**23CSE111**

**LAB MANUAL**



**Department of CSE**

**Amrita School of Engineering**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Verified By :- Name: G.lakshmi karthikeya**

**Roll No: 24049**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.NO | Programs | Date | Pg:No | Signature |
| 1 | 1. Download and Install Java Software. 2. Write a java program to print message “Welcome to java programming”. 3. Write a java program that prints name, roll number, section of a student. |  |  |  |
| 2 | i. To calculate the area of the rectangle  ii. Program to convert the temperature in celsius to Fahrenheit.  iii. Program to calculate the simple interest.  iv. Program to find the largest of three numbers using the ternary operators.  v. Program to find the factorial of the number |  |  |  |
| 3 | i. Create the java program for the cars with constructor and methods.  ii. Create the java program to withdraw and deposit money in the bank account. |  |  |  |
| 4 | i. Create the java program for the books by using the constructor and display its details using methods.  ii. Program to explain the final and the static variables. |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

***Week 1:-***

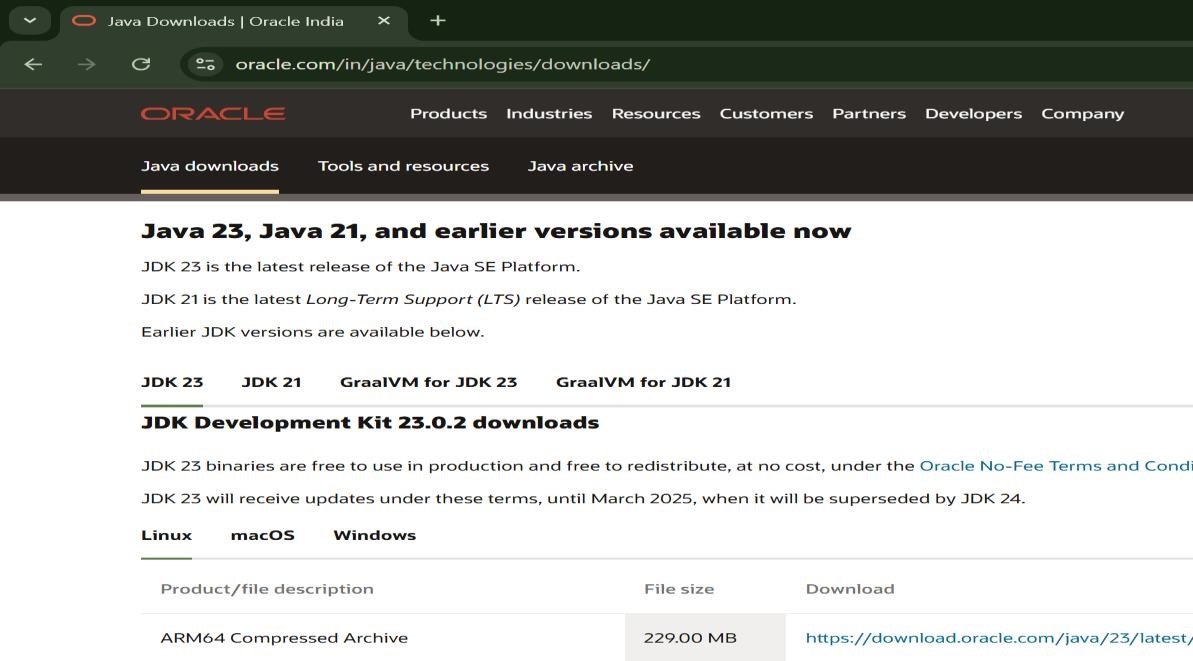
# Program-1:-

## Aim:-Download and Instal the Java Software Procedure

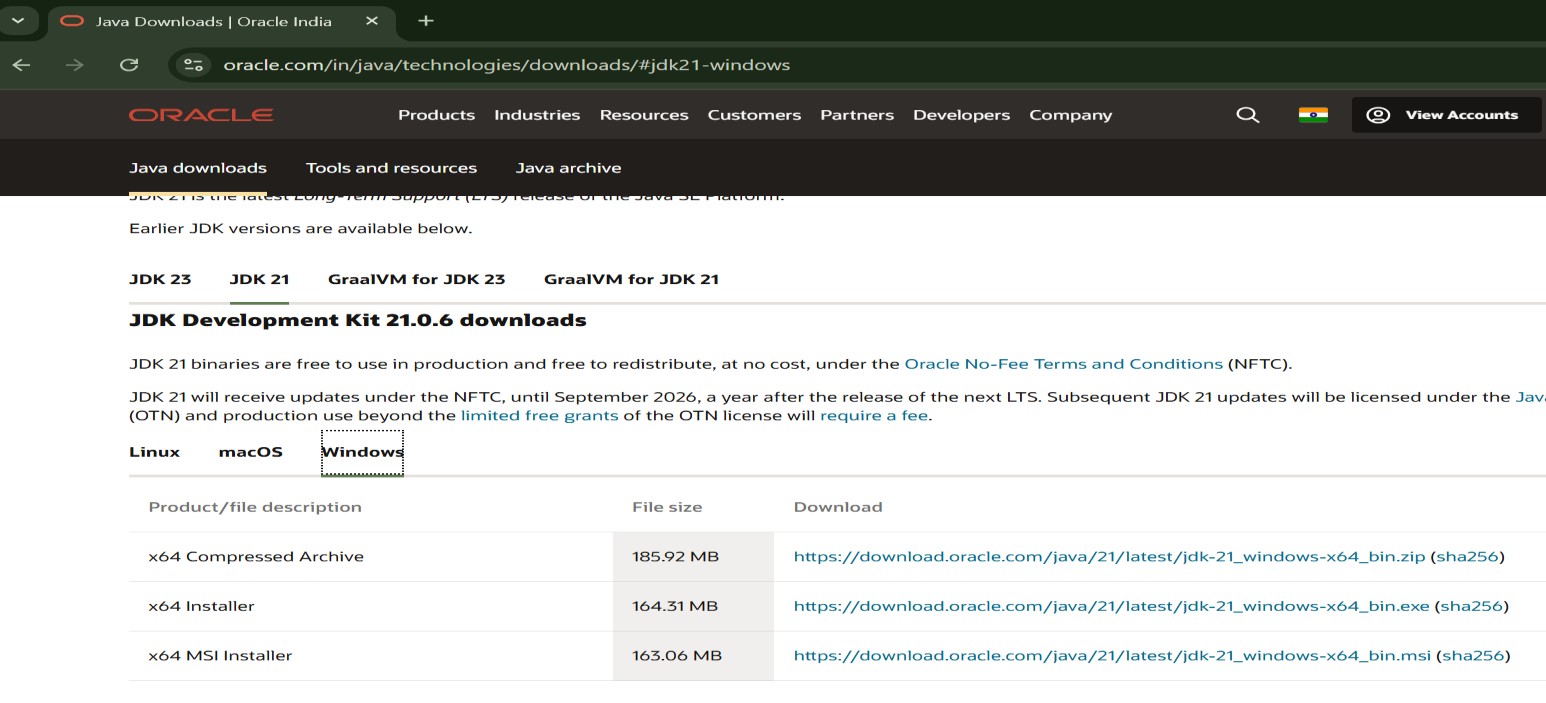
## Step-1:- Type Java download in search

## 

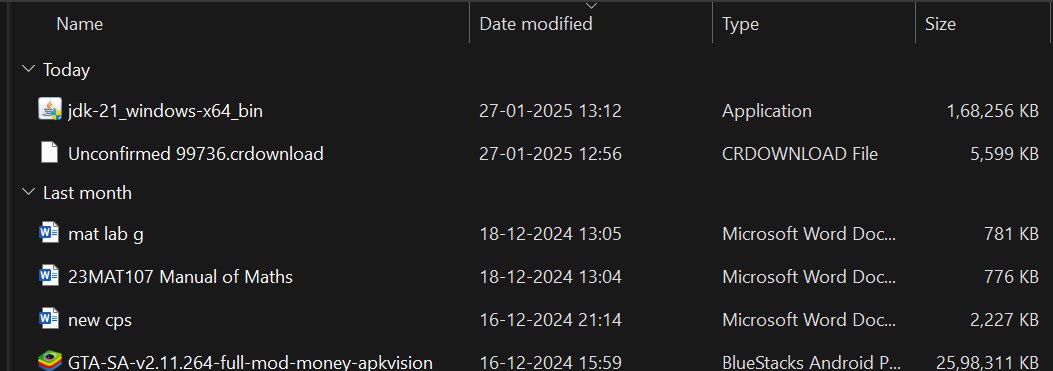
## **Step-2:-click on oracle java download and enter into oracle website**



## **Step-3:-click on JDK21 and click on windows and later click on x64 instalier link to download**

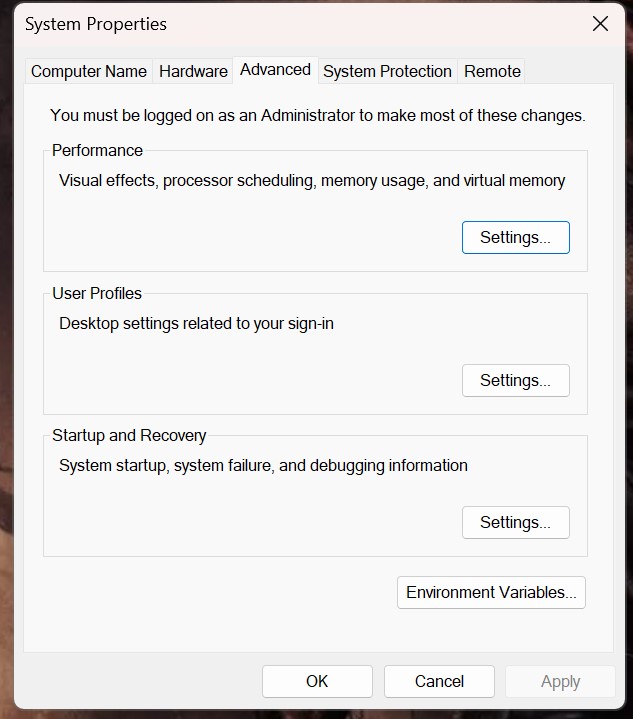


## **Step-4:-After completing download click on it’s file and then give permission to install**



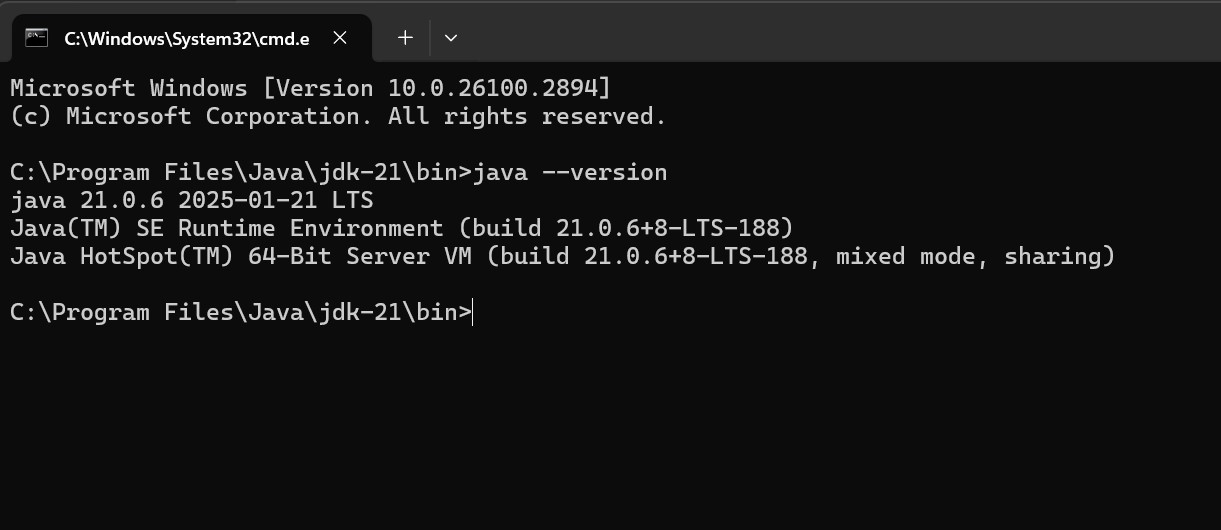
Step-5:-Then go to (This pc) in that click (windows{c}) in that click (Program files) in that click (Java) in that click (jdk-21) in that click (bin)

## **Step-6:-Select and copy path of opening the file and then press windows and search System Environmental**



Step-7:-After opening Environment variables then past path of opening file in user variable and click on ok

## **Step-8:-To verify version open CMD and type java --version**



**Program : 2**

## **Aim:-write a java program to print[welcome to java programming Input:-**

**class ex\_1{**

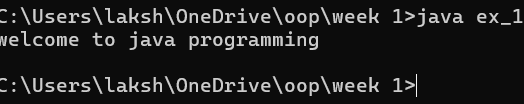
**public static void main(String[] args){**

**System.out.println("welcome to java programming");**

**}**

**}**

## Output:-



### Program : 3

Aim:-write a java program that prints name, roll no, section of the student Input:-

class Studentdet{

public static void main(String[] args){

System.out.println("Name: G.l.karthikeya”);

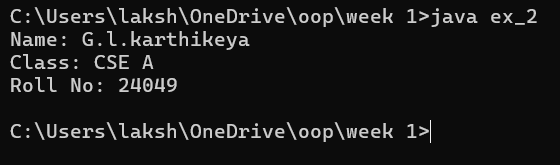
System.out.println("Class: CSE A");

System.out.println("Roll No: 24049");

}

}

### Output:-

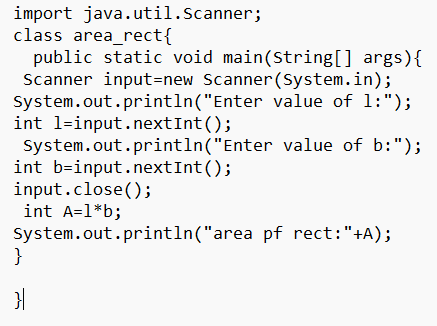


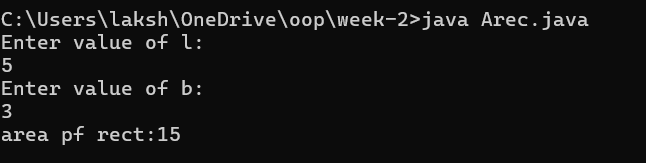
**Week-2**

Program-1:

Aim: to write a java program to find area of rectangle

Input:



Output: 

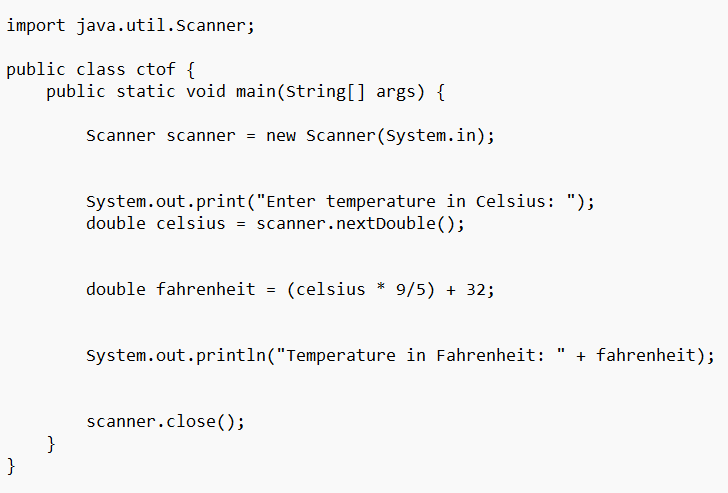
Error table:

|  |  |  |
| --- | --- | --- |
| Error | Error cause | Error rectification |
| error: ';' expected | ‘;’ is missed in the end of print statement | Placed ‘; ’ at the end of the statement |
| error:cannot find symbolScannerinput=new scanner(System.in); | Placed small s in place of capital S | Replaced capital s in place of small s to rectifiy the error |
| error: cannot find symbol  int b=input.nextstr(); | Placed str in place of int | Rectified by replacing int in place of int |

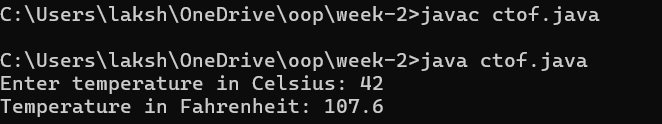
Program-2

Aim: write a java program to convert temp from celsius to farenheit

Input:



Output:



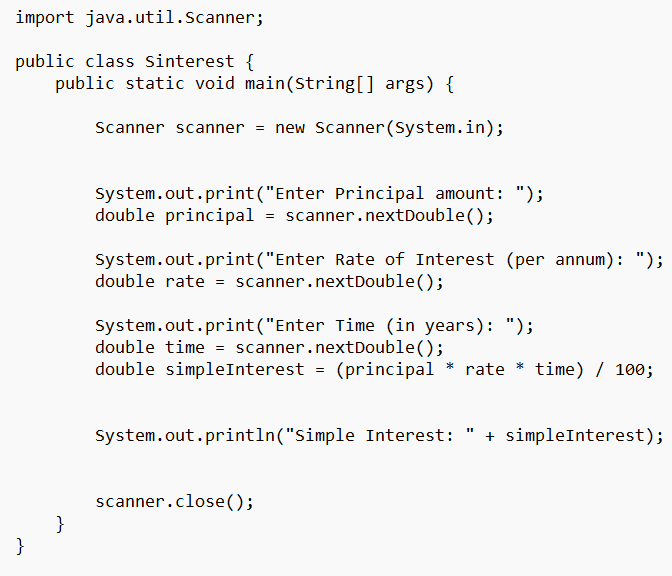
Error table:

|  |  |  |
| --- | --- | --- |
| Error | Error cause | Error rectification |
| error: ';' expected  public static void main(String[] args) | Forgot flower brace in the end | Rectified by replacing flower brace |
| error: ';' expected  scanner.close() | Forgot ‘;’ at the end the statement | Rectified by placing ’;’  it |
| error: incompatible types: possible lossy conversion from double to int  int celsius = scanner.nextDouble(); | Placed int in place of double | Rectified by replacing double |

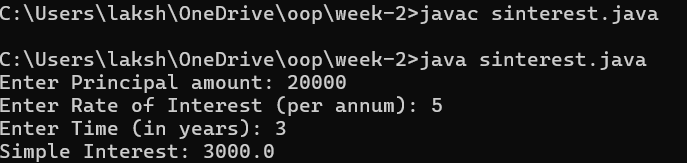
Program-3:

Aim: write a java program to calculate simple interest

Input:



Output:



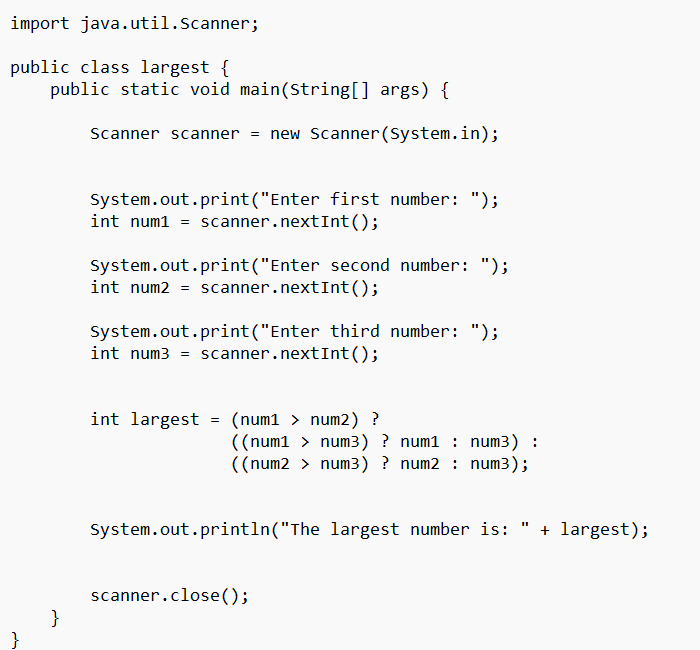
Error table:

|  |  |  |
| --- | --- | --- |
| Error | Error cause | Error rectification |
| error: class Sinterest is public, should be declared in a file named Sinterest.java  public class Sinterest {  ^ | Placed capital s I place of small ‘s’ | Replaced by placing small ‘s’ |

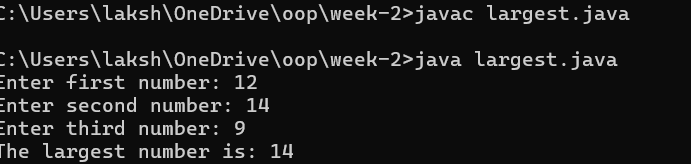
Program-4

Aim: write a java program to find the largest of 3 numbers using terenary operator

Input:



Output:



Error table:

|  |  |  |
| --- | --- | --- |
| Error | Error cause | Error rectification |
| error: reached end of file while parsing  ((num1 > num3) ? num1 : num3) | Missed ‘} ’ in the end of the program | Rectified by replacing it |
| error: illegal start of expression  }  ^ | Missed’}’in the starting | Rectified by replacing it |

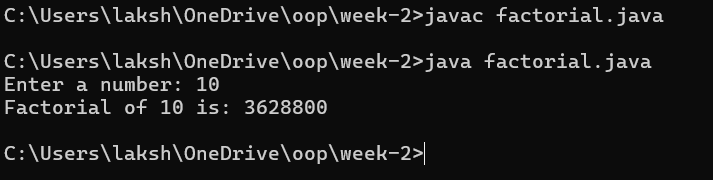
Program-5

Aim: write a java program to find the factorial of a number

Input:



Output:



Error table:

|  |  |  |
| --- | --- | --- |
| Error | Error cause | Error rectification |
| error: unclosed string literal  System.out.print("Enter a number: );  ^ | Missed “ in the end | Rectified by replacing “ |

***Week-3***

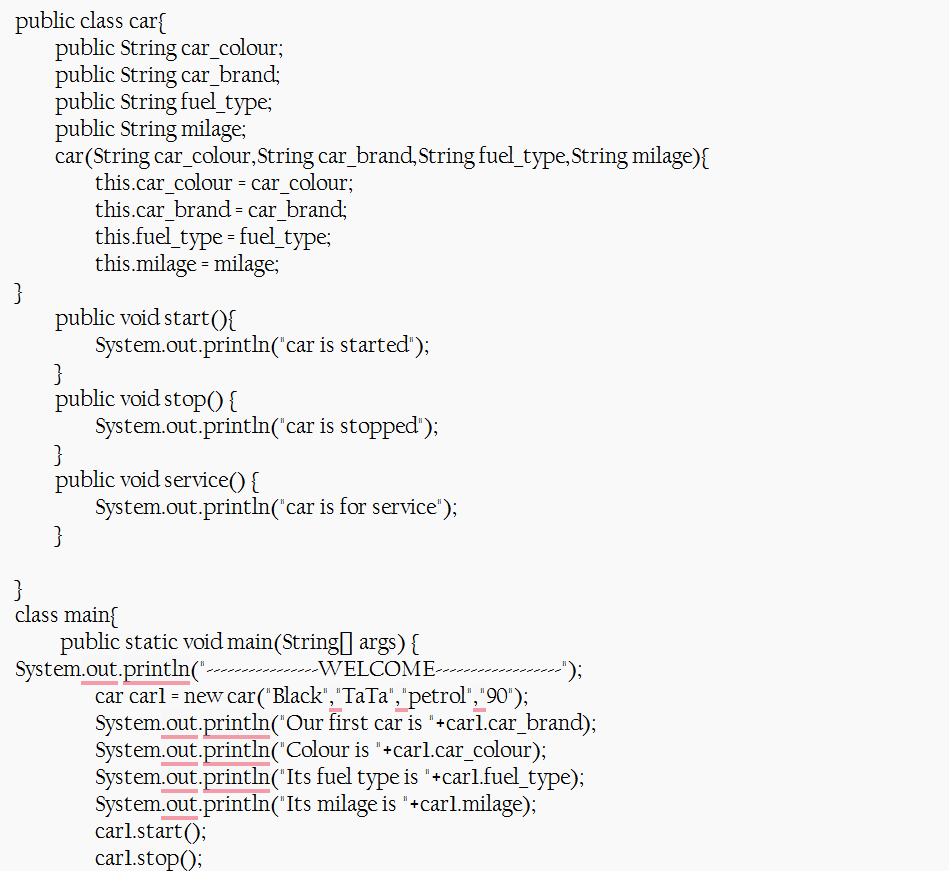
***Program -1:***

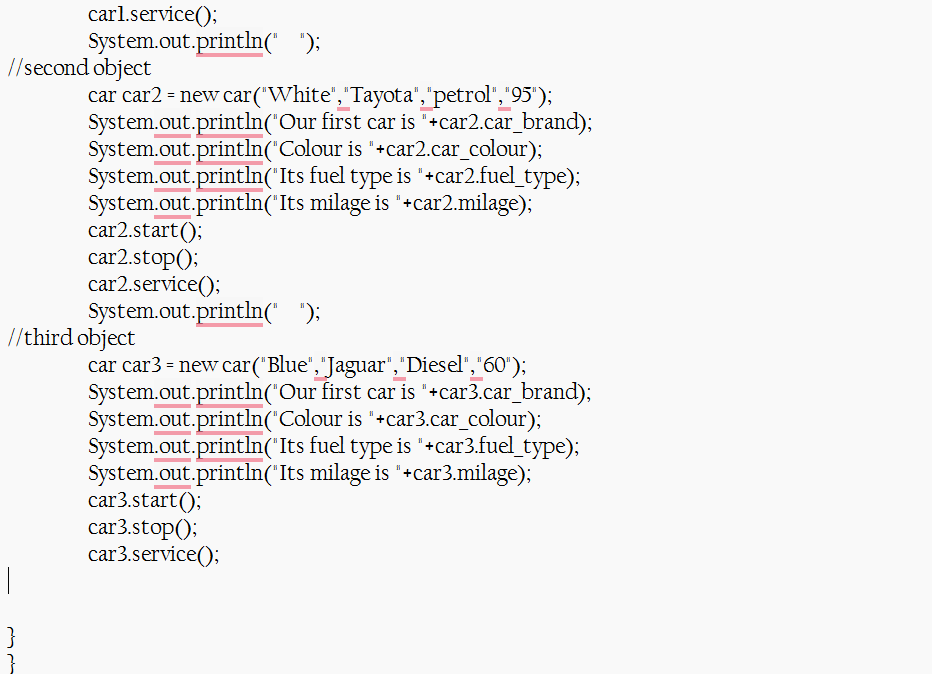
**Aim: (i)** Create the java program for the cars with constructor and methods

***Class diagram:***

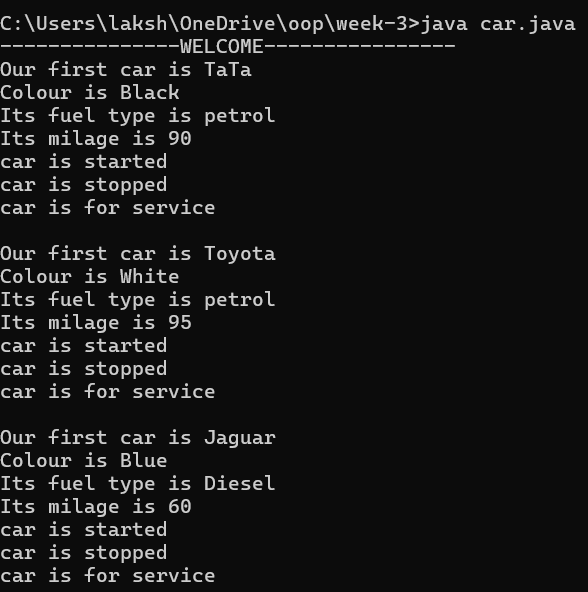
|  |
| --- |
| ***Car*** |
| * ***carColor: String*** |
| * ***carBrand: String*** |
| * ***fuelType: String*** |
| * ***topSpeed: int*** |
| ***+ Car(String,String,String,int)*** |
| ***+ startRacing()*** |
| ***+ endRace()*** |

***Code:***





***Output:***



**Error:**

|  |  |  |
| --- | --- | --- |
| **Error Type** | **Incorrect Code** | **Corrected Code** |
| **Class Naming Issue** | class main{ | class Main{ |
| **Incorrect Object Description** | "Our first car is "+car2.car\_brand; | "Our second car is "+car2.car\_brand; |

***Program-2:***

B ) Write a class by writing java program named Bank Account

with two methods “deposits and withdraw”.

a) In deposit method whenever an amount is deposited it

has to be updated with current amount (logic C.A+D.A).

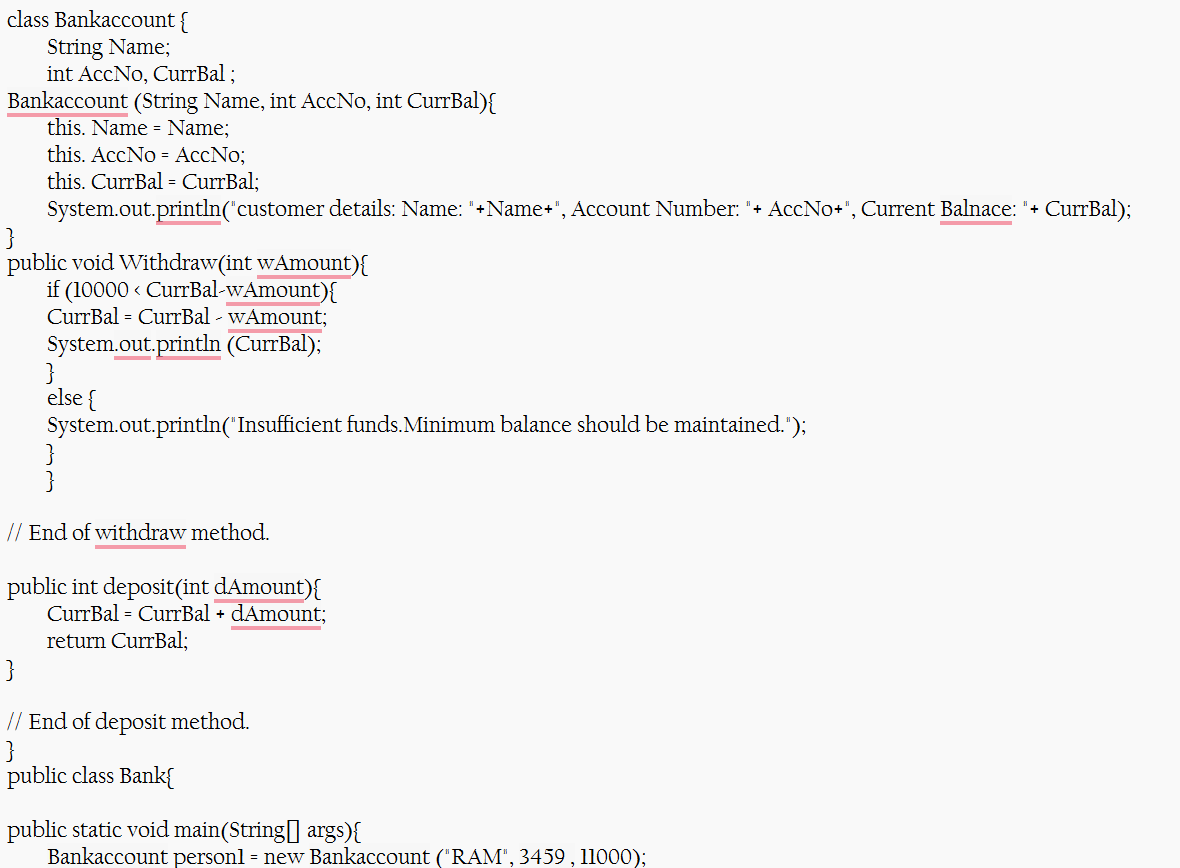
b) With draw amount whenever an amount is being

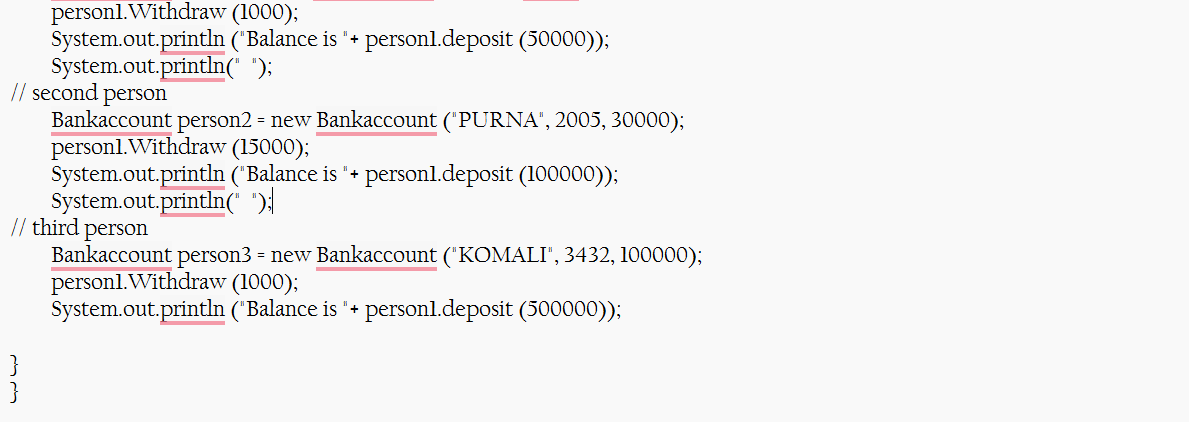
withdraw it has to be less than the current amount less

than the amount else print “Insufficient funds”.

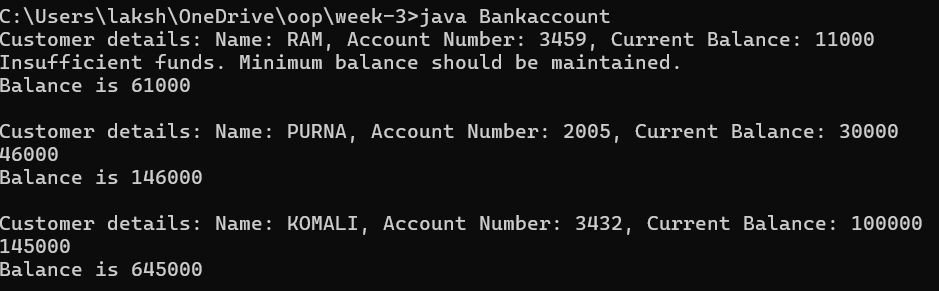
**Class Diagram:**

|  |
| --- |
| Bank Account |
| * currentAmount: double |
| + BankAccount(initialAmount:double) |
| + deposit(amount: double):void |
| + withdraw(amount: double):void |
| + getCurrentAmount():double |





Output:



**Error:**

|  |  |  |
| --- | --- | --- |
| **Error Type** | **Incorrect Code** | **Corrected code** |
| **Class Name Capitalization** | class Bankaccount | class BankAccount (Java follows PascalCase for class names) |
| **Object Naming Issue** | BankAccount person-1 (hyphen is not allowed) | BankAccount person1 |
| **Missing Semicolon** | System.out.println ("Balance is "+ person-1.deposit (50,000)) | System.out.println ("Balance is "+ person1.deposit (50000)); (semicolon added) |

***Week-4:***

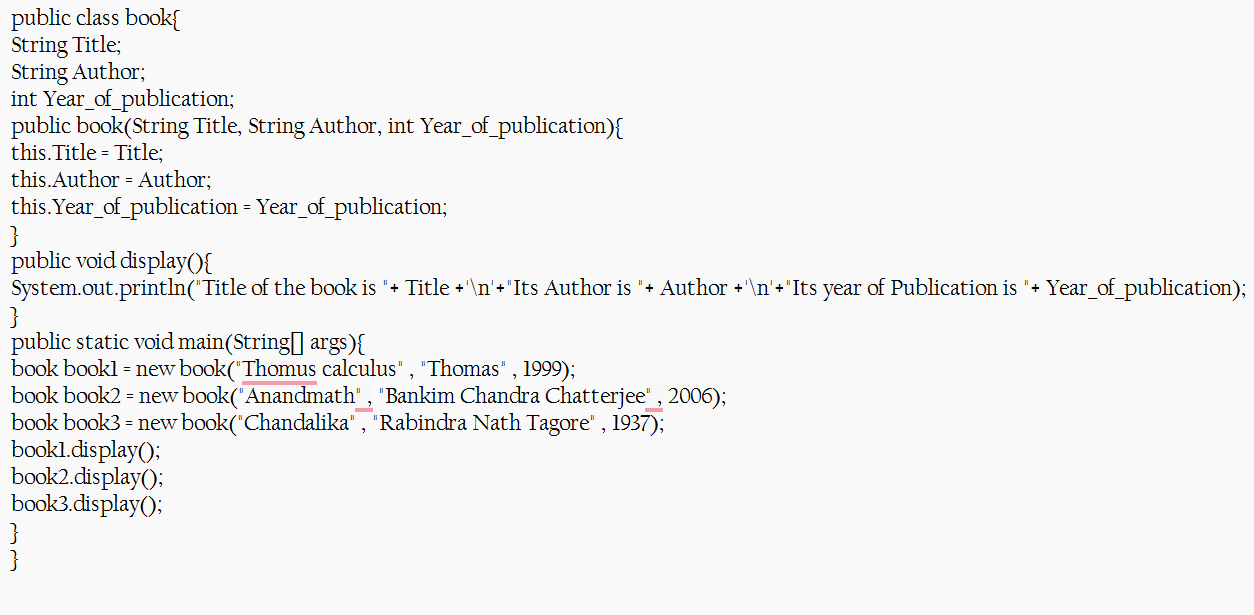
***Program 1:***

**Aim:** (i) Create the java program for the books by using the constructor and display its details using methods

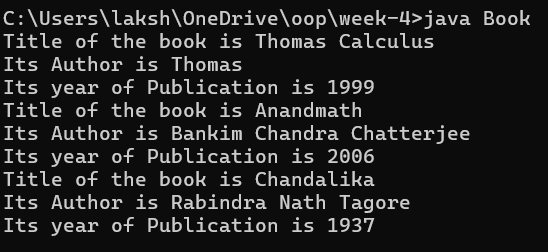
**Class diagram:**

|  |
| --- |
| **Book** |
| **-title: String**  **-author: String**  **-year: int** |
| **+ Book(title: String, author:String, year: int) + displayDetails(): void** |

**Code:**



**Output:**



**Errors:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Error Type** | |  | | --- | | **Incorrect Code** |  |  | | --- | |  | | **Corrected Code** |
| **Class Name Capitalization** | public class book | public class Book (Java follows PascalCase for class names) |
| **Constructor Name Mismatch** | new book(...) | new Book(...) (Constructor name must match class name) |

**IMPORTANT POINTS:**

1. **Constructor**:

* The constructor Book(String, String, int) is used to initialize the object when it is created.
* The keyword **this** is used to differentiate between class attributes and constructor parameters.

2.**Method**:

* The method displayDetails() is used to display the book details.
* The *System.out.println()* method prints the details to the console.

3. **Object Creation**:

* Two objects b1 and b2 are created using the constructor.

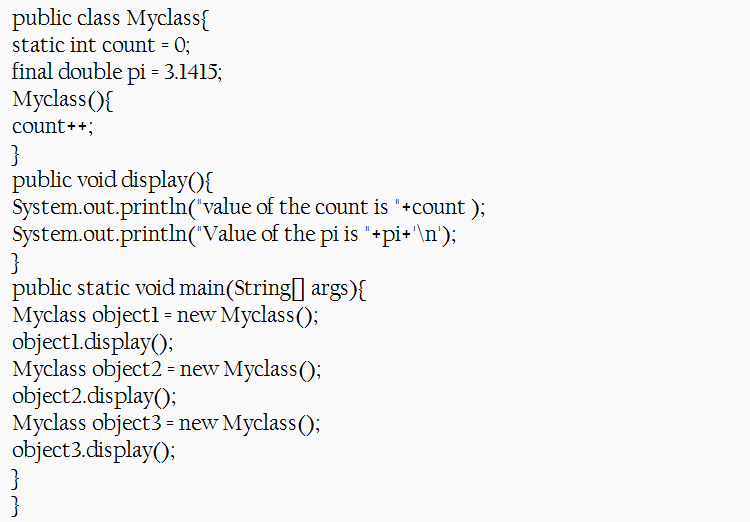
Program 2:

**Aim: (ii)** Program to explain the final and the static variables.

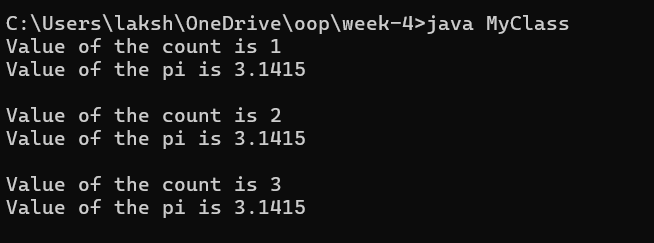
**CLASS DIAGRAM:**

|  |
| --- |
| **MyClass** |
| **-count: int (static)**  **-pi: double (static, final)** |
| **+MyClass()**  **+main(args: String[]):void** |

Code:



Output:



**Error:**

|  |  |  |
| --- | --- | --- |
| **Error Type** | **Incorrect Code** | **Corrected Code** |
| **Attempt to Modify final Variable** | pi = 3.14; (if added inside the constructor or method) | Remove this line (final variables cannot be reassigned) |
| **Incorrect Class Name** | public class Myclass | public class MyClass (Java follows PascalCase for class names) |

**IMPORTANT POINTS:**

**1.Static Keyword**

* Static members belong to the **class, not to individual objects**.
* Only one copy of the static variable is maintained for all objects.

**2.Static Variable**

* **static int count**:
  + Shared among all objects of the class.
  + It is initialized only once and not for every object.
  + It increments every time the constructor is called.

**3.Final Variable**

* **static final double pi**:
  + The **final** keyword makes the variable constant.
  + Its value **cannot be changed** once assigned.
  + It must be initialized at the time of declaration.

**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

***Input:***

class addition

{

public int add(int a, int b)

{

int addition = a + b;

return addition;

}

}

class subtraction extends addition

{

public int sub(int a, int b)

{

int subtraction = a - b;

return subtraction;

}

}

class multiplication extends subtraction

{

public int mult(int a, int b)

{

int multiplication = a \* b;

return multiplication;

}

}

class division extends multiplication

{

public int div(int a, int b)

{

int division = a / b;

return division;

}

}

class calculator

{

public static void main(String args[])

{

division obj = new division();

System.out.println("Addition is: " + obj.add(10, 2));

System.out.println("Subtraction is: " + obj.sub(8, 4));

System.out.println("Multiplication is: " + obj.mult(12, 4));

System.out.println("Division is: " + obj.div(8, 4));

}

}

**CLASS DIAGRAM:-**

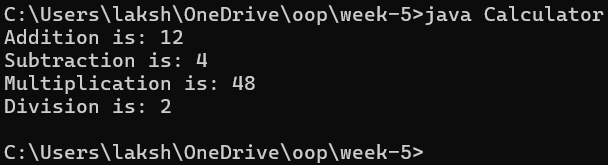
|  |
| --- |
| CLASS ADDITION |
| +add(int a, int b):int |

|  |
| --- |
| Class Subtraction |
| +sub(int a, int b):int |

|  |
| --- |
| Class Multiplication |
| +mult(int a, int b):int |

|  |
| --- |
| Class Division |
| +div(int a, int b):int |

***Output:***



**Error-table:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Reason for error** | **Rectification** |
| **1.** | Constructive error | Invalid method name declared | Created class name |
| **2.** | Syntax error | Haven’t included ‘;’ | Added ‘;’ |

**Important Points:-**

**Inheritence:**

The concept of OOP where a class inherits the properties and behaviours from

Another class (parent class) which promotes code reusability and hieratchical relationships

**Multilevel Inheritence:**

This is a type of inheritance in which a class inherited from another class, and

That superclass, in turn, inherits from yet another class, creating a chain of

Inheritance

**PROGRAM-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 cars, bikes and truck and they need a program to

Store details about each vehicle, such as brand and speed

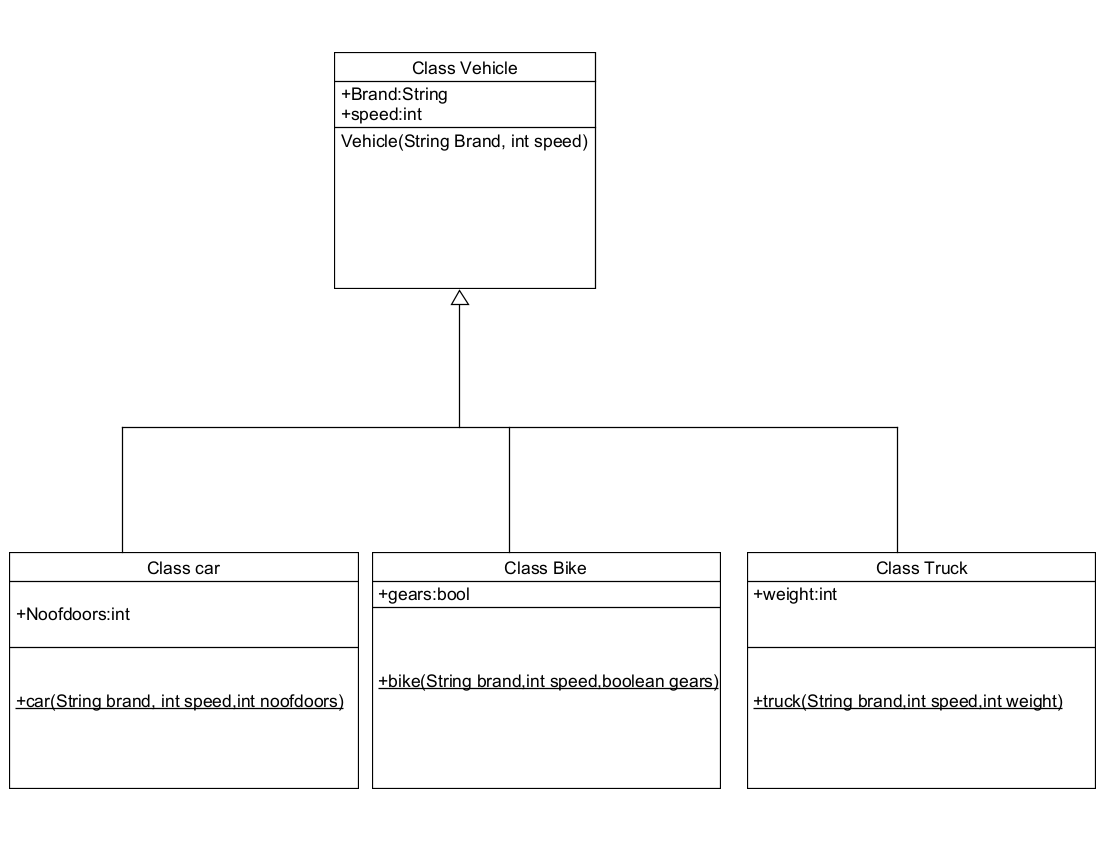
Cars should have an additional property: number of doors

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

The system should also include a function to display details about each vehicle

And indicate when a vehicle is starting.

Class diagram:



**INPUT:-**

class vehicle {

    public String brand;

    public int speed;

    public vehicle(String brand, int speed) {

        this.brand = brand;

        this.speed = speed;

    }

    public void start() {

        System.out.println(brand + " is starting");

    }

    public void showDetails() {

        System.out.println("Brand: " + brand);

        System.out.println("Speed: " + speed + " km/h");

    }

}

class Car extends vehicle {

    private int noOfDoors;

    public Car(String brand, int speed, int noOfDoors) {

        super(brand, speed);

        this.noOfDoors = noOfDoors;

    }

    public void showDetails() {

        super.showDetails();

        System.out.println("Number of Doors: " + noOfDoors);

    }

}

class Bike extends vehicle {

    private boolean hasGears;

    public Bike(String brand, int speed, boolean hasGears) {

        super(brand, speed);

        this.hasGears = hasGears;

    }

    public void showDetails() {

        super.showDetails();

        System.out.println("Has Gears: " + (hasGears ? "Yes" : "No"));

    }

}

class Truck extends vehicle {

    private int capacity;

    public Truck(String brand, int speed, int capacity) {

        super(brand, speed);

        this.capacity = capacity;

    }

    public void showTruck() {

        super.showDetails();

        System.out.println("Capacity: " + capacity + " tons");

    }

}

public class rent {

    public static void main(String[] args) {

        Car car = new Car("Porshe", 150, 4);

        Bike bike = new Bike("Discover", 120, true);

        Truck truck = new Truck("Ashok Leyland", 90, 10);

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

        car.start();

        car.showDetails();

        System.out.println("Bike Details");

        bike.start();

        bike.showDetails();

        System.out.println("Truck Details");

        truck.start();

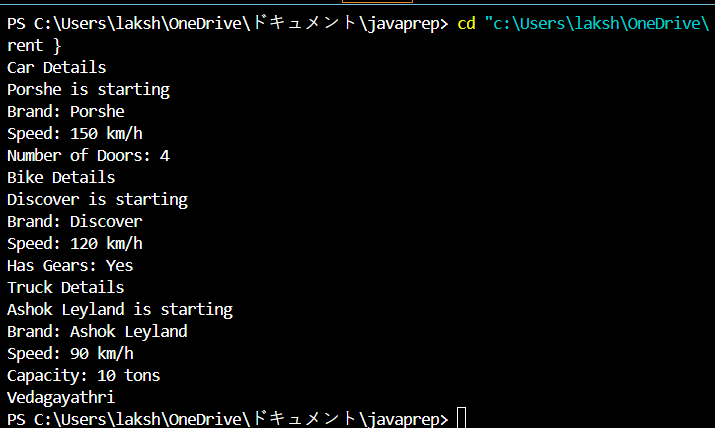
        truck.showTruck();

        System.out.println("Vedagayathri");

    }

}

Output:



Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| S No | Error Type | Cause | Rectification |
| 1 | Syntax Error | Semicolon missing | Added ; |

**Important Points**

**Hierarchical Inheritence:**

This is a type of inheritance occurs when multiple subclasses inherit from a

Single parent class

***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 a

About car

Class diagram:

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

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

Code:

class Vehicle1 {

String car\_company;

String car\_model;

long car\_price;

int seating\_capacity;

boolean petrol;

Vehicle1(String car\_company, String car\_model, long car\_price, int seating\_capacity, boolean petrol) {

this.car\_company = car\_company;

this.car\_model = car\_model;

this.car\_price = car\_price;

this.seating\_capacity = seating\_capacity;

this.petrol = petrol;

}

void displayInfo() {

System.out.println("Car company: " + car\_company);

System.out.println("Car model: " + car\_model);

System.out.println("Car price: " + car\_price);

System.out ("Car seating capacity: " + seating\_capacity);

System.out.println("Car uses petrol: " + petrol);

}

}

class Car extends Vehicle1 {

Car(String car\_company, String car\_model, long car\_price, int seating\_capacity, boolean petrol) {

super(car\_company, car\_model, car\_price, seating\_capacity, petrol);

}

}

public class Main {

public static void main(String[] args) {

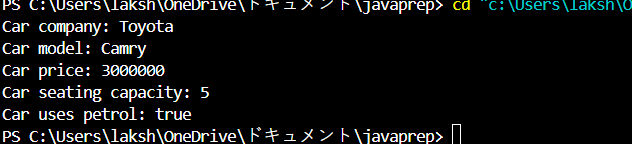
Car c1 = new Car("Toyota", "Camry", 3000000, 5, true);

c1.displayInfo();

}

}

Output:



Error table:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Program-2:

Aim: A college is developing an automated admission system that verifies

Student eligibility for UG and PG programs. Each program has different

Eligibility criteria based on the student’s percentage in their previous

Qualification

UG admissions require a minimum of 60%

PG admissions require a minimum of 70%

Class diagram:

|  |
| --- |
| **adm** |
| elg():void |

|  |  |
| --- | --- |
| ug | pg |
| +elg():void | +elg():void |

Code:

class Student {

public String name;

public double percentage;

Student(String name, double percentage) {

this.name = name;

this.percentage = percentage;

}

void eligibility() {

System.out.println("Amrita Vishwa Vidyapeetham");

}

}

class UGStudent extends Student {

UGStudent(String name, double percentage) {

super(name, percentage);

}

void eligibility() {

if (percentage > 60) {

System.out.println(name + " is selected to the college");

} else {

System.out.println(name + " is not matching the required criteria");

}

}

}

class PGStudent extends Student {

PGStudent(String name, double percentage) {

super(name, percentage);

}

void eligibility() {

if (percentage > 70) {

System.out.println(name + " is selected to college");

} else {

System.out.println(name + " is not matching the required criteria");

}

}

}

public class School {

public static void main(String args[]) {

PGStudent obj1 = new PGStudent("Bhanu", 90.0);

UGStudent obj2 = new UGStudent("Teja", 80.0);

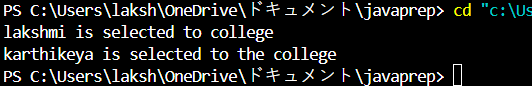
obj1.eligibility();

obj2.eligibility();

}

}

Output:



Error table:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Program-3:

Aim:

Create a calculator class with overloaded methods to perform addition

1. Add two integers 2. Add two doubles 3. Add three integers

Class Diagram:

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

Code:

class calculator

{

    public int add(int a, int b)

    {

        return a+b;

    }

    public double add(double a, double b)

   {

       return a+b;

    }

    public int add(int a, int b, int c)

    {

       return a+b+c;

    }

}

class Overloading

{

      public static void main(String args[]){

       calculator c = new calculator();

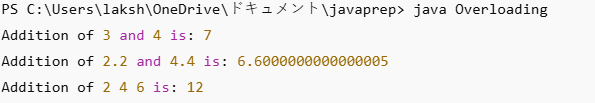
       System.out.println("Addition of 3 and 4 is:"+c.add(3,4));

       System.out.println("Addition of 2.2 and 4.4 is:"+c.add(2.2,4.4));

       System.out.println("Addition of 2 4 6 is:"+c.add(2,4,6));

}}

Output:



Error table:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Program-4:

Aim: Create a Shape class with a method calculateArea() that is overloaded for

Different shapes. Then, create a subclass circle that overrides the

calculateArea() method for a circle

**Class Diagram:**

|  |
| --- |
| shape |
| +calarea(float side):float  +calarea(float l,float b):float  +calarea(float c):float |

|  |
| --- |
| Circle |
| +calarea(double r):double |

code:

class Shape {

    public double calculateArea(double side) {

        return side \* side;

    }

    public int calculateArea(int length, int width) {

        return length \* width;

    }

}

class Circle extends Shape {

    public double calculateArea(double radius) {

        return 3.14 \* radius \* radius;

    }

}

class Example {

    public static void main(String args[]) {

        Circle obj1 = new Circle();

        Shape obj2 = new Shape();

        System.out.println("The area of side 6 is: " + obj2.calculateArea(6.0));

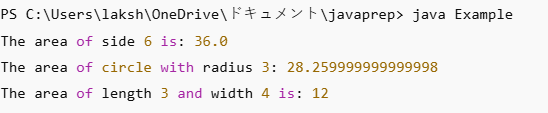
        System.out.println("The area of circle with radius 3: " + obj1.calculateArea(3.0));

        System.out.println("The area of length 3 and width 4 is: " + obj2.calculateArea(3, 4));

    }

}

Output:



**Error table:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Expected Error** | **Reason** |
| **1** | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| **2** | } | Ending the class and main method is required |

Week-7:

**Program-1:**

**Write a java program to create an abstract class Animal with an abstract method sound().Create Subclass Tiger and Lion extends the Animal class and implement the sound() method to make a specific sound for each animal**

**Program:**

**abstract class Animal {**

**public abstract void sound();**

**}**

**class Lion extends Animal {**

**@Override**

**public void sound() {**

**System.out.println("Lion: Roar!");**

**}**

**}**

**class Tiger extends Animal {**

**@Override**

**public void sound() {**

**System.out.println("Tiger: Growl!");**

**}**

**}**

**public class main2 {**

**public static void main(String[] args) {**

**Animal lion = new Lion();**

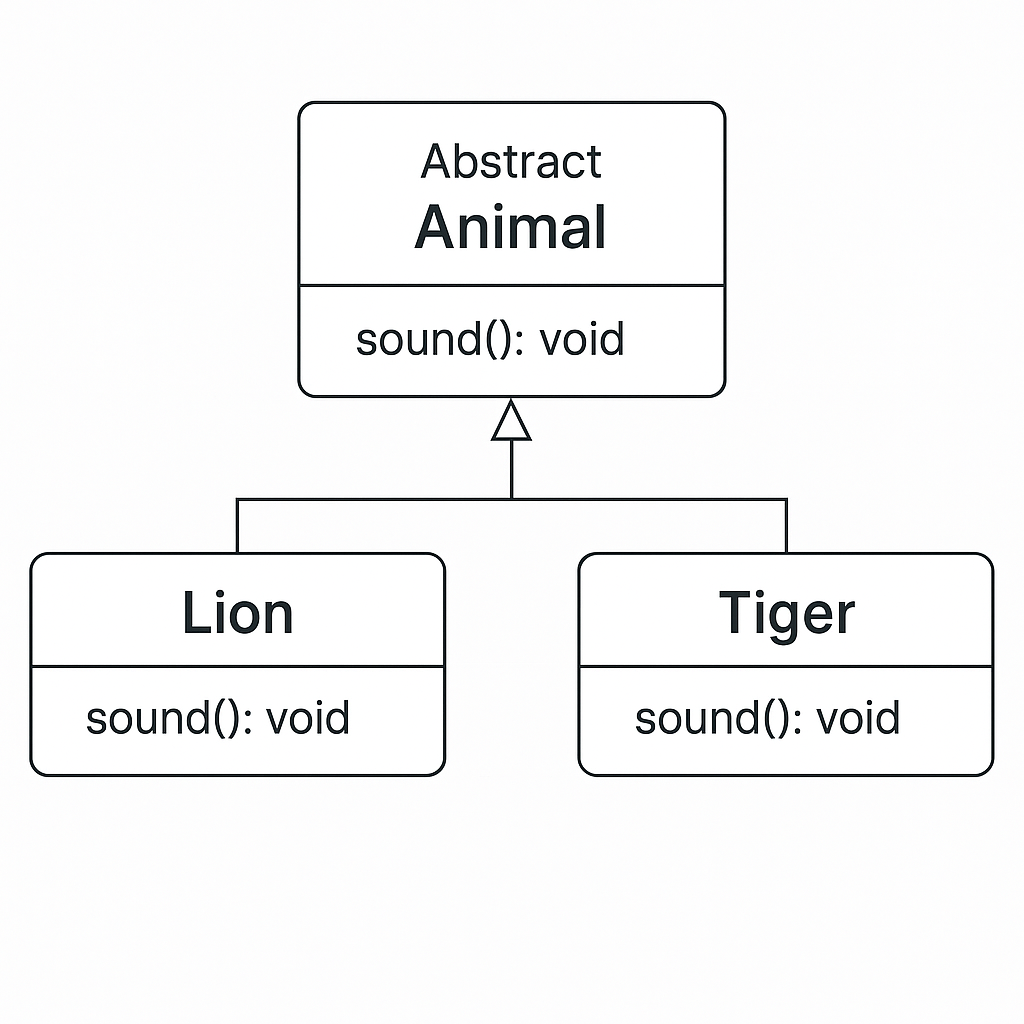
**Animal tiger = new Tiger();**

**lion.sound();**

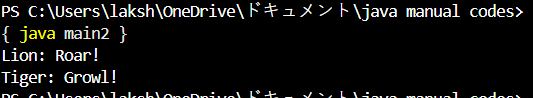
**tiger.sound();**

**}**

**}**

****

**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **Setting the parameters inside the constructor** | **We cannot pass the values inside constructor without setting them first** |
| **2** | **}** | **Ending the class and main method is required** |

**Program-2:**

**Write a java program to create an abstract class Shape3D with an abstract methods Calculate\_volume() and Calculate\_Surface\_area.Create Subclass Sphere and Cube extends the Shape3D class and implement the respective methods to calculate the volume and surface\_area of each shape.**

**Program:**

**abstract class Shape3D {**

**public abstract void calculate\_volume();**

**public abstract void calculate\_surf\_a();**

**}**

**class Sphere extends Shape3D {**

**private double radius;**

**public Sphere(double radius) {**

**this.radius = radius;**

**}**

**@Override**

**public void calculate\_surf\_a() {**

**double surfaceArea = 4 \* Math.PI \* Math.pow(radius, 2);**

**System.out.printf("Surface Area of Sphere: %.2f%n", surfaceArea);**

**}**

**@Override**

**public void calculate\_volume() {**

**double volume = (4.0 / 3) \* Math.PI \* Math.pow(radius, 3);**

**System.out.printf("Volume of Sphere: %.2f%n", volume);**

**}**

**}**

**class Cube extends Shape3D {**

**private double side;**

**public Cube(double side) {**

**this.side = side;**

**}**

**@Override**

**public void calculate\_surf\_a() {**

**double surfaceArea = 6 \* Math.pow(side, 2);**

**System.out.printf("Surface Area of Cube: %.2f%n", surfaceArea);**

**}**

**@Override**

**public void calculate\_volume() {**

**double volume = Math.pow(side, 3);**

**System.out.printf("Volume of Cube: %.2f%n", volume);**

**}**

**}**

**public class main3{**

**public static void main(String[] args) {**

**Shape3D sphere = new Sphere(5);**

**Shape3D cube = new Cube(3);**

**sphere.calculate\_surf\_a();**

**sphere.calculate\_volume();**

**cube.calculate\_surf\_a();**

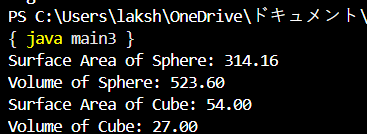
**cube.calculate\_volume();**

**}**

**}**

****

**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **Setting the parameters inside the constructor** | **We cannot pass the values inside constructor without setting them first** |
| **2** | **}** | **Ending the class and main method is required** |

**Program-3:**

**Write a java program using an abstract class to define a method for pattern printing**

**-->create an abstract class named patternprinting with an abstract method print pattern (int n) and a concrete method to display the pattern title**

**-->impletment two sub classes**

1. **star pattern**

**Prints a right angled triangle of stars**

1. **Number pattern**

**Prints a right angled triangle of increasing numbers**

**-->in the main()method create objects of both sub classes and print the patterns for a given number of rows**

**Program:**

**import java.util.Scanner;**

**abstract class Pattern {**

**public abstract void printPattern(int n);**

**}**

**class RightTrianglePattern extends Pattern {**

**@Override**

**public void printPattern(int n) {**

**System.out.println("Right Triangle Pattern:");**

**for (int i = 1; i <= n; i++) {**

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

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

**}**

**System.out.println();**

**}**

**}**

**}**

**class NumberPattern extends Pattern {**

**@Override**

**public void printPattern(int n) {**

**System.out.println("number pattern:");**

**for (int i =1; i <= n; i++) {**

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

**System.out.print( j);**

**}**

**System.out.println();**

**}**

**}**

**}**

**public class main4 {**

**public static void main(String[] args) {**

**Scanner input= new Scanner(System.in);**

**System.out.println("enter the n value to select number of rows");**

**int n=input.nextInt();**

**Pattern rightTriangle = new RightTrianglePattern();**

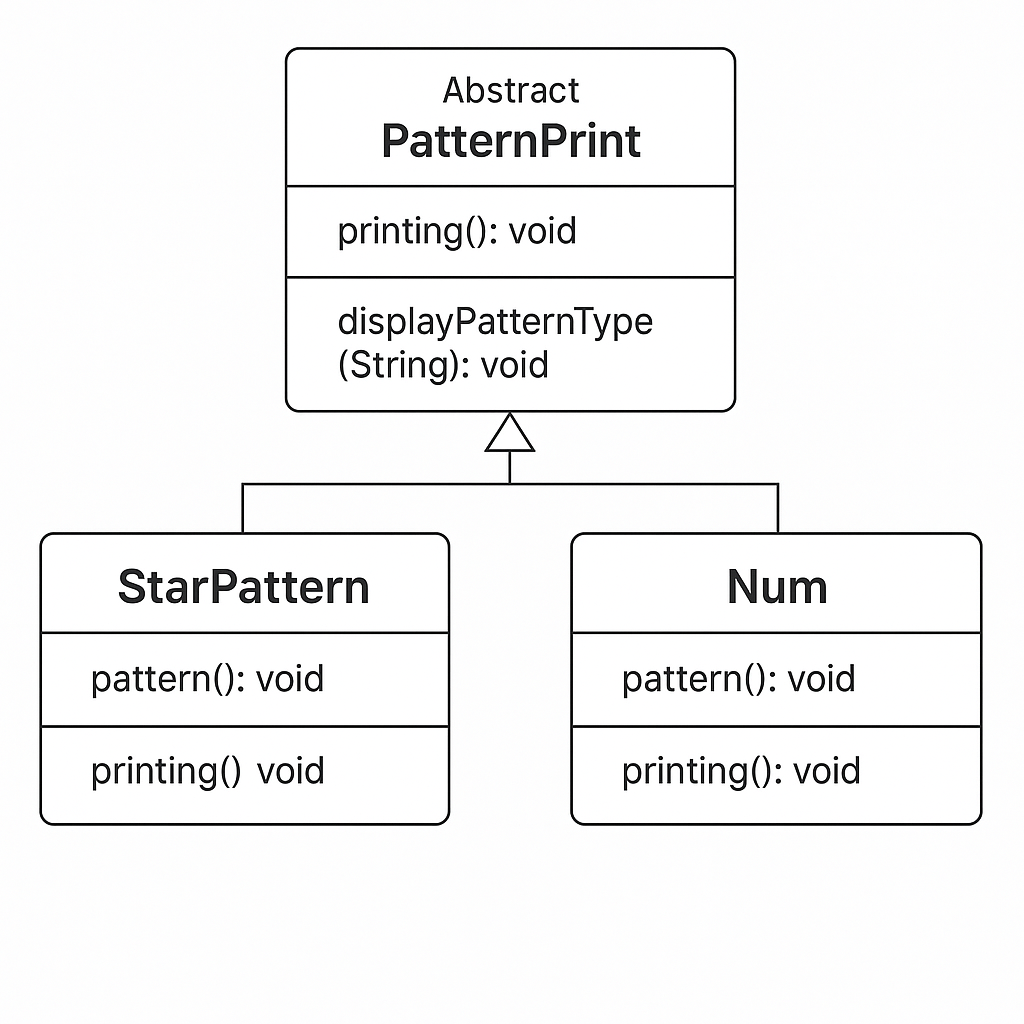
**Pattern numberpattern = new NumberPattern();**

**rightTriangle.printPattern(n);**

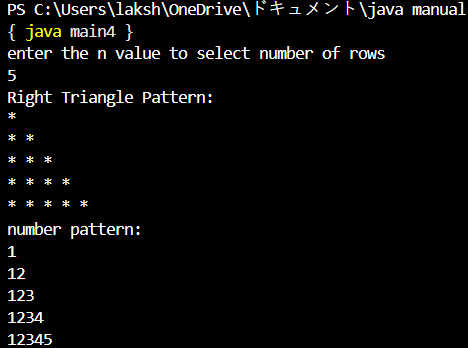
**numberpattern.printPattern(n);**

**}**

**}**

****

**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **Setting the parameters inside the constructor** | **We cannot pass the values inside constructor without setting them first** |
| **2** | **}** | **Ending the class and main method is required** |

**WEEK-8**

**PROGRAM-1:**

write a java program creating an interface Shape with the get perimeter method create 3 classes rectangle,triangleand circle that implements the shape interface ,implement the getperimeter method for each of the three classes

**code:**

**import java.util.\*;**

**interface Shape {**

**int getPerimeter();**

**}**

**class Rectangle implements Shape {**

**int length, breadth;**

**Rectangle(int length, int breadth) {**

**this.length = length;**

**this.breadth = breadth;**

**}**

**public int getPerimeter() {**

**return 2 \* (length + breadth);**

**}**

**}**

**class Triangle implements Shape {**

**int side1, side2, side3;**

**Triangle(int side1, int side2, int side3) {**

**this.side1 = side1;**

**this.side2 = side2;**

**this.side3 = side3;**

**}**

**public int getPerimeter() {**

**return side1 + side2 + side3;**

**}**

**}**

**class Circle implements Shape {**

**double radius;**

**Circle(double radius) {**

**this.radius = radius;**

**}**

**public int getPerimeter() {**

**return (int) (2 \* Math.PI \* radius);**

**}**

**}**

**public class Main16 {**

**public static void main(String[] args) {**

**Shape rect = new Rectangle(5, 10);**

**Shape tri = new Triangle(3, 4, 5);**

**Shape cir = new Circle(4);**

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

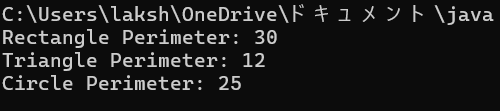
**System.out.println("Triangle Perimeter: " + tri.getPerimeter());**

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

**}**

**}**

**output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **Setting the parameters inside the constructor** | **We cannot pass the values inside constructor without setting them first** |
| **2** | **}** | **Ending the class and main method is required** |

**PROGRAM-2:**

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 implements the playable and override the play method to play the respective sports

**Code:**

**import java.util.\*;**

**interface games{**

**void play();**

**}**

**class football implements games{**

**public void play() {**

**System.out.println("boys are Playing football!");**

**}**

**}**

**class basketball implements games {**

**public void play() {**

**System.out.println("boys are Playing basketball!");**

**}**

**}**

**class volleyball implements games {**

**public void play() {**

**System.out.println("boys are Playing volleyball!");**

**}**

**}**

**public class Main17{**

**public static void main(String[] args) {**

**games foot = new football();**

**games basket = new basketball();**

**games volley= new volleyball();**

**foot.play();**

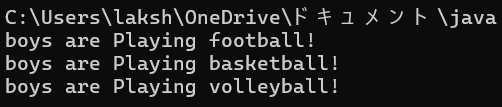
**basket.play();**

**volley.play();**

**}**

**}**

**Output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **Setting the parameters inside the constructor** | **We cannot pass the values inside constructor without setting them first** |
| **2** | **}** | **Ending the class and main method is required** |

**Program-3:**

**write a java program to implement a login system using interfaces**

**code:**

**interface LoginSystem {**

**boolean login(String id, String password);**

**}**

**class UniversityPortal implements LoginSystem {**

**public boolean login(String id, String password) {**

**if (id.equals("student123") && password.equals("pass123")) {**

**return true;**

**} else {**

**System.out.println("Invalid credentials");**

**return false;**

**}**

**}**

**}**

**public class Main18 {**

**public static void main(String[] args) {**

**UniversityPortal portal = new UniversityPortal();**

**boolean loginSuccess1 = portal.login("student123", "pass123");**

**System.out.println("Login successful: " + loginSuccess1);**

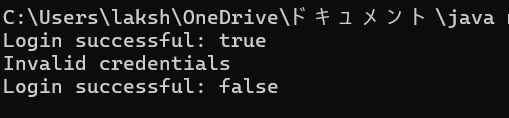
**boolean loginSuccess2 = portal.login("student123", "wrong pass");**

**System.out.println("Login successful: " + loginSuccess2);**

**}**

**}**

**output:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Expected Error** | **Reason** |
| **1** | **Setting the parameters inside the constructor** | **We cannot pass the values inside constructor without setting them first** |
| **2** | **}** | **Ending the class and main method is required** |