = side³

Surface Area = 6 \* side²

**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).

Create concrete subclasses:

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

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

**CLASS DIAGRAM:**

**PatternPrinter**

**- rows: int**

**+displayTitle()**

**+printPattern()**

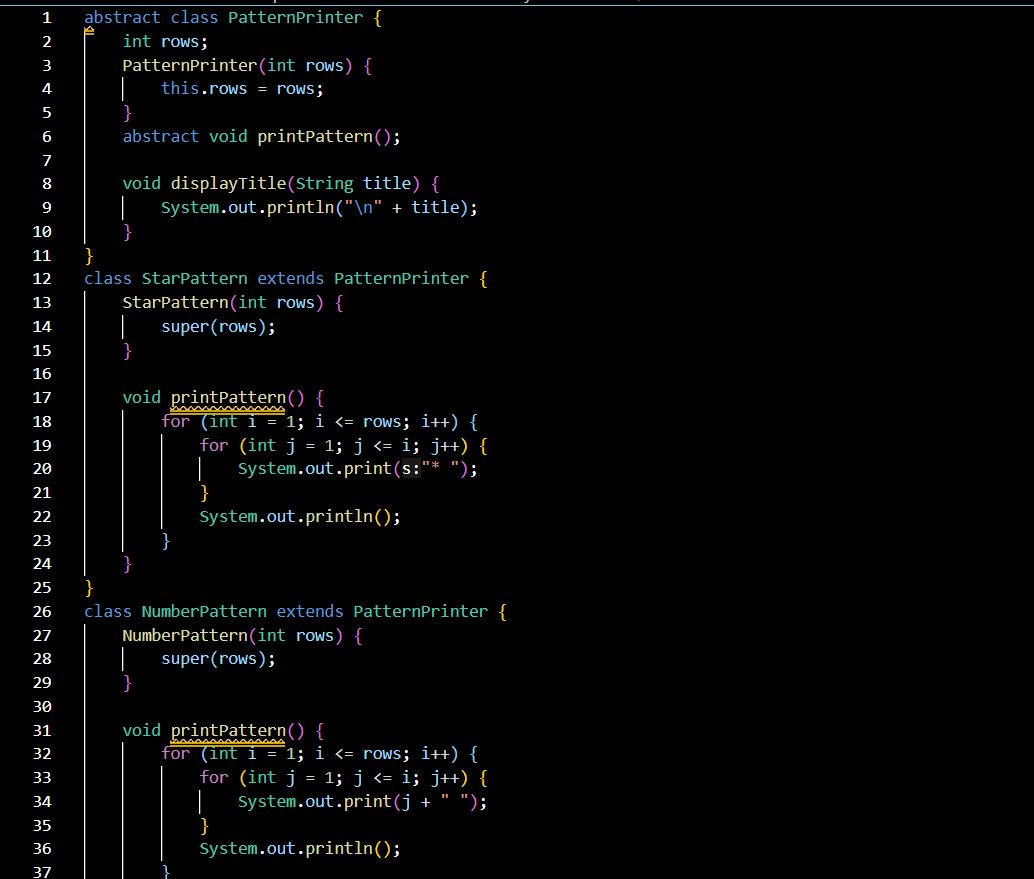
**St**arPattern

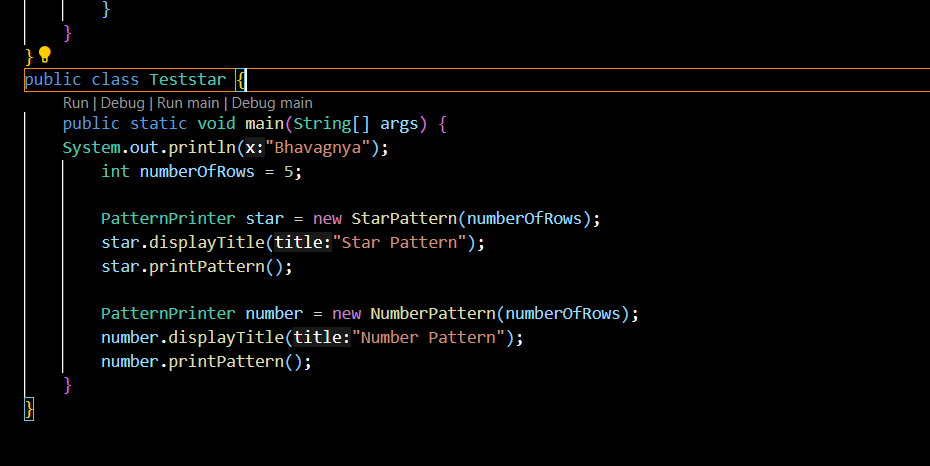
+printPattern()

NumberPattern

+printPattern()

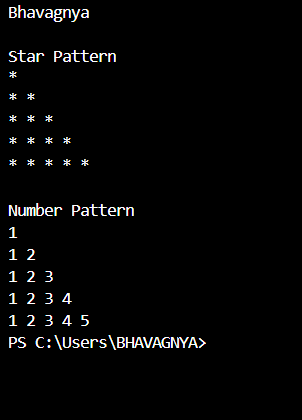
**CODE:**

****

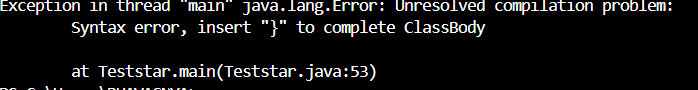
****

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

 **ERROR TABLE:**

|  |  |
| --- | --- |
| CODE ERROR:   1. Class name and file name should match 2. Subclass doesn’t override abstract method | ERROR RECTIFICATION   1. Save file as main.java   2)implement printpattern()in all subclasses |

**IMPORTANT POINTS:**Use abstract classes to enforce a common structure for pattern printing.

PatternPrinter is the abstract class defining the common template.

Subclasses (StarPattern, NumberPattern) provide specific implementations.

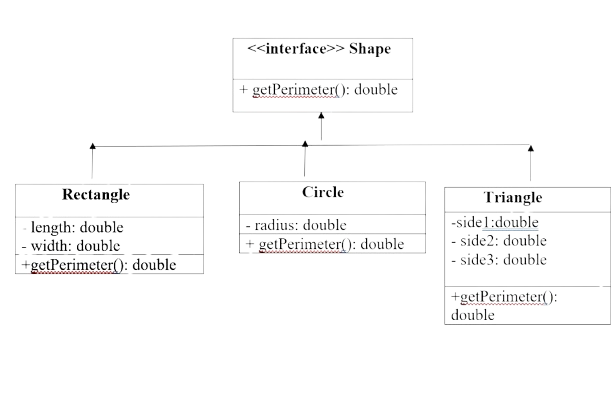
displayTitle() is a concrete method shared by all subclasses

**WEEK-08**

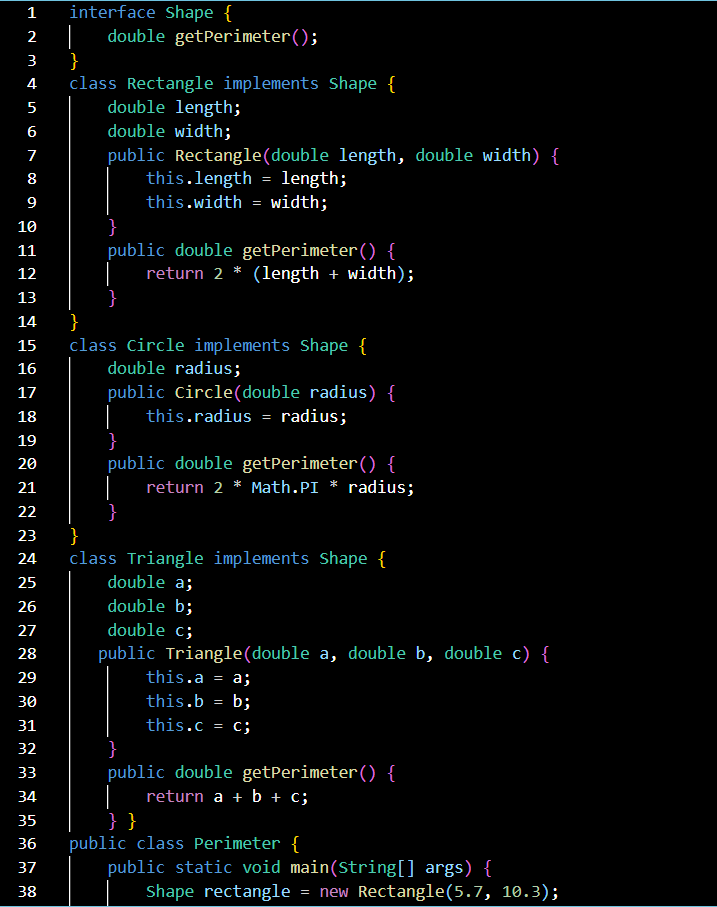
**PROGRAM-1:**

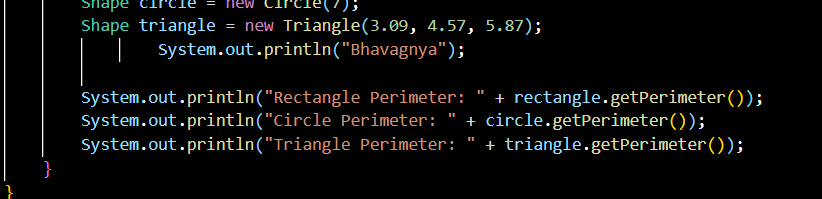
**AIM:** Write a Java program to create an interface Shape with the getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes**.**

**CLASS DIAGRAM:**

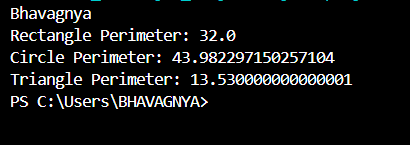
****

**CODE:**

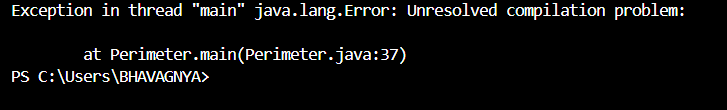
****

**  
OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Error Found** | **Error Rectified** |
| **System.out.println("The perimeter of the circle is: "+perimeter)** | **System.out.println("The perimeter of the circle is: "+perimeter);** |

**IMPORTANT POINTS:**

1. No Implementation: Interfaces cannot contain method implementations (in many languages), only method signatures.

2. Multiple Inheritance: A class can implement multiple interfaces, providing a way to achieve multiple inheritance.

3. Abstract Methods: All methods in an interface are implicitly abstract (in many languages) and must be implemented by the implementing class.

4. No Constructors: Interfaces cannot have constructors because they cannot be instantiated directly.

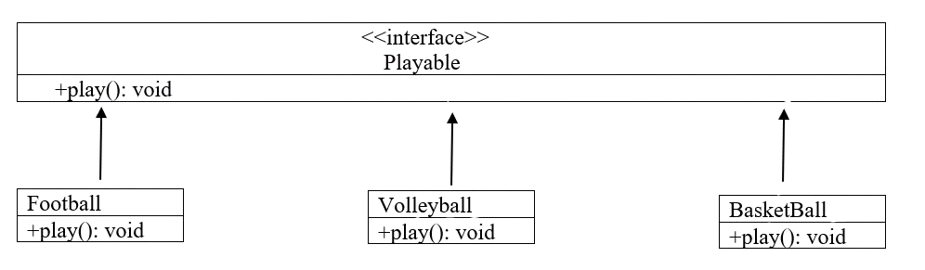
**PROGRAM-2**

**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.

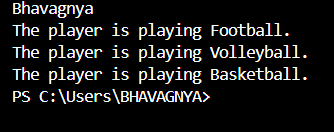
**CODE:**

**CLASS DIAGRAM:**

****



**OUTPUT:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| VolleyBall = new VolleyBall();  ^ 1 error | VolleyBall v1 = new VolleyBall(); |

**IMPORTANT POINTS:**

* The program demonstrates the use of interfaces in Java through the Playable interface, which defines a single method play() that must be implemented by any class that uses it.
* Three different sports classes (FootBall, VolleyBall, and BasketBall) implement the Playable interface, each providing its own version of the play() method, showcasing polymorphism.