**23CSE111**

**OBJECT-ORIENTED**

**PROGRAMMING**

**LAB MANUAL**



**Department of computer science and engineering**

**Amrita School of Engineering**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

# **Name:** E . Aamuktha

**Verified By: Roll No:** 24047

**Date of Submission: Class:** CSE-A

**Semester:** 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No. | Title | Date | Page No. | Signature |
| Week 1 |  | 27-01-2025 |  |  |
| 1. | How to download and install Java Software. |  | 3 |  |
| 2. | Write a Java Program to print the message “Welcome to Java Programming”. |  | 6 |  |
| 3. | Write a Java program that prints: Name, Roll.no. , section of a student. |  | 7 |  |
| Week 2 |  | 10-02-2025 |  |  |
| 1. | Write a java program to calculate the area of a rectangle. |  | 8 |  |
| 2. | Write a java program to temperature from Celsius to Fahrenheit and vica-versa. |  | 9 |  |
| 3. | Write a java program to calculate the simple interest. |  | 10 |  |
| 4. | Write a java program to find the largest of three numbers, using ternary operator. |  | 11 |  |
| 5. | Write a java program to find the factorial of a number. |  | 12 |  |
| Week 3 |  | 24-02-2025 |  |  |
|  | To create a java program with the following instructions:   1. Create a class with name “Car” 2. Create 4 attributes, named: car\_color, car\_brand, fuel\_type, mileage 3. Create 3 methods, named: start(), service(), stop() 4. Create 3 objects, named: car1, car2, car3 5. Create a constructor, which should print, “Welcome to car garage”. |  | 13 |  |
|  | To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().   1. deposit(): Whenever an amount is deposited, it has to be update the current amount. 2. withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”) |  | 15 |  |
|  |  |  |  |  |

**WEEK 1**

**Program 1**

**Aim:** How to download and install Java software

**Procedure:**

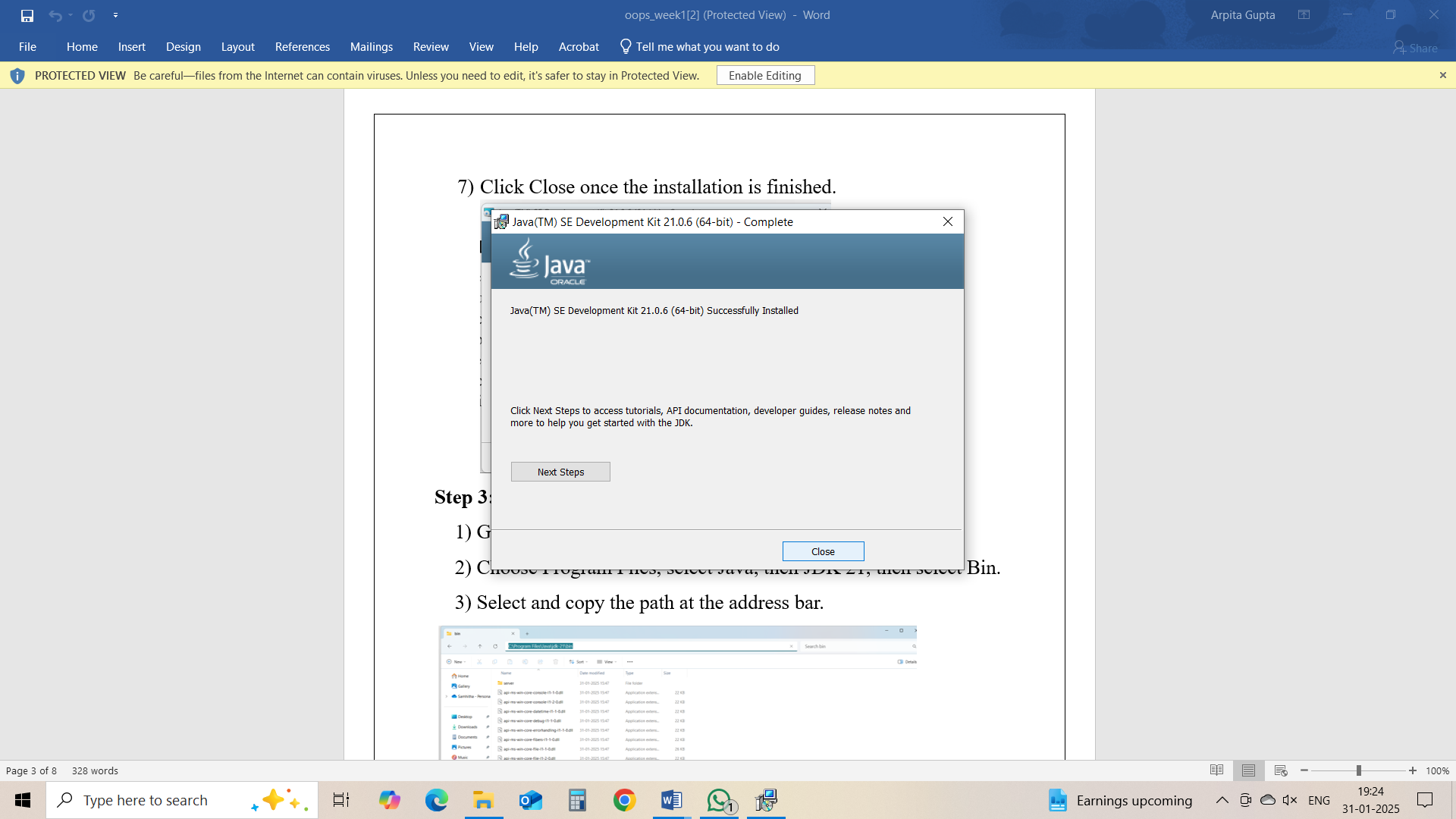
**Step 1: To download Java software**

1. Open your web browser, and search for Java Oracle download in the search bar.
2. Open Java Downloads | Oracle India.
3. Scroll down and click on JDK 21(Java SE Development Kit 21.0.6 downloads)
4. Select your operating system.(Windows)
5. Download X64 Installer.



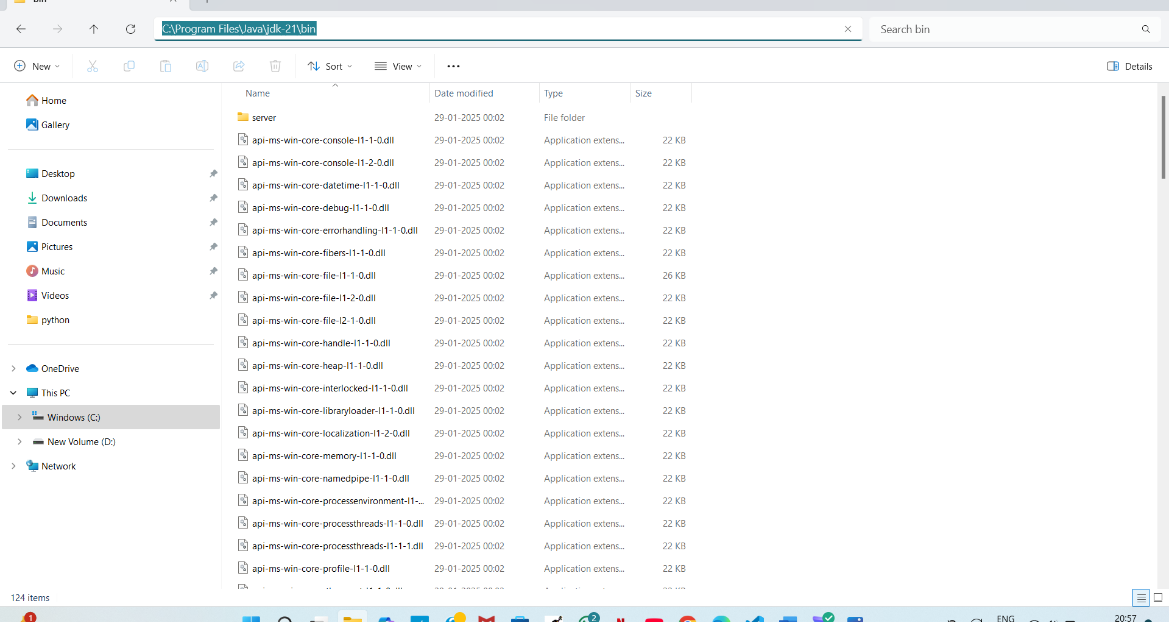
**Step 2: To install Java on your system**

1. For installation, locate the downloaded jdk-21\_windows-x64\_bin.exe file.
2. Double-click to launch the installer.
3. Click next on setup wizard.
4. Choose the installation path. (C:\Program Files\Java\jdk-21)
5. Click next, then click install.
6. Wait for installation to complete.
7. Click on close, once the installation is complete.



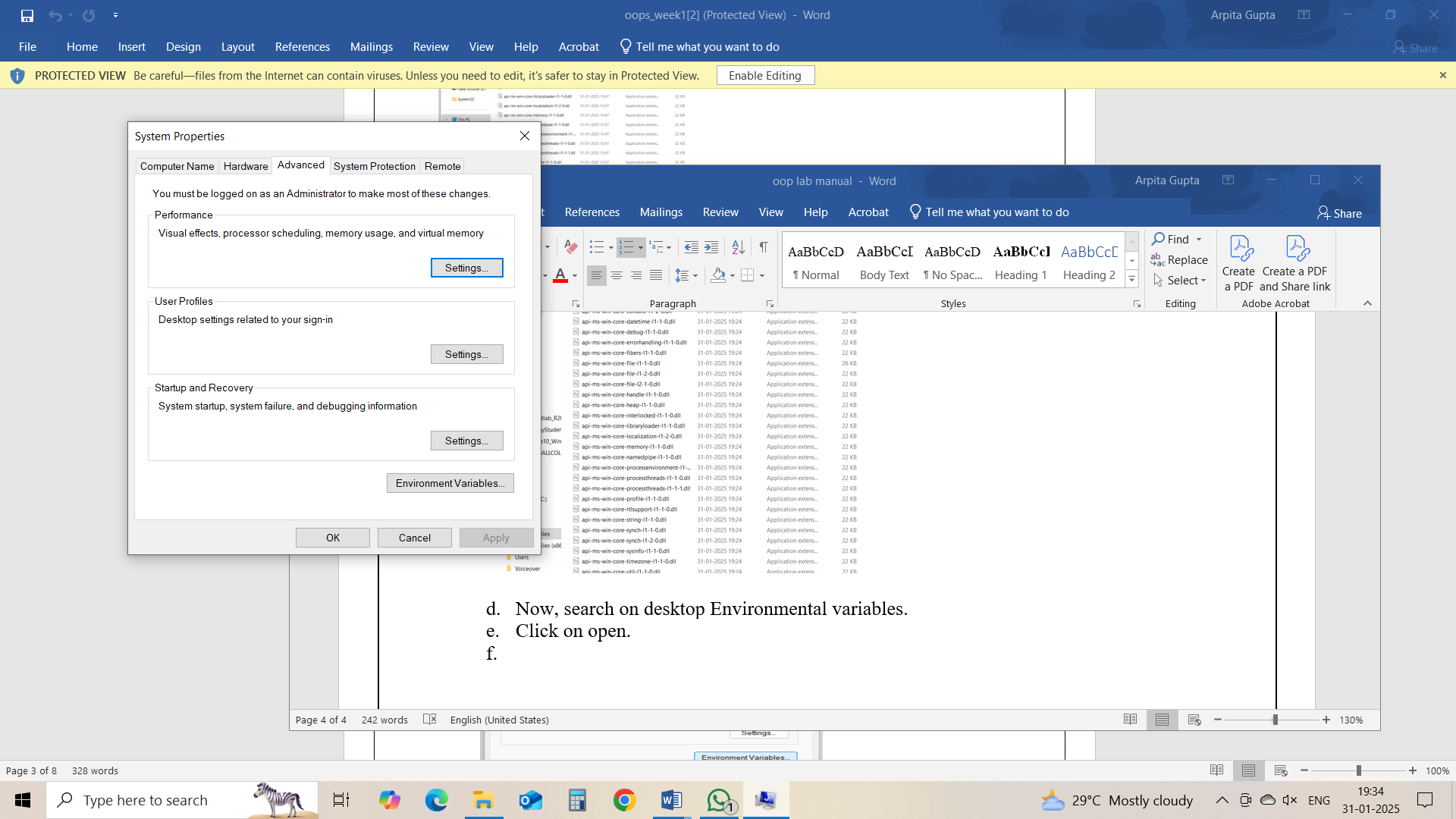
**Step 3: Setup a path**

1. Go to C drive on desktop.
2. Choose Program Files > Java > jdk21 > bin
3. Select and copy the path at the address bar.



**Step 4: Open System Properties**

1. Press Windows + R, type sysdm.cpl, and click Ok.
2. The System Properties window will open.
3. Navigate to the Advanced tab.
4. Click on Environment Variables at the bottom.



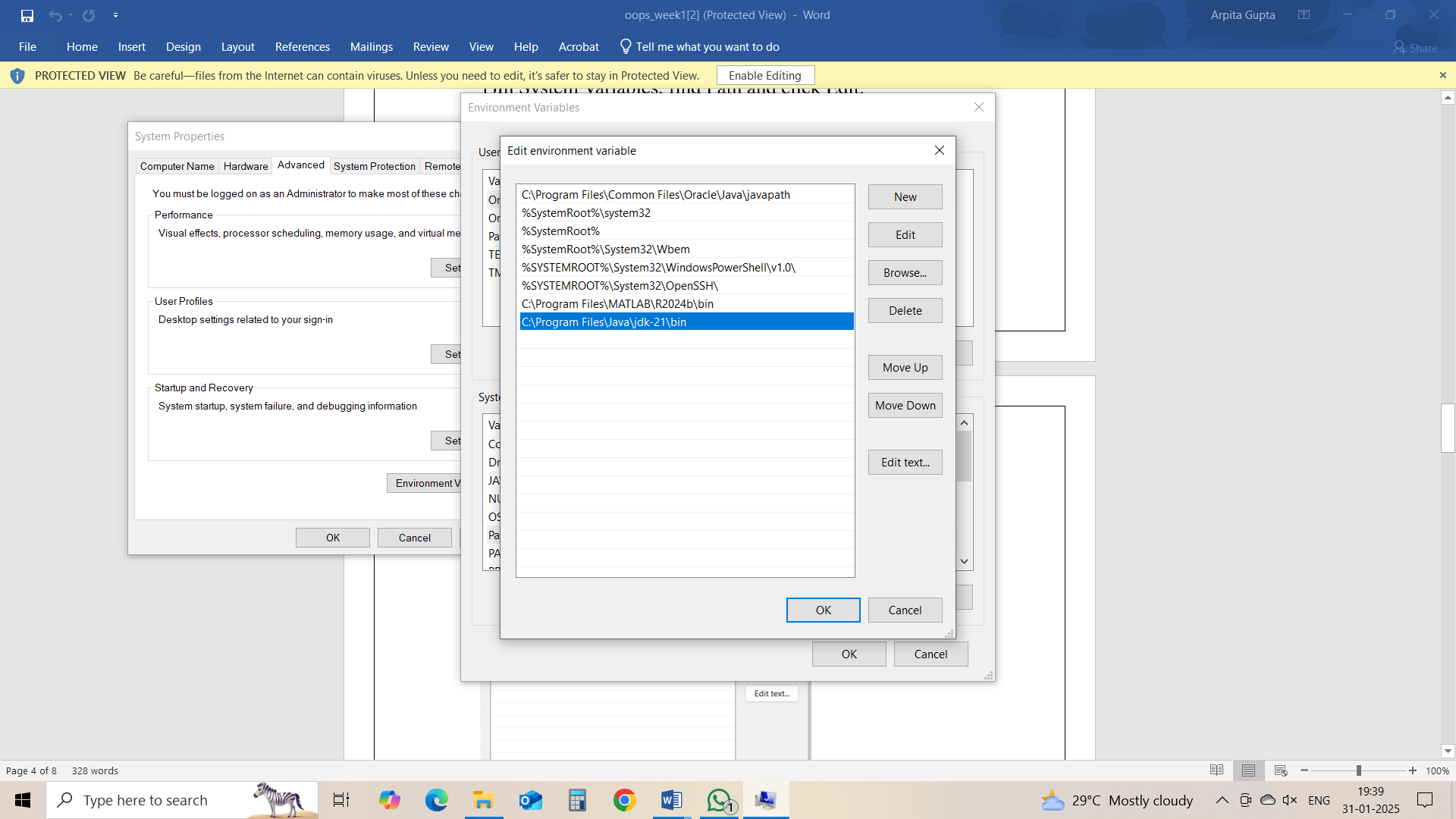
**Step 5: Set JAVA\_HOME**

1. Under System Variables, click New.
2. Set the Variable name as JAVA\_HOME.
3. Set Variable value as C:\Program Files\Java\jdk-21 (or your installation path).
4. Click OK.



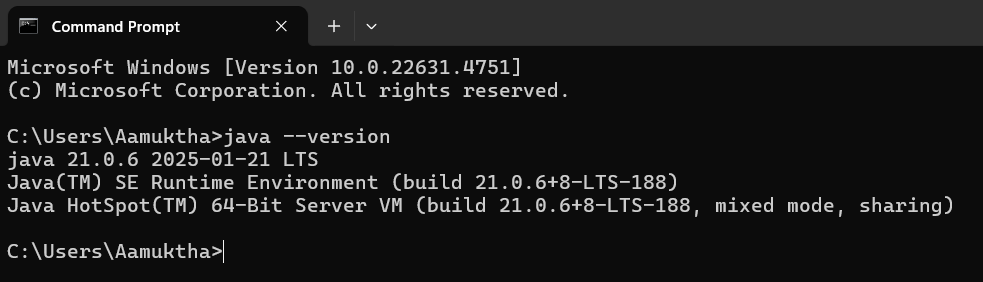
**Step 6: Update PATH Variable**

1. In System Variables, find Path and click Edit.
2. Click New and add: C:\Program Files\Java\jdk-21\bin
3. Click OK to save.

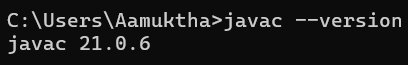


**Step 7:Verify Installation**

1. Open Command Prompt.
2. Type the following command: **java --version** and press Enter.



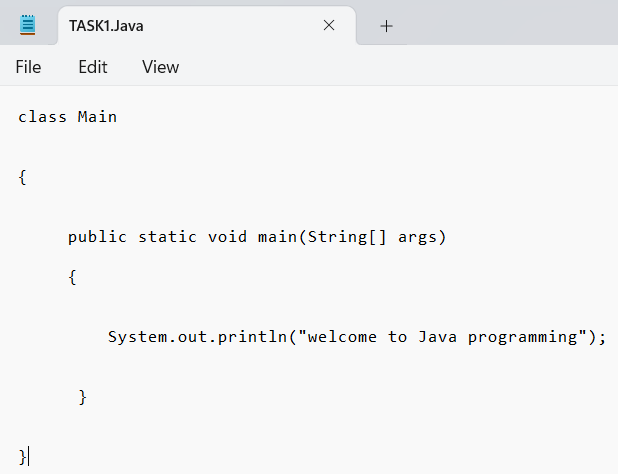
1. To check the java compiler type: **javac –version.**



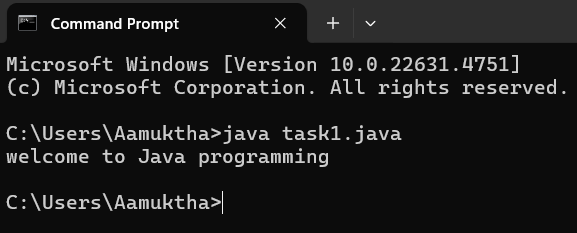
**Program 2:**

**Aim:** Write a Java Program to print the message “Welcome to Java Programming”

**Code:**

****

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  System.out.println("Welcome to Java Programming") | Adding a semi-colon at the last  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 |

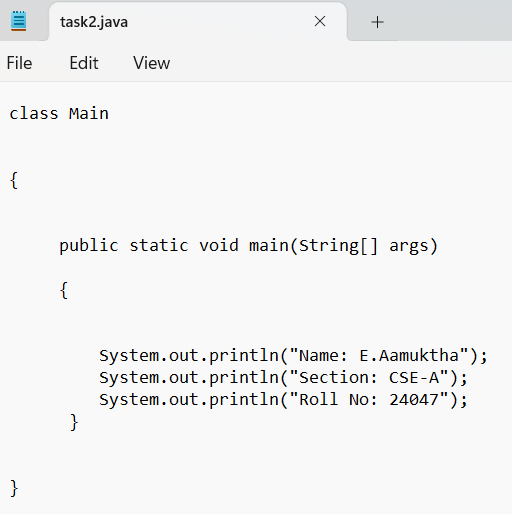
**Concepts to be known:**

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

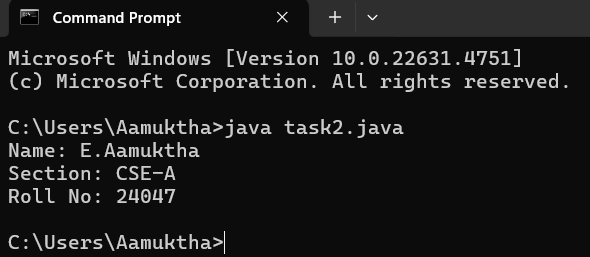
**Program 3:**

**Aim:** Write a Java Program to print the Name, Roll.no. and Section of a student.

**Code:**

****

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: incompatible types: int cannot be converted to String  String sc=input.nextInt(); | String sc=input.next() |
| 2. | error: incompatible types: String cannot be converted to char  char sc=input.next(); | Change the data type to String |

**Concepts to be known:**

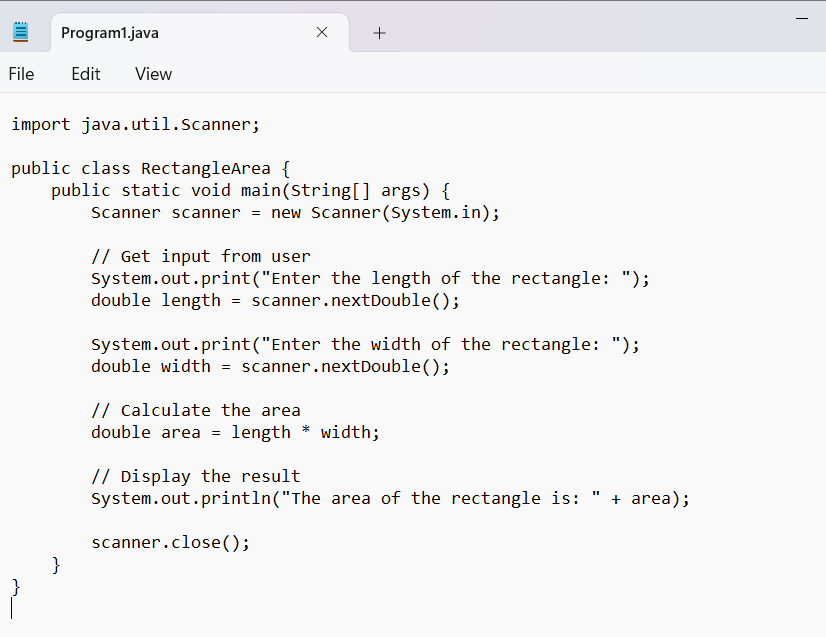
1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. String nm=input.next(); - Used to read the String data type stored under the object created
4. int rn=input.nextInt(); - Used to read the integer data type stored under the object created
5. 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 2**

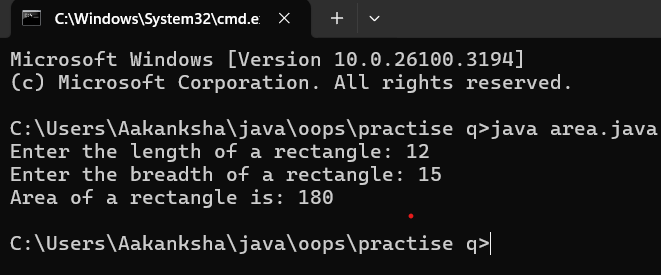
**Program 1**

**Aim:** 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 |

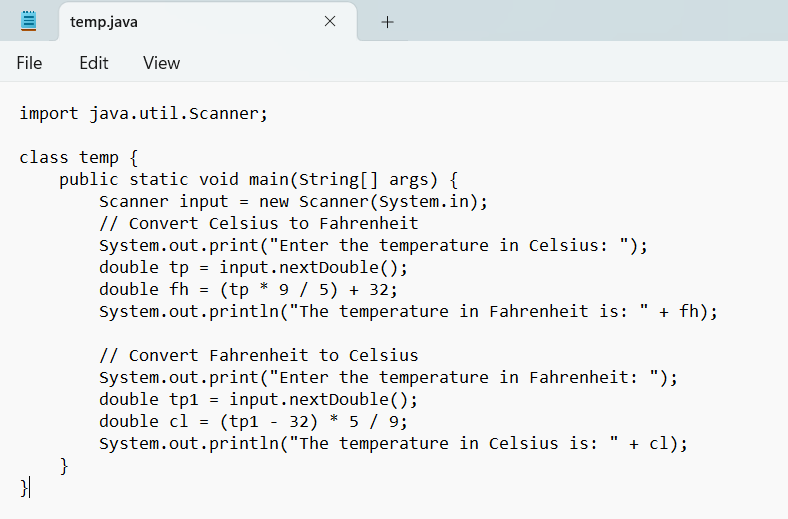
**Concepts to be known:**

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.

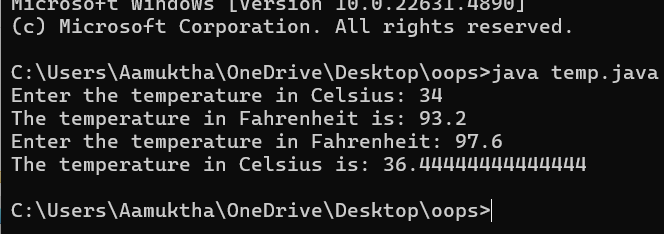
**Program 2**

**Aim:** Write a java program to convert temperature from Celsius to Fahrenheit and vica-versa.

**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:"); |

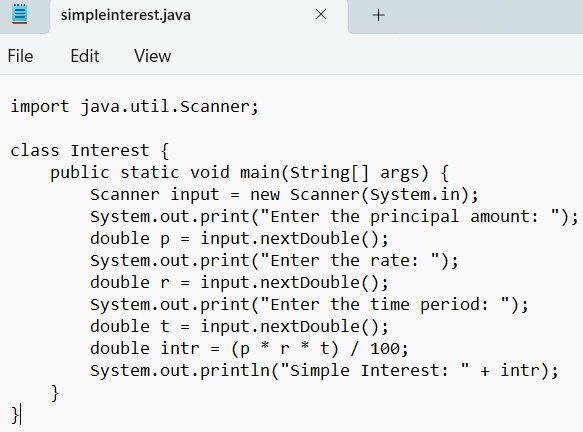
**Concepts to be known:**

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.

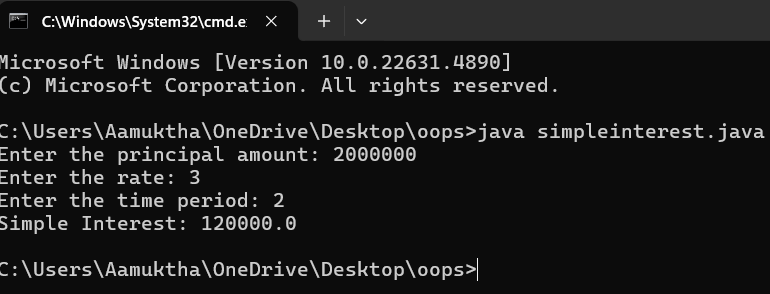
**Program 3:**

**Aim:** 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(); |

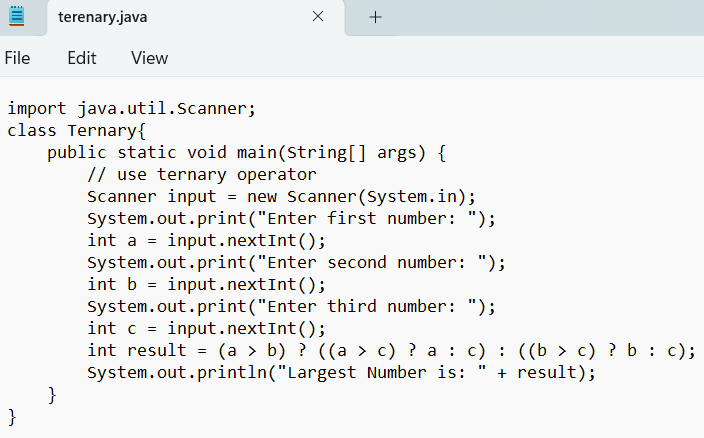
**Concepts to be known:**

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.

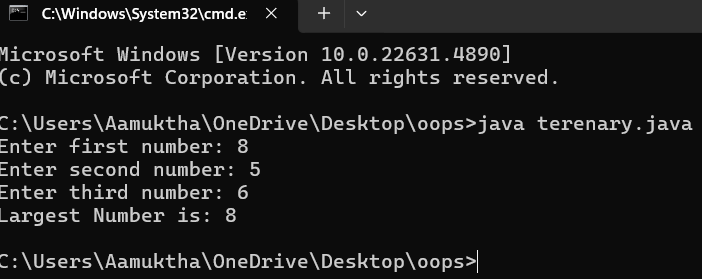
**Program 4**

**Aim:** 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); |

**Concepts to be known:**

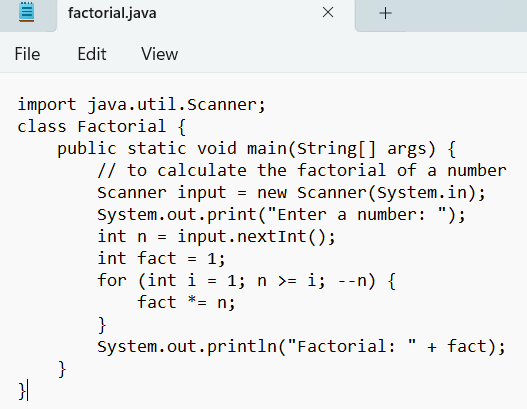
1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. int a=input.nextInt (); - Used to read integer data type stored under the object created
4. int result=(a>b)? ((a>c)?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

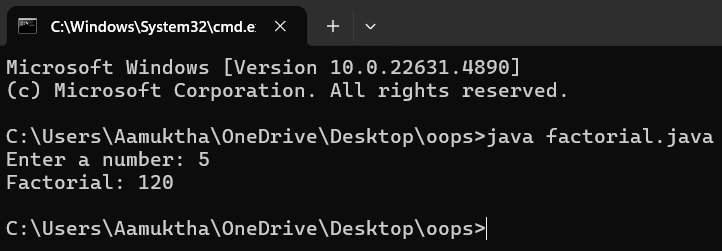
**Program 5**

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

**Code:**



**Output:**



**Error:**

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

**Concepts to be known:**

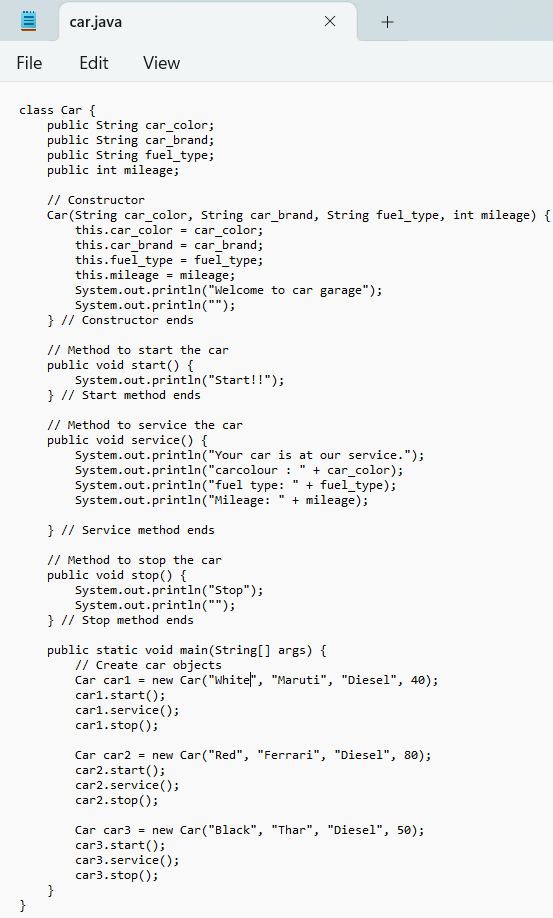
1. for (int i=1; n>=i;--n){ } - For loop syntax: for(initial expression; test expression; update expression){} The loop is executed, until the test expression evaluates to be false.

**WEEK 3**

**Program 1**

**Aim:** To create a java program with the following instructions:

1. Create a class with name “Car”
2. Create 4 attributes, named: car\_color, car\_brand, fuel\_type, mileage
3. Create 3 methods, named: start(), service(), stop()
4. Create 3 objects, named: car1, car2, car3
5. Create a constructor, which should print, “Welcome to car garage” .

**Code:****Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  car1.start() | Add a “;”  car1.start(); |
| 2. | error: illegal start of type  public void stop({ | Add a “)”  public void stop(){ |
| 3. | error: cannot find symbol thiscar\_brand=car\_brand; | Add a “.”  this.car\_brand=car\_brand; |

**Class Diagram:**

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

**Concepts to be known:**

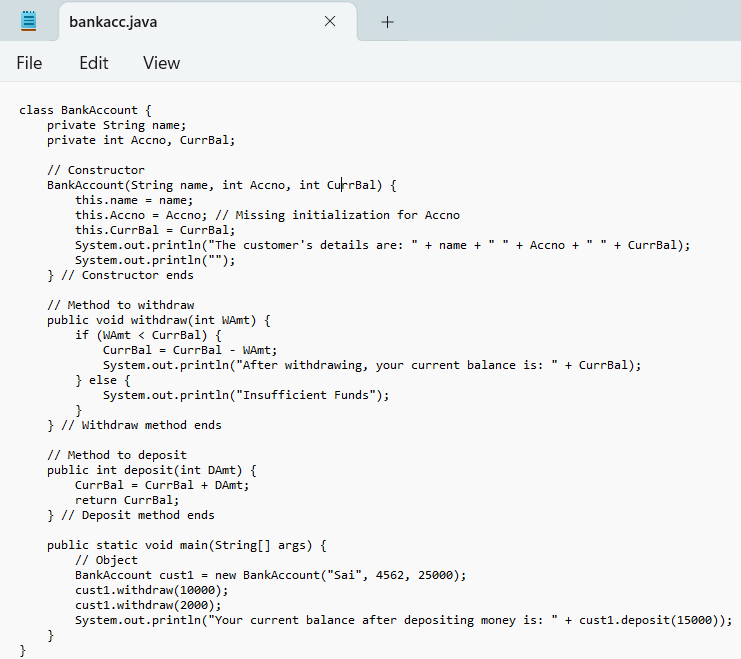
1. public String car\_color; - Used to declare a variable named car\_color, with data type as String with public accessibility.
2. Car(String car\_color,String car\_brand,String fuel\_type,int mileage){ } – It is a constructor (method with name same as class), which requires parameters such as car\_color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

**Program 2**

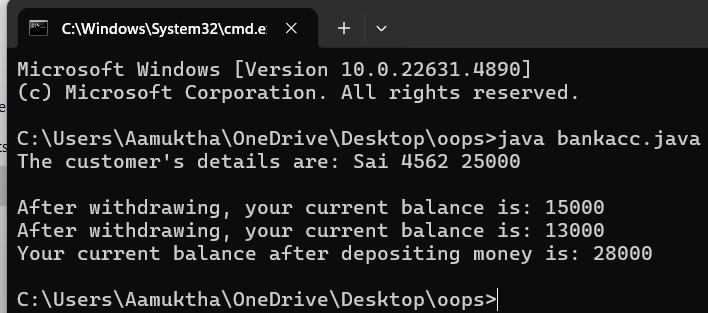
**Aim:** To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().

1. deposit(): Whenever an amount is deposited, it has to be update the current amount.
2. withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”)

**Code:**



**Output:**



**Error:**

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

**Class Diagram:**

|  |
| --- |
| BankAccount |
| - name: String  - Accno: int  - CurrBal: int |
| BankAccount: void  + withdraw(int WAmt): void  + deposit(int DAmt): int |

**Concepts to be known:**

1. private String name; - Used to declare a variable named name, with data type as String with private accessibility.
2. BankAccount(String name,int Accno,int CurrBal){ } – It is a constructor (method with name same as class), which requires parameters such as name (String data-type) and so on.
3. this.CurrBal=CurrBal; - “this” is a default method, which is used to point to the instance variables.
4. public void withdraw(int WAmt){ } – used to declare a method, which will return nothing(void) in public accessibility, which requires a parameter WAmt(integer data type).
5. public int deposit(int DAmt){} - used to declare a method, which will return integer data type in public accessibility, which requires a parameter DAmt(integer data type).
6. BankAccount cust1=new BankAccount("Ram",5587,20000); - used to create a object in class BankAccount, with object name as cust1.
7. cust1.withdraw(50000); - Calling a method, under object cust1, by passing a parameter.
8. System.out.println("Your current balance after depositing money is:"+cust1.deposit(25000)); - Deposit method will return the value, which will be directly printed.

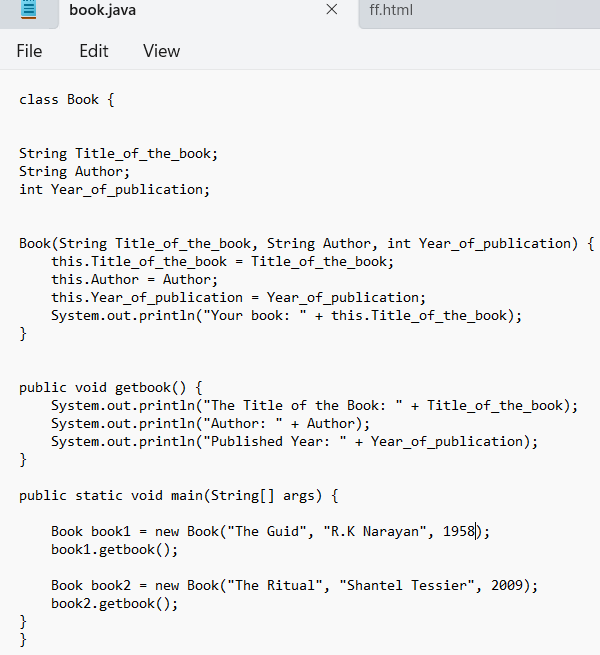
**WEEK 4**

**Program 1**

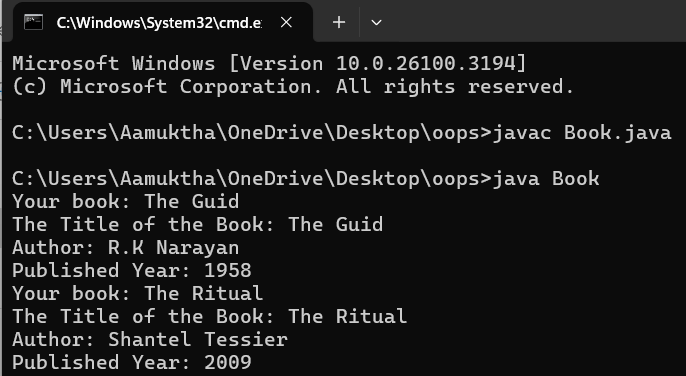
**Aim:** Write a java program with class named “book”, the class should contain various attributes such as title, author, year of publication it should also contain a constructor with parameters which initializes, title, author, and year of publication.

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

**Code:**

****

**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error Message** | **Error rectification** |
| 1. | System.out.println(" Your book: "+this Title\_of\_the\_book); - Syntax error | Change this Title\_of\_the\_book to this.Title\_of\_the\_book. |
| 2. | book.getbook(); - Error in calling method | Replace book.getbook(); with book1.getbook(); and book2.getbook();. |
| 3. | book book1 = new book("The Guid", "R/K Narayan", 1958); - Class name issue | Change book to Book everywhere in the file |

**Class Diagram:**

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

**Concepts to be known:**

A)(public Book(String titleOfTheBook, String author, int yearOfPublication)):

•This method initializes the attributes of the Book class when an object is created.

•It ensures that all necessary details (title, author, year of publication) are provided when creating a new book object.

B)Public String getTitle():Return the title of the book.

C)Public String get()Author(): the author of the book.

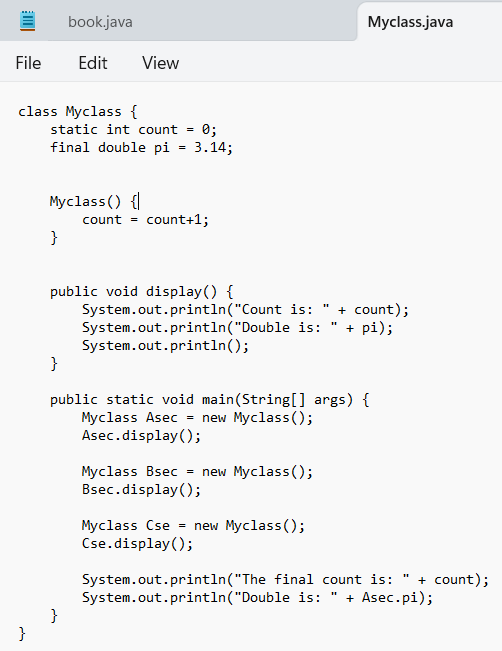
D)Public String getYearOfPublication():Return the year of publication of the book.

E)public void getBook(): This method prints out the details of the book, including its title, author, and year of publication.

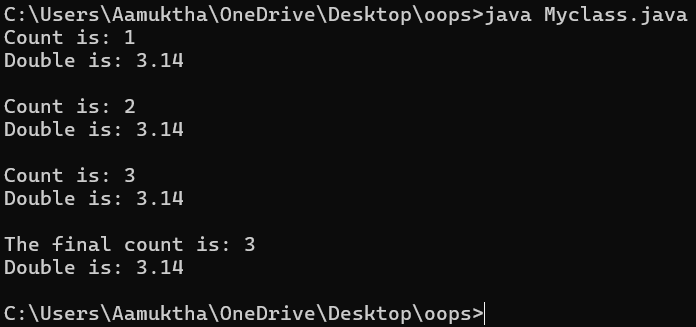
**Program 2**

**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. |

**Class Diagram:**

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

**Concepts to be known:**

•Asec.display() and Bsec.display() access the instance methods and variables through their respective object references.

•System.out.println(“Double is :”+Bsec.pi); accesses that pi variable of the Bsec object.

•new keyword followed by the class constructor. This allocates memory for the object and initializes its attributes.

•new is necessary for creating objects and invoking constructors.

•Object References are needed to access instance variables and methods.

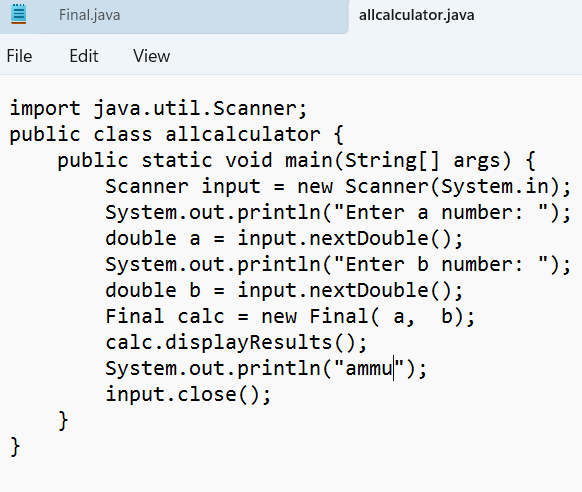
•final double pi means that once pi is initialized with the value 3.14, it cannot be changed.

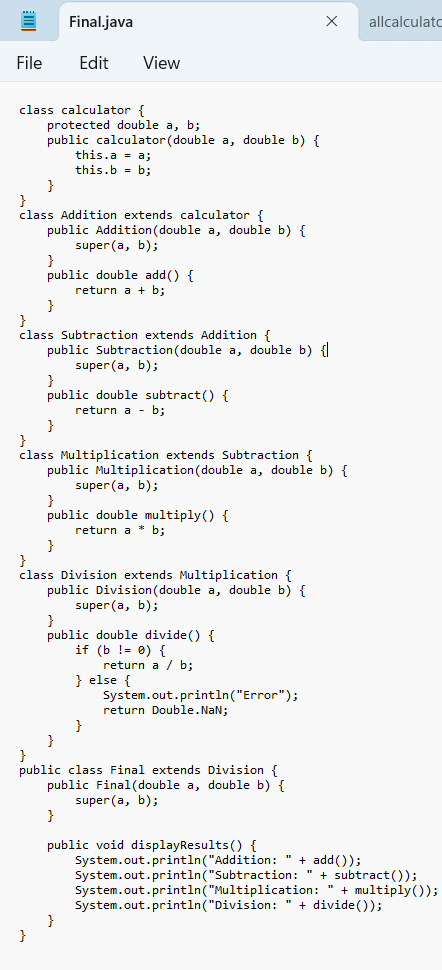
**WEEK 05**

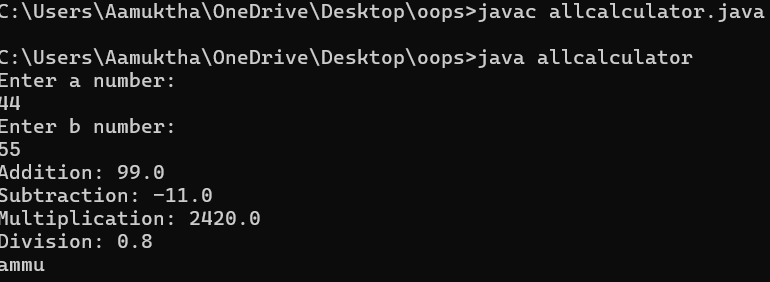
**PROGRAM-1:**

**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:** ****

**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:**

|  |  |
| --- | --- |
| **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. |

**CLASS DIAGRAM:**

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

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

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

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

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

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

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

**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.

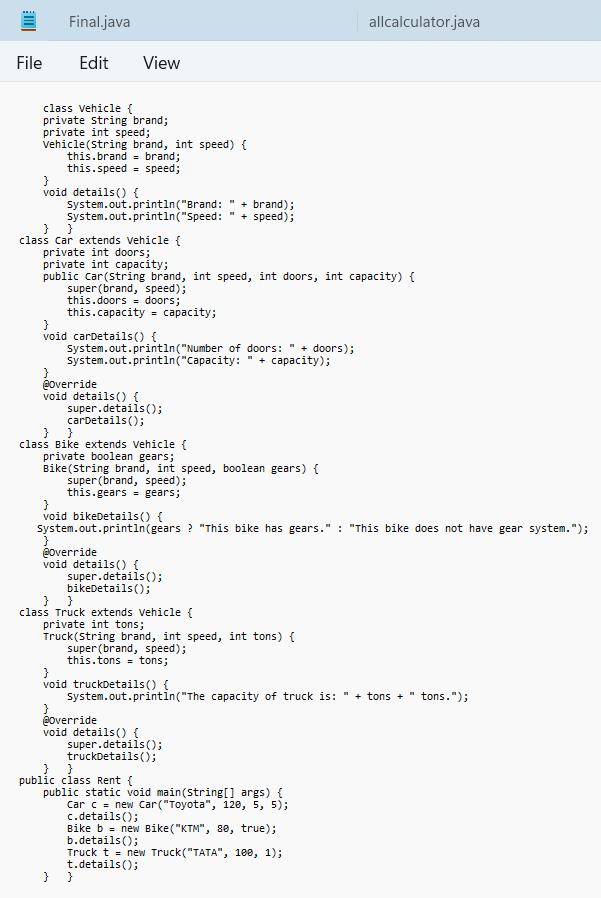
**IMPORTANT POINTS:**

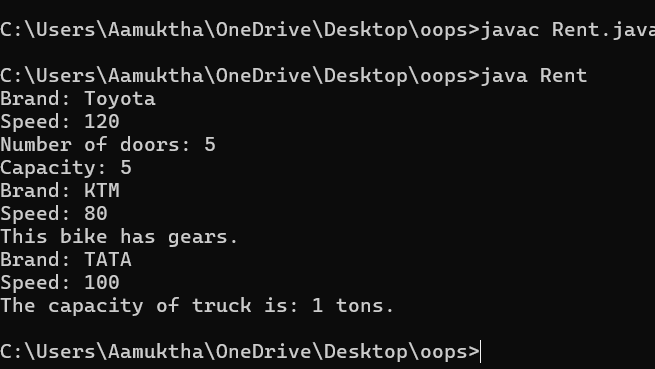
1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. a double method can represent more decimal point numbers than float method.
4. the void keyword in java is used to specify that a method does not return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

|  |  |
| --- | --- |
| 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. |

**CLASS DIAGRAM:**

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

|  |
| --- |
| Bikes |
| -has gears:bool |
| +int (brand, speed,  has gears);  +display deatails(); |

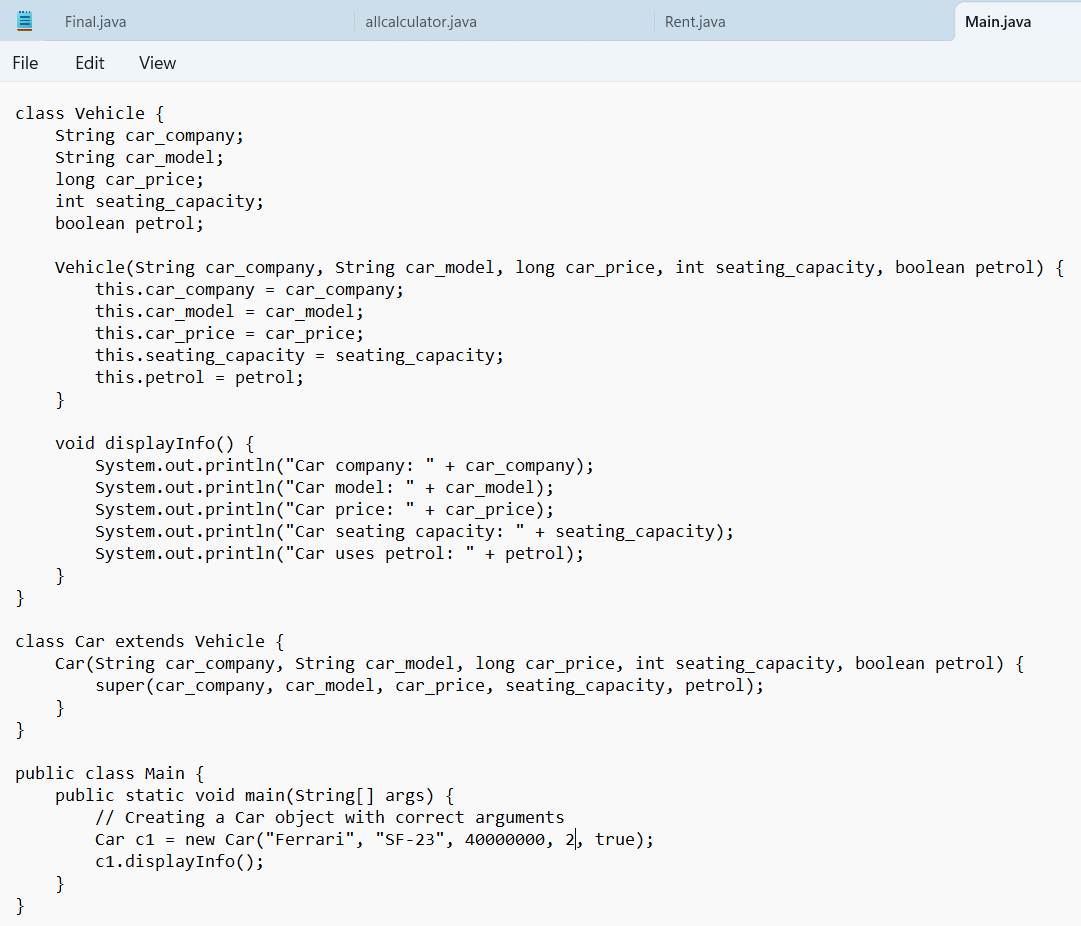
|  |
| --- |
| Car |
| -no.of.doors:int |
| +int (brand, speed,  No.of doors);  +display deatails(); |

|  |
| --- |
| Truck |
| -Capacity:float |
| -Show truck detais();  +display deatails(); |

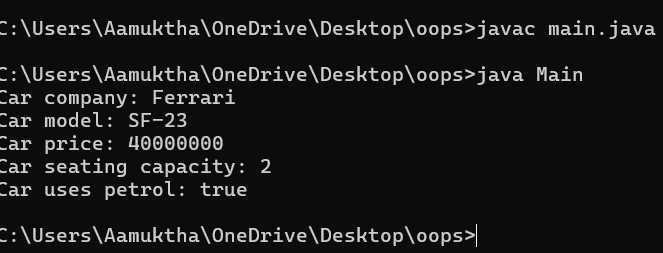
**WEEK-06**

**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:** ****

**CODE:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Incorrect class name for main method(Truck).  2. Inconsistent car model output in displayinfo(). | 1.Rename Truck to Main or place main inside car or vehicle.  2. Ensure Car correctly passes Toyota” to super(car\_model,color,fueltype) |

**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:**

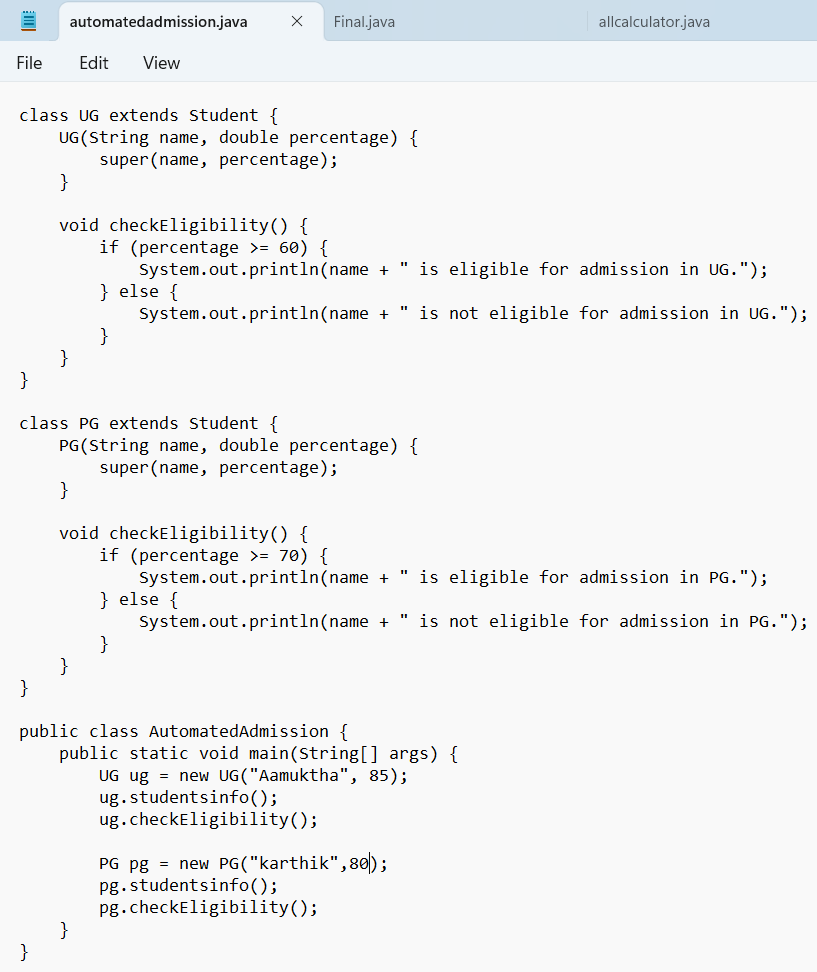
|  |
| --- |
| **Vehicle** |
| * Brand: String * Speed: int |
| + vehicle(brand: string  Speed: int)  +start vehicle(): void  +displaydetails():void |

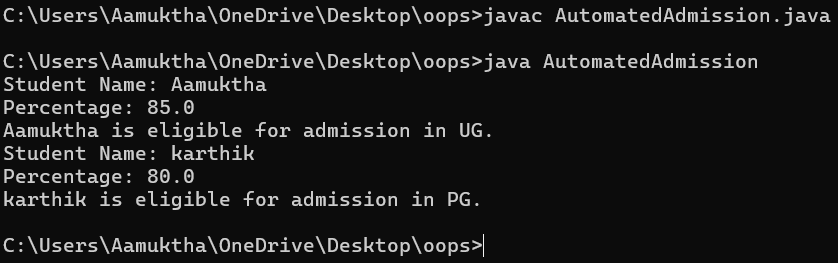
**PROGRAM-2:**

**AIM:** A college is developing an automated admission system that verifies students eligibility(UG) and postgraduation(PG) programs. Each program hasdifferent 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:**

|  |  |
| --- | --- |
| **Code error** | **Code 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. |

**IMPORTANT POINTS:**

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

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

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

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

|  |
| --- |
| **AutomatedAdmission** |
| * Scanner: scanner * Name: String * Percentage : double * Program: stirng |
| + main(args:String[]): void  +takeInput(): void  +checkEligibility(): void  +closeScanner(); void |

**CLASS DIAGRAM:**

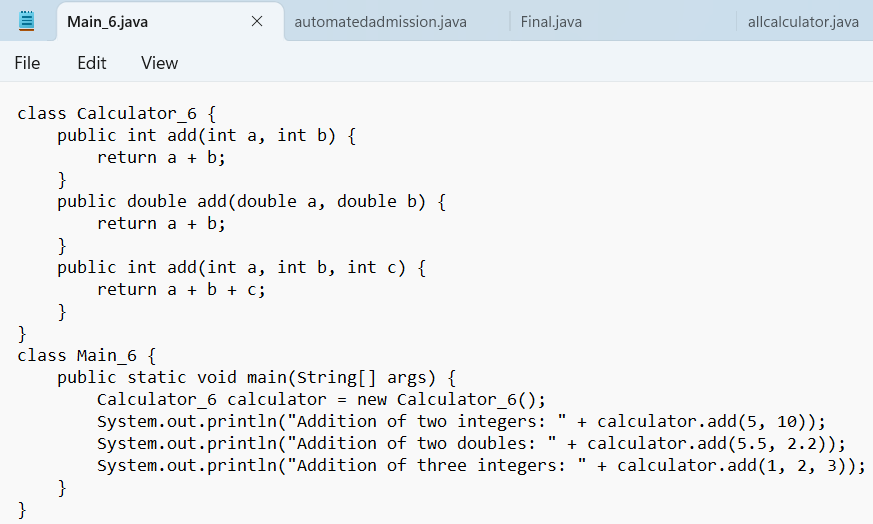
**PROGRAM-3:**

**AIM:** Create a calculator class with overloaded methods to perform addition of:

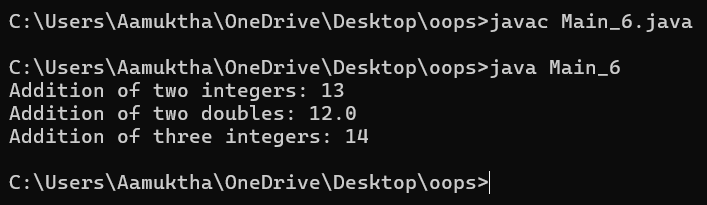
1. Add two integers

2. Add two doubles

3. Add three integers

**CODE:** ****

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1.Method parameters missing spaces. E.g.,”inta, intb”should be “int a, int b”  2.Inconsistent indentation in method bodies | 1**.** Add proper spacing between parameters: (int a, int b)  2.Fix indentation:  Consistent 4 space o indentation. |

**IMPORTANT POINTS:**

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

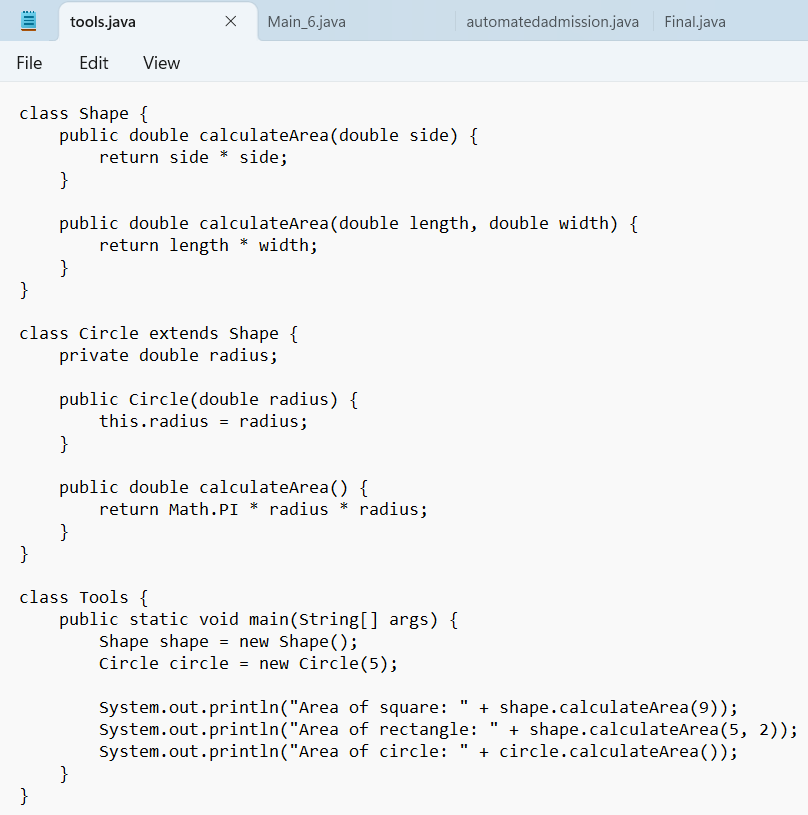
**2.Automatic Method Selection:** Java selects the appropriate add method based on the argument types during compilation.

**CLASS DIAGRAM:**

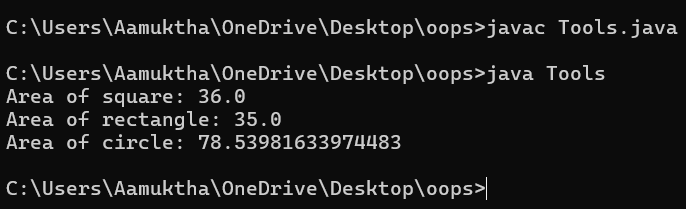
|  |
| --- |
| **Calculator** |
| + add(int, int): int  +add(double, double): double  +add(int,int,int): int |

**PROGRAM-4:**

**AIM:** Create a shape class with a method to calculate area i.e., overloaded for different shapes eg: Squares, Recatangle. Then create a subclass circle that overrides the calculateArea() method for a circle.

**CODE:** ****

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Method calls in main are missing an object reference (e.g., calculateArea(4) instead of s.calculateArea(4)).  2. Circle class method does not override theparent 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. |

**CLASS DIAGRAM:**

|  |
| --- |
| **SHAPE** |
| + CalculateArea(side:double): double  +CalculateArea(width: double, length: double): double |

|  |
| --- |
| **CIRCLE** |
| + CalculateArea(radius: double): double |

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

**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)).

**WEEK – 07**

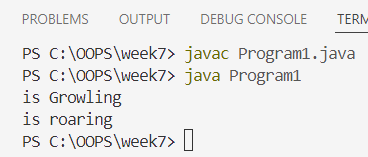
**PROGRAM1**

**AIM:** create a Java program to create an abstract class 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:**

****

**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. |

|  |
| --- |
| **<<abstract>>**  **ANIMAL** |
| + sound(): void |

**CLASS DIAGRAM:**

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

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

**IMPORTANT POINTS:**

* Animal is an abstract class with an abstract method sound().
* Lion and Tiger classes extend Animal and override sound().
* Abstract methods must be implemented in child classes.
* Abstract classes cannot be instantiated directly.
* Method overriding allows each subclass to define its own version of sound()

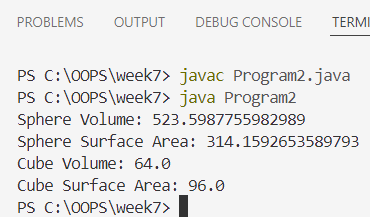
**PROGAM 2**

**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 Shape 3D class and implement the respective methods to calculate volume and surface area of each shape.

**CODE:**

****

**OUTPUT:**

****

**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 |

**IMPORTANT POINTS:**

* Shape3D is an abstract class with two abstract methods: calculateVolume() and calculateSurface().
* Sphere and Cube classes extend Shape3D and provide specific implementations.
* Sphere Volume: (4/3) \* π \* radius³ using Math.pow(radius, 3).
* Sphere Surface Area: 4 \* π \* radius² using Math.pow(radius, 2).

**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

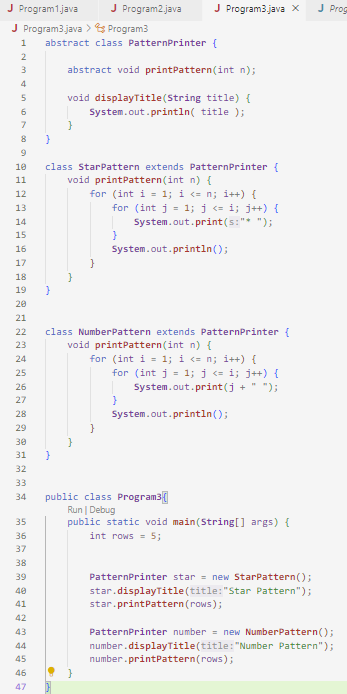
Create two subclasses:

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

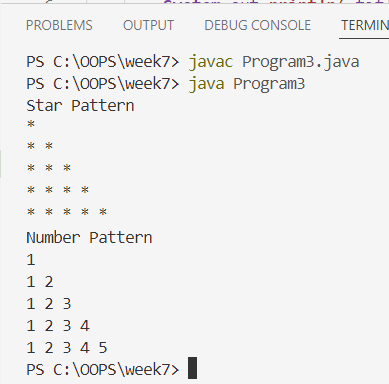
2.NumberPattern: Prints a right-angled triangle of increasing numbers

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

**CODE:**

****

**OUTPUT:**

****

**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 |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<abstract>> PatternPrinter** |
| +displayTitle(title)  +printPattern(n) |

|  |
| --- |
| **StarPattern** |
| +Printpattern (n) |

|  |
| --- |
| **NumberPattern** |
| +Printpattern (n) |

**IMPORTANT POINTS:**

1.PatternPrinter is an abstract class with an abstract method printPattern(int n).

2.It also has a concrete method displayTitle(String title) to print a title.

3.StarPattern and NumberPattern extend PatternPrinter and implement printPattern().

4.StarPattern prints a right-angled triangle using "\*" characters.

**WEEK – 8**

**PROGRAM-1:**

**AIM:** Write a java program shape with get perimeter() method and create three class rectangle circle triangle that implement the shape interface and implement the get perimeter for all the 3 classes.

**CODE:**

interface Shape{

    double getPerimeter();

}

class Rectangle implements Shape{

    double width;

    double length;

 Rectangle(double length,double width){

    this.width=width;

    this.length=length;

 }

 public double getPerimeter(){

    return 2\*(length=width);}

}

class Circle implements Shape{

    double radius;

Circle (double radius){

    this.radius=radius;

}

public double getPerimeter(){

    return 2\*Math.PI\*radius; }

}

class Triangle implements Shape{

    double side1;

    double side2;

    double side3;

    Triangle(double s1,double s2 ,double s3){

        this.side1=s1;

        this.side2=s2;

        this.side3=s3;

    }

public double getPerimeter(){

    return side1+side2+side3; }

}

public class Program1{

    public static void main(String[] args) {

        Shape rect= new Rectangle(5,4);

        Shape circle=new Circle(3);

        Shape tri=new Triangle(2,5,3);

        System.out.println("Name: E.Aamuktha, Roll No.:24047, Class: Sec A");

        System.out.println("Rectangle Perimeter: " + rect.getPerimeter());

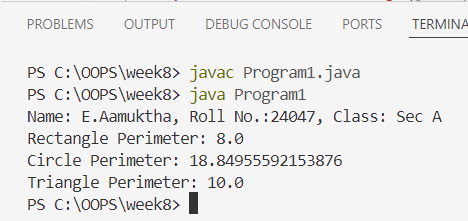
        System.out.println("Circle Perimeter: " + circle.getPerimeter());

        System.out.println("Triangle Perimeter: " + tri.getPerimeter());

    }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1. Missed Math.PI in Circle perimeter,  2.Did not give meaningful class name. | 1. Rectified to Math.PI in Circle perimeter. 2. Rectified |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<interface >> SHAPE** |
| + getPerimeter(): double |

|  |
| --- |
| **Rectangle** |
| - length: double  - width: double |

|  |
| --- |
| **Circle** |
| - radius: double |

|  |
| --- |
| + getPerimeter(): double |

|  |
| --- |
| **Triangle** |
| - side1: double  - side 2: double  - side 3: double |

|  |
| --- |
| + getPerimeter(): double |

|  |
| --- |
| + getPerimeter(): double |

**IMPORTANT POINTS:**

* Uses an interface Shape with a method getPerimeter().
* Implements Shape in Rectangle, Circle, and Triangle classes.
* Demonstrates polymorphism by referencing different shapes via the Shape interface.
* Contains a bug in Rectangle's perimeter calculation (uses length = width instead of length + width).

**PROGRAM – 2**

**AIM: :** java program to create an interface playable with method play() that takes no argurments and return void create subclasses volleyball basketball football that implements playabale interface and override the play.

**CODE:**

interface Playable {

    void play();

}

class Volleyball implements Playable {

    public void play() {

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

    }

}

class Basketball implements Playable {

    public void play() {

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

    }

}

class Football implements Playable {

    public void play() {

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

}

public class Program2 {

    public static void main(String[] args) {

        System.out.println("Name: E.Aamuktha, Roll No.:24047, Class: Sec A");

        Playable v = new Volleyball();

        Playable b = new Basketball();

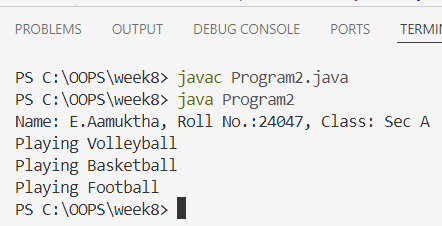
        Playable f = new Football();

        v.play();

        b.play();

        f.play();

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1.Wrote System.out.println(play);  2. forgot public in Method play()   |  | | --- | |  | | 1.Rectified to v.play();  2.used public void play() in all implementing classes |

**CLASS DIAGRAM:**

|  |
| --- |
| **<<interface>>**  **Playable** |
| + play(): void |

|  |
| --- |
| **Basketball** |
| + play(): void |

|  |
| --- |
| **Volleyball** |
| + play(): void |

|  |
| --- |
| **Football** |
| + play(): void |

**IMPORTANT POINTS:**

* Interface Usage – Playable interface defines a play() method.
* Implementation – Volleyball, Basketball, and Football all implement Playable.
* Polymorphism – Different objects are referenced using the Playable interface.
* Correct Output Method – play() is called using the object, not printed directly.

**PROGRAM-3**

**AIM:** Write a java program to create a login system using interface.

**CODE:**

import java.util.Scanner;

interface LoginSystem {

    void login();}

class UserLogin implements LoginSystem {

    String correctUsername = "amrita\_student";

    String correctPassword = "amrita@123";

    public void login() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Amrita Portal Username: ");

        String username = sc.nextLine();

        System.out.print("Enter Password: ");

        String password = sc.nextLine();

        if (username.equals(correctUsername) && password.equals(correctPassword)) {

            System.out.println("Login successful! Welcome to Amrita University Portal, " + username + "!");

        } else {

            System.out.println(" Login failed! Incorrect Amrita credentials.");}

    }

}

public class Program3{

    public static void main(String[] args) {

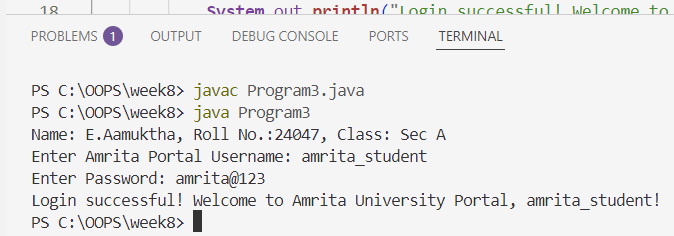
        System.out.println("Name: E.Aamuktha, Roll No.:24047, Class: Sec A");

        LoginSystem user = new UserLogin();

        user.login(); }

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| **Code error** | **Code rectification** |
| 1.Forgot to use .equals() for string comparison  2. Didn't import java.util.Scanner.  3. Called user.login() with parameters   |  | | --- | |  | | 1. Used username.equals(correctUsername) not ==  2. Added import java.util.Scanner;  3. Correct is user.login(); with no parameters |

**CLASS DIAGRAM:**

|  |
| --- |
| <<interface>>  LoginSystem |
| + login(): void |

|  |
| --- |
| UserLogin |
| **-** correctUsername: String  - correctPassword: String |
| + login(): void |

**IMPORTANT POINTS:**

* Interface LoginSystem defines a login() method.
* UserLogin class implements LoginSystem and provides logic for user authentication.
* Scanner class is used to take user input for username and password.
* Hardcoded credentials simulate actual login verification.