**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: E.Kundan Sri Vyshnavi**

**Verified By Roll No: 24105**

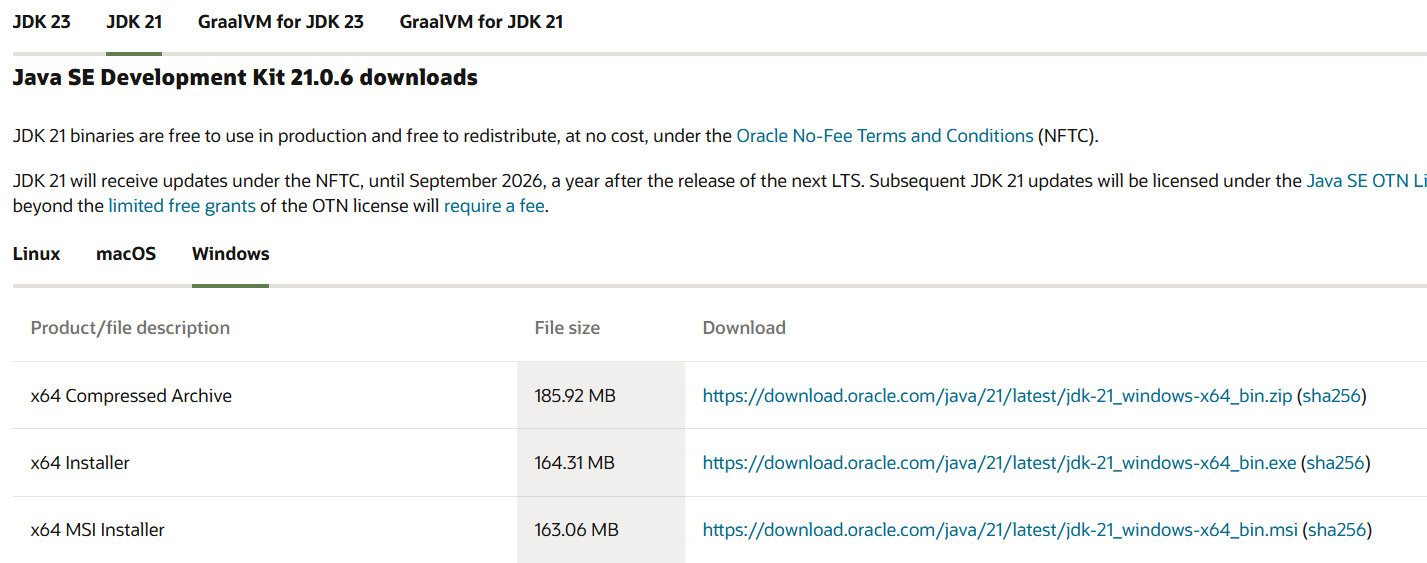
**Class : CSE-B**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | | **Title** | **Date** | | **Page No.** | | **Signature** | |
| **Week 1** | |  | 29-01-2025 | |  | |  | |
| **1.** | | How to download and install Java Software. |  | |  | |  | |
| **2.** | | Write a Java Program to print the message “Welcome to Java Programming”. |  | |  | |  | |
| **3.** | | Write a Java program that prints: Name, Roll.no. , section of a student. |  | |  | |  | |
| **Week 2** | |  | 12-02-2025 | |  | |  | |
| **1.** | | Write a java program to print Fibonacci series |  | |  | |  | |
| **2.** | | Write a java program to temperature from Celsius to Fahrenheit and vica-versa. |  | |  | |  | |
| **3.** | | Write a java program to calculate the simple interest. |  | |  | |  | |
| **4.** | | Write a java program to print Area of triangle using heron’s formula |  | |  | |  | |
| **5.** | | Write a java program to find the factorial of a number. |  | |  | |  | |
| **Week 3** | |  | 05-03-2025 | |  | |  | |
| **1.** | | 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”. |  | |  | |  | |
| **2.** | | 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 withdraw(): Whenever an amount is withdrawn, it has to be less than the current amount , else print (“Insufficient funds”) |  | |  | |  | |
| **Week 4** | |  | 8-03-2025 | |  | |  | |
| **1.** | | Write a java Program with class named Book:   1. The class should contain various attributes such as “title\_of\_book, Author, year\_of\_publication”. 2. It should also contain a constructor with parameters which initializes “title\_of\_book, Author, year\_of\_publication”. 3. Create a method which displays the details of the book “title\_of\_book, Author, year\_of\_publication”. 4. Display the details of the two books by creating two objects. |  | |  | |  | |
| **2.** | | To create a java program with class named Myclass:   1. with “static variable-count” of int type, initialize to zero and a constant variable “pi-double” to initialize to 3.1415 as attributes of that class. 2. Now define a constructor for Myclass that increments the count variable each time object for Myclass is created. Finally print values of “count” and “pi” variables. |  | |  | |  | |
| Week 5 |  | | | 12-03-2025 | |  | |  |
| 1. | Create a calculator, using the operations including addition, subtraction, multiplication and division using multilevel inheritance and display the desired output. | | |  | |  | |  |
| 2. | A vehicle rental company wants to develop a system, that maintains information about different types of vehicles available for rent. The company rents out the cars and bikes and they need a program to store details about each vehicle, such as brand and speed.   1. Cars should have additional properties (attributes/variables):no\_doors, seating capacity. 2. Bikes should have a property indication of 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. Each class should have a constructor. | | |  | |  | |  |
| Week 6 |  | | | 26-03-2025 | |  | |  |
| 1. | 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. [car\_company, car\_model, car\_prize, seating\_capacity,diesel] | | |  | |  | |  |
| 2. | A college is developing an automated admission system that verifies student’s eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility criteria based on the student’s percentage in their previous qualifications.   * UG admission requires minimum 60% * PG admission requires minimum 70% | | |  | |  | |  |
| 3. | Create a calculator class with overloaded methods to perform addition.   * add 2 integers * add 2 doubles * add 3 integer | | |  | |  | |  |
| 4. | Create a shape class with a method calculateArea() that is overloaded for different shape (eg: square, rectangle). Then, create a subclass Circle that overrides the calculateArea() method for a class. | | |  | |  | |  |
|  |  | | |  | |  | |  |

**OBJECT ORIENTED PROGRAM**

**1) Process of installing JDK**

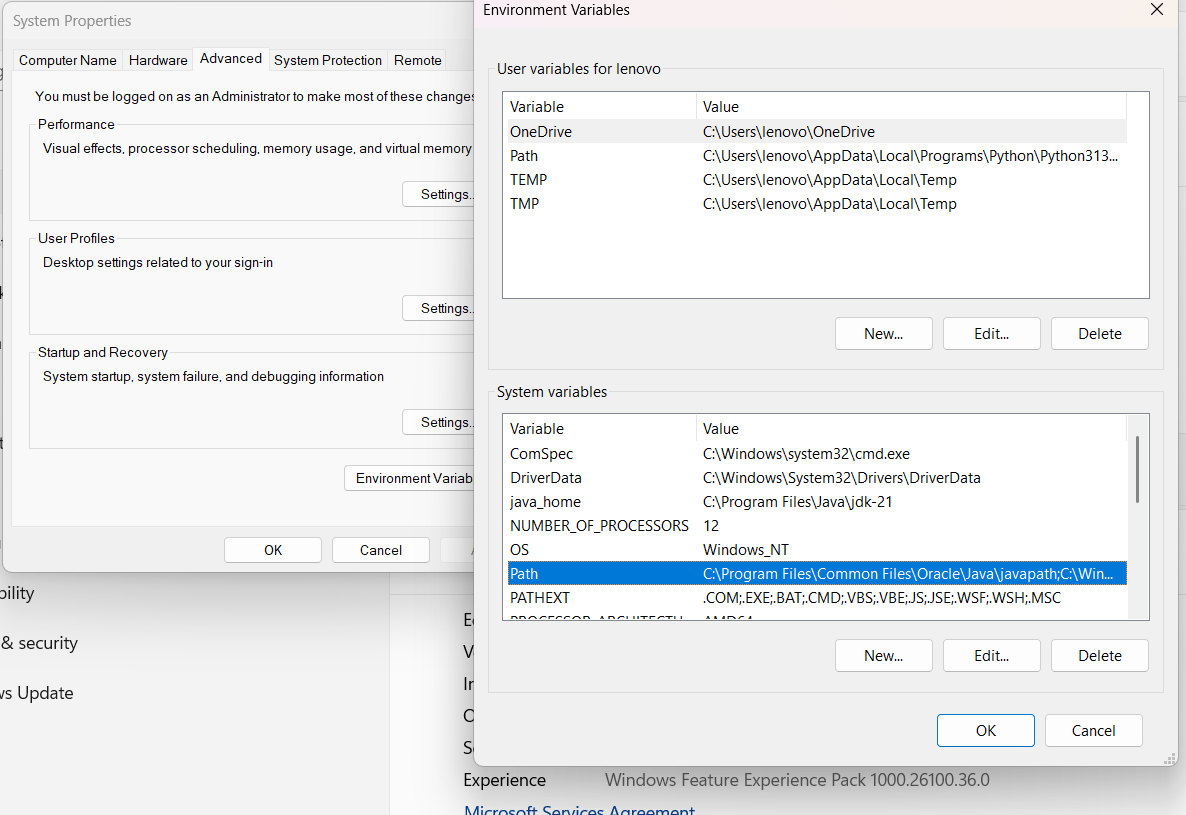
* Open browse and search JDK download,next click the first link(oracle).
* Open and choose JDK 21 version,there choose the operating system windows.
* Click the x64 installer download link.



**INSTALLATION:**

* After downloading,run the installer.
* Follow the instructions and click ‘Next’ until it’s done.

**Setting environmental variables:**

* Open file explorer, then right click on the PC.next click 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 Variablesand Set Variable name as: java\_home
* Variable value: The folder address where JDK is installed (C:\Program Files\Java\jdk-21)
* Find Path under **System Variables**, click **Edit**, and add the path of the jdk-21(C:\Program Files\Java\jdk-21) 

**Checking of JDK Version:**

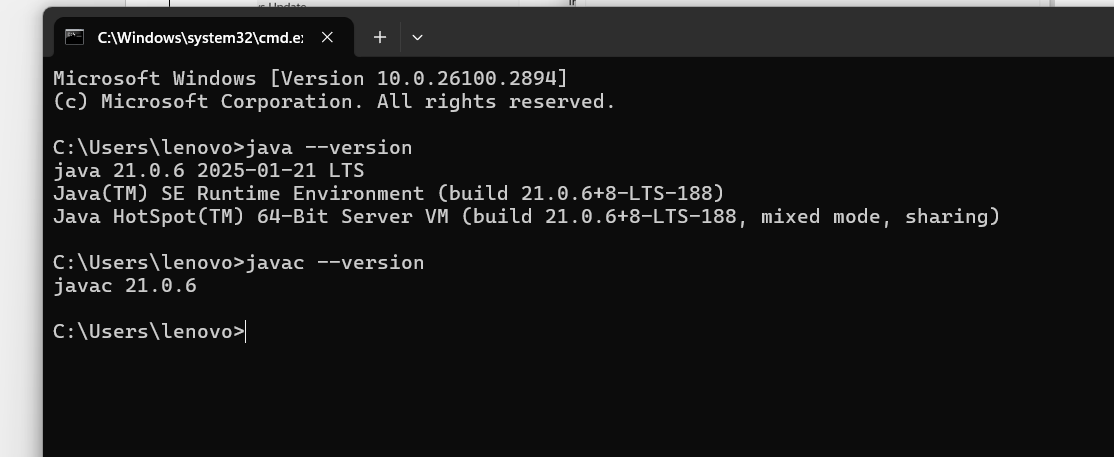
1. Open Command Prompt:

Press window+R, type cmd, and press Enter.

1. Check Version:

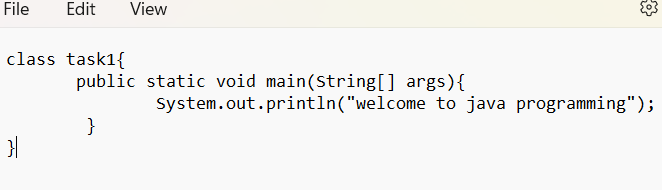
Type java --version and press Enter.

Type javac --version and press Enter.

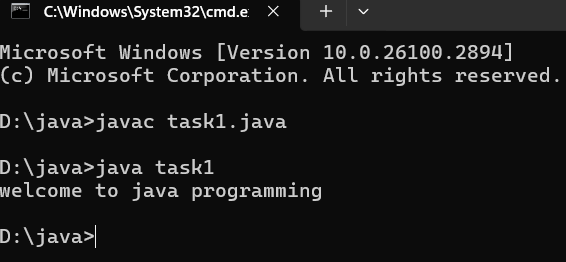


**2) Write a Java Program to print the message “Welcome to Java Programming”**

**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.

**3) printing Name, Class, Roll No, of a Student by using java.**

Write your code in Notepad and execute in cmd prompt

**CODE:**

public class main

{

public static void main(String[]args)

{

System.out.println("name:E.kundan sri vyshnavi");

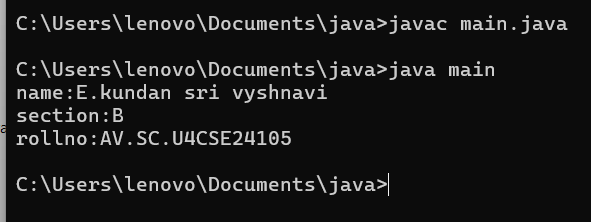
System.out.println("section:B");

System.out.println("rollno:AV.SC.U4CSE24105");

}

}

**Output:**

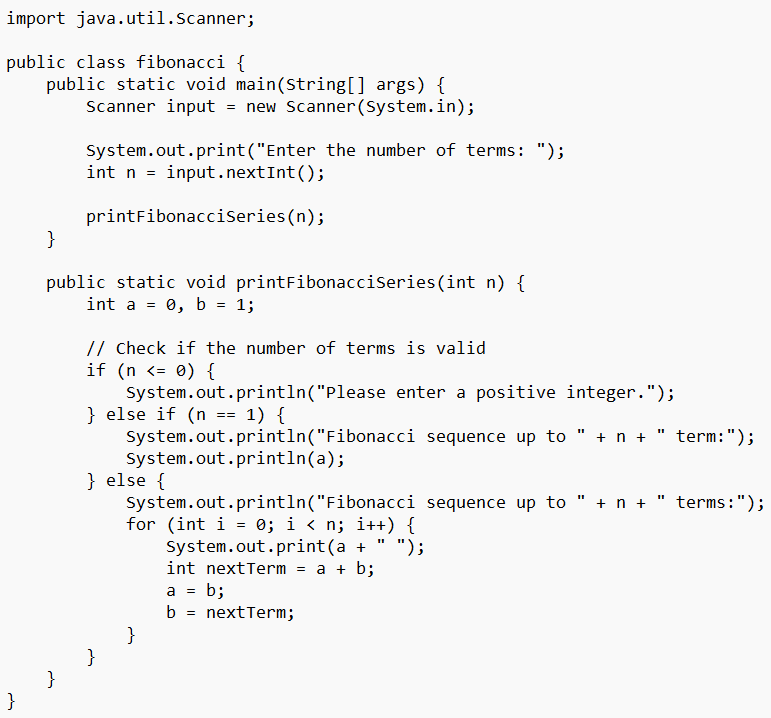
****

**WEEK 2**

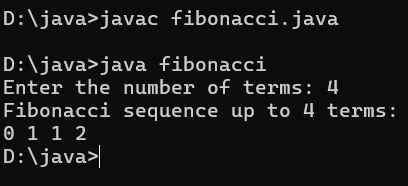
**Program 1**

**Write a java program to print Fibonacci series**

**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 |

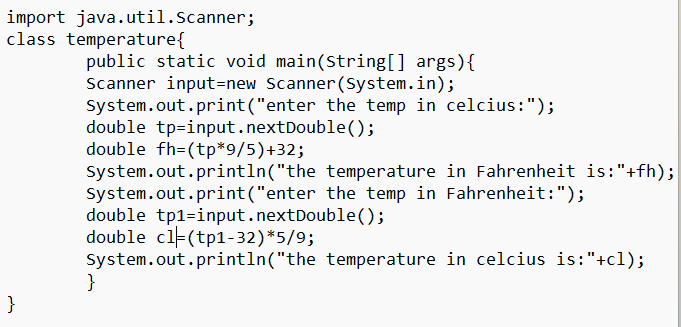
**Concepts to be known:**

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.

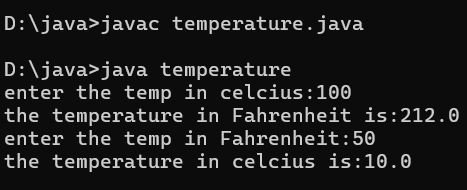
**Program 2**

**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:"); |

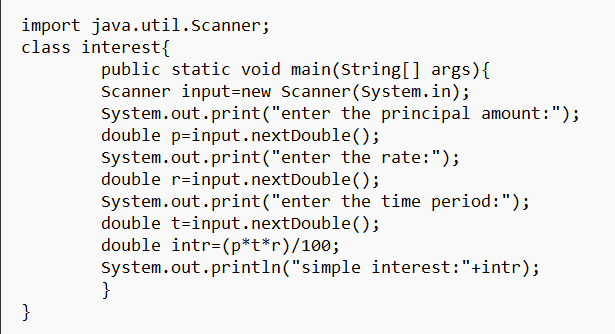
**Concepts to be known:**

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.

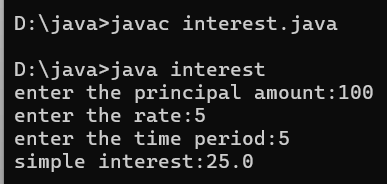
**Program 3:**

**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(); |

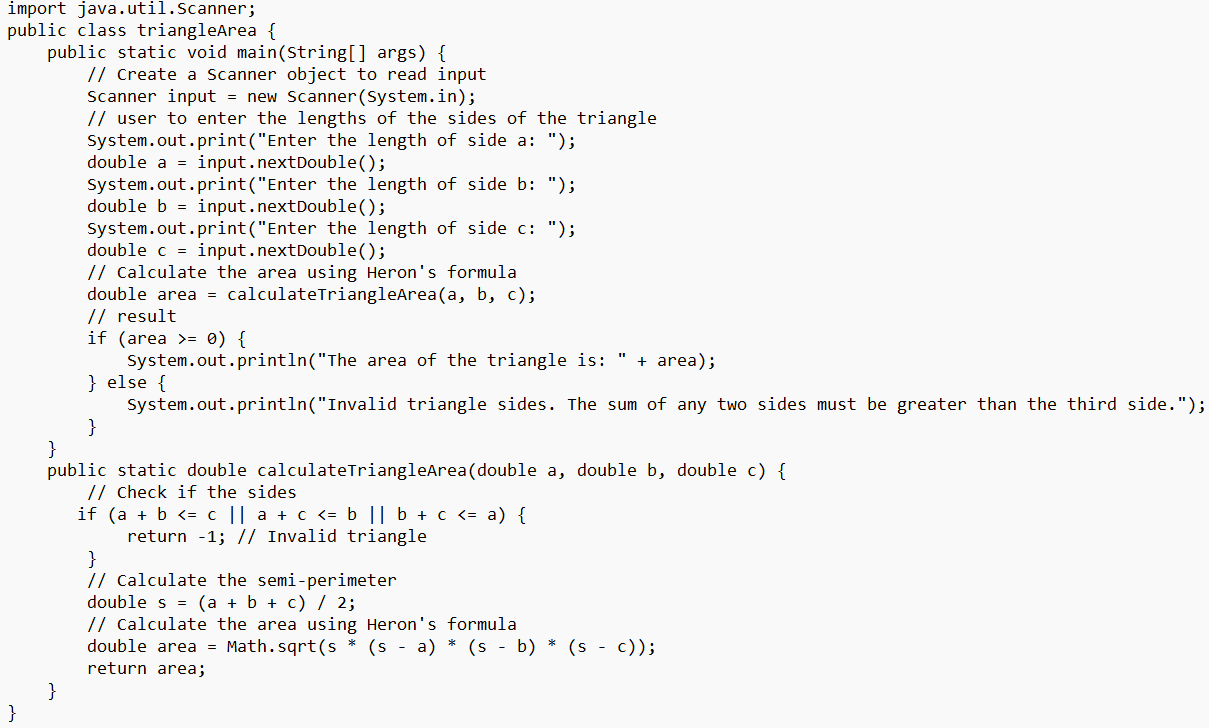
**Concepts to be known:**

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.

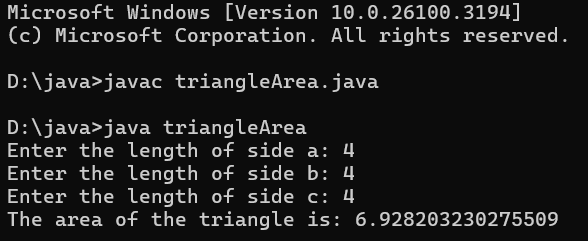
**Program 4**

**Write a java program to print Area of triangle using heron’s formula**

**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:**

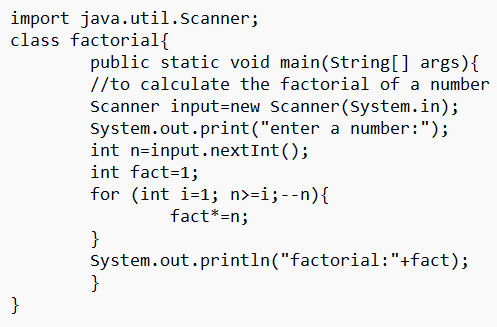
1. import java.util.Scanner; - To accept input from user, Scanner class under util package has to be imported.
2. Scanner input=new Scanner(System.in); - Used to create a Scanner object
3. int a=input.nextInt (); - Used to read integer data type stored under the object created
4. int result=(a>b)? ((a>c)?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

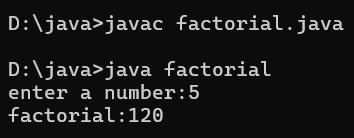
**Program 5**

**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.

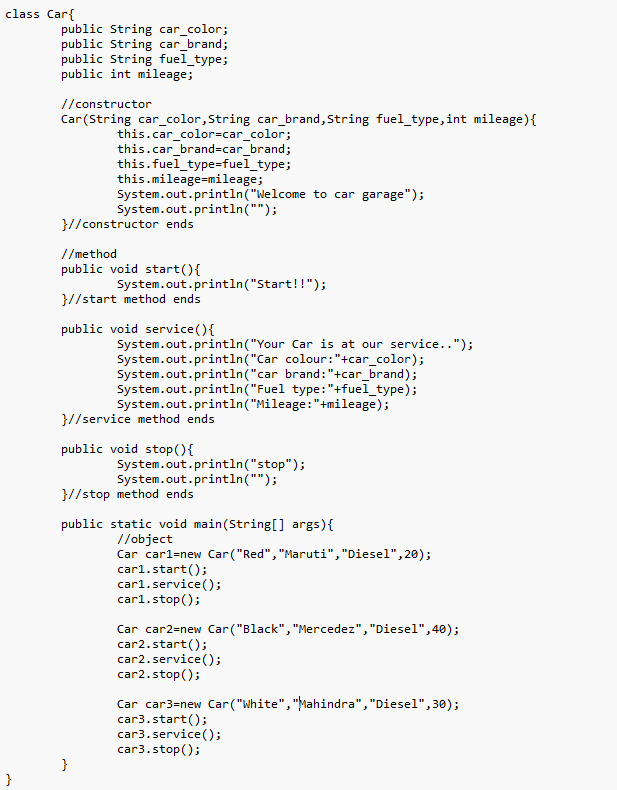
**WEEK 3**

**Program 1**

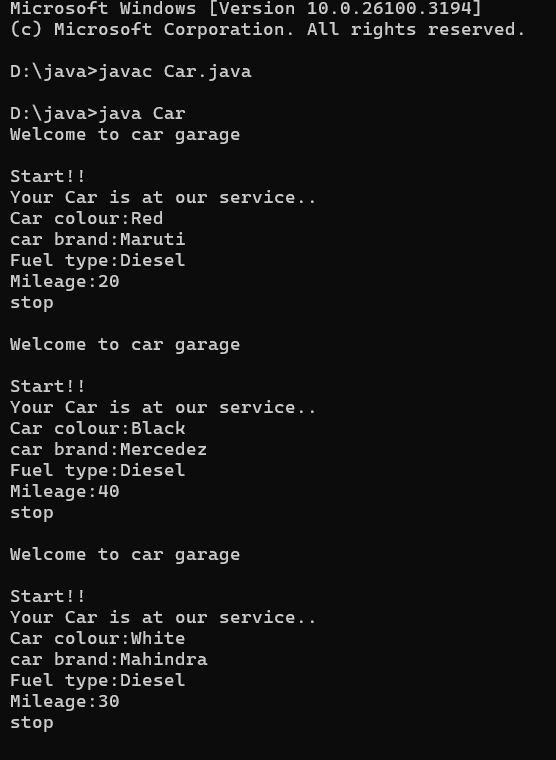
**Aim:** To create a java program with the following instructions:

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:**

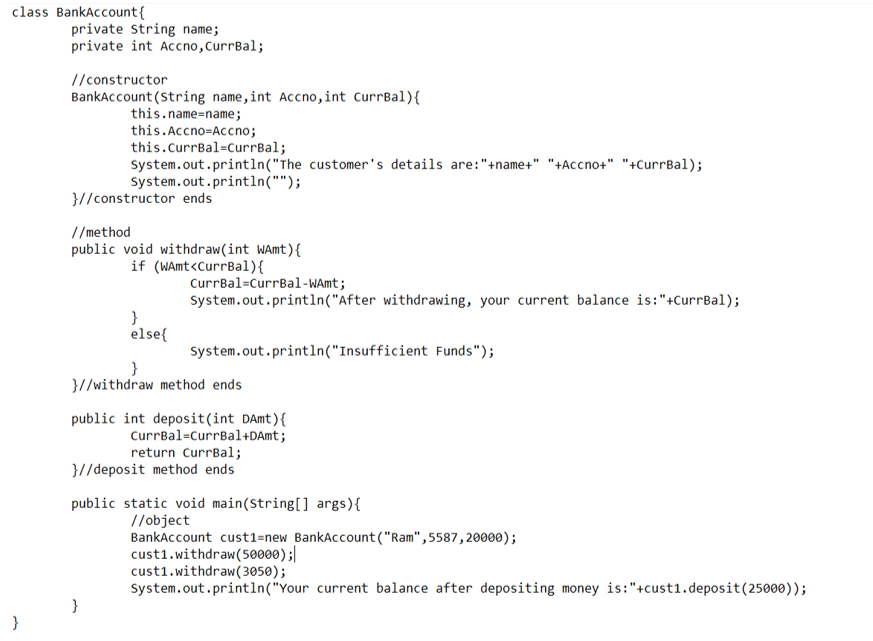
1. public String car\_color; - Used to declare a variable named car\_color, with data type as String with public accessibility.
2. Car(String car\_color,String car\_brand,String fuel\_type,int mileage){ } – It is a constructor (method with name same as class), which requires parameters such as car\_color (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**

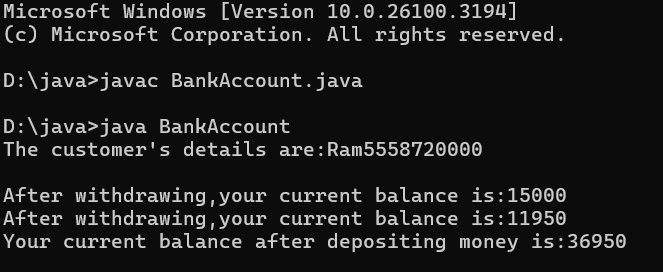
**Aim:** To write a java program to create a class named BankAccount, with 2 methods deposit() and withdraw().

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.

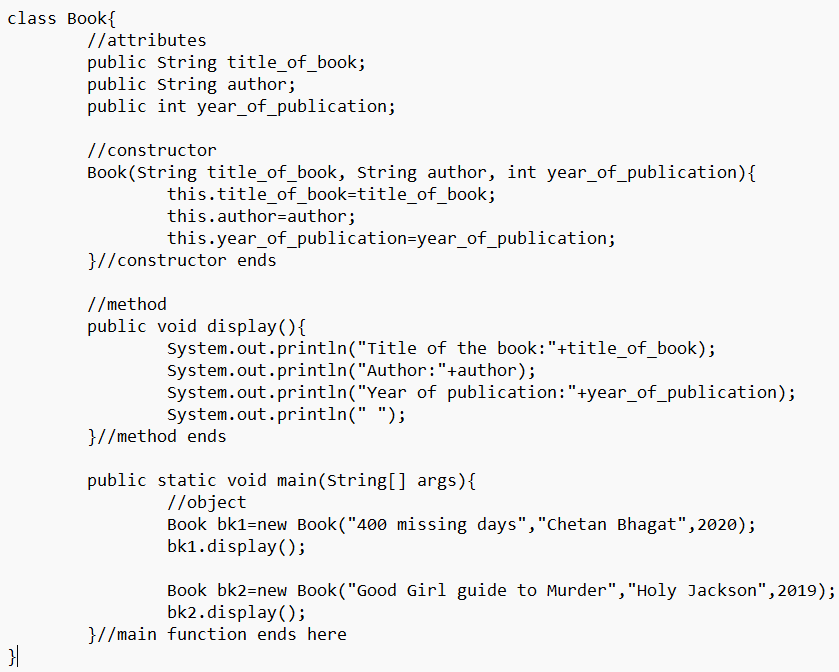
**WEEK 4**

**Program 1**

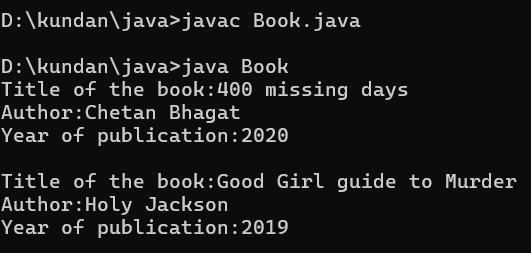
**Aim:** Write a java Program with class named Book:

1. The class should contain various attributes such as “title\_of\_book, Author, year\_of\_publication”.
2. It should also contain a constructor with parameters which initializes “title\_of\_book, Author, year\_of\_publication”.
3. Create a method which displays the details of the book “title\_of\_book, Author, year\_of\_publication”.
4. Display the details of the two books by creating two objects.

**Code:**



**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: illegal start of type  public void display({ | Add a “)”  public void display(){ |
| 2. | error: cannot find symbol thisauthor=author; | Add a “.”  this.author=author; |

**Class Diagram:**

|  |
| --- |
| Book |
| + title\_of\_book : String  + author: String  + year\_of\_publication: int |
| Book():void  + display():void |

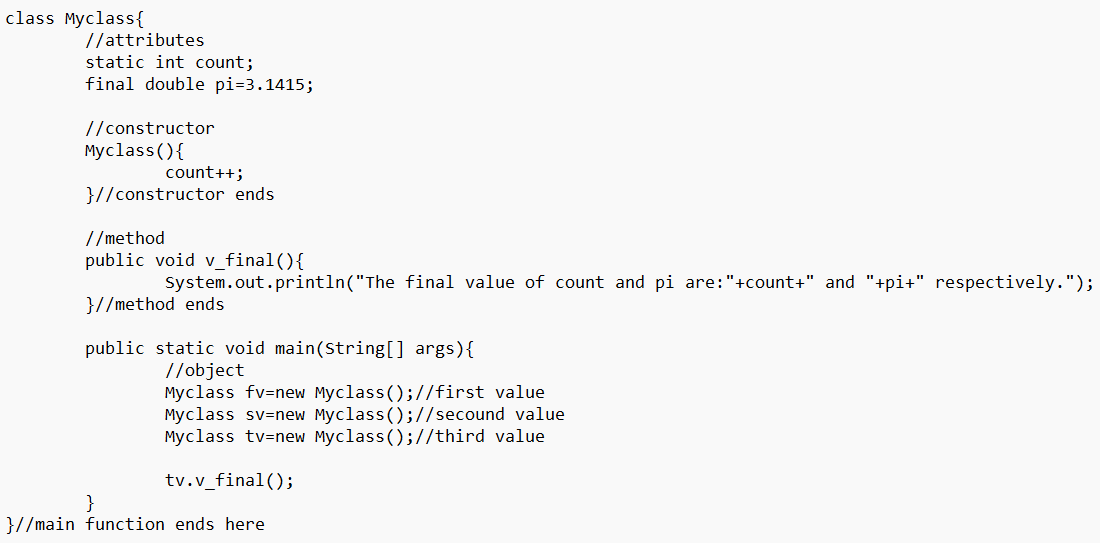
**Concepts to be known:**

1. public String title\_of\_book; - Used to declare a variable named title\_of\_book, with data type as String with public accessibility.
2. Book(String title\_of\_book, String author, int year\_of\_publication){} – It is a constructor (method with name same as class), which requires parameters such as year\_of\_publication (int data-type) and so on.
3. this.author=author; - “this” is a default method, which is used to point to the instance variables.
4. public void display() {} – used to declare a method, which will return nothing(void) in public accessibility, which requires a parameter WAmt(integer data type).
5. Book bk1=new Book("400 missing days","Chetan Bhagat",2020); - used to create a object in class Book, with object name as bk1.
6. bk1.display();- Calling a method, under object bk1

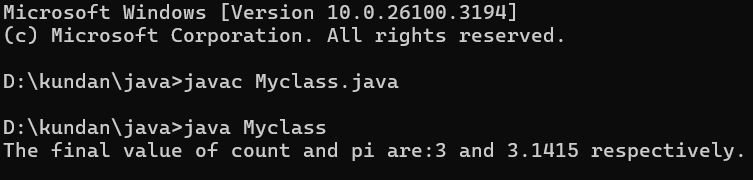
**Program 2**

**Aim:** To create a java program with class named Myclass:

1. with “static variable-count” of int type, initialize to zero and a constant variable “pi-double” to initialize to 3.1415 as attributes of that class.
2. Now define a constructor for Myclass that increments the count variable each time object for Myclass is created. Finally print values of “count” and “pi” variables.

**Code:**

**Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  tv.v\_final() | Add a “;”  tv.v\_final(); |
| 2. | error: illegal start of type  public void v\_final({ | Add a “)”  public void v\_final (){ |

**Class Diagram:**

|  |
| --- |
| Myclass |
| count: int  pi : double |
| Myclass()  + v\_final():void |

**Concepts to be known:**

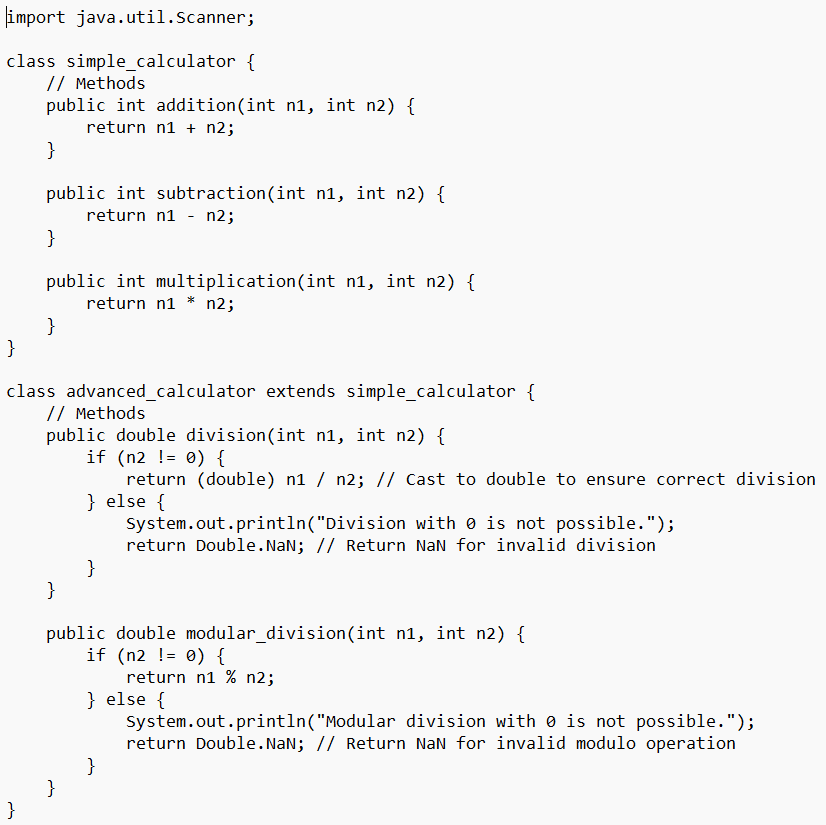
1. static int count; - Used to declare a variable named count, with data type as integer, which is static(class-level variable, i.e. it can be accessed via class).
2. final double pi=3.1415; - Used to declare a variable named pi, with data type as double, whose value is fixed throughout the program(final-is a reserved keybord for such conditions).
3. Myclass(){} – It is a constructor (method with name same as class).
4. count++; - Used to increment the value of variable count by 1.
5. public void v\_final() {} – Used to declare a method, which will return nothing(void) in public accessibility.
6. Myclass tv=new Myclass(); - Used to create an object in class Myclass, with object name as tv.
7. tv.v\_final();- Calling a method, under object tv.

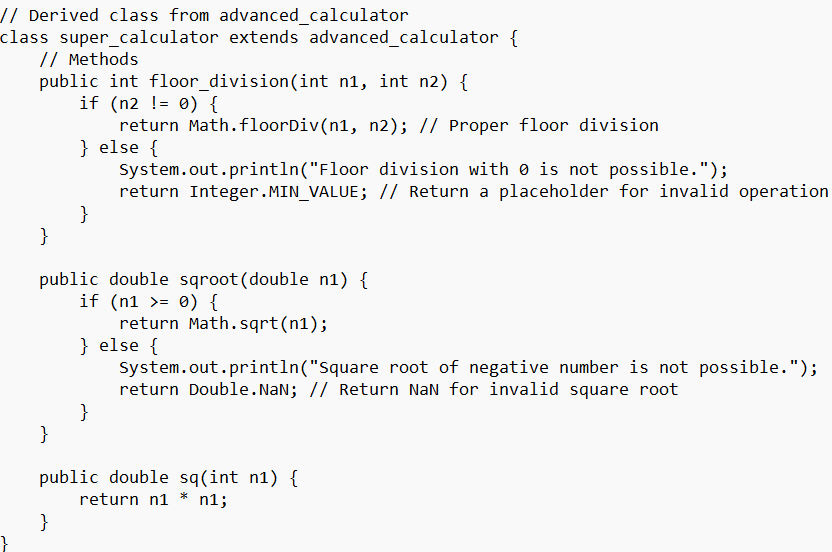
**WEEK 5**

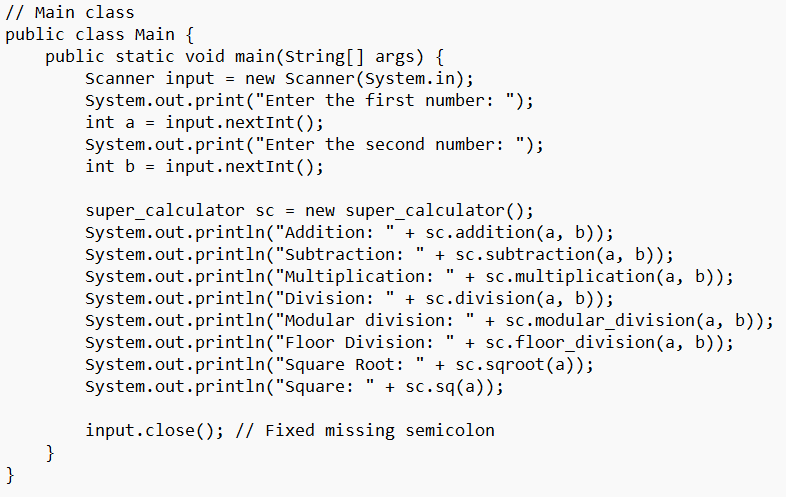
**Program 1**

**Aim:** Create a calculator, using the operations including addition, subtraction, multiplication and division using multilevel inheritance and display the desired output.

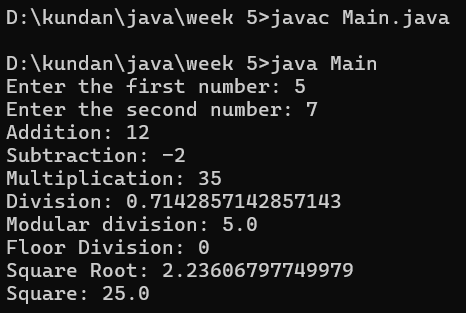
**Code:**

****

****

****

**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  tv.v\_final() | Add a “;”  tv.v\_final(); |
| 2. | error: illegal start of type  public void v\_final({ | Add a “)”  public void v\_final (){ |

**Class Diagram:**

|  |
| --- |
| simple\_calculator |
| **+** addition(n1:int,n2:int):int  +subtraction(n1:int,n2:int):int  +multiplication(n1:int,n2:int):int |

|  |
| --- |
| advanced\_calculator |
| + division(n1:int,n2:int):double  +modular\_division(n1:int,n2:int):double |

|  |
| --- |
| super\_calculator |
| + floor\_division(n1:int,n2:int):double  + sqroot(n1:double): double  +sq (n1:int):double |

**Concepts to be known: (Multi-level inheritance)**

1. public int addition(int n1, int n2){ - Method name is addition, whose accessibility is public. It takes it’s parameters in integer data type and returns an integer data type.
2. class advanced\_calculator extends simple\_calculator{ - Class named advanced\_calculator inherits it’s some of the properties from parent class simple\_calculator
3. return Double.NaN; - Nan stands for Not-a-Number. This is returned when mathematical operation results in an undefined value.It is a part of double class and is used in floating point calculation.

**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 the cars and bikes and they need a program to store details about each vehicle, such as brand and speed.

1. Cars should have additional properties (attributes/variables):no\_doors, seating capacity.
2. Bikes should have a property indication of 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. Each class should have a constructor.

**Additional question:** i) Which OOP concept is used in the above program? Explain why is it useful in this scenario?

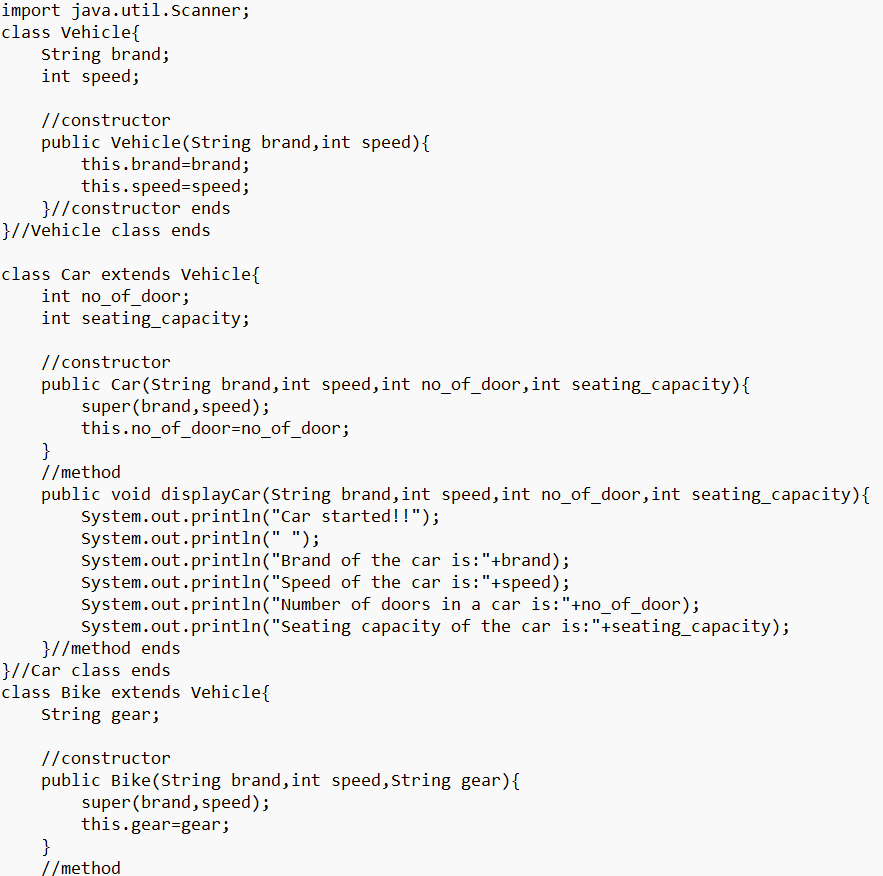
ii) If the company decides to add a new type of vehicle: “Truck”. How would you modify the above program?

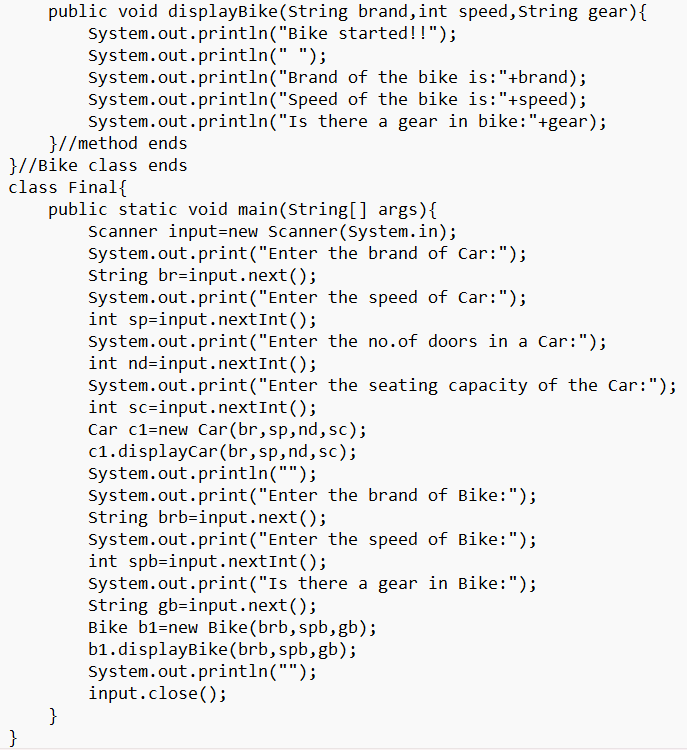
* Truck should include an additional property called capacity(in tons)
* Create a showTruckdetails() to display the Truck’s capacity
* Write a constructor for Truck that initializes all properties.

iii) Implement the Truck class and update the main method to create a truck object and also create an object for car and bike subclasses.

Finally display all the details

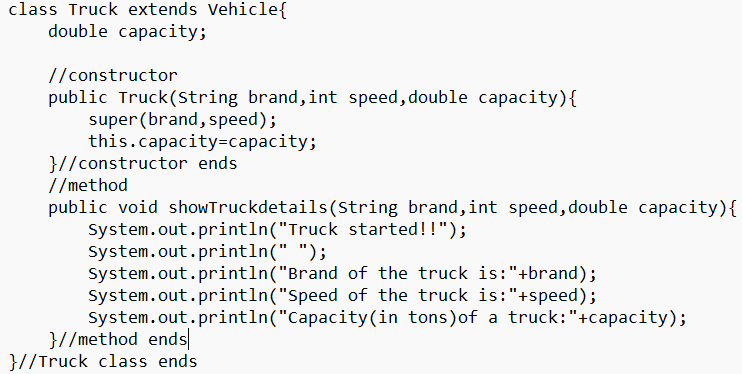
**Code:**

****

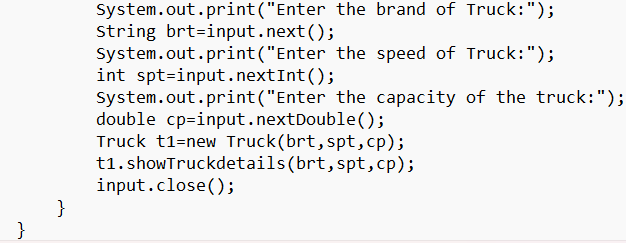
****

**Additional Part:**

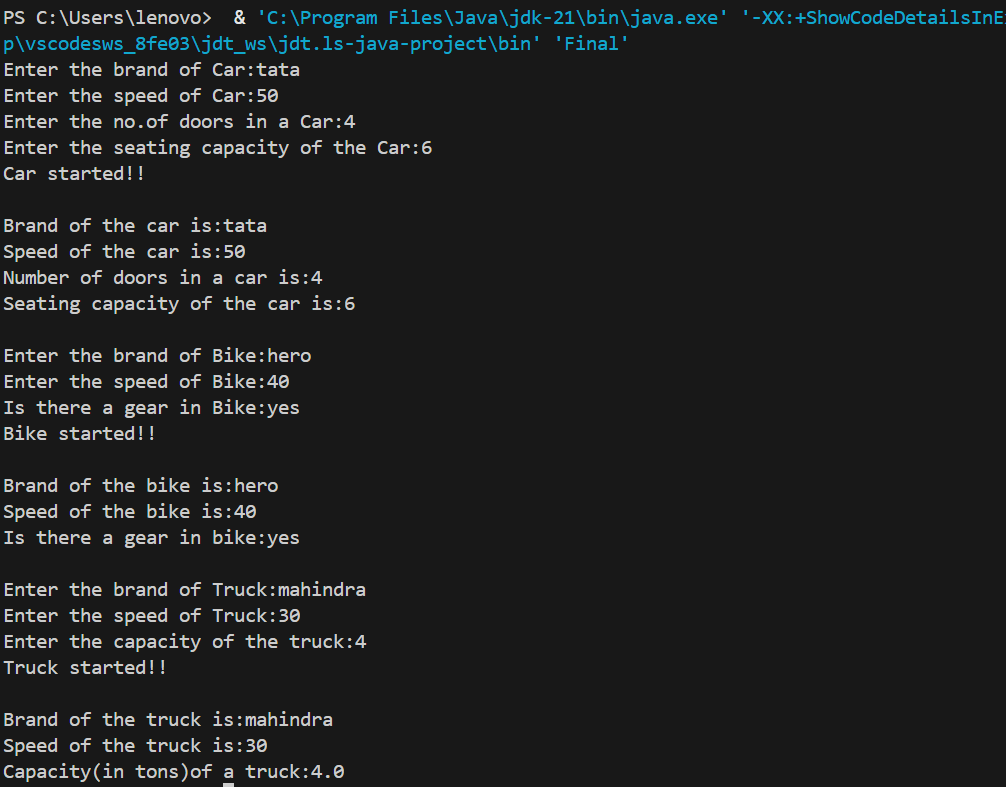
1. Hierarchial Inheritance is used.
2. Modified code:



1. In the Final class:

****

**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Error Message** | **Error Rectification** |
| 1. | error: ';' expected  this capacity=capacity | Add a “.”  this.capacity=capacity |
| 2. | error: illegal start of type  super(brand,speed; | Add a “)”  super(brand,speed){ |

**Class Diagram:**

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

|  |
| --- |
| Truck |
| +capacity:double |
| +Truck(brand:String,  speed:int,capacity:double)  :void  +showTruckdetails( brand:String,  speed:int,capacity:double)  :void |

|  |
| --- |
| Car |
| +no\_of\_door:int  +seating\_capacity:int |
| +Car(brand:String,speed:int,  No\_of\_doors:int,seating\_  capacity:int):void  +displayCar(brand:String,speed:int,  No\_of\_doors:int,seating\_  capacity:int):void |

|  |
| --- |
| Bike |
| +gear:String |
| +Bike(brand:String,  speed:int,gear:String)  :void  +displayBike(brand:String,  speed:int,gear:String):void |

**Concepts to be known: (Hierarchial Inheritance)**

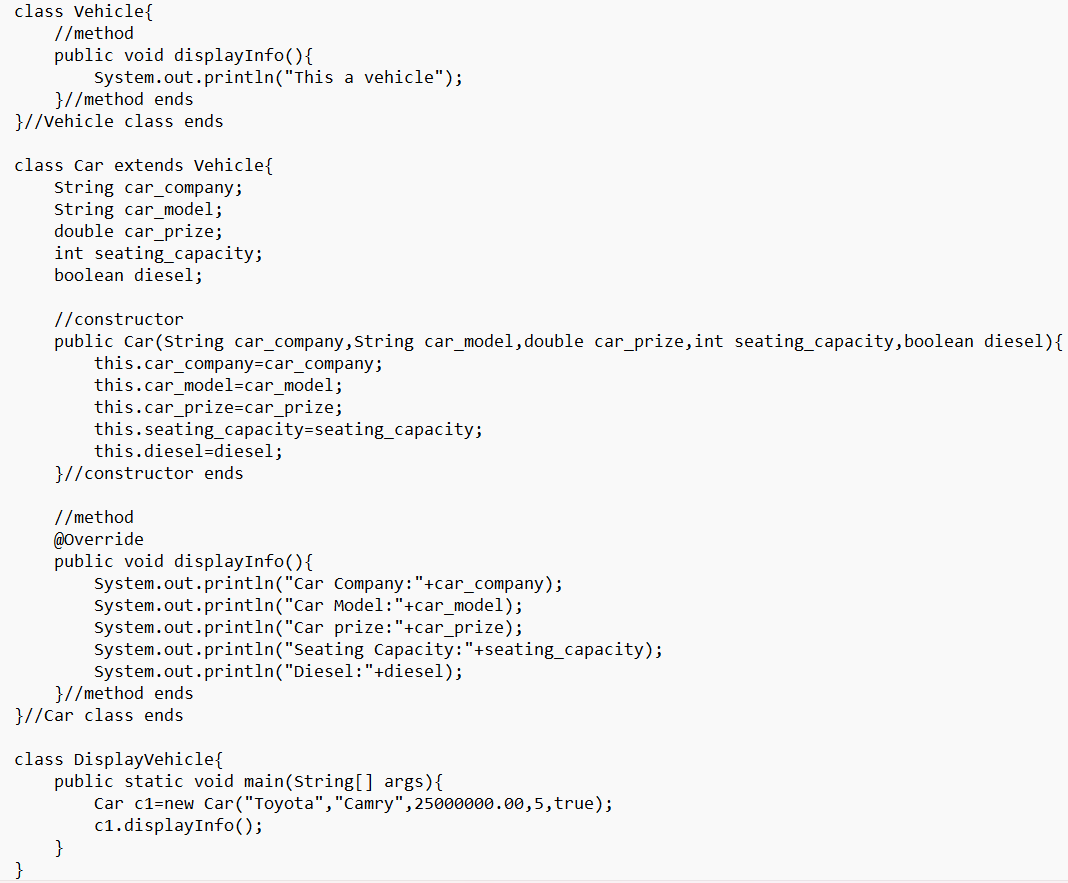
1. super(brand,speed); - This line of code, corresponds to calling the constructor of a super class, which requires parameters such as brand and speed.
2. input.close()- Used to closed the Scanner.

**WEEK 6**

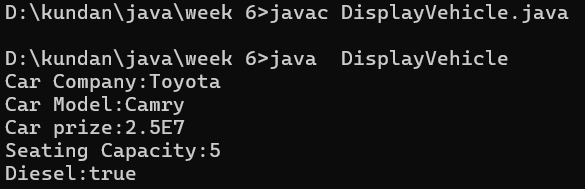
**Program 1**

**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. [car\_company, car\_model, car\_prize, seating\_capacity,diesel]

**Code:**



**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Error Message** | **Error Rectification** |
| 1. | The constructor Car(int, String, double, int, boolean) is undefined | Changed the parameter passed to String, i.e. from 25 to “Toyota”. |
| 2. | Seating\_capacity cannot be resolved to a variable | Changed the uppercase S into lowercase ‘ s’ |

**Class Diagram:**

|  |
| --- |
| Vehicle |
| + displayInfo(): void |

|  |
| --- |
| Car |
| car\_company: String  car\_model: String  car\_prize: double  seating\_capacity: int  diesel: Boolean |
| + Car(String car\_company,String car\_model,double car\_prize,int seating\_capacity,boolean diesel)  :void  + displayInfo() : void |

**Concepts to be known:**

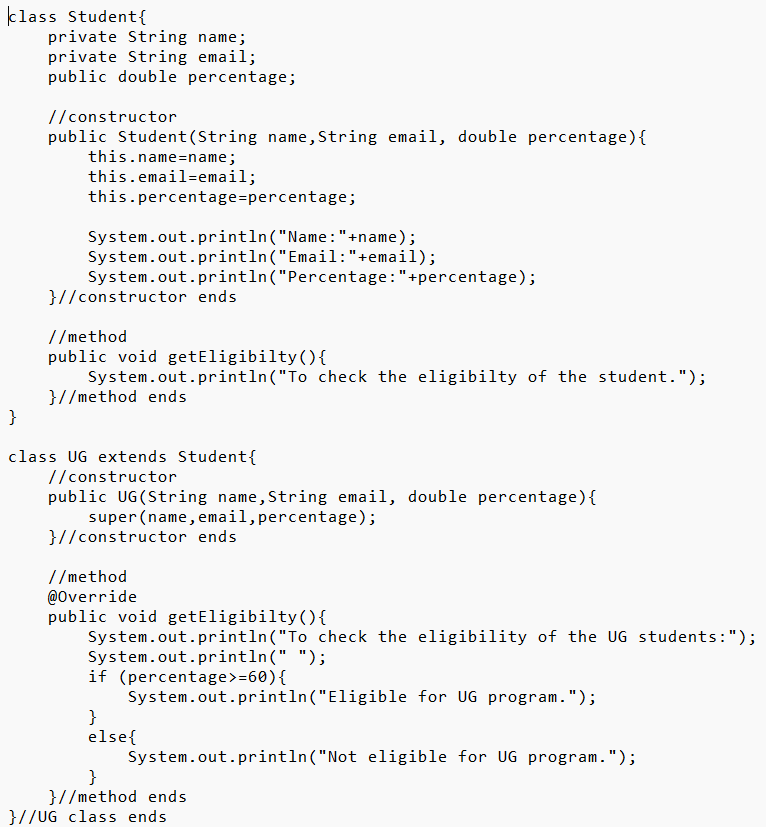
1. class Car extends Vehicle{} – Single-level Inheritance, where Vehicle is the parent class and Car is the subclass or child class.
2. Overriding – The method displayInfo() is defined in both the parent class and the subclass. Since we create an object of the subclass, the method in the subclass is given priority over the one in the parent class when called, i.e. The displayInfo() method is overridden in the subclass.

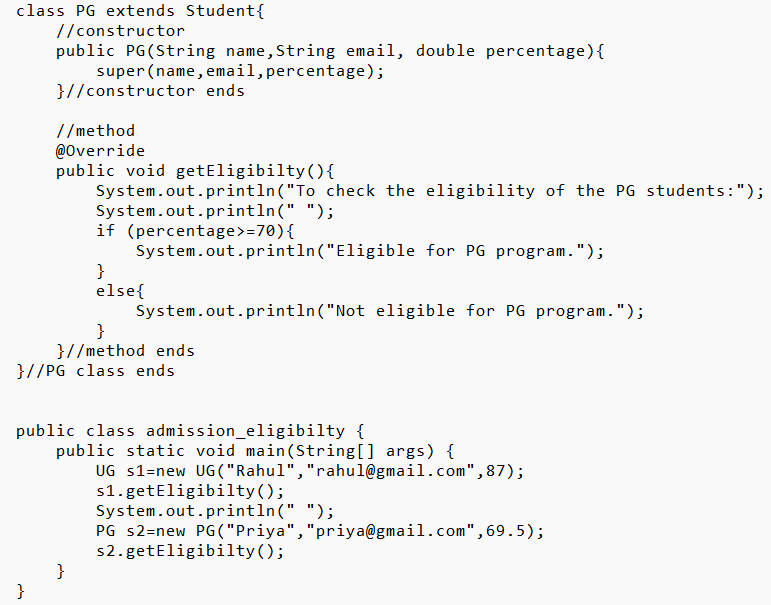
**Program 2**

**Aim:** A college is developing an automated admission system that verifies student’s eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility criteria based on the student’s percentage in their previous qualifications.

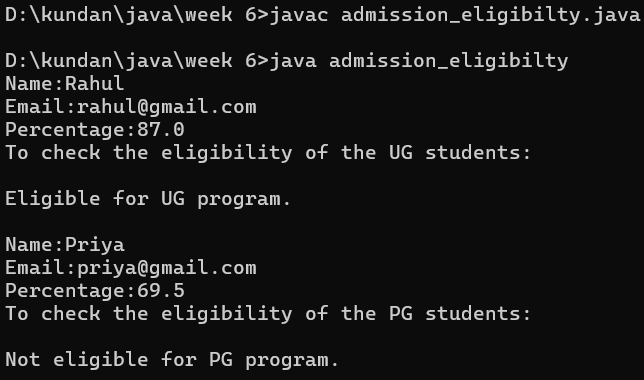
* UG admission requires minimum 60%
* PG admission requires minimum 70%

**Code:**

****

****

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Error Message** | **Error Rectification** |
| 1. | Syntax error on token "if", ( expected after this token | Changed if percentage>=70  To if (percentage>=70) |
| 2. | percentage cannot be resolved or is not a field | Added a variable int percentage in the Student Class. |

**Class Diagram:**

|  |
| --- |
| Student |
| - name:String  - email: String  -percentage: double |
| + Student(String name,String email, double percentage): void  + getEligibilty(): void |

|  |
| --- |
| UG |
| + UG(String name,String email, double percentage):void  + getEligibilty():void |

|  |
| --- |
| PG |
| + PG(String name,String email, double percentage):void  + getEligibilty():void |

**Concepts to be known:**

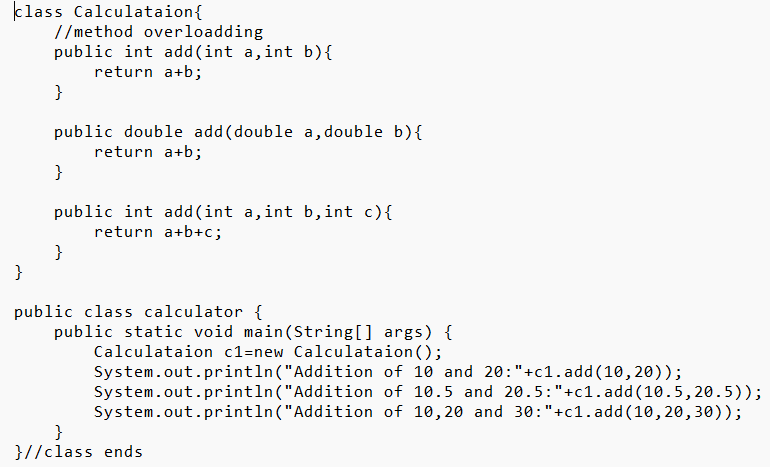
1. Overriding – The method getEligibilty() is defined in both the parent class and the subclass. Since we create an object of the subclass, the method in the subclass is given priority over the one in the parent class when called, i.e. The getEligibilty() method is overridden in the subclass.

**Program 3**

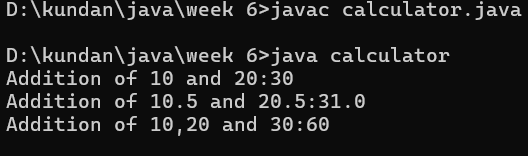
**Aim:** Create a calculator class with overloaded methods to perform addition.

* add 2 integers
* add 2 doubles
* add 3 integer

**Code:**

****

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Error Message** | **Error Rectification** |
| 1. | Void methods cannot return a value | Modified the return data type as int. : public int add(int a,int b,int c) |
| 2. | The method Calculataion() is undefined for the type calculator. | Calculataion c1=new Calculataion();  Added ‘new’ |

**Class Diagram:**

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

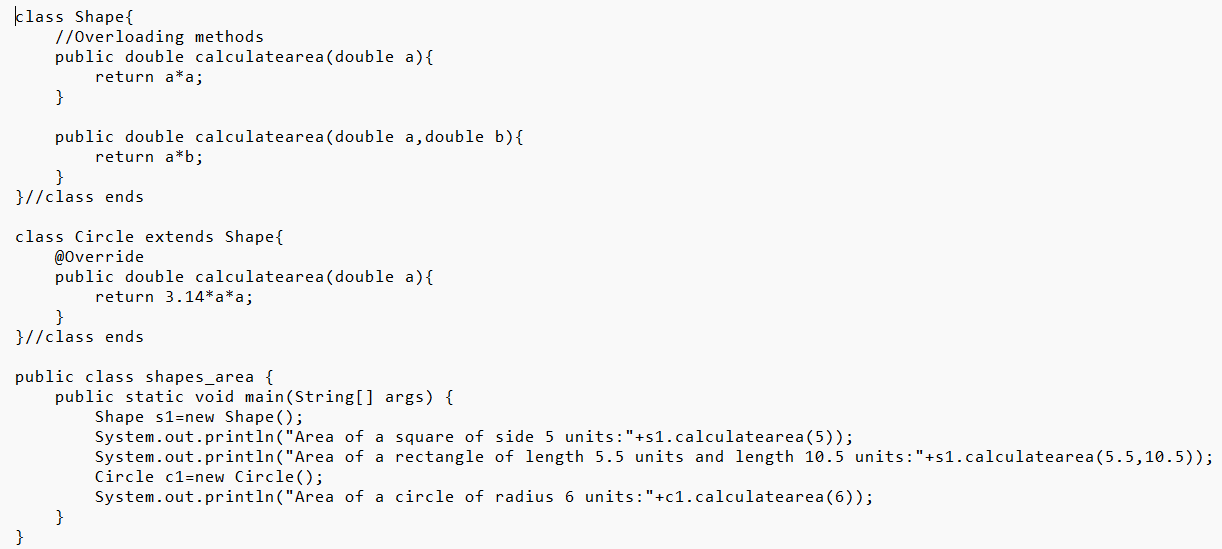
**Concepts to be known:**

1. Overloading- Defining **multiple methods with the same name** but with **different parameters** in the **same class**. Here, there are Multiple add() methods present, but all with different parameters. Depending upon the parameters passed, method is called.

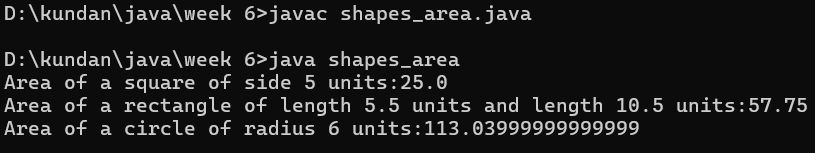
**Program 4**

**Aim:** Create a shape class with a method calculateArea() that is overloaded for different shape (eg: square, rectangle). Then, create a subclass Circle that overrides the calculateArea() method for a class.

**Code:**

****

**Output:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Error Message** | **Error Rectification** |
| 1. | Return type for the method is missing | Added the return data type as double : public double calculatearea(double a) |
| 2. | Syntax error, insert "}" to complete ClassBody | Added a semi-colon to the end of Circle class. |

**Class Diagram:**

|  |
| --- |
| Shape |
| + calculatearea(a: double): double + calculatearea(a: double, b: double): double |

|  |
| --- |
| Circle |
| + calculatearea(a: double): double |

**Concepts to be known:**

1. The above code explains method Overriding and method Overloading. Method calculatearea() which returns a double data type has been given different kinds of parameters. As per the condition, the method is executed.

On the other hand, calculatearea() described in the Circle class serves as Overriding.

**WEEK 7**

**Program 1**

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

**Code:**abstract class Animal {

abstract void sound();

}

class Lion extends Animal {

@Override

void sound() {

System.out.println("Roar");

}

}

class Tiger extends Animal {

@Override

void sound() {

System.out.println("Tiger growls");

}

}

class Testsound {

public static void main(String[] args) {

System.out.println("Name: E.Kundan Sri Vyshnavi, Rollno: AV.SC.U4CSE24105, Section: B");

Lion l = new Lion();

Tiger t = new Tiger();

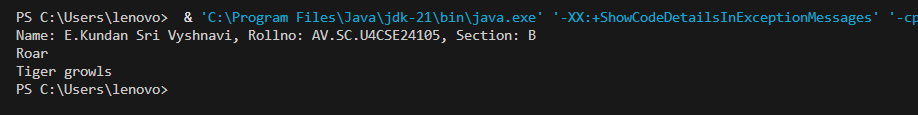
l.sound();

t.sound();

}

}

**OUTPUT:**



**CLASS DIAGRAM:**

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

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

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

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Error while printing the variables. 2. Incorrect declaration of integer. | 1. Give the plus sign while printing. 2. Give input.nextInt(), where I should be capital. |

**IMPORTANT POINTS:**

1. We override the methods in the superclass.

Here we are using the heirarchial inheritance.

**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 the volume and surface area of each shape.

**Code:**

abstract class Shape3D {

abstract double volume();

abstract double surfaceArea();

}

class Sphere extends Shape3D {

double radius;

Sphere(double radius) {

this.radius = radius;

System.out.println("Sphere created with radius: " + radius);

}

@Override

double volume() {

return (4.0/3.0) \* 3.14 \* radius \* radius \* radius;

}

@Override

double surfaceArea() {

return 4 \* 3.14 \* radius \* radius;

}

}

class Cylinder extends Shape3D {

double radius;

double height;

Cylinder(double radius, double height) {

this.radius = radius;

this.height = height;

System.out.println("Cylinder created with radius: " + radius + " and height: " + height);

}

@Override

double volume() {

return 3.14 \* radius \* radius \* height;

}

@Override

double surfaceArea() {

return (2 \* 3.14 \* radius \* height) + (2 \* 3.14 \* radius \* radius);

}

}

class ShapeTest {

public static void main(String[] args) {

System.out.println("Name: E.Kundan Sri Vyshnavi, Rollno: AV.SC.U4CSE24105, Section: B");

Sphere s1 = new Sphere(5.0);

System.out.println("Sphere Volume: " + s1.volume());

System.out.println("Sphere Surface Area: " + s1.surfaceArea());

Cylinder c1 = new Cylinder(3.0, 7.0);

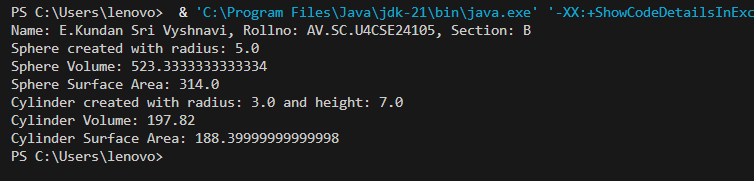
System.out.println("Cylinder Volume: " + c1.volume());

System.out.println("Cylinder Surface Area: " + c1.surfaceArea());

}

}

**Output:**



**CLASS DIAGRAM:**

|  |
| --- |
| Shape3D  +calculatevolume(): double  +calculatesurfacearea(): double |

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Wrong datatype entered. 2. Object not defined. | 1. Enter the correct datatype i.e double instead of int. 2. Enter the correct object and if not create new one. |

1. Here we used the abstract to declare an abstract class.
2. Abstract classes and methods help us to declare the methods without declaring the return type in them.

To get the values, we declared a constructor for each subclass and initialized values for them

**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)and a concrete method to display the pattern title

Implement 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.

Example Output for n = 5:

Star Pattern

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

Number Pattern

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**Code:**

abstract class PatternPrinter {

int rows;

PatternPrinter(int rows) {

this.rows = rows;

}

abstract void printPattern();

void displayTitle(String title) {

System.out.println("\n" + title);

}

}

class StarPattern extends PatternPrinter {

StarPattern(int rows) {

super(rows);

}

void printPattern() {

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

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

System.out.print("\* ");

}

System.out.println();

}

}

}

class NumberPattern extends PatternPrinter {

NumberPattern(int rows) {

super(rows);

}

void printPattern() {

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

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

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

}

System.out.println();

}

}

}

public class Teststar {

public static void main(String[] args) {

System.out.println("Name: E.Kundan Sri Vyshnavi, Rollno: AV.SC.U4CSE24105, Section: B");

int numberOfRows = 5;

PatternPrinter star = new StarPattern(numberOfRows);

star.displayTitle("Star Pattern");

star.printPattern();

PatternPrinter number = new NumberPattern(numberOfRows);

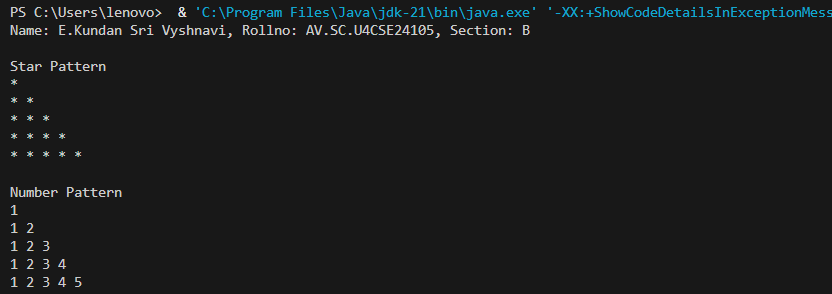
number.displayTitle("Number Pattern");

number.printPattern();

}

}

**Output:**

****

**CLASS DAIGRAM:**

PatternPrinter

- rows: int

+displayTitle()

+printPattern()

StarPattern

+printPattern()

NumberPattern

+printPattern()

**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.

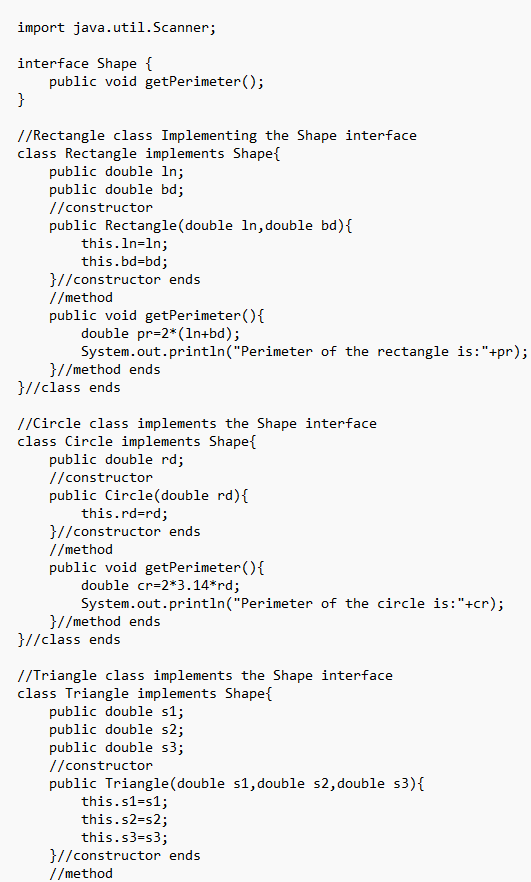
displayTitle() is a concrete method shared by all subclasses.

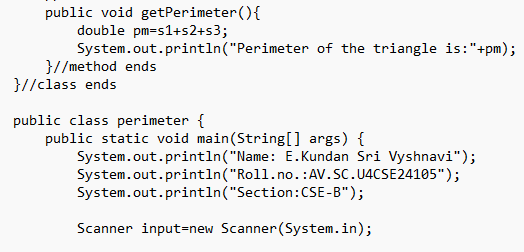
**WEEK 8**

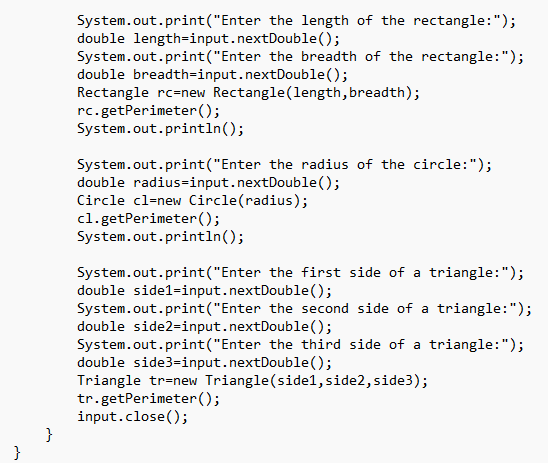
**Program 1**

**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.

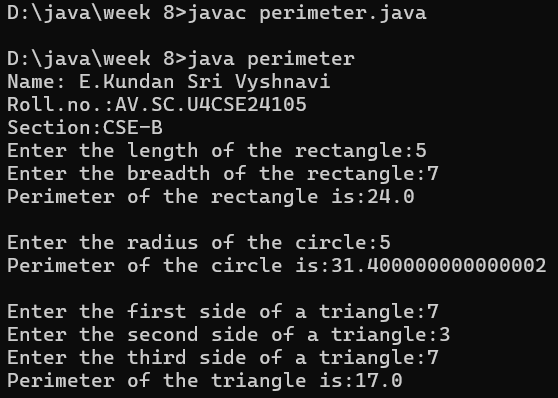
**Code:**







**Output:**

****

**CLASS DIAGRAM:**

Shapes

+ area(): double

Triangle

- base

- height

+ area():double

Circle

-radius

Rectangle

-length

-width

+ area(): double

**ERROR TABLE:**

|  |  |
| --- | --- |
| CODE ERROR  1)Class name "Shapes" is inconsistently used (should be consistent capitalization)    2)Base class method area() returns 0 by default - better to make it abstract | ERROR RECTIFICATION   1. Change to consistent capitalization (either all "Shapes" or all "Shapes")   2)Consider making Shapes abstract with abstract area() method |

IMPORTANT POINTS:

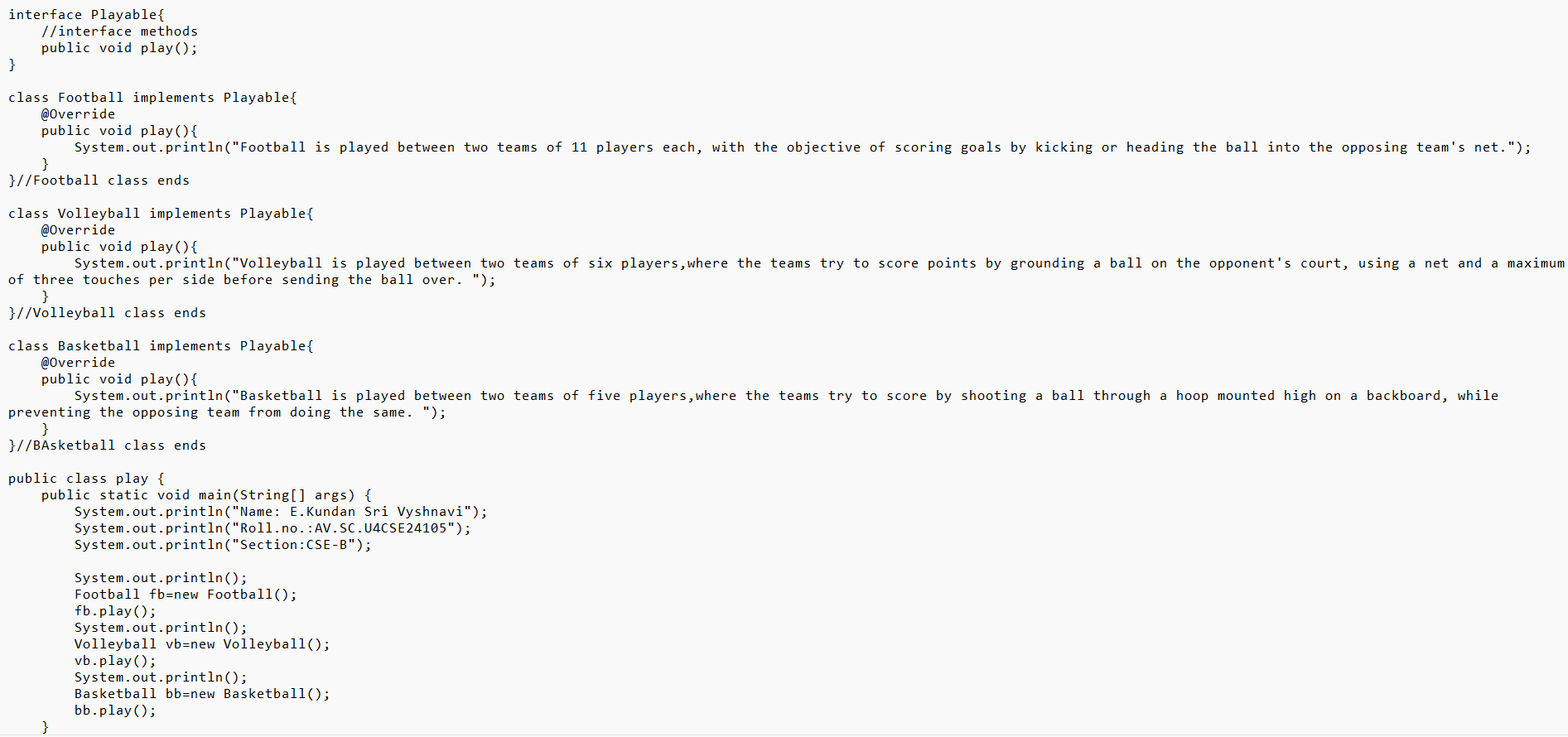
**Inheritance Hierarchy**: The Traingle, Circle and Rectangle classes all inherit from the base Shapes class (note: class name is misspelled as "Shapes" in some places and "Shapes" in others).

**Polymorphism**: Each subclass overrides the area() method to provide its own implementation, demonstrating polymorphic behavior.

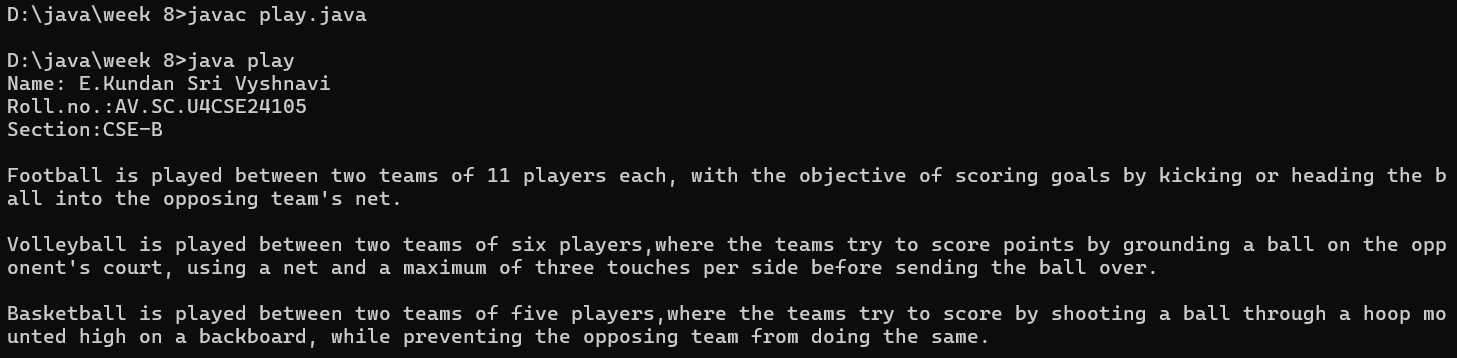
**PROGRAM-2:**

**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:**



**Output:**

****

**CLASS DAIGRAM:**

Interface: Playable

+ play(): void

FOOTBALL

+ play(): void

Volleyball

+ play(): void

Basketball

+ play(): void

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. Declaring an abstract class instead of interface class. 2. Not declaring public in each class. | 1. Declare an interface class instead of abstract class. 2. Declare public infront of each class. |

**IMPORTANT POINTS:**

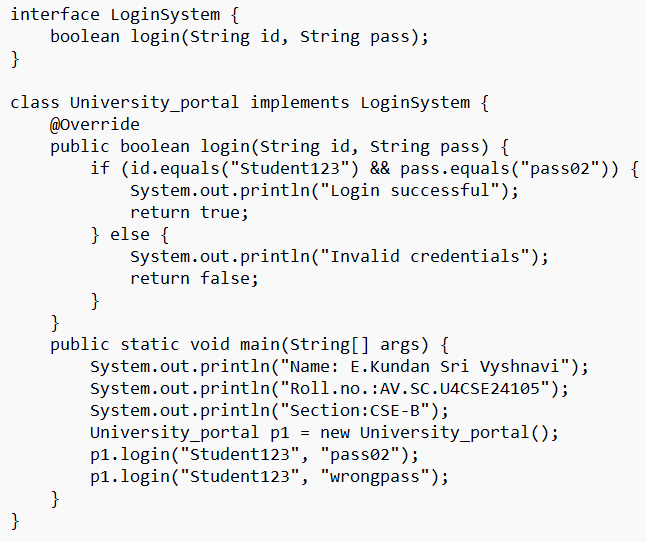
1. The playable interface abstracts the play() method, ensuring different classes implement it differently
2. The play() method behaves differently based on the object type football, volleyball, basketball.

Each class encapsulates its own implementation of how the sport is played, hiding the details from the user

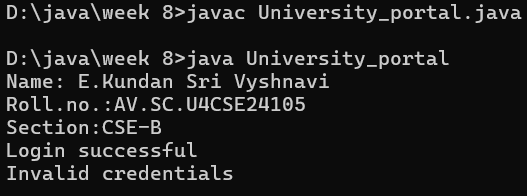
**PROGRAM-3:**

**AIM:**Write a java program to implements login System using interfaces.

**CODE:**



**Output:**



CLASS DAIGRAM:

interface :Login system

+ login(String,String): Boolean

University\_portal

+ login(StringString): boolean

+ main(String[]): void

ERROR TABLE:

|  |  |
| --- | --- |
| CODE ERROR   1. Saving file with the interface name is error | ERROR RECTIFICATION   1. save with implement name. |

**IMPORTANT POINTS:**

**Interface Implementation:**

University\_potral correctly implements Login System interface

Uses Override annotation for the login() method

**Main Method**:

Demonstrates both successful and failed login attempts

Includes student information print statement