**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name:T.Harini**

**Verified By Roll No: 24337**

**WEEK-1**

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

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

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

**Checking of JDK Version:**

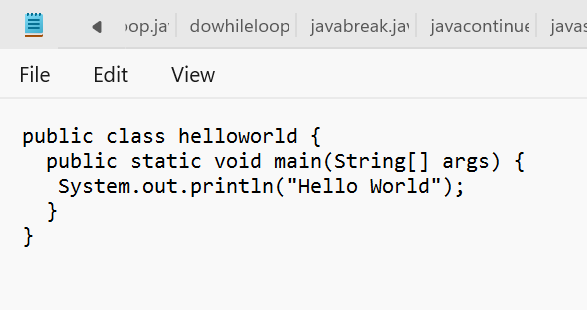
1. **Open Command Prompt:**
   * Press win+R, type cmd, and press Enter.
2. **Check Version:**
   * Type java --version and press Enter.
   * Type javac --version and press Enter.



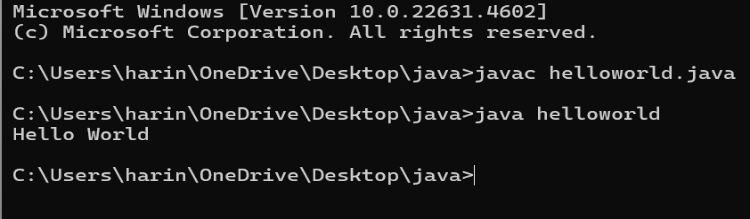
**PROGRAM-2:**

AIM: Write a java program to print the message “Welcome to java programming.”

CODE:



**Output:**



**ERRORS:**  None found

**IMPORTANT POINTS:**

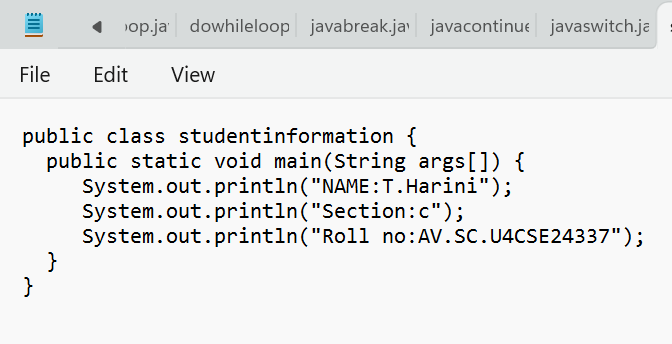
* + 1. Make sure that the file and the class name are the same to avoid confusion.

**PROGRAM-3:**

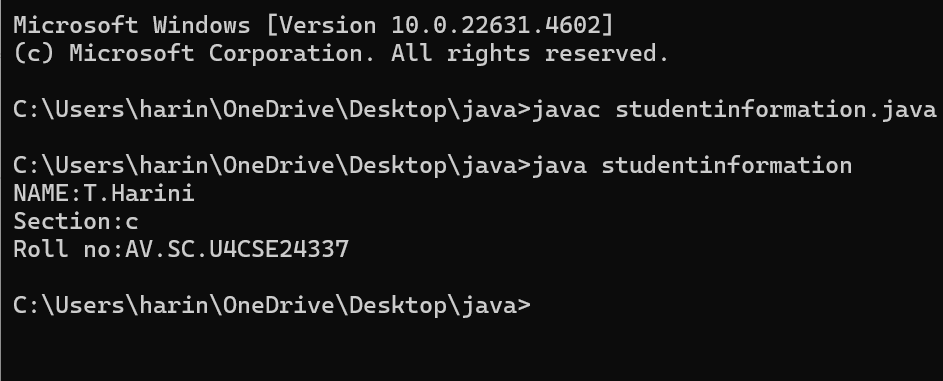
AIM: Write a java program that prints Name, Roll no, Section of a student.

CODE:

**STUDENT DETAILS:**



**OUTPUT:**



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. writing small “S” in place of “S” In system.out.println() 2. not giving strings to the name and section | 1. code is rectified by keeping capital “S” 2. Giving strings to name and section |

**IMPORTANT POINTS:**

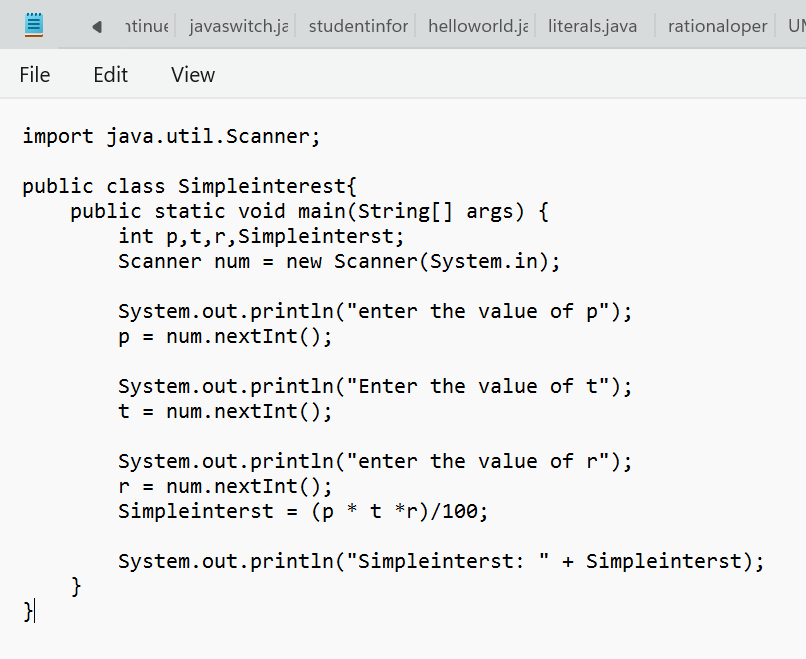
1. When printing the statements, everything should be inside double quotes.

**WEEK 2:**

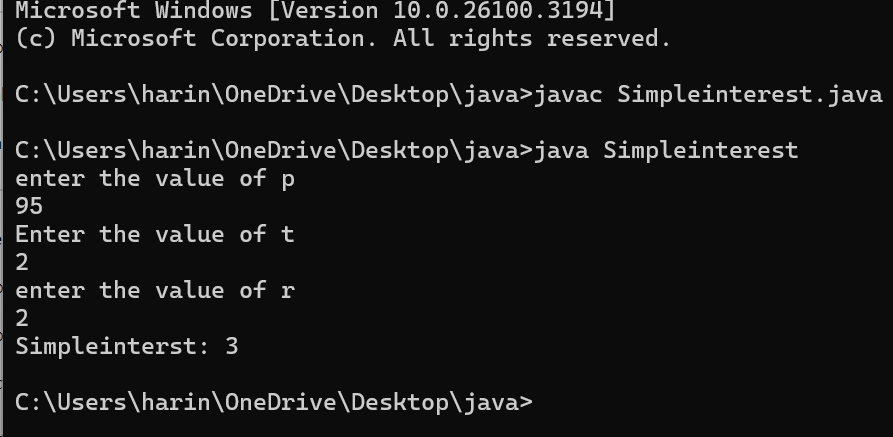
**Program 1:**

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

Code:



Output:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Giving space between next and Double. 2. Not giving parenthesis after closing the input. | 1. Should not give space between next and Double. 2. We must put parenthesis after closing the input. |

**IMPORTANT POINTS**:

1. Simple interest formula is: (p\*t\*r)/100, where:

P: Principal amount

R: Rate of interest

T: Time period

1. The data type double indicates the floating points in the integers.
2. The line “import java.util.Scanner” indicates:

Import: tells the java compiler that we want to use a specific class or package in your code.

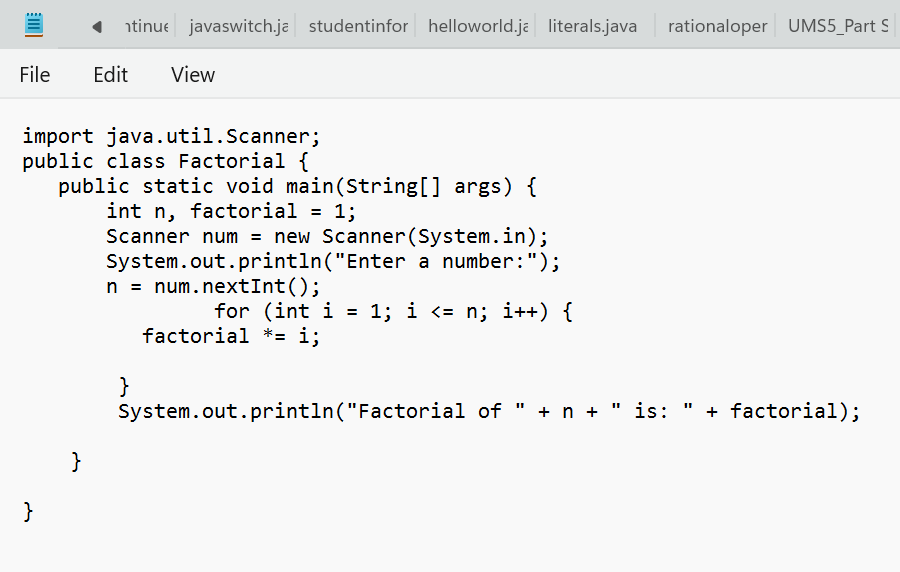
Java.util : This is the package that contains utility classes for Java programming, including the “Scanner” class.

Scanner: this is the class that allows you to read input from the keyboard.

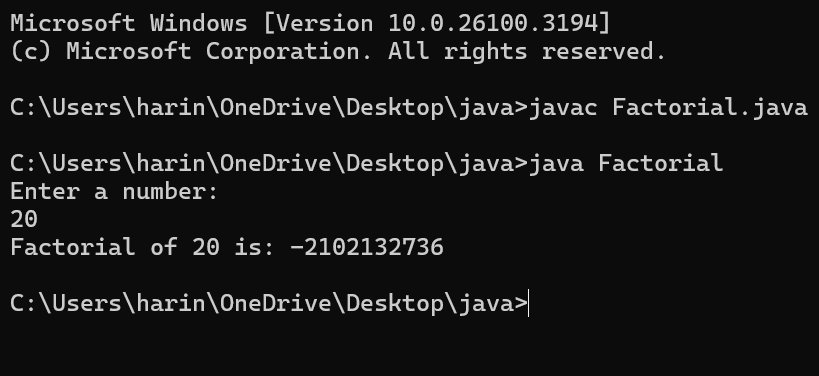
**PROGRAM 2:**

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

CODE:



OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While using for iteration, not giving the conditions correctly. 2. Declaring the data type as double instead of int. | 1. We should give iterative statements correctly. 2. We should give the data type as int for integers. |

IMPORTANT POINTS:

1. While the for loop the data inside the parenthesis indicates the Initial expression

Test expression and

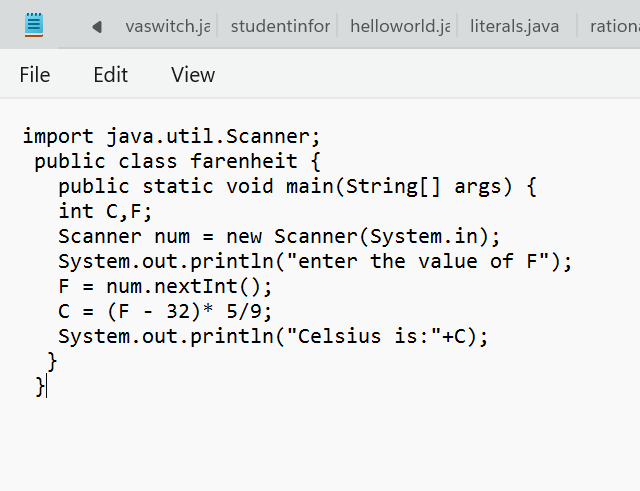
Update expression.

1. Here “factorial\*=I” means factorial = factorial\*I.
2. Here we are using the data type “int” just to calculate the integer values and it doesn’t support floating points.

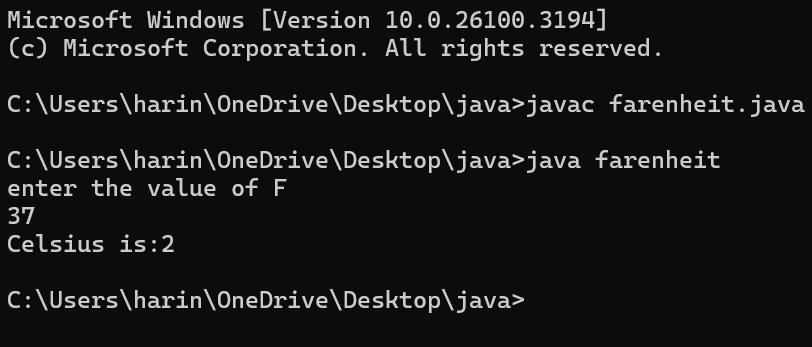
PROGRAM 3:

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

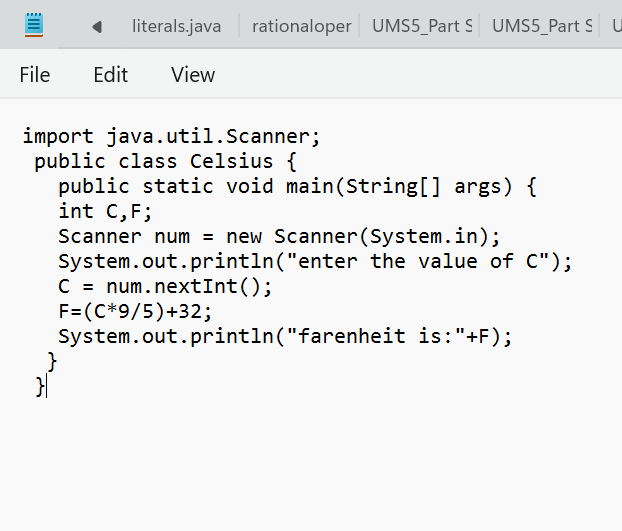
CODE:



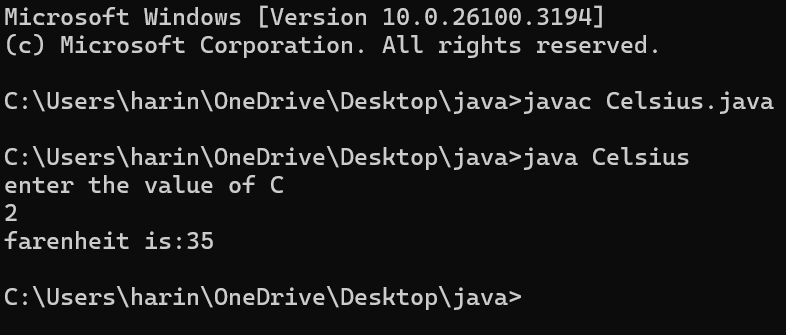
OUTPUT:



CODE:



OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While printing the variable not giving + sign. 2. Not closing the scanner. | 1. We should give correct indentation. 2. Closing the scanner is must. |

IMPORTANT POINTS:

1. The formula to convert a Fahrenheit to Celsius is

Celsius = (Fahrenheit-32)\*5/9

1. The formula to convert a Celsius to Fahrenheit is

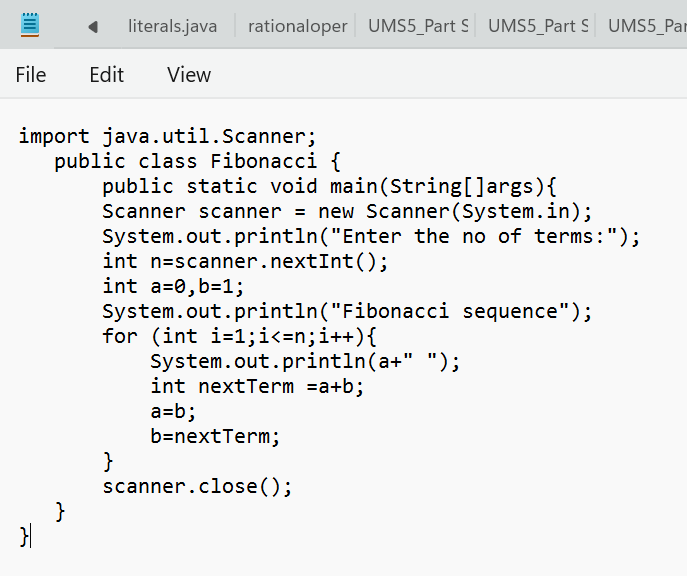
Fahrenheit = (Celsius\*9/5)+32.

1. The line “Scanner input = new Scanner(System.in),” tends to create a new Scanner object named “input” that reads input from the standard input stream (System.in), like keyboard.

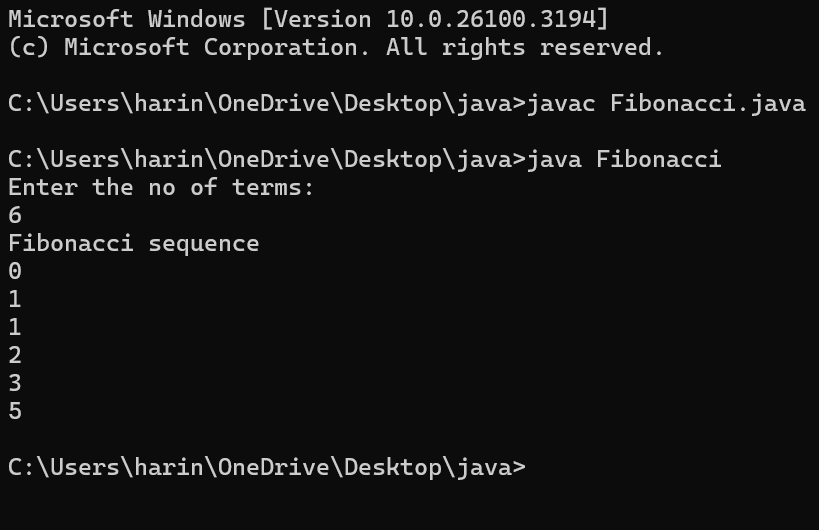
**PROGRAM 4:**

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

CODE:



OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Giving space between next and Double. 2. Not giving parenthesis after closing the input. | 1. Should not give space between next and Double. 2. We must put parenthesis after closing the input. |

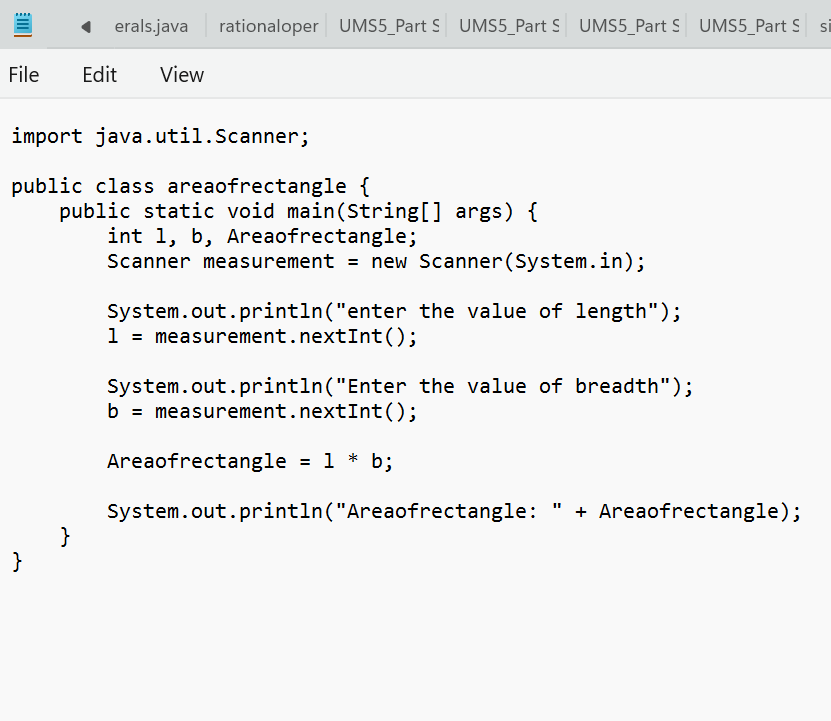
IMPORTANT POINTS:

1. In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.
2. This process is repeated a certain number of times until the conditions are met.

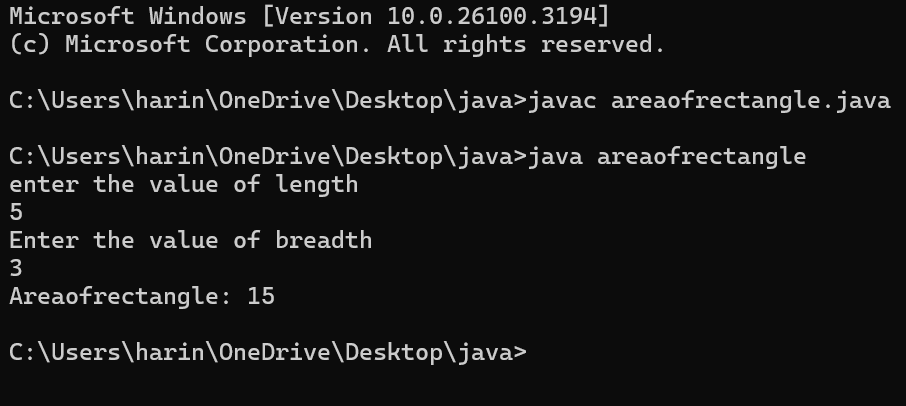
PROGRAM-5:

**AIM**: Write a java program to find the Area of a rectangle, where all the inputs are taken from the user.

CODE:



OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While using for iteration, not giving the conditions correctly. 2. Declaring the data type as double instead of int. | 1. We should give iterative statements correctly. 2. We should give the data type as int for integers. |

IMPORTANT POINTS:

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

L = length of a side of the rectangle,

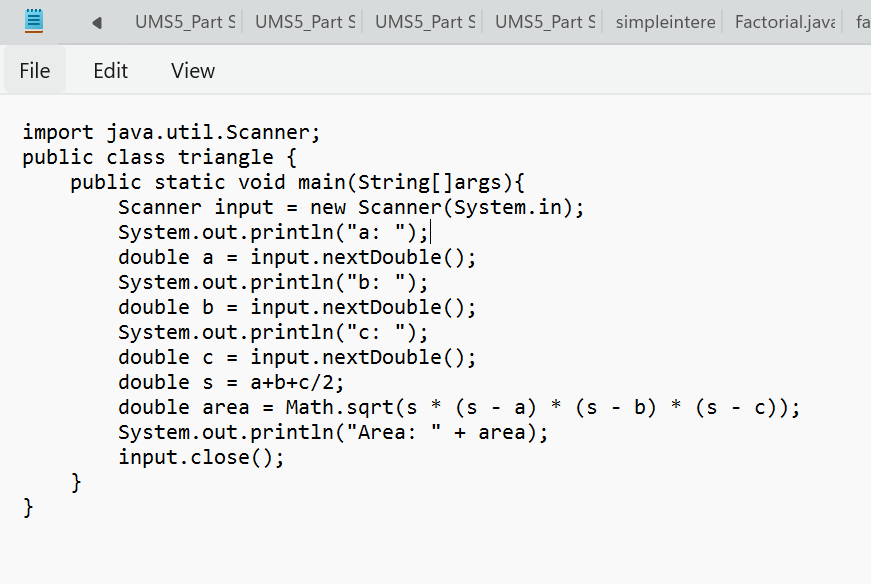
B= breadth of a side of the rectangle.

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

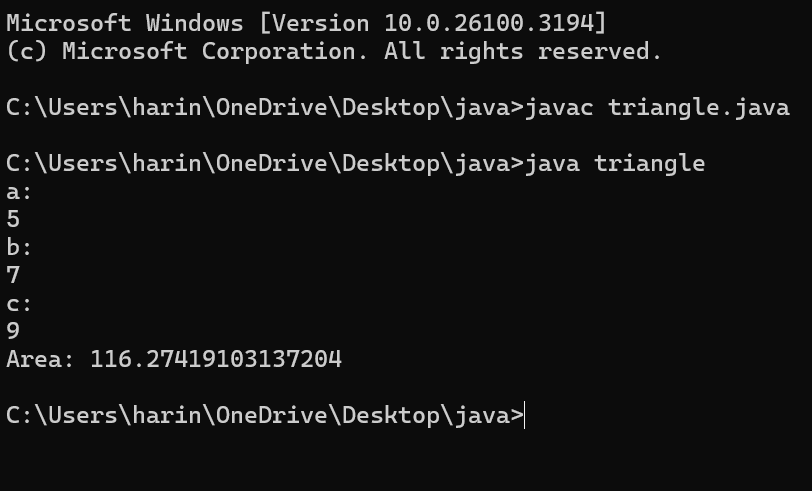
**PROGRAM – 6:**

**AIM**: Write a java program to find the Area of a Triangle using heron’s formula, where all the inputs are taken from the user.

CODE:



OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. While printing the variable not giving + sign. 2. Not closing the scanner. | 1. We should give correct indentation. 2. Closing the scanner is must. |

IMPORTANT POINTS:

1. Here, we’re finding the area of a triangle using heron’s formula.
2. Heron’s formula for finding a triangle is:

S = (a +b +c)/2

Where S is the semi-perimeter of the triangle.

Now the area formula is:

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

**Week 3:**

**AIM:** To create java program with following instructions:

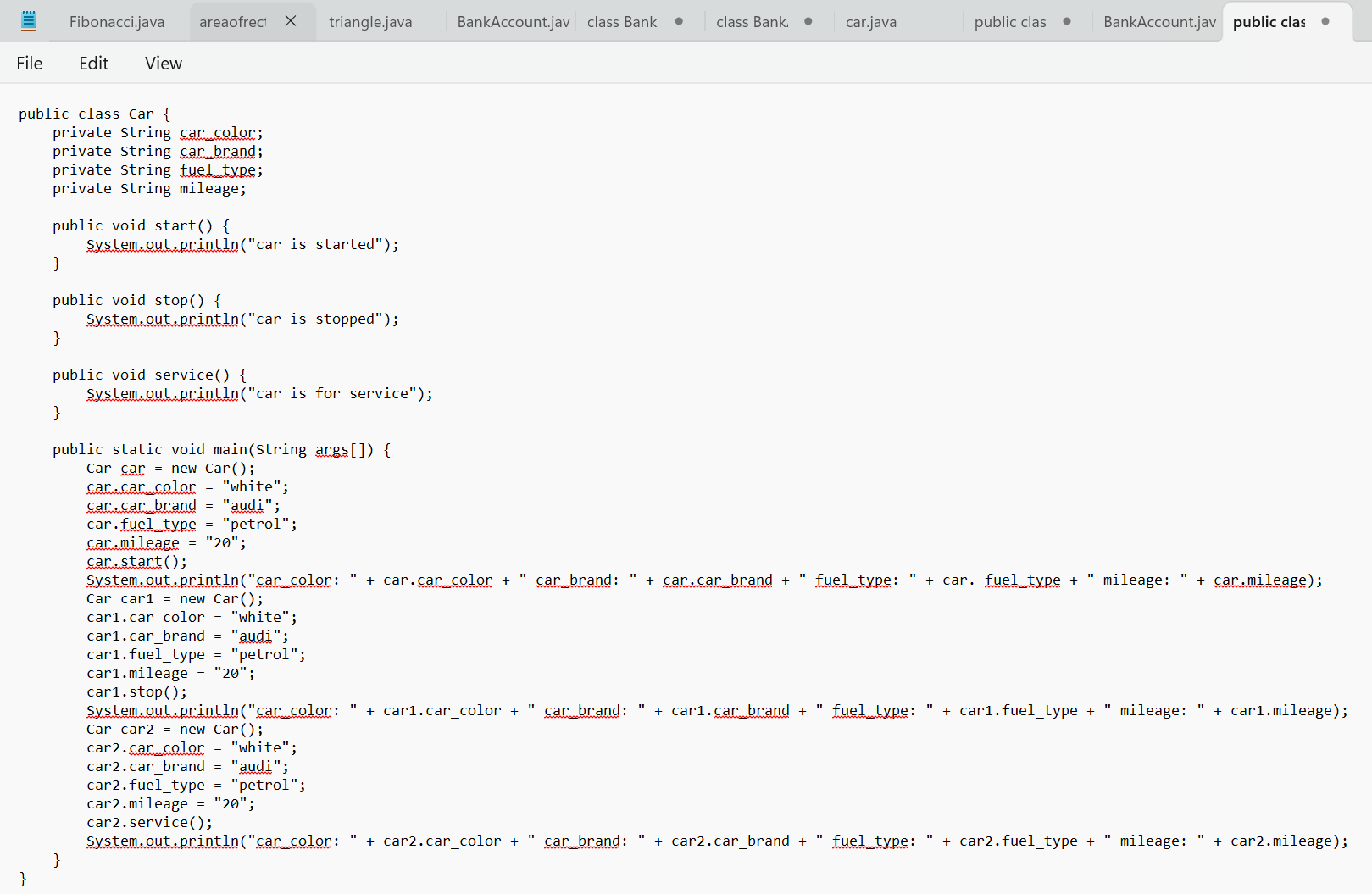
1.Create a class with name Car

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

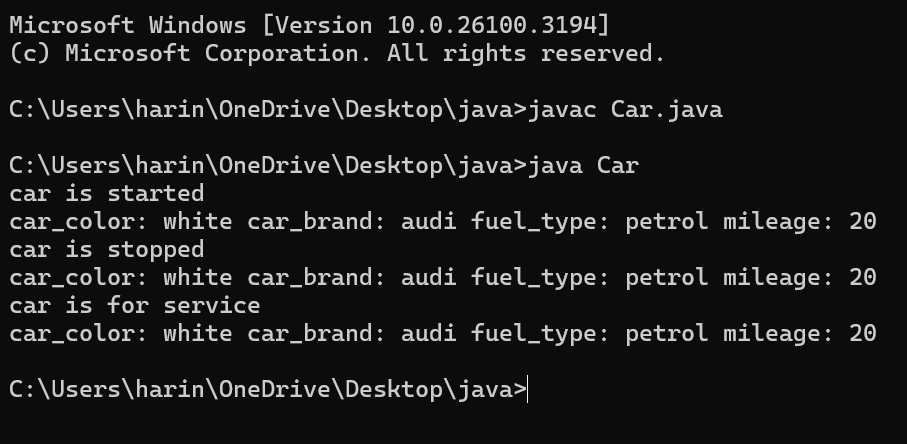
3.Create these methods named start(), stop(),service()

4.Create the objects named car, car1, car2

**CODE:**

****

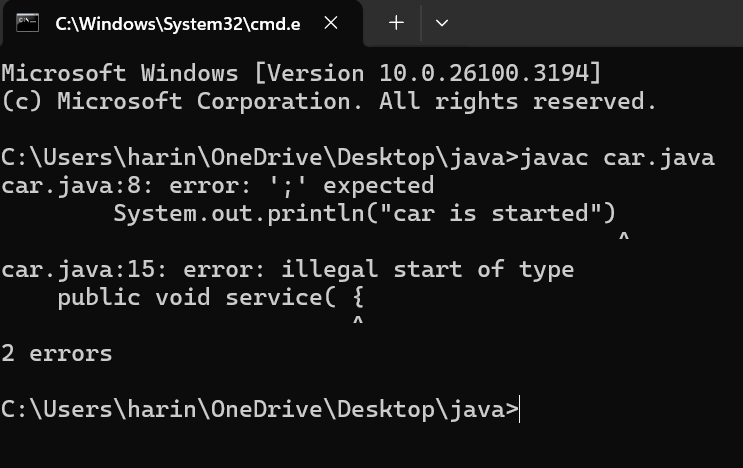
**OUTPUT:**

****

**ERROR TABLE:**

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

NEGATIVE CASE:



IMPORTANT POINTS:

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

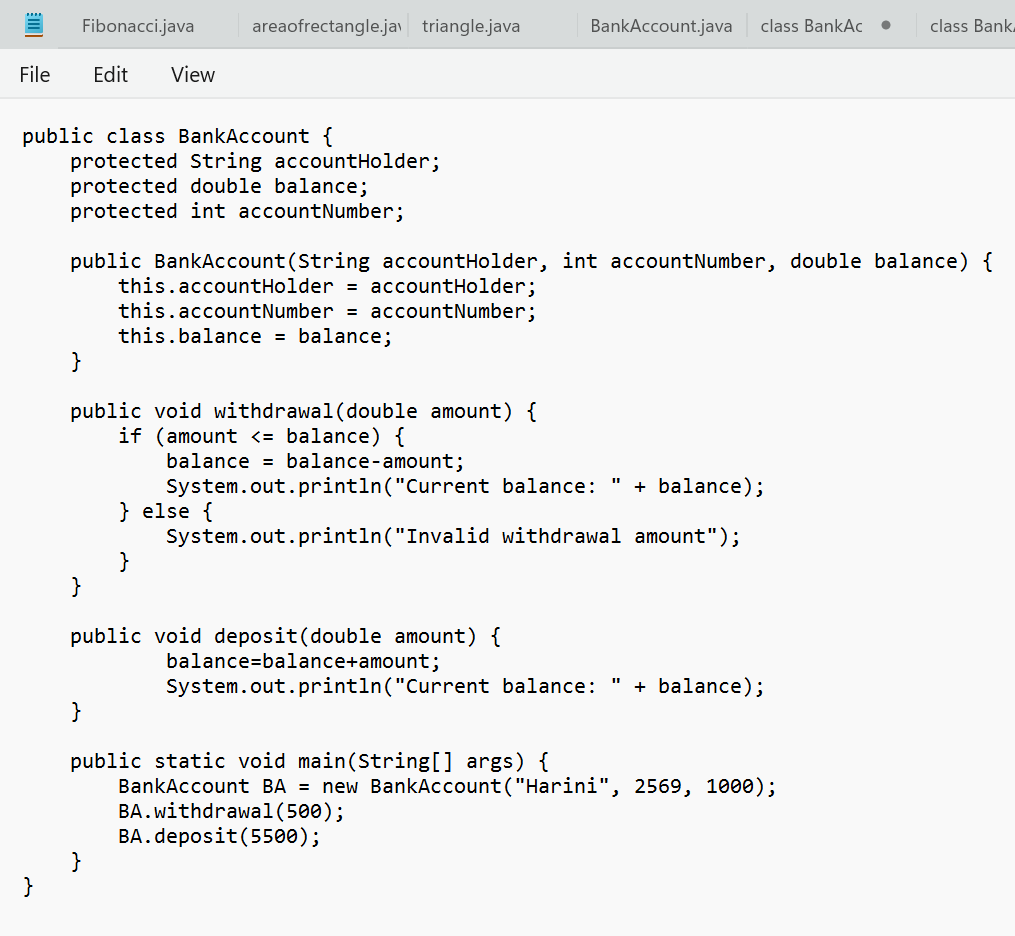
**CLASS DIAGRAM:**

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

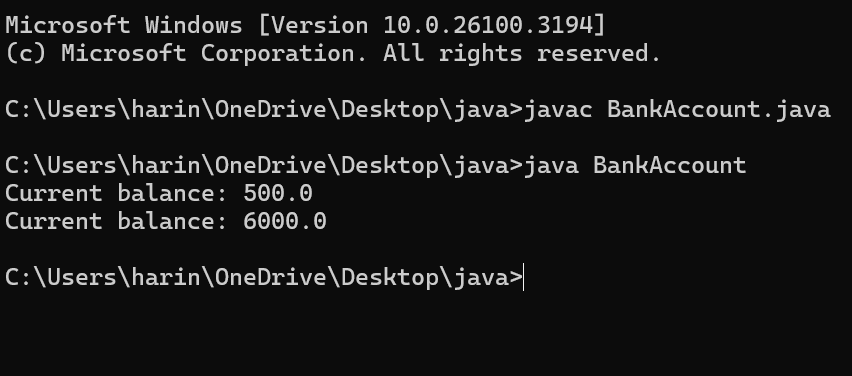
**PROGRAM:2**

**2.AIM:** To create a class Bank Account with methods deposit() and withdraw () . create two subclasses savings account and checking account override the withdraw () method in each subclass to impose different withdrawal limits and fees.

**CODE:**



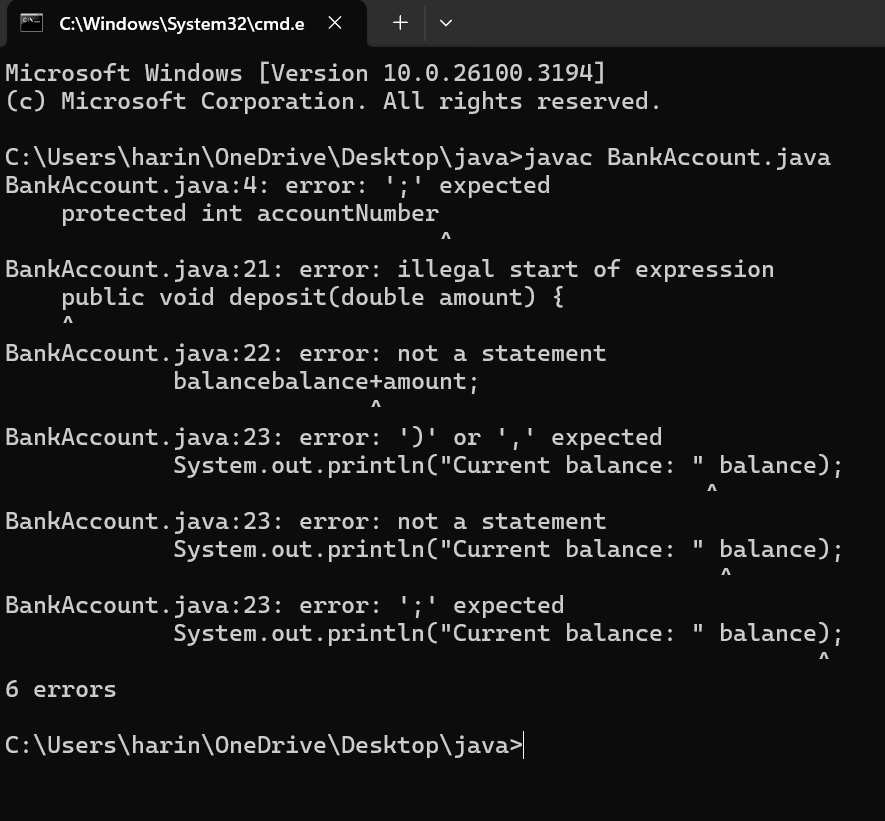
**OUTPUT:**

****

**ERROR TABLE:**

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

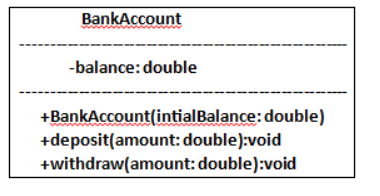
NEGATIVE CASE:



IMPORTANT POINTS:

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

CLASS DIAGRAM:



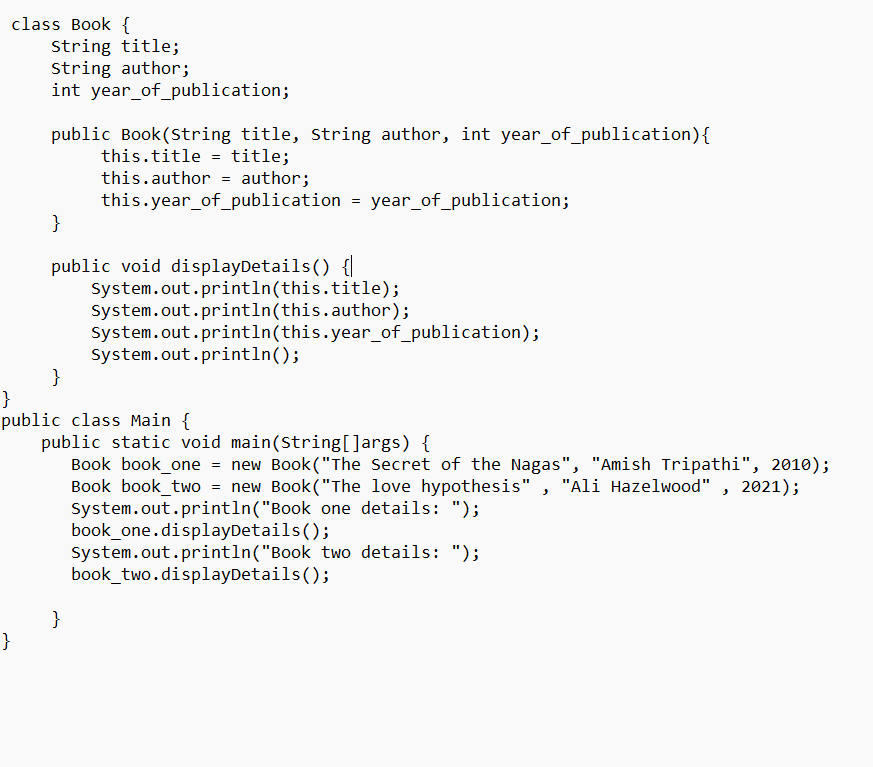
**WEEK -4**

**PROGRAM – 1:**

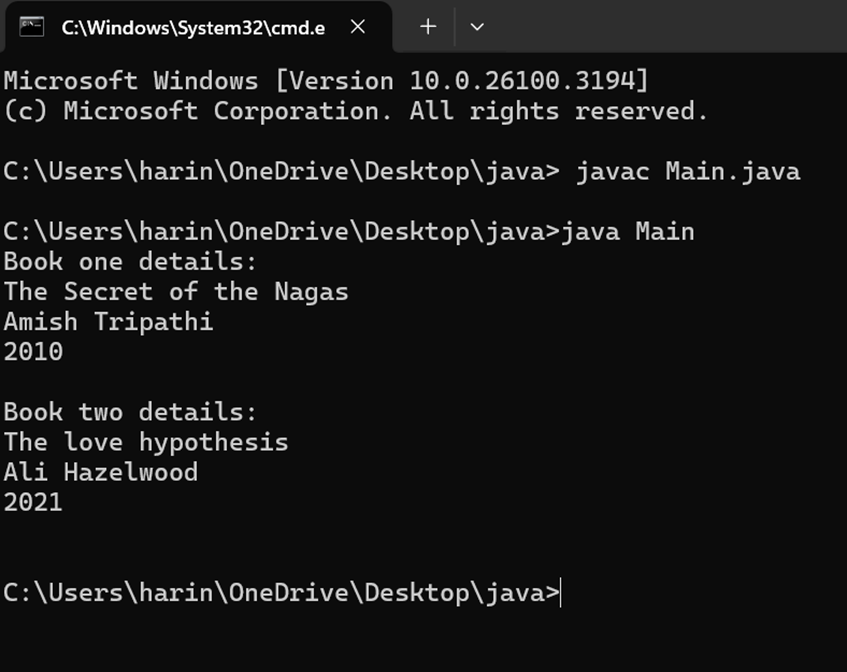
**AIM:** Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

Create a method which displays the details of the book and display the details of two books.

CODE:

****

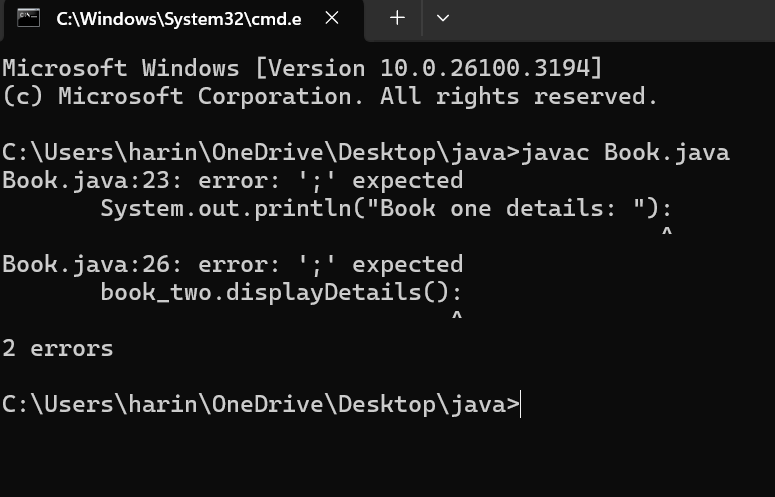
OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not defining the function in a file. 2. Two public class files should not be saved in the same file. | 1. To call the method we must define a function in a file. 2. Two public class files should be saved in different files. |

NEGATIVE CASE:



IMPORTANT POINTS:

1. While defining two classes for a code, we must be sure that we save both the classes in separate files.
2. While defining a method we should also define a function to call that method.

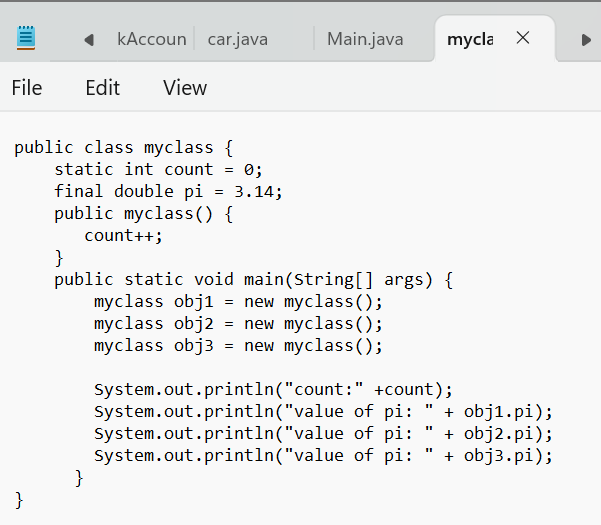
CLASS DIAGRAM:

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

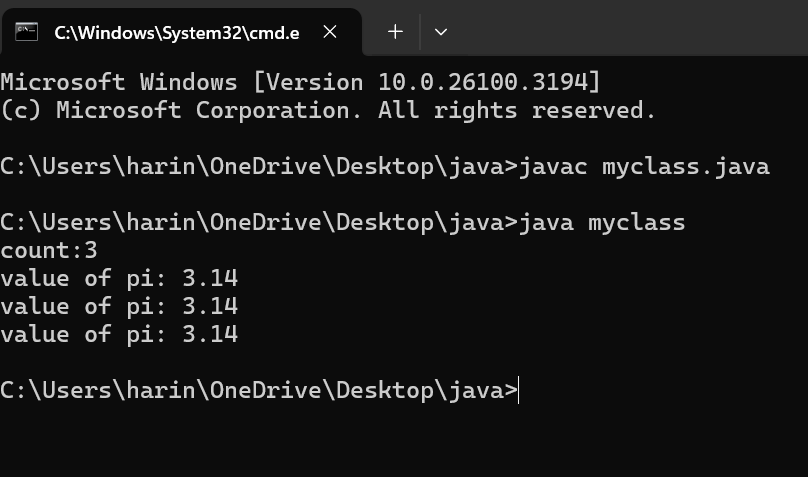
**PROGRAM – 2:**

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

CODE:



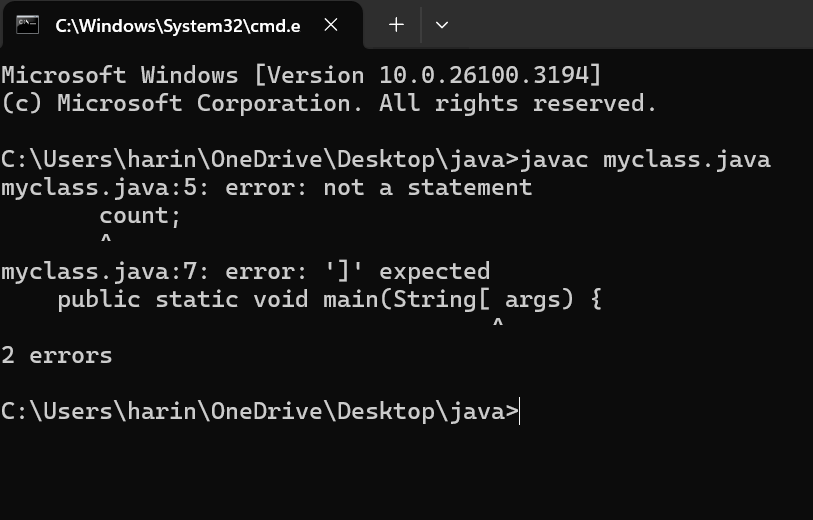
OUTPUT:



ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Not Putting the semi-colon after calling a function, 2. Not giving the indentation properly. | 1. Put the semi-colon after calling a function. 2. All the indentation must be correct to run the code correct. |

NEGATIVE CASE:



IMPORTANT POINTS:

1. We must declare the initial value of the variable before declaring the final one.
2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

CLASS DIAGRAM:

|  |
| --- |
| Myclass   * Count: int * Pi: double   + myclass( )  + main(args: String []): void |

**WEEK-5:**

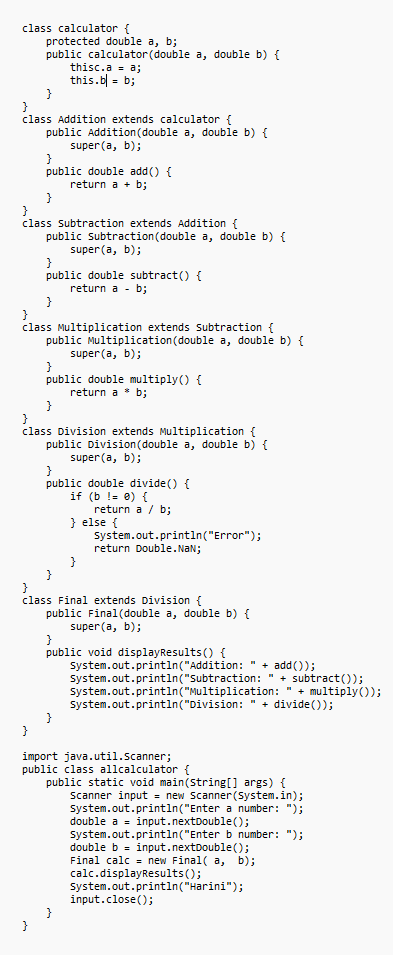
**PROGRAM 1:**

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

Hint: collect required variables using super class,

Create each class for a parameter and each class must contain a method.

**CODE:**



OUTPUT:



CLASS DIAGRAM:

|  |
| --- |
| Calulator   * a : double * b : double   + Calculator(a,b) |

|  |
| --- |
| Addition  + add(): double |

|  |
| --- |
| Subtraction  + subtract(): double |

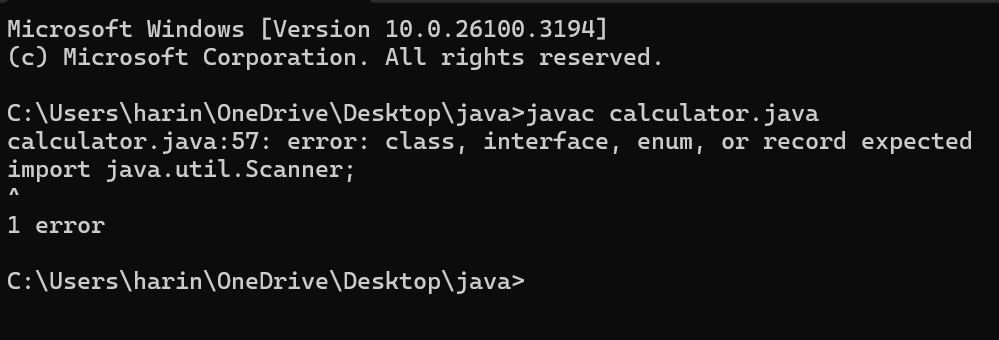
|  |
| --- |
| Multiplication  +multiply(): double |

|  |
| --- |
| Division  +divide(): double |

ERROR TABLE:

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

NEGATIVE CASE:



**IMPORTANT POINTS:**

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

**PROGRAM-2:**

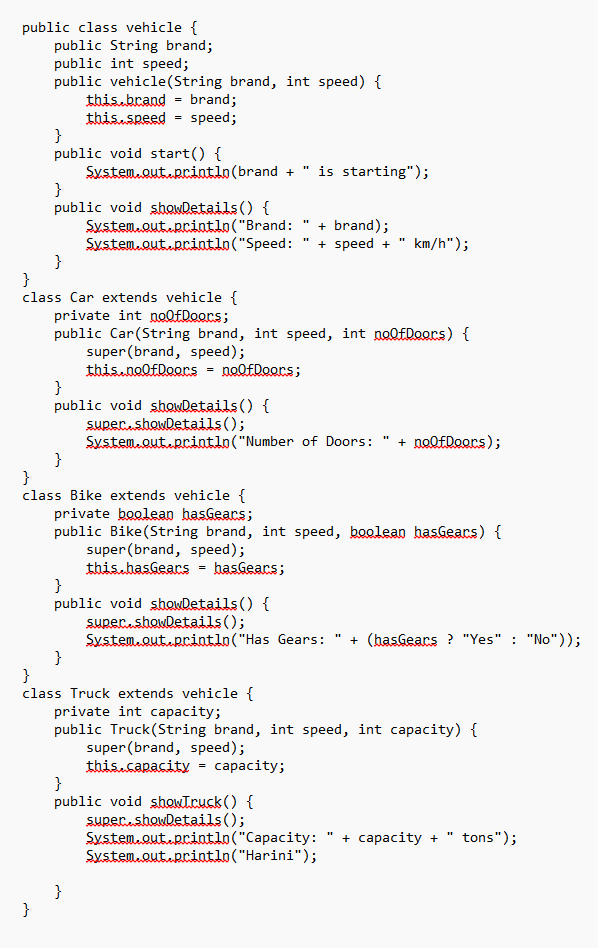
**AIM:** A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed( should be in super class)

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

Question:

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

**CODE:**

****

OUTPUT:



CLASS DIAGRAM:

|  |
| --- |
| Vehicle  -Brand: str  -speed: int  +int(brand, speed)  +start\_vehicle()  +display\_details() |

|  |
| --- |
| Car  -no.of.doors:  Int float  +int(brand,  Speed,no.of doors)  +displaydetails() |

|  |
| --- |
| Bike  -has gears:bool  +int(brand,speed,has gears)  +displaydetails() |

|  |
| --- |
| Truck  -capacity: |

ERROR TABLE:

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

IMPORTANT POINTS:

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

Answer:

The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

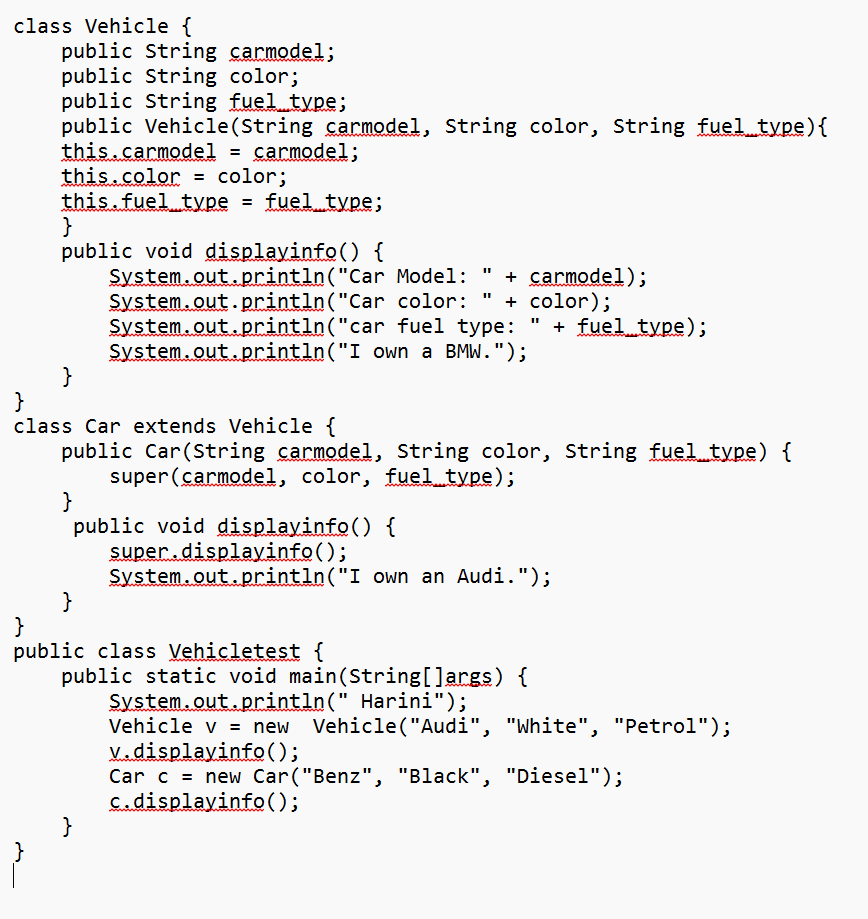
To add a new vehicle type truck we need to create a truck class that will:

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

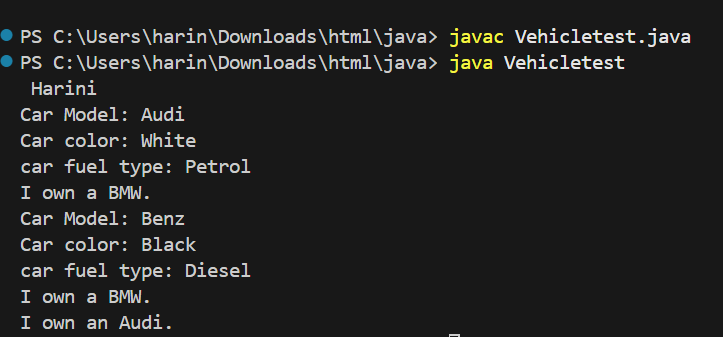
**WEEK-6:**

**Program-1:** Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to display specific information about a car.

**Code:**



OUTPUT:



CLASS DIAGRAM:

|  |
| --- |
| vehicle  -brand:String  -speed:int  +vehicle(brand:String, speed:int)  +startvehicle():void  +displayDetails():void |

ERROR TABLE:

|  |  |
| --- | --- |
| Error | Error Rectification |
| 1.Incorrect class name  for main method(Truck)  2.Incosistent car model output in displayInfo() | 1.Rename Truck to main or place main inside car or vehicle  2.Ensure car correctlypasses Benz” to super(car\_model,  Color, fuel\_type); |

Negative case: 

IMPORTANT POINTS:

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

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

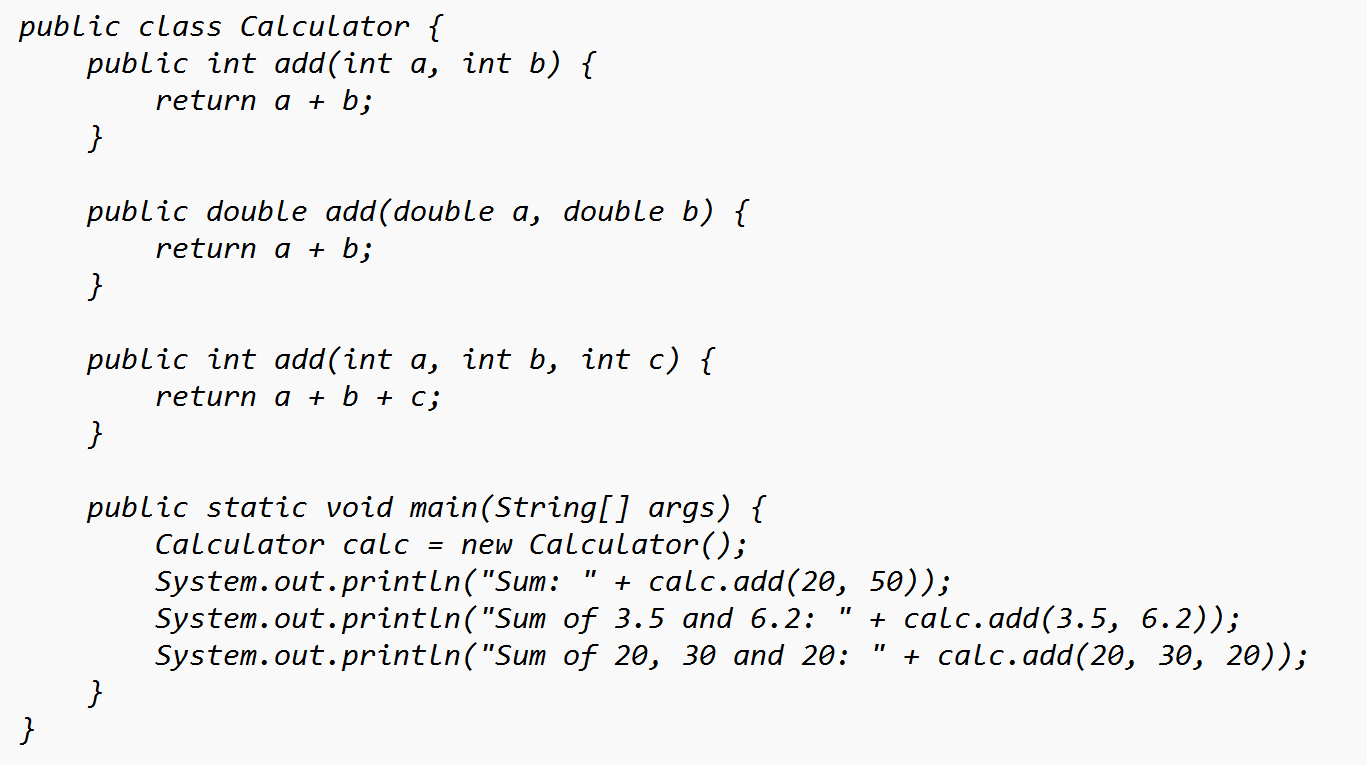
**Method Overriding:**The Car class overrides the displayInfo() method from Vehicle and calls super.displayInfo() to reuse the parent method before adding its own output.

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

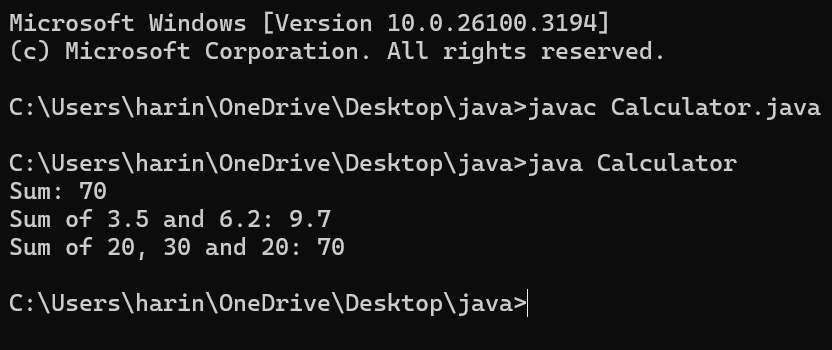
**Program-2:**

Create a calculator with overloaded methods to perform addition:  
(i) Add two integers.  
(ii) Add two doubles.  
(iii) Add three integers.

**CODE:**



**Output:**



**CLASS DIAGRAM:**

Calculator

+add(int, int): int

+add(double, double): double

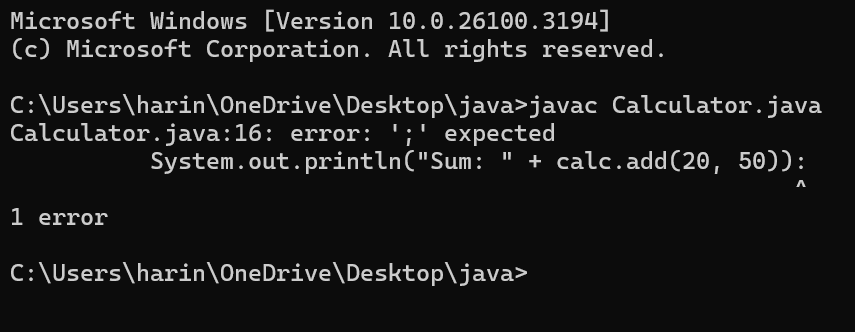
+add(int, int, int): int

+main(String[]): void

**ERROR TABLE:**

|  |  |
| --- | --- |
| Code Errors | Code Rectifications |
| 1.Method parameter missing spaces  (e.g. “int a, int b” should be “int a, int b”)  2.Inconsistent indentation in method bodies  (some lines not properly aligned) | 1.Add proper spacing between  Parameters:  -odd:add(int a, int b)  -new:add(int a, int b)  2.Fix indentation:  Consist 4 space of indentation |

**NEGATIVE CASE:**



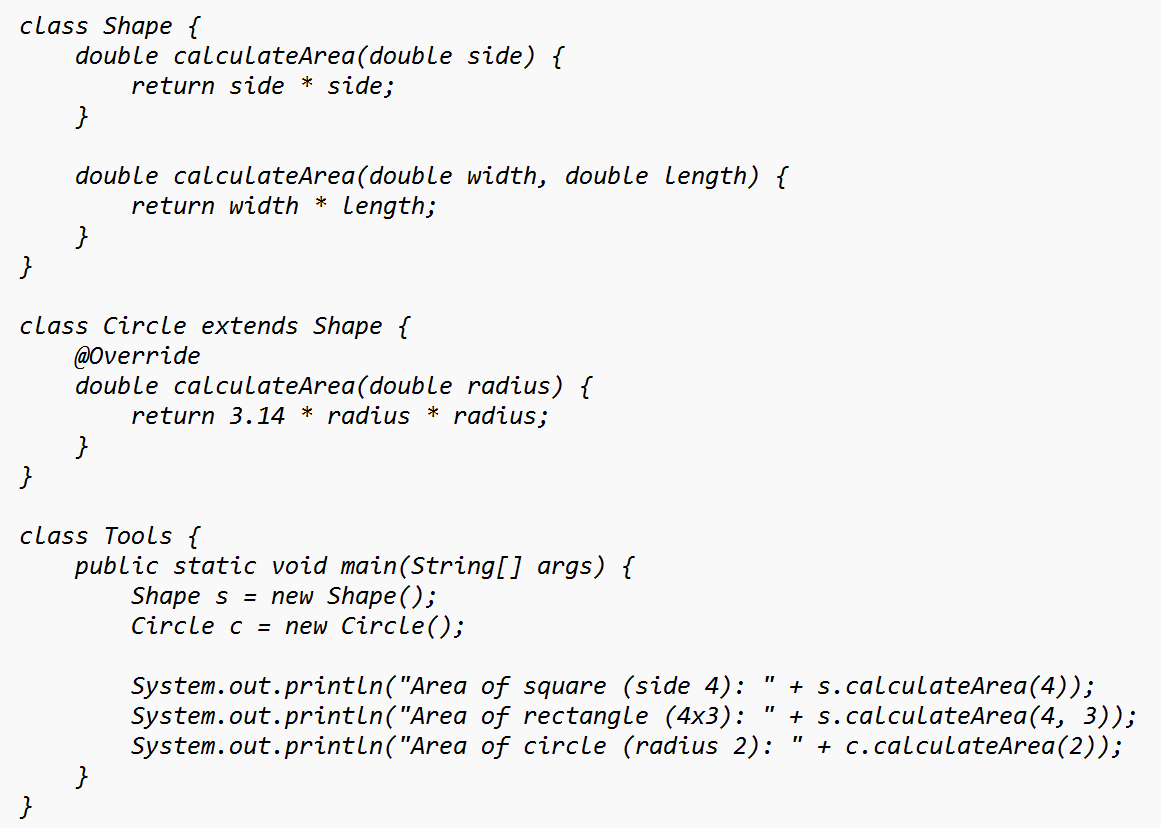
IMPORTANT POINTS:

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

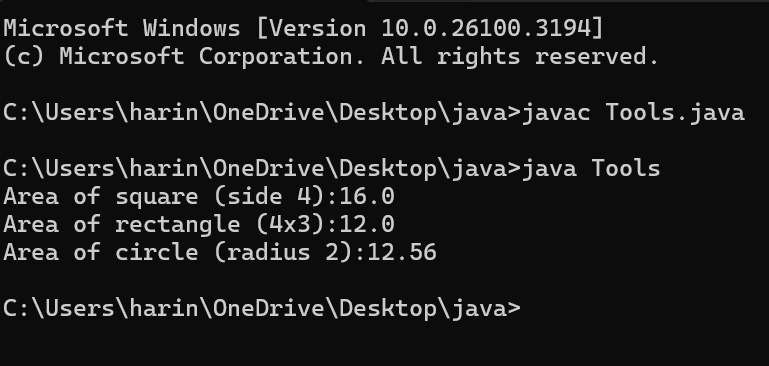
**2.Automatic Method Selection:** Java selects the appropriate add method based on the argument types during compilation.

**Program-3**: Create a class with a method `calculateArea()` that is overloaded for different shapes (e.g., square, rectangle). Then create a subclass `Circle` that extends this class and overrides the method to calculate the area of a circle.

**CODE:**



**Output:**

****

**Class Diagram:**

|  |
| --- |
| SHAPE |
| +calculateArea(side: double): double  +claculateArea(width: double, length: double): double |

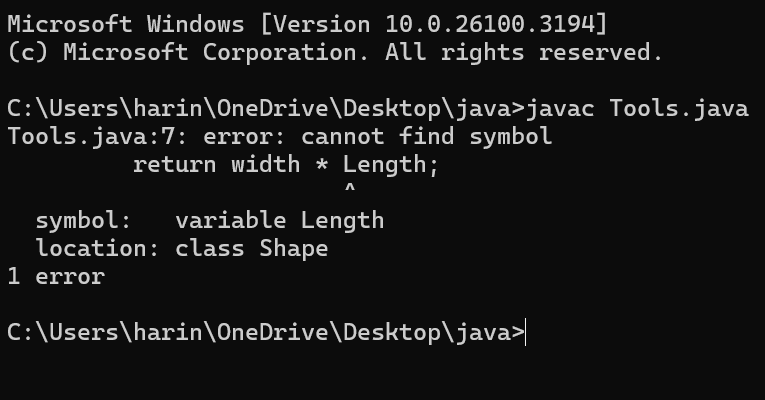
|  |
| --- |
| CIRCLE |
| +calculateArea(radius: double): double |

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

ERROR TABLE:

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

Negative case:



**Important Points:**

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

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

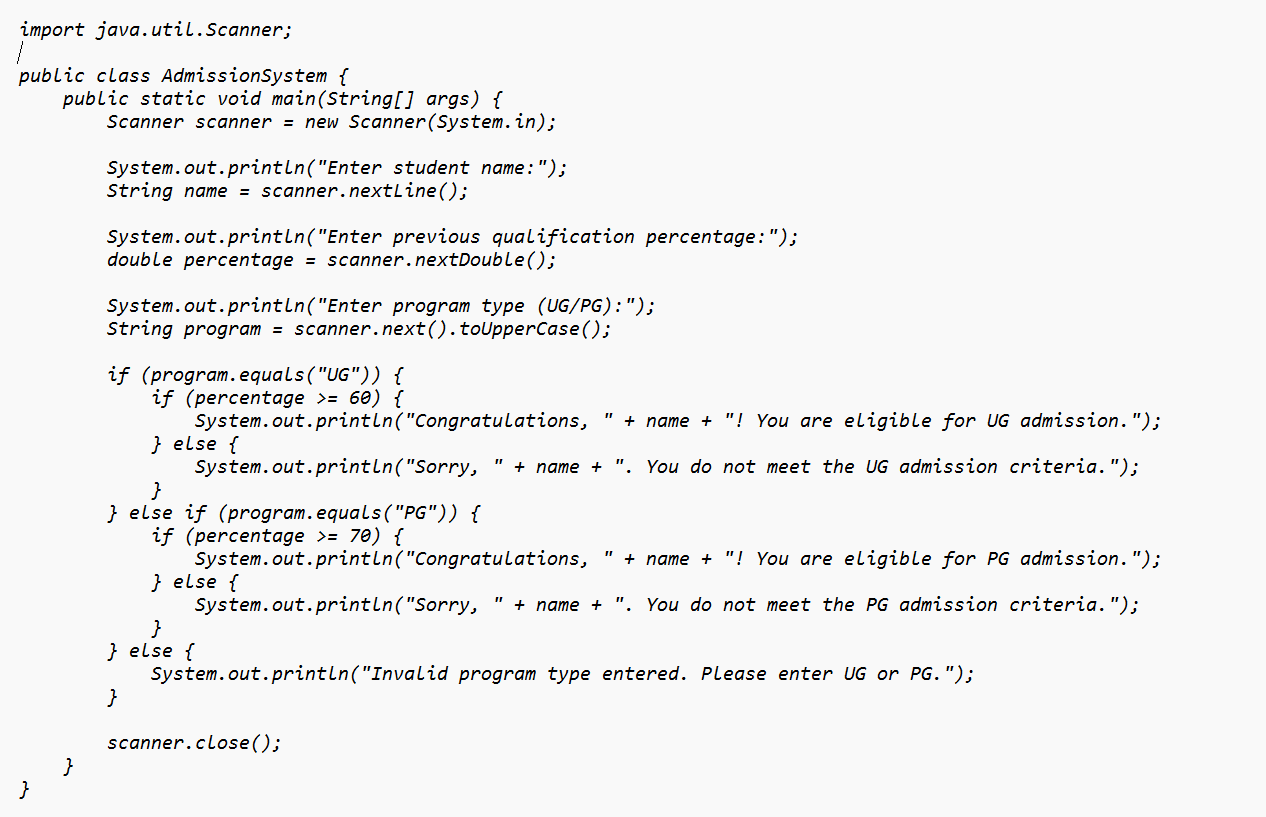
**3.Method Overriding**: Circle overrides calculateArea from Shape to implement its own formula.

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

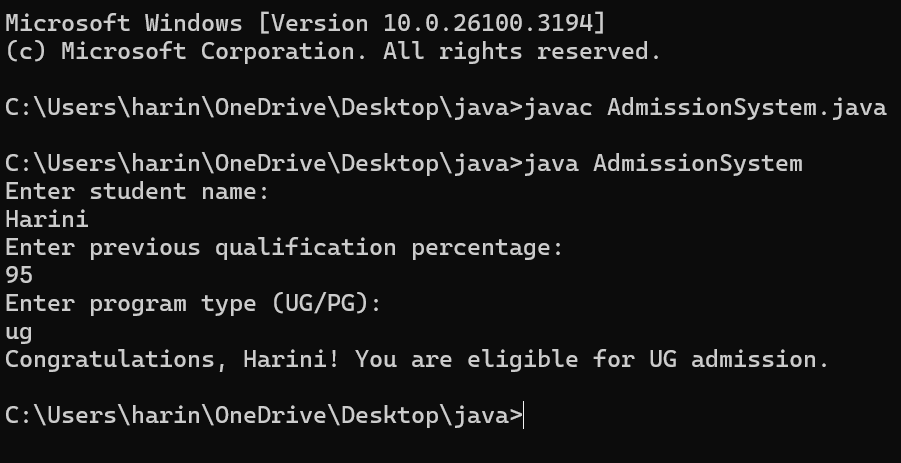
**5.Proper Object Reference**: Methods should be called using an object (s.calculateArea(4), c.calculateArea(2)).

**Program-4:** Create a class with a method `calculateArea()` that is overloaded for different shapes (e.g., square, rectangle). Then create a subclass `Circle` that extends this class and overrides the method to calculate the area of a circle.

**Code:**



OUTPUT:



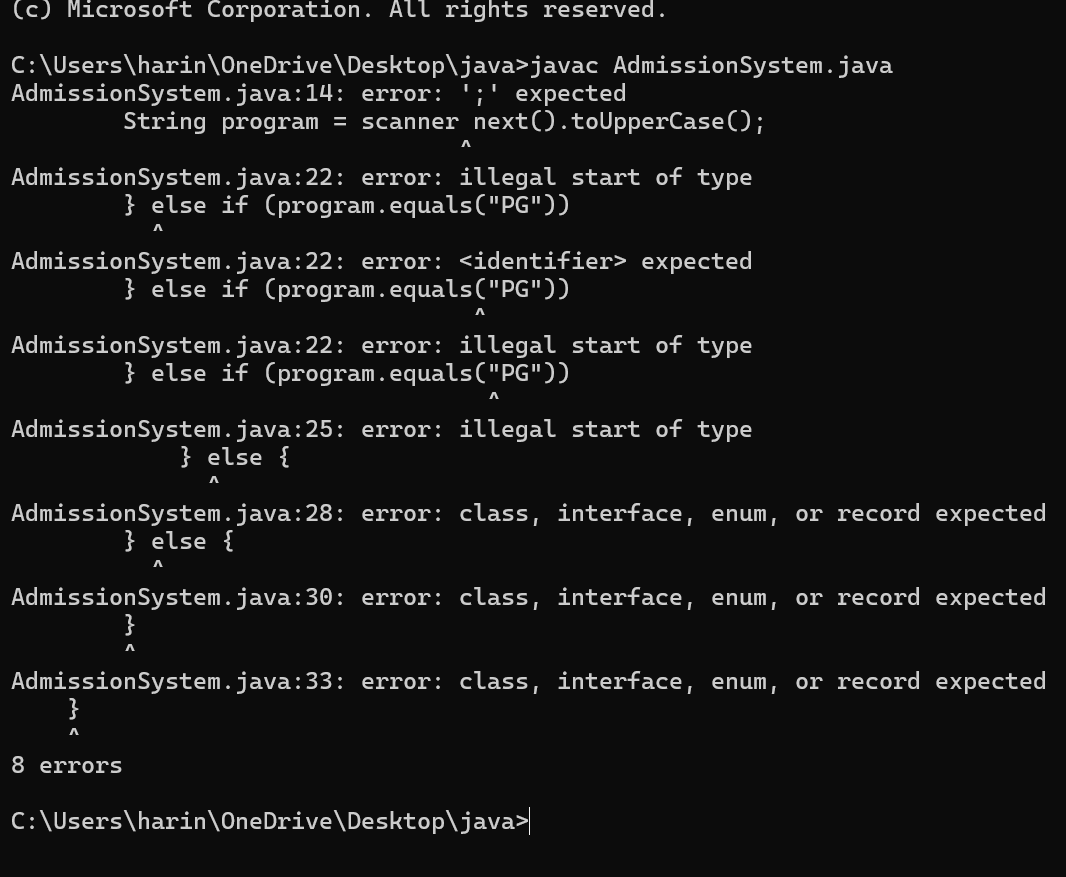
CLASS DIAGRAM:

|  |
| --- |
| AdmissionSystem |
| -scanner:Scanner  -name: String  -percentage: double  -program: String |
| +main(args: String[]): void  +takeInput():void  +checkEligibility(): void  +closeScanner(): void |

**ERROR TABLE:**

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

NEGATIVE CASE:



**Important Points:**

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

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

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

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