**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**   **Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: P. Komal Yashwanth**

**Roll No: 24243**

**Verified By :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **Title** | **Page No** | **Date** | **Signature** |
|  | **WEEK -1** |  |  |  |
| **1** | Explain the process of download & Installation of JDK | **4** |  |  |
| **2** | Write a program to print Student’s name, Roll no, Section | **7** |  |  |
|  |  |  |  |  |

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **WEEK-2** |  |  |  |
| **1** | Write a simple java program to calculate factorial of a number | **9** |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **2** | Write a simple java program to find the simple interest by taking iinputs from the user | **10** |  |  |
| **3** | Write a program to calculate the Fibonacci sequence and take the input from the user | **13** |  |  |
| **4** | Write a program to find the area of triangle using hereon’s formula | **15** |  |  |
| **5** | Write a program to convert temperature from celssius to fahrenheit | **17** |  |  |

# 

# WEEK-1

1. **Process of Installing JDK (Java Development Kit)**

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

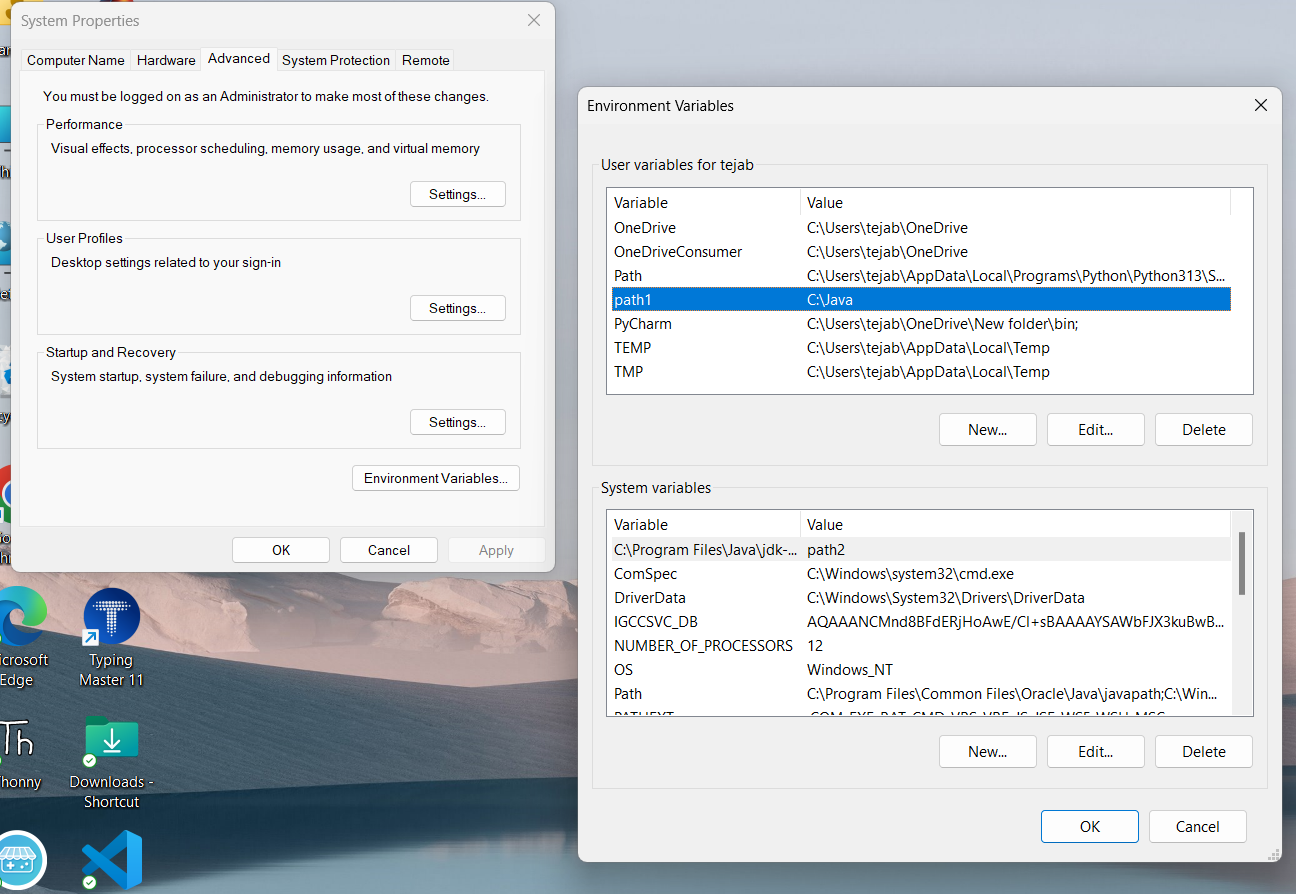
* 1. **Download JDK:**
* Go to the Oracle JDK download page in google and click on JDK-21 version which is Long term support (LTS) version.
* Click the download link as your operating system (Windows, macOS, or Linux).
  1. **Install JDK:**
* Once downloaded, run the installer.
* Follow the given instructions and keep clicking "Next" until it is done.
  1. **Set Environment Variables (Windows):**
* Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on **Environment Variables**.
* Click on path and new under **System Variables**:

**Variable value:** The folder address where JDK is installed (like 4

C:\Program Files\Java\jdk-21\bin)

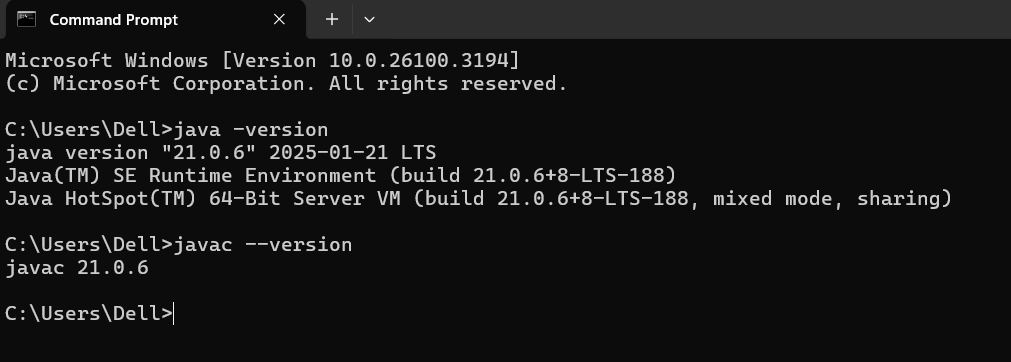
* Find Path under **System Variables**, click **New**, and add the path of the jdk-21(C:\Program Files\Java\jdk-21\bin)

001235456789



**Checking JDK Version: -**

* 1. **Open Command Prompt:**
* Presswin+R, typecmd, and press Enter.
  1. **Check Version:**
* Type java -version and press Enter.
* Type javac --version and press Enter.



1. **Simple Java Program for printing Name, Class, Roll No, of a Student**

Write your code in Notepad and execute it in cmd prompt

**CODE: -**

class Main

{

public static void main(String[] args)

{

System.out.println("Name: Komal");

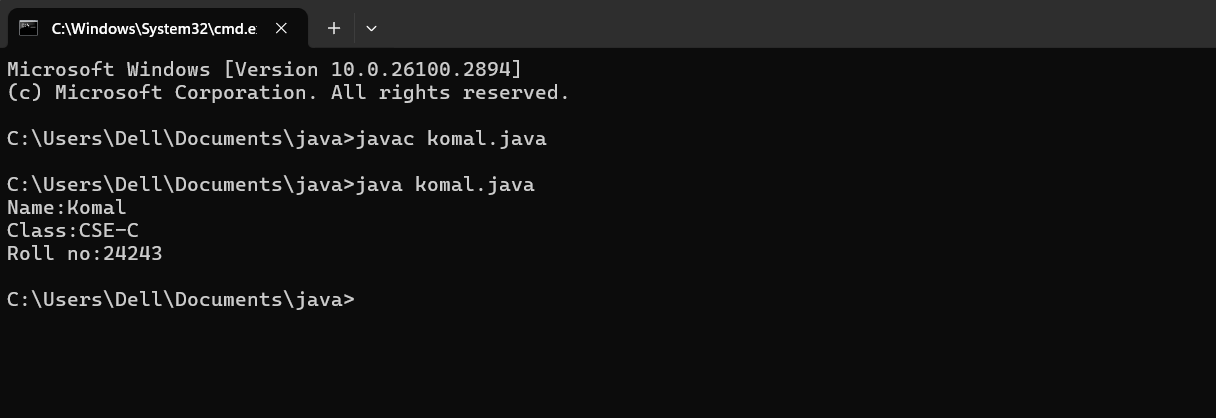
System.out.println("Class :CSE-C");

System.out.println("Roll No:24243");

}

}

**Output: -**



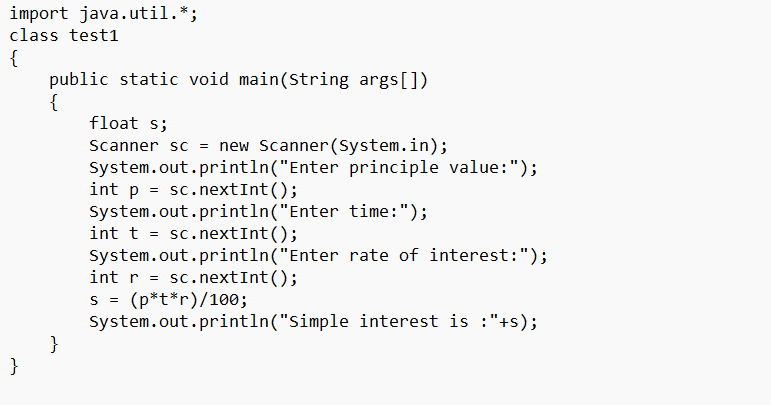
Errors

|  |  |  |
| --- | --- | --- |
| 1 | Syntax error | Semicolon added |
| 2 | Runtime error | Copied correct path |
| 3 | Name error | rectified |

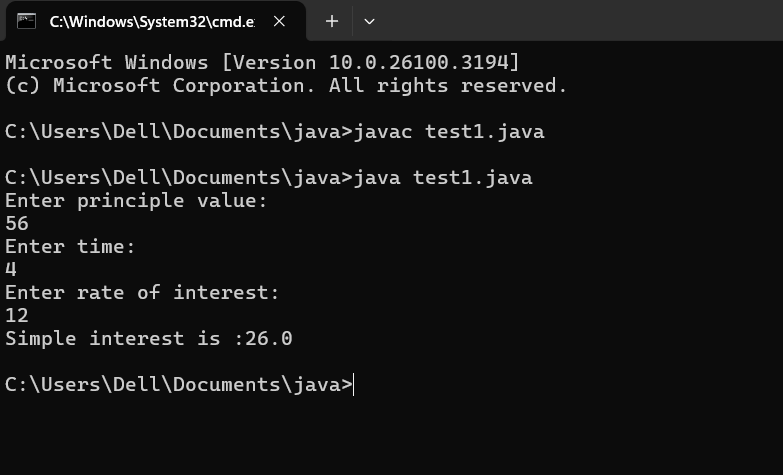
1. **Simple Java Program for finding simple interest by taking input from**

**User**

**Code:**

****

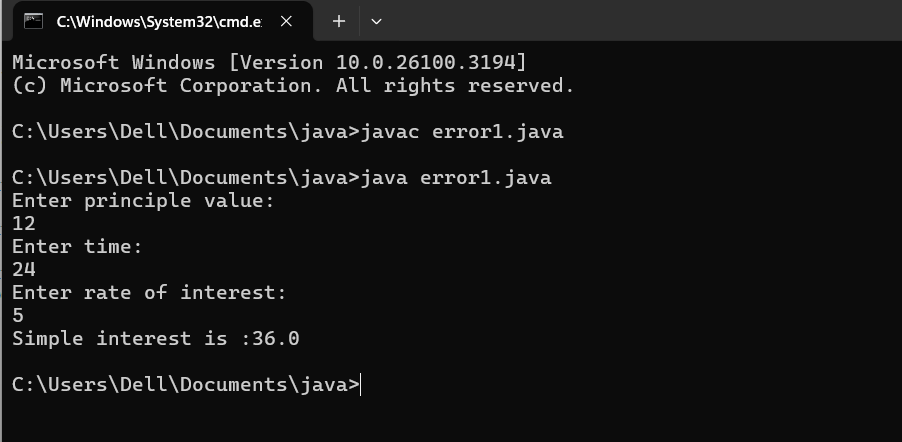
**Output:- Positive Case:**

****

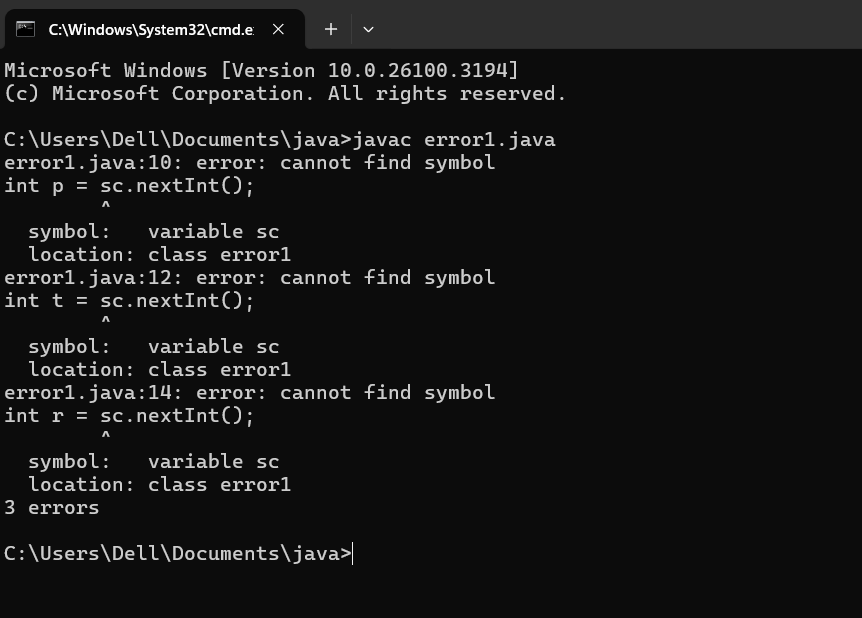
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Logical Error** | **Due to incorrect formula the result may be invalid, In this case int s variable is assigned as (p+t+r)/100** | **We can achieve valid result by assigning proper formula s=(p\*t\*r)/100** |
| **2** | **Compilation Error** | **Scanner code is declared as SC, Java is case-sensitive.** | **Scanner Code is rectified by replacing sc instead of SC** |
| **3** | **Syntax Error** | **The line Scanner SC is missing the semicolon** | **The statement should end with semicolon** |

**Negative case:**

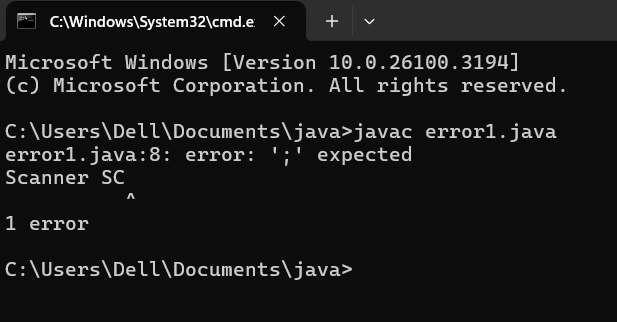
**I)**



**II)**

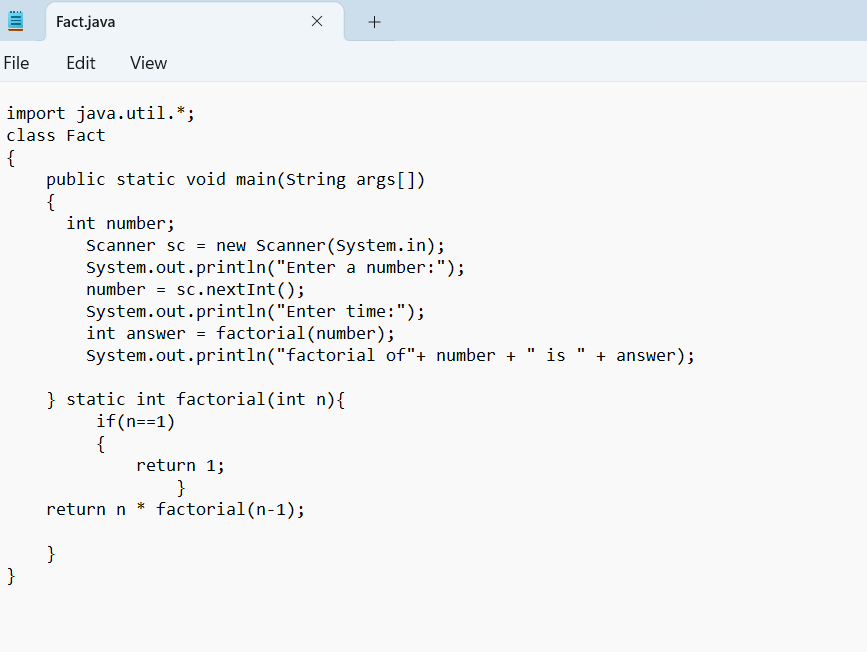
****

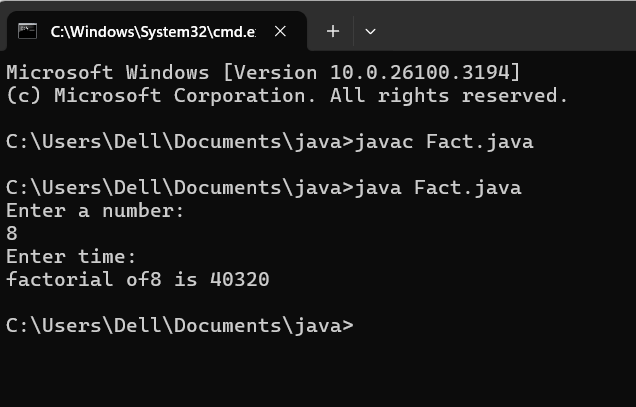
**III)**

****

1. **Write a simple program to calculate factorial of a number and read the**

**input from user Code:**

****

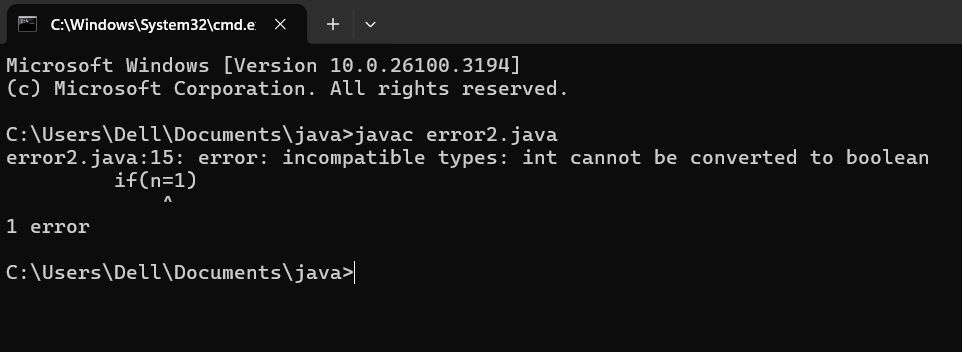
****

**ERRORS:**

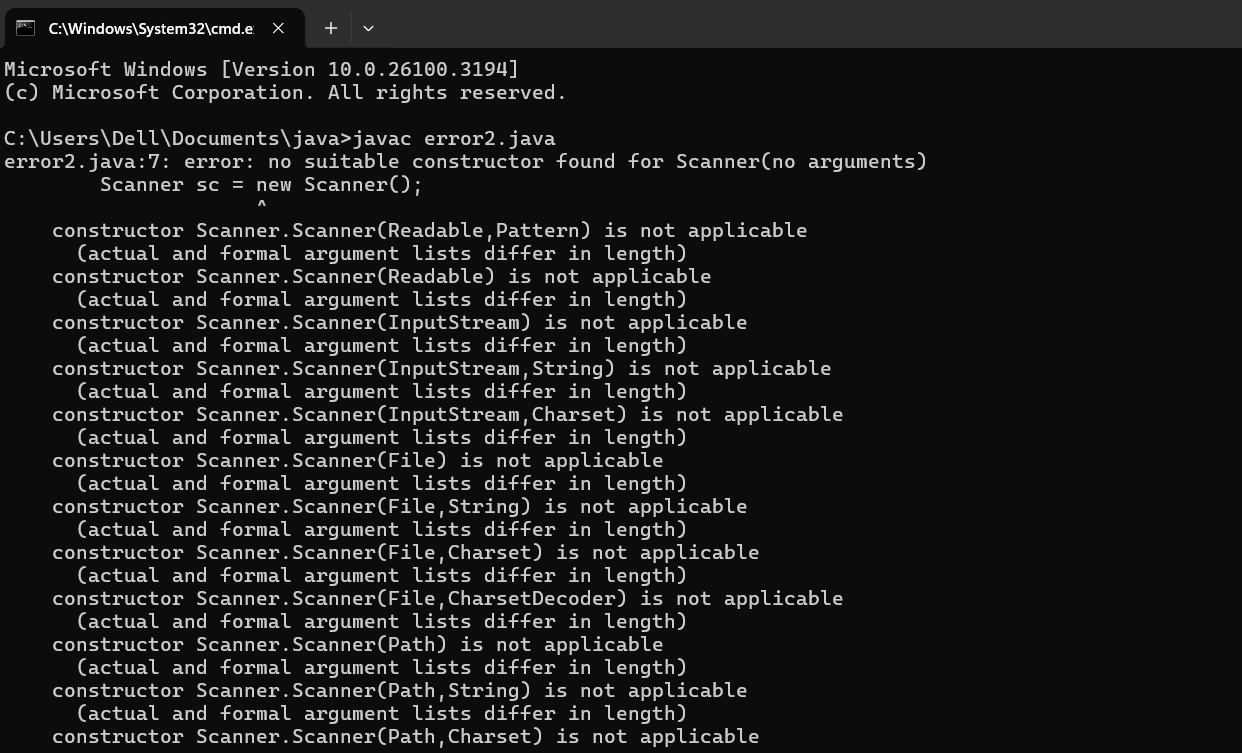
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax Error** | **Incorrect Conditional Statement .n=1** | **The “=” is assignment operator, So in order to compare , the “==” should be used** |
| **2** | **Scanner Constructor Error** | **The Scanner constructor requires input like System.in inside the parenthesis , but It is missing** | **This error could be rectified by typing System.in inside parenthesis.** |
| **3** | **Logical error** | **if(n == 1)**  **{**  **return 0;**  **}** | **The factorial of 1 is 1 , so the if statement should be**  **if(n==1)**  **{**  **return 1;**  **}** |

**Negative Case:**

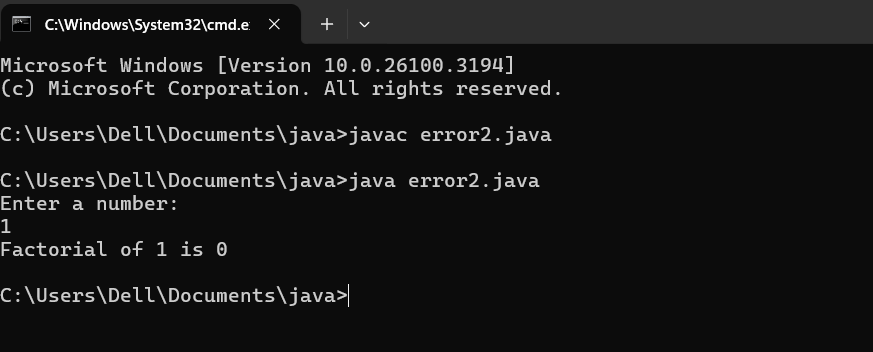
**I)**

****

**II)**

****

**III)**

****

**3.Write a program to to calculate the fibonacii sequence and take the input from user**

**Code:**

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

**class fibo**

**{**

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

**{**

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

**int num;**

**int f3;**

**int f1 = 0;**

**int f2 = 1;**

**int i = 2;**

**System.out.print("Enter a number:");**

**num = sc.nextInt();**

**System.out.println(f1);**

**System.out.println(f2);**

**while(i<num)**

**{**

**f3 = f1+f2;**

**f1 = f2;**

**f2 = f3;**

**System.out.println(f3);**

**i = i+1;**

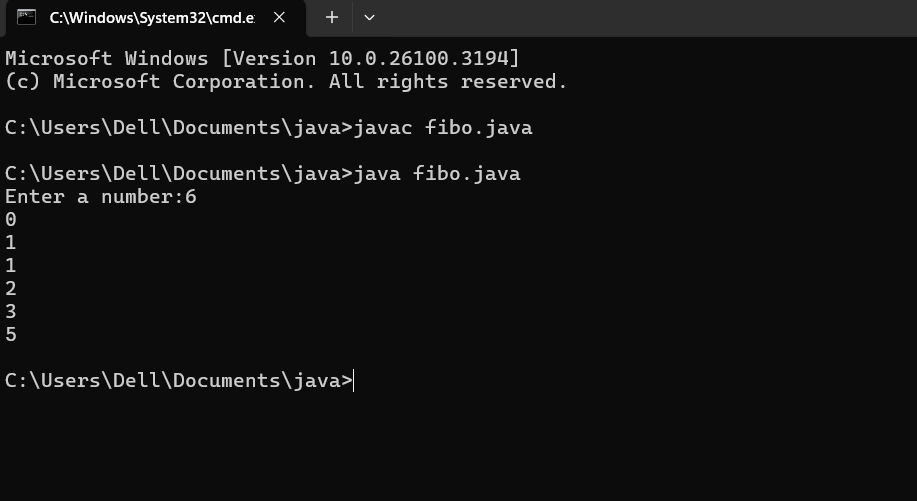
**}**

**}**

**}**

**Output:**

**POSITIVE CASE:**

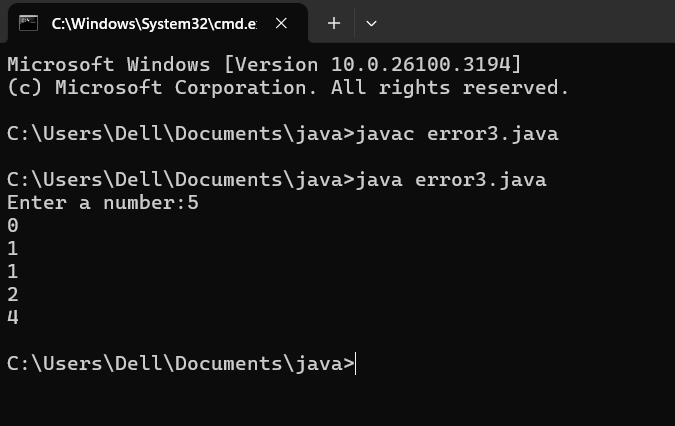
****

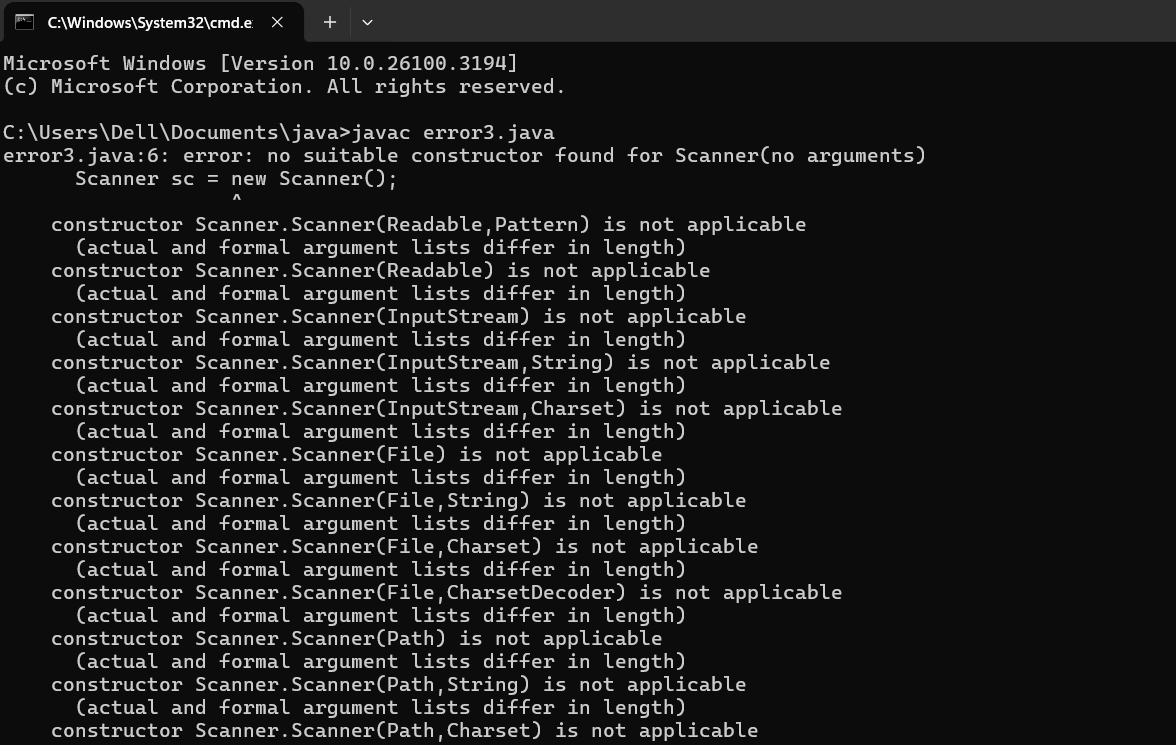
**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **The variables are assigned as f1=f3**  **f2=f1** | **But the variables should be assigned as f1=f2 and f2=f3** |
| **2** | **Run-time error** | **The Scanner constructor requires input like System.in inside the parenthesis , but It is missing** |  |

**NEGATIVE ERRORS:**

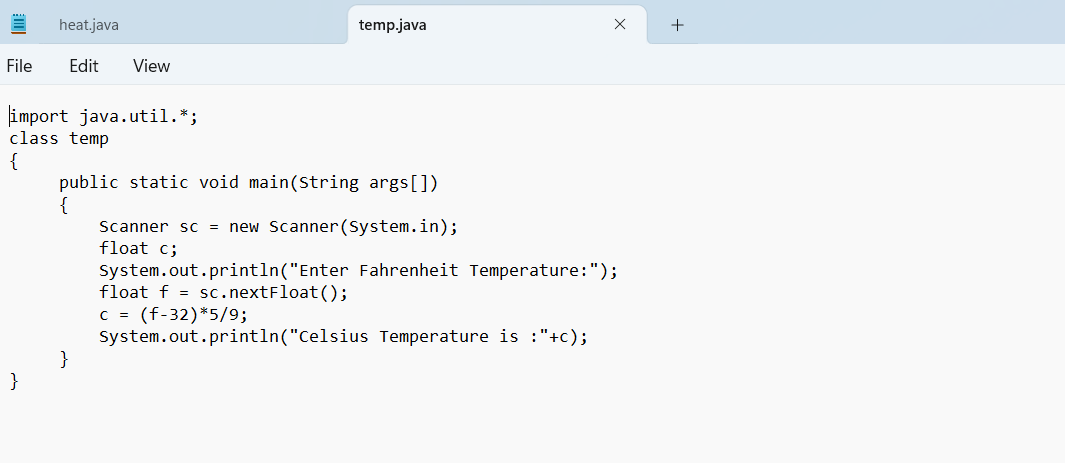
**I)**

****

**II)**

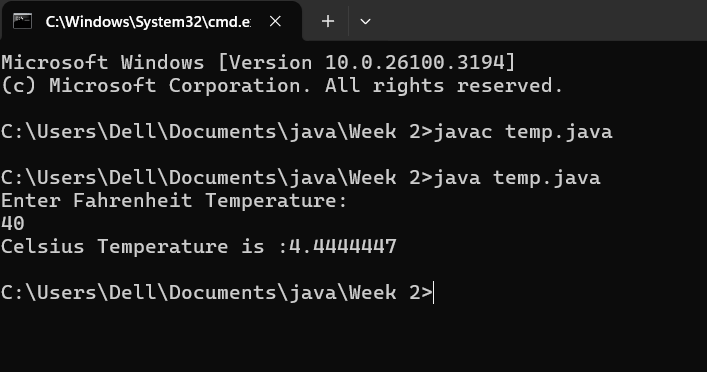
**4.Write a java program to convert temperature from Fahrenheit to Celsius**

**CODE:**

****

**Output:**

**POSTIVE CASE:**

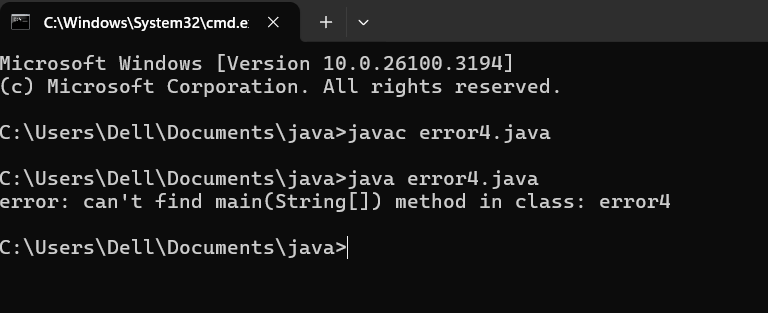
****

**ERRORS:**

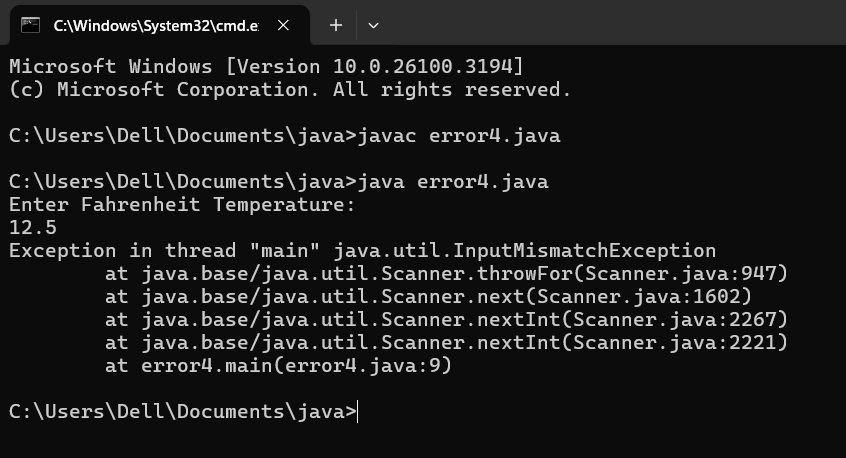
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Syntax error** | **The argument should be String args[] instead of String arg.** | **The code should be**  **public static void main(String args[])** |
| **2** | **Incorrect Input Method for float** | **nextInt() is used instead of nextFloat()** | **float f = sc.nextFloat(); should be typed to rectify the error** |
| **3** | **Logical error** | **The formula for conversion of Fahrenheit to Celsius is written as c=(f-32)\*5//9** | **This error could be rectified by using the right formula: c=(f-32)\*5/9** |

**NEGATIVE CASE:**

**I)**

****

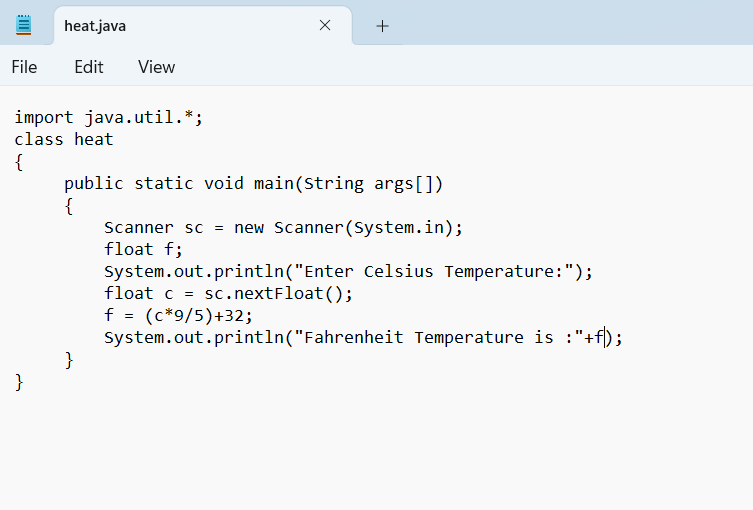
**II)**

****

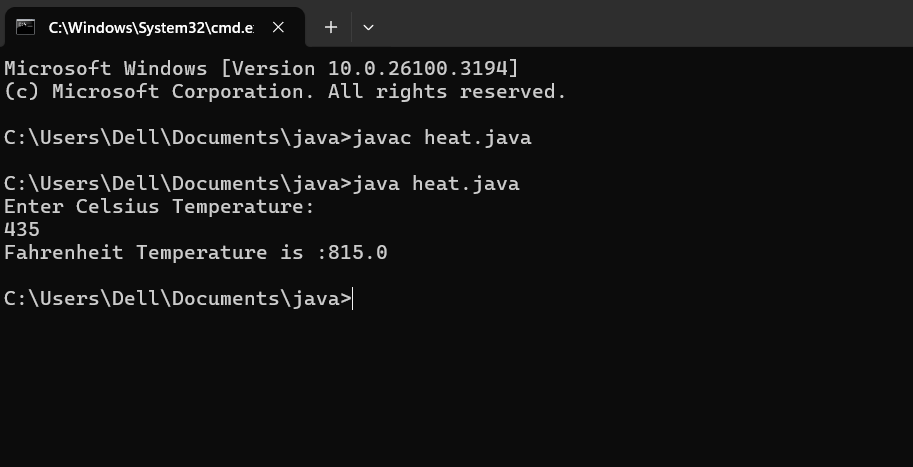
**III)**

**5.Write a java program to convert temperature from Celsius to Fahrenheit**

**Code**

****

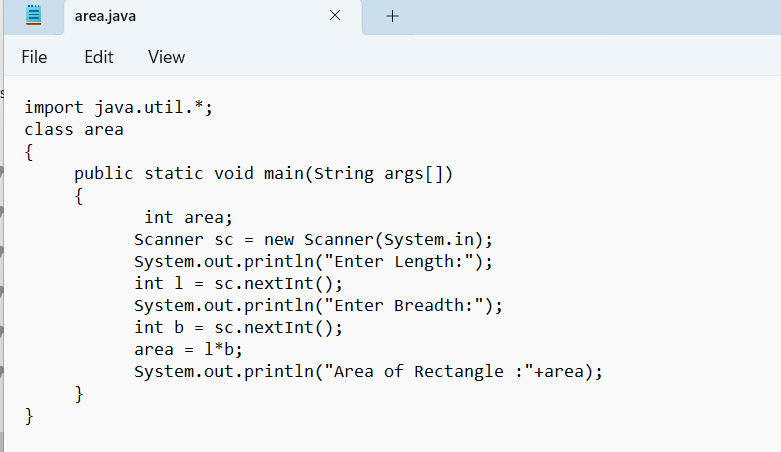
**Output:**

****

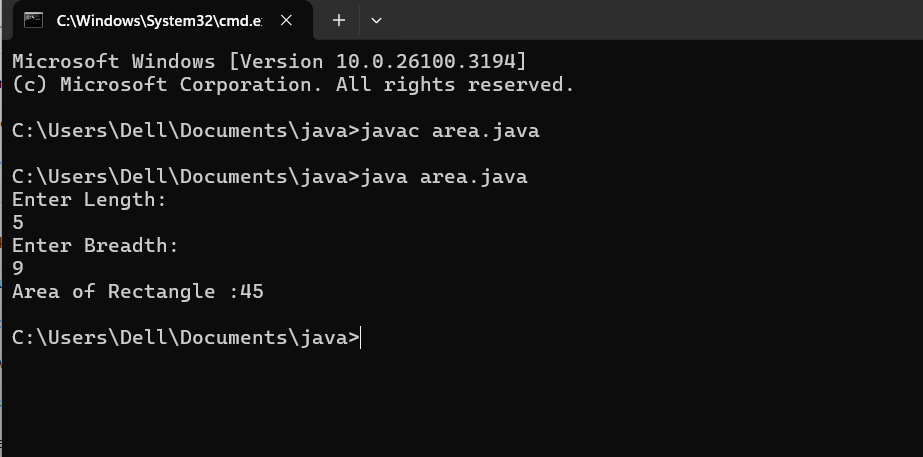
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Syntax error** | **Missing ”** | **“ is added** |
| **2** | **Missing import error** | **Util package missing** | **Util package added** |
| **3** |  |  |  |

**6.Write a simple program to find the area of rectangle:**

**Code:**

****

**Output:**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **Incorrect formula** | **Formula rectified** |
| **2** | **Name error** | **Undeclared variable** | **Variable declared** |
|  |  |  |  |

**WEEK - 3**

**Aim:**

**To create java program with following instructions**

**1.Create a class with name car**

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

**3. Create three methods named start(), stop(). Service()**

**4. Create three objects named car1,car2 and car3**

**Code:**

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

**class car**

**{**

**public String Car\_color;**

**public String Car\_brand;**

**public String fuel\_type;**

**public int mileage;**

**public void start()**

**{**

**System.out.println("Car Started:");**

**System.out.println("Car color is :"+Car\_color);**

**System.out.println("Car Brand is:"+Car\_brand);**

**System.out.println("Car fuel type is:"+fuel\_type);**

**System.out.println("Car mileage is:"+mileage);**

**}**

**public void service()**

**{**

**System.out.println("Car Started:");**

**System.out.println("Car color is :"+Car\_color);**

**System.out.println("Car Brand is:"+Car\_brand);**

**System.out.println("Car fuel type is:"+fuel\_type);**

**System.out.println("Car mileage is:"+mileage);**

**}**

**public void stop()**

**{**

**System.out.println("Car Started:");**

**System.out.println("Car color is :"+Car\_color);**

**System.out.println("Car Brand is:"+Car\_brand);**

**System.out.println("Car fuel type is:"+fuel\_type);**

**System.out.println("Car mileage is:"+mileage);**

**}**

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

**{ System.out.println("\n komal\n\n");**

**car car1 = new car();**

**car1.Car\_color = "Blue";**

**car1.Car\_brand = "BMW";**

**car1.fuel\_type = "Deisel";**

**car1.mileage = 10;**

**car1.start();**

**car car2 = new car();**

**car2.Car\_color = "Red";**

**car2.Car\_brand = "Tesla";**

**car2.fuel\_type = "EV";**

**car2.mileage = 300;**

**car2.stop();**

**car car3 = new car();**

**car3.Car\_color = "Yellow";**

**car3.Car\_brand = "MAHINDRA";**

**car3.fuel\_type = "Petrol";**

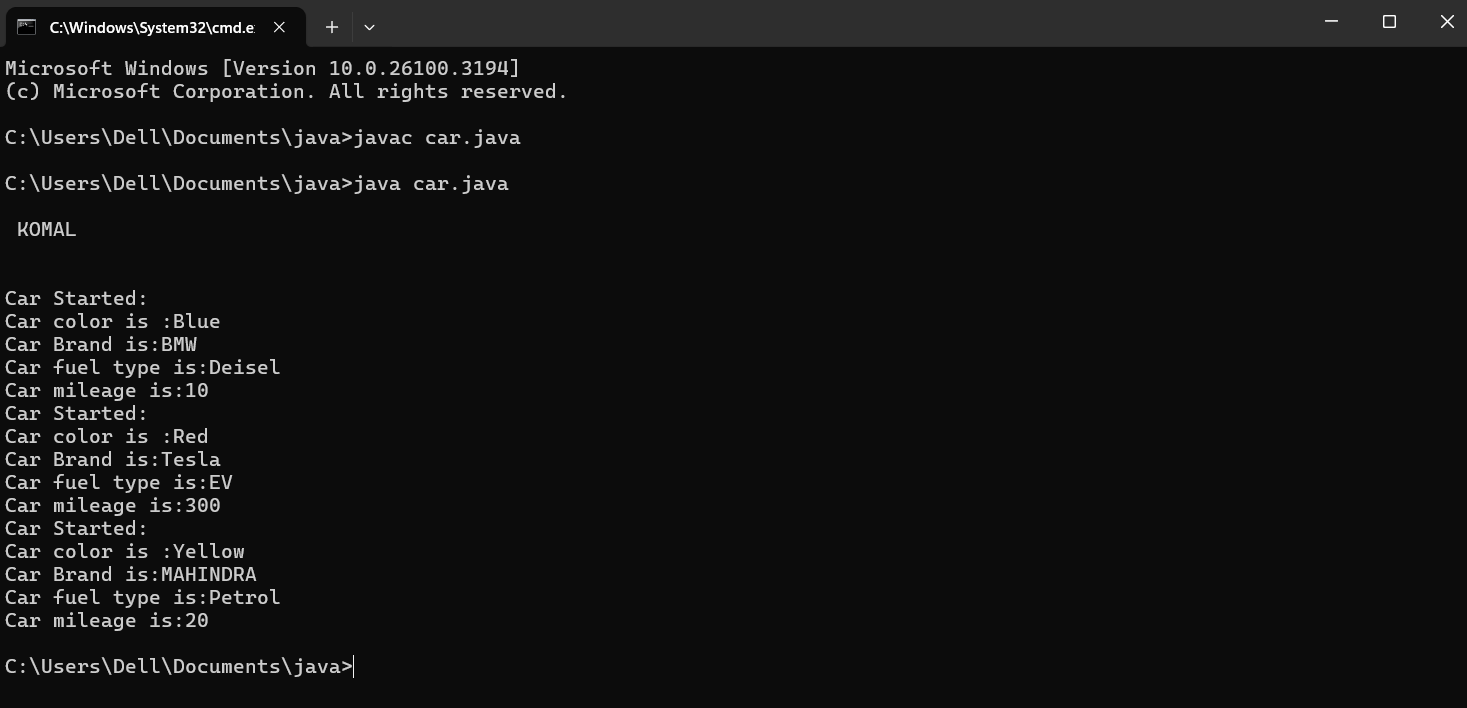
**car3.mileage = 20;**

**car3.service();**

**}**

**}**

**Output:**

****

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.

car1.start(); - Calling a method, under object car1.

**2. To create a class bankAccount with methods deposit() and withdrawl**

**Code:**

**class BankAccount**

**{**

**private double balance;**

**public BankAccount(double initialBalance)**

**{**

**if(initialBalance > 0)**

**{**

**this.balance = initialBalance;**

**}**

**else**

**{**

**this.balance = 0;**

**}**

**}**

**public void deposit(double amount)**

**{**

**if(amount>0)**

**{**

**balance = balance+amount;**

**System.out.println("Deposited $:"+amount);**

**}**

**else**

**{**

**System.out.println("Deposited amount must be positive");**

**}**

**}**

**public double getBalance()**

**{**

**return balance;**

**}**

**}**

**public class Main1**

**{**

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

**{**

**BankAccount account = new BankAccount(1000);**

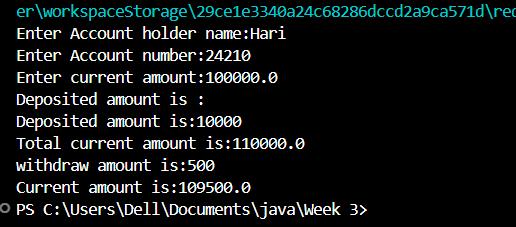
**account.deposit(500);**

**System.out.println("Current Balance is:"+account.getBalance());**

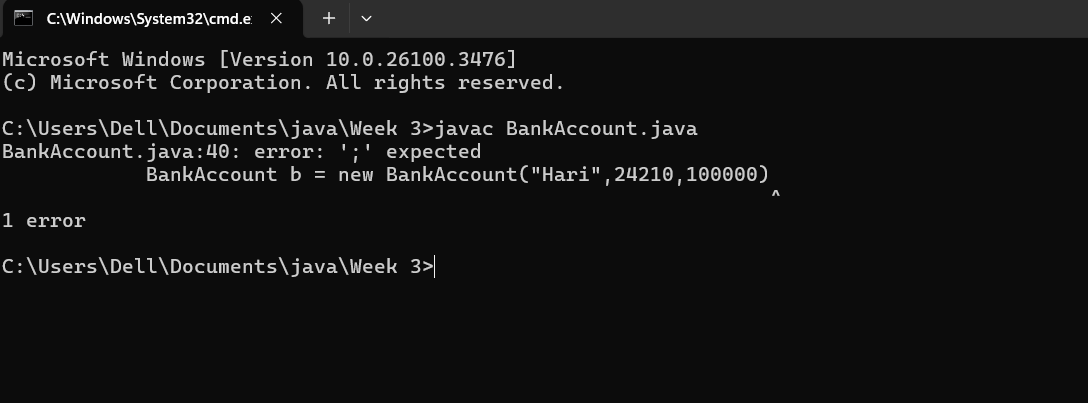
**}**

**}**

Output:



Negative Case:



**Important points:**

**Constructor: The Constructor creates and initializes objects of a class. They are called**

**when an object is created to a class.**

**This Keyword: The This keyword refers to the current instance of a class.It is used to**

**Access class variables and methods.**

**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **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; |

**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.

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

WEEK-4

**1.AIM:**

**WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOUKD CONTAIN VARIOUS ATTRIBUTES SUCH AS TITLE, AUTHOR, YEAR OF**

**PUBLICATION. IT SHOULD ALSO CONTAIN A CONSTRUCTOR WITH**

**PARAMETERS WHICH INITIALIZES TITLE, AUTHOR, YEAR OF PUBLICATION**

**AND CREATE A METHOD WHICH DISPLAYS THE DETAILS OF 2 BOOKS.**

**PROGRAM:**

**public class Book {**

**public String title;**

**public String author;**

**public int year;**

**Book(String title, String author, int year) {**

**this.title = title;**

**this.author = author;**

**this.year = year;**

**}**

**public void displayDetails() {**

**System.out.println("Title: " +title);**

**System.out.println("Author: " +author);**

**System.out.println("Year of Publication" +year);**

**}**

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

**Book b1 = new Book("Math", "Ramanujan", 1950);**

**Book b2 = new Book("Physics", "CV Raman", 1960);**

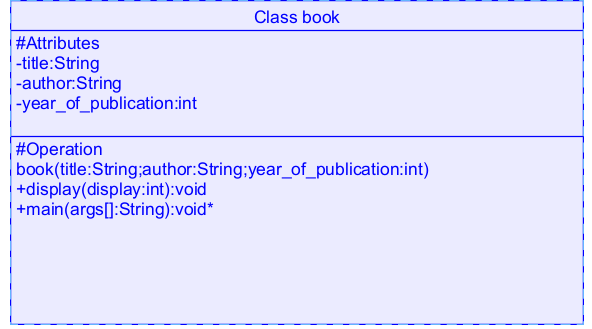
**b1.displayDetails();**

**b2.displayDetails();**

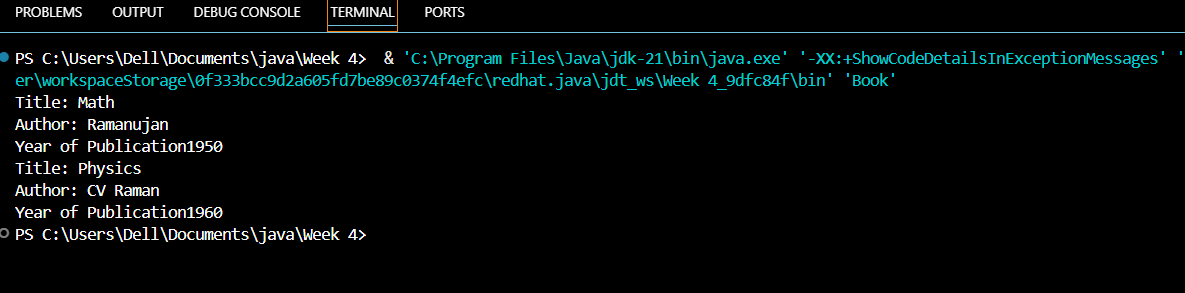
**}**

**}**

**Class Diagram:**

****

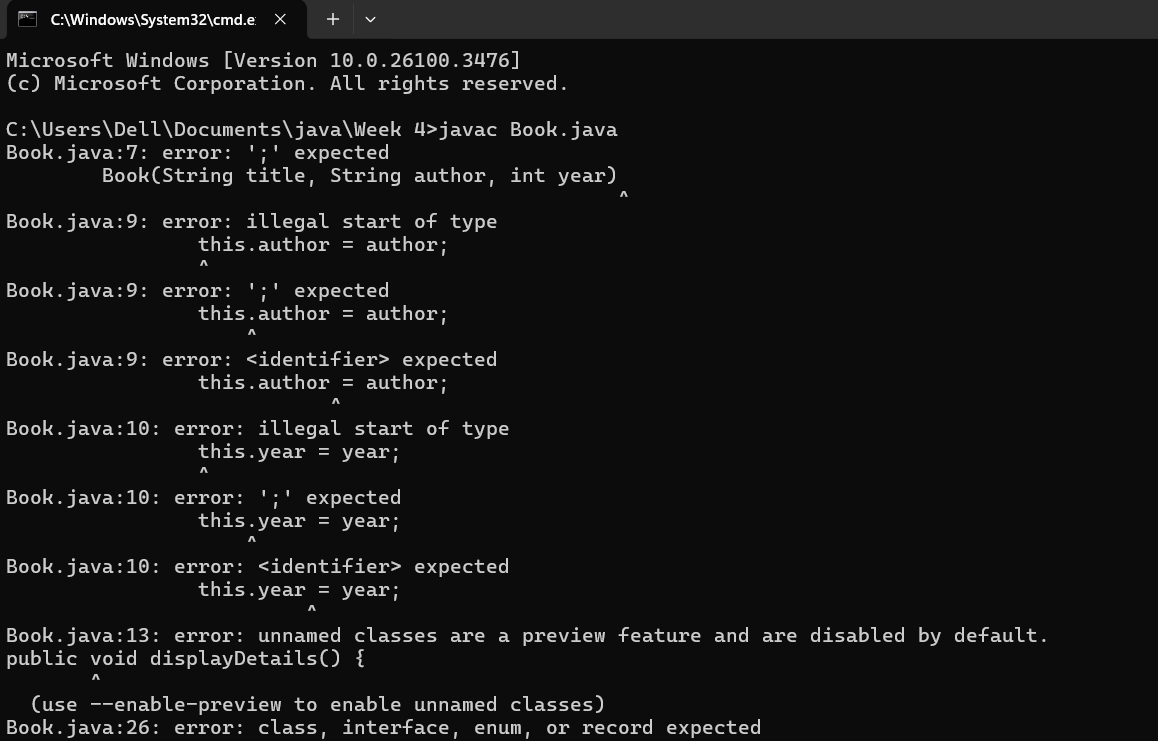
OUTPUT:



**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **s.no** | **Error name** | **Cause of error** | **Rectification** |
| **1** | **Name Error** | **Undefined name** | **Correct variable**  **Name replaced** |
| **2** | **Syntax Error** | **Missing Parenthesis** | **Parenthesis Added** |
| **3** | **Logical Error** | **Incorrect Condition** | **Condition Rectified** |

**Negative Case:**

****

**Important points:**

**Constructor: The Constructor creates and initializes objects of a class. They are called**

**when an object is created to a class.**

**This Keyword: The This keyword refers to the current instance of a class.It is used to**

**Access class variables and methods.**

**2.Create a java program with class named “myclass” with a static variable**

**“count” of int type, initialized to zero and a constant variable “pi” of type**

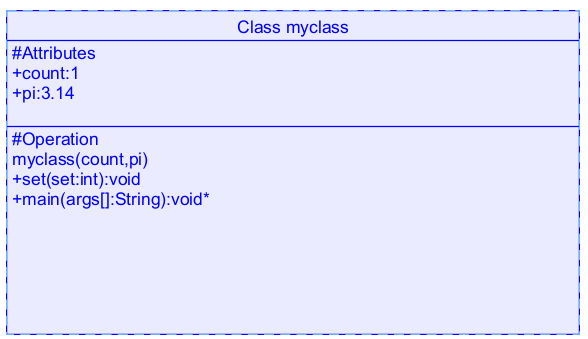
**Double initialized to 3.14 as attributes of the class. Now define a constructor**

**For “myclass” that increments the count variable each time an object of**

**“myclass” is created**

**Finally Print the final values of count and pi variables. Create three objects**

**Class Diagram:**



**Code:**

**class myclass**

**{**

**static int count=0;**

**static double pi=3.14;**

**myclass()**

**{**

**count = count+1;**

**}**

**public void set()**

**{**

**System.out.println("Count is:"+count);**

**System.out.println("Pi value is:"+pi);**

**}**

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

**{**

**myclass m = new myclass();**

**m.set();**

**myclass m1 = new myclass();**

**m1.set();**

**myclass m2 = new myclass();**

**m2.set();**

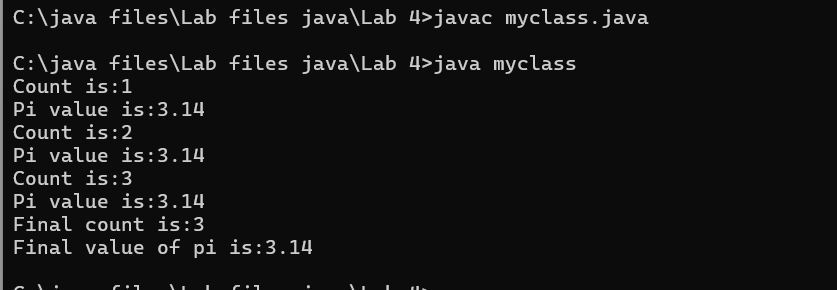
**System.out.println("Final count is:"+count);**

**System.out.println("Final value of pi is:"+pi);**

**}**

**}**

**Output:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error Type** | **Reason for error** | **Rectification** |
| **1.** | **No class** | **No class name declared** | **Created class named ‘MyClass’** |
| **2.** | **Syntax error** | **Not added keyword** | **Added keyword named ‘new’** |

Negative Case:



**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**

**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));

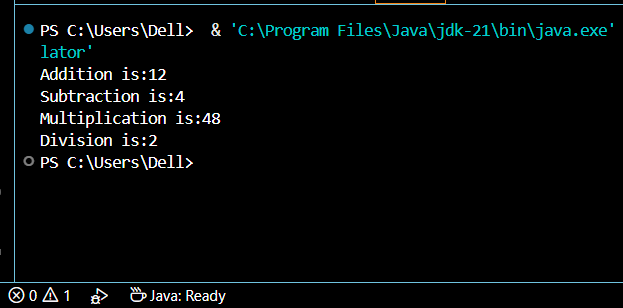
           System.out.println("Multiplication is:"+obj.mult(12,4));

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

    }

}

Output:



**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 |

**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

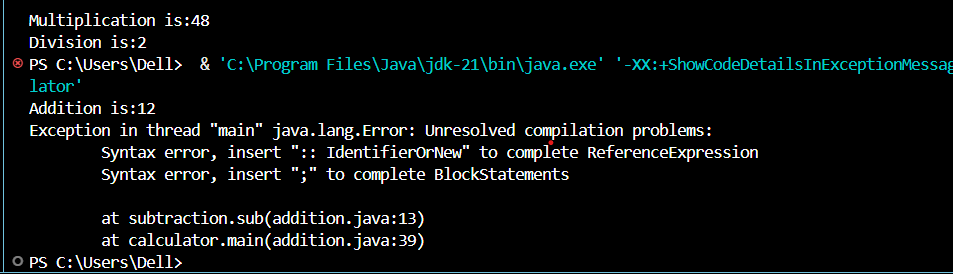
**Multilevel Inheritence:**

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

**ERROR:**

****

**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.

**INPUT:-**

class Vehicle {

String brand;

int speed;

public Vehicle(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

public static void main(String[] args) {

Car obj1 = new Car("Ford", 34, 4);

Bike obj2 = new Bike("Hero", 100, true);

Truck obj3 = new Truck("Tata", 60, 40);

}

}

class Car extends Vehicle {

int noOfDoors;

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

super(brand, speed);

this.noOfDoors = noOfDoors;

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

System.out.println("Speed of car is: " + speed);

System.out.println("No of doors of car: " + noOfDoors);

}

}

class Bike extends Vehicle {

boolean gears;

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

super(brand, speed);

this.gears = gears;

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

System.out.println("Speed of bike is: " + speed);

System.out.println("Gears of bike: " + gears);

}

}

class Truck extends Vehicle {

int weight;

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

super(brand, speed);

this.weight = weight;

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

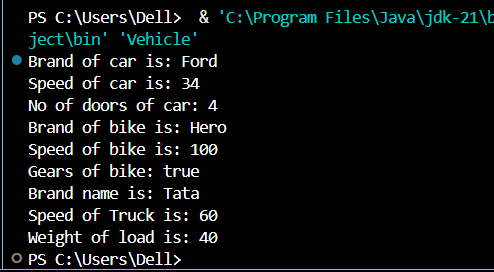
System.out.println("Speed of Truck is: " + speed);

System.out.println("Weight of load is: " + weight);

}

}

OUTPUT:



**ERROR TABLE:-**

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

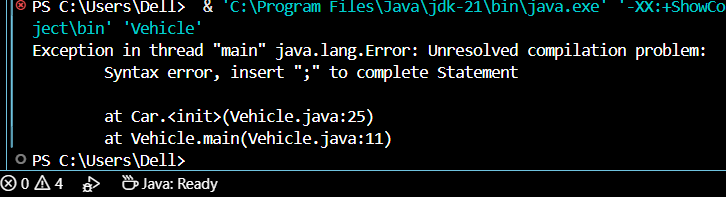
**Important Points**

**Hierarchical Inheritence:**

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

Single parent class

ERROR:



**WEEK - 6**

**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.**

**Program :**

class vehicle{

    String company;

    String model;

    String fuel;

    int capacity;

    void displayInfo(String company,String model,String fuel,int capacity){

        System.out.println("The details of vehicle: ");

        this.company=company;

        this.model=model;

        this.fuel=fuel;

        this.capacity=capacity;

    }

}

class car extends vehicle{

    void displayInfo(String company,String model,String fuel,int capacity){

        System.out.println("Company: "+company);

        System.out.println("Model: "+model);

        System.out.println("Fuel: "+fuel);

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

    }

}

class poly1{

    public static void main(String[] args){

        car car1=new car();

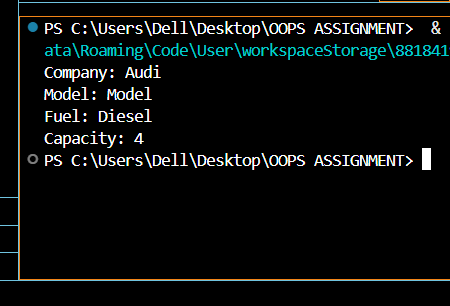
        car1.displayInfo("Audi","Model","Diesel",4);

    }

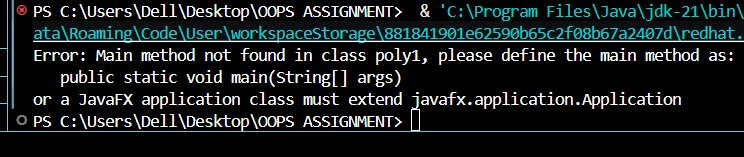
}

OUTPUT:

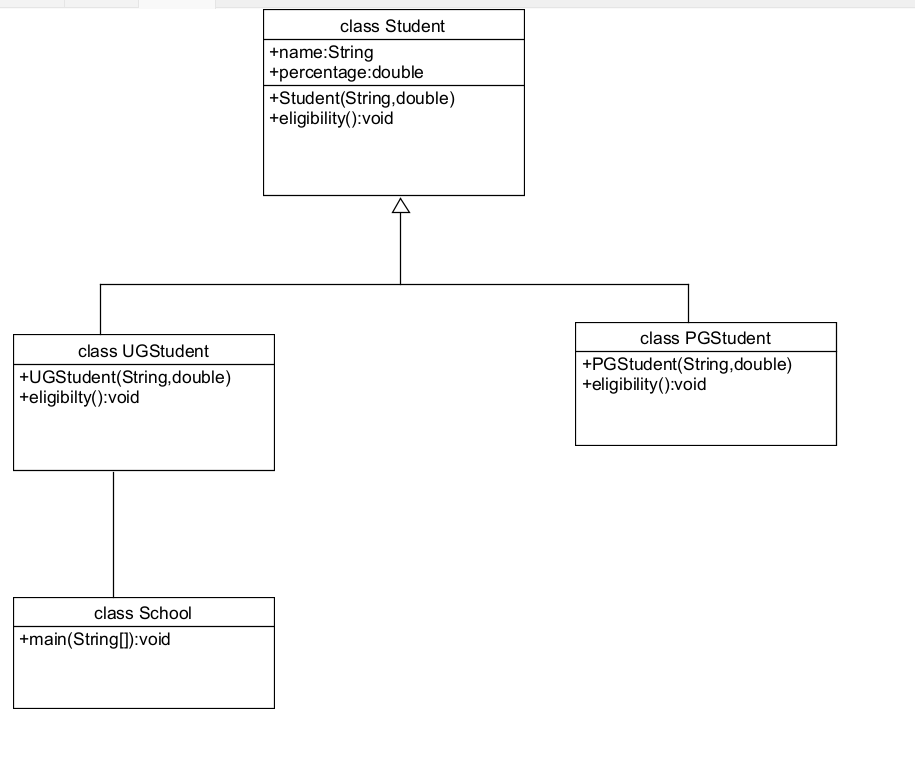
POSITIVE CASE:



NEGATIVE CASE:



CLASS DIAGRAM:



**ERRORS :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | [] is missed | [] is added |
| 2 | Logical error | Incorrect logic | Correct logic |

Important points:

1. **Inheritance** allows the Car class to reuse the Vehicle class functionality without repeating code.

2. **Method Overriding** enables the Car class to provide its own implementation of the displayInfo() method.

3. **Polymorphism** makes the code flexible, allowing different classes (e.g., Car, Bike) to provide customized behavior for displayInfo().

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

(i)UG admissions require a minimum of 60%.

(ii)PG admissions require a minimum of 70%

**Program :**

class College{

    String name;

    int percentage;

    void geteligibility(String name,int percentage){

        this.name=name;

        this.percentage=percentage;

    }

}

class UG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=60){

            System.out.println(name+" is eligible");

        }

        else{

            System.out.println(name+" is not eligible");

        }

    }

}

class PG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=70){

            System.out.println(name+" is eligible");

        }

        else{

            System.out.println(name+" is not eligible");

        }

    }

}

class poly2{

    public static void main(String[] args){

        UG ug=new UG();

        ug.geteligibility("Person-1",40);

        PG pg=new PG();

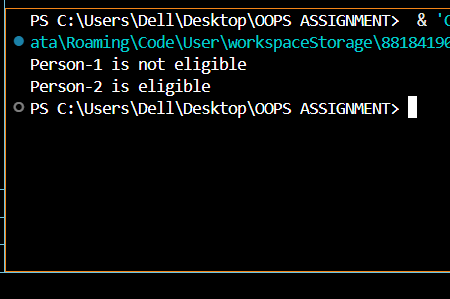
        pg.geteligibility("Person-2",80);

    }

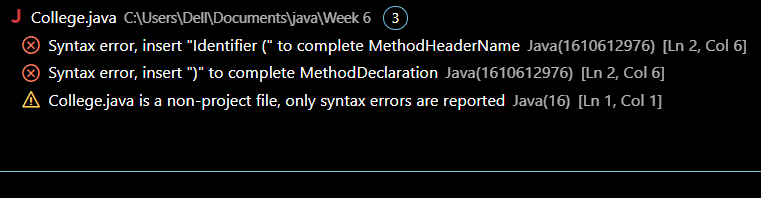
}

**OUPUT :**

POSITIVE CASE:



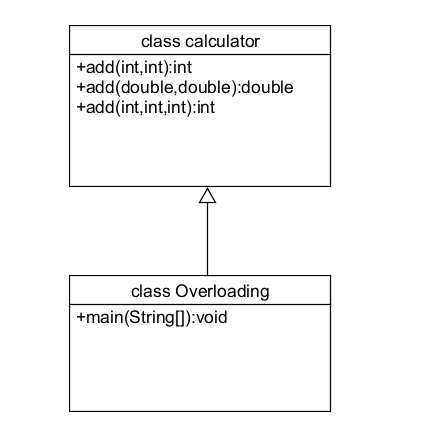
NEGATIVE CASE:



**ERROR :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | syntax error | String forgot in main function | String is added |
| 2 | Logical error | Incorrect logic | Correct logic |

CLASS DIAGRAM:



IMPORTANT POINTS:

1.**Inheritance** allows PG and UG classes to reuse the University class’s attributes and constructor.

2. **Method Overriding** customizes the office() method in PG and UG classes to implement specific admission criteria.

3. **Polymorphism** enables the same office() method to behave differently for PG and UG objects based on their type

**3) Aim :** Create a Calculator class with overloaded methods to perform addition:

(i) Add two integers.

(ii) Add two doubles.

(iii) Add three integers.

**Program :**

class Calcee{

    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 poly3{

    public static void main(String[] args){

        Calcee C1=new Calcee();

        System.out.println("Sum of 2 and 5 is: "+C1.add(2,5));

        System.out.println("Sum of 3.2 and 6.5 is: "+C1.add(3.2,6.5));

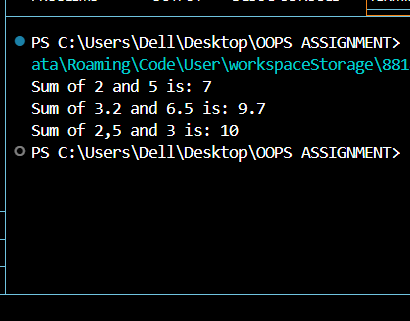
        System.out.println("Sum of 2,5 and 3 is: "+C1.add(2,5,3));

    }

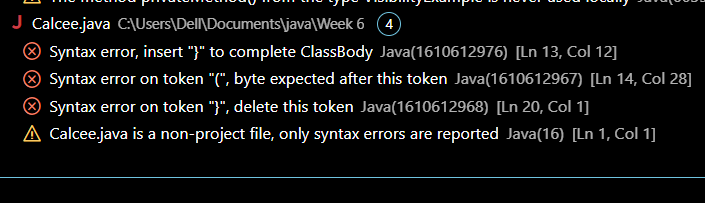
}

**OUTPUT :**

POSITIVE CASE:



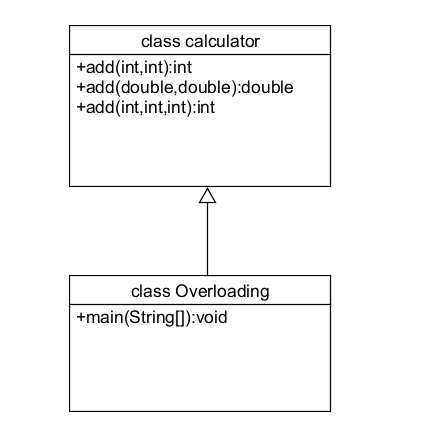
NEGATIVE CASE:



**ERROR :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | syntax error | String forgot in main function | String is added |
| 2 | Logical error | Incorrect logic | Correct logic |

**CLASS DIAGRAM:**



IMPORTANT POINTS :

1. **Method Overloading** allows the Shape class to calculate areas for different shapes (rectangle, triangle, square) using multiple calculateArea() methods with different parameters.
2. **Method Overriding** in the Circle class provides a custom implementation of calculateArea() for calculating the area of a circle.
3. **Polymorphism** enables objects of different types (e.g., Shape, Circle) to call the appropriate version of calculateArea() based on the object type and input parameters.

**4)Aim :** Create a Shape class with a method calculateArea() that is overloaded for different shapes (e.g., square, rectangle). Then, create a subclass Circle that overrides the calculateArea() method for a circle.

**Program :**

class Shape { // class shape

    void calculateArea( int a) { // method 1

        System.out.println("The area of Square is :" + (a\*a) );

    }

    void calculateArea(int a , int b) { // method 2

        System.out.println("The area of rectangle is :" + (a\*b));

    }

}

class circle extends Shape { // inheritance class

    void calculateArea(double a){ // method overloading

        System.out.println("The area of circle is :" + (3.14\*a\*a));

  } }

class main { // main program

    public static void main(String[] args) {

        // creating objects for class

        Shape s = new Shape();

        circle c = new circle();

        // calling methods

        s.calculateArea(4);

        System.out.println("    ");

        s.calculateArea(4, 5);

        System.out.println("    ");

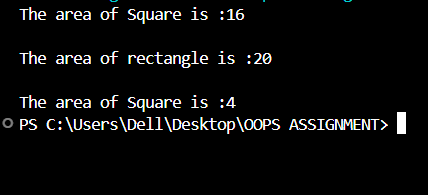
        c.calculateArea(2);

    }

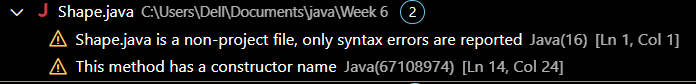
}

**OUTPUT :**

POSITIVE CASE:

****

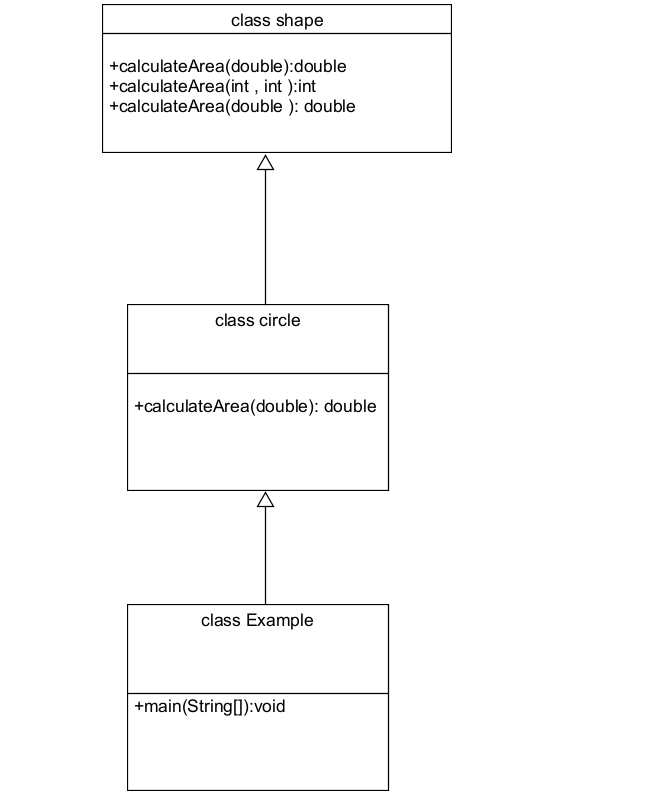
NEGATIVE CASE:



**ERRORS :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | ; is missed | ; is added |
| 2 | Logical error | Incorrect logic | Correct logic |

CLASS DIAGRAM:



**WEEK-7**

**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.

**PROGRAM:**

abstract class Animal{

    abstract void sound();

}

class Tiger extends Animal{

    @Override

    public void sound(){

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

    }

}

class Lion extends Animal{

    @Override

    public void sound(){

        System.out.println("The Lion Roars");

    }

}

class AK{

public static void main(String args[]){

    Tiger t1 = new Tiger();

    t1.sound();

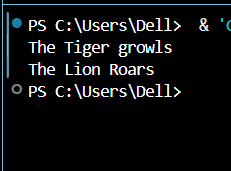
    Lion l1 = new Lion();

    l1.sound();

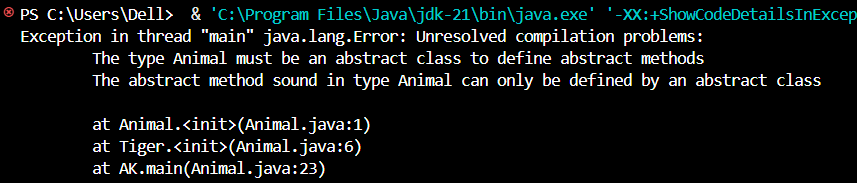
}

}

**OUTPUT:**



**NEGATIVE CASE:**

****

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | Abstract key word is missed before class | Abstract keyword is added |
| 2 | Logical error | Incorrect logic in subclass method | Corrected logic in subclass method |

**CLASS DIAGRAM:**

+main(args:[])

AK

+sound()

+sound()

Tiger

Lion

* sound()

<<abstarct>>

Animal

**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.

**PROGRAM :**

abstract class Shape3D {

    public abstract double calculateVolume();

    public abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

    private double radius;

    public Sphere(double radius) {

        this.radius = radius;

    }

    @Override

    public double calculateVolume() {

        return (4.0 / 3.0) \* Math.PI \* Math.pow(radius, 3);

    }

    @Override

    public double calculateSurfaceArea() {

        return 4 \* Math.PI \* Math.pow(radius, 2);

    }

}

class Cube extends Shape3D {

    private double side;

    public Cube(double side) {

        this.side = side;

    }

    @Override

    public double calculateVolume() {

        return Math.pow(side, 3);

    }

    @Override

    public double calculateSurfaceArea() {

        return 6 \* Math.pow(side, 2);

    }

}

public class Shape3DTest {

    public static void main(String[] args) {

        Shape3D sphere = new Sphere(5.0);

        Shape3D cube = new Cube(4.0);

        System.out.println("Sphere Volume: " + sphere.calculateVolume());

        System.out.println("Sphere Surface Area: " + sphere.calculateSurfaceArea());

        System.out.println("Cube Volume: " + cube.calculateVolume());

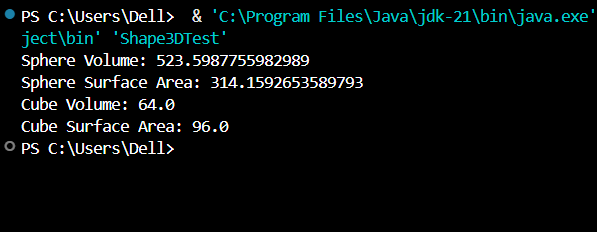
        System.out.println("Cube Surface Area: " + cube.calculateSurfaceArea());

    }

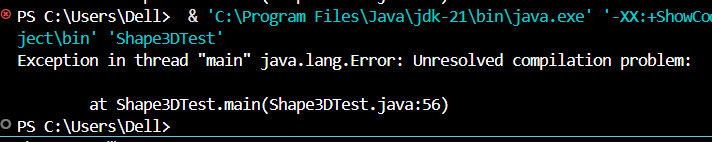
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **package is missed before abstract class** | **Package is imported** |
| **2** | **Logical error** | **Incorrect logic in subclass method** | **Corrected logic in subclass method** |

**CLASS DIAGRAM:**

<<abstract>>

Shape3D

+calculateVolume(): double

+calculateSurfaceArea(): double

+main(args: String[]): void

Shape3DTest

+calculateVolume(): double

+calculateSurfaceArea(): double

- side: double

- radius: double

Cube

Sphere

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | package is missed before abstract class | Package is imported |
| 2 | Logical error | Incorrect logic in subclass method | Corrected logic in subclass method |

**3)Aim :** Write a java program using an abstract class to define a method for pattern printing Create an abstract class named pattern printer with an abstract method printpattern(int n) and a concrete method to display the pattern title.

Implement two subclasses:

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

2) Number pattern - Prints a right- angled triangles of increasing numbers.

In the main() method, create Objects

Star Pattern Number pattern

\* 1

\*\* 1 2

\*\*\* 1 2 3

\*\*\*\* 1 2 3 4

\*\*\*\*\* 1 2 3 4 5

**CLASS DIAGRAM :**

|  |
| --- |
| <<abstract>>  PatternPrinter |
| +printpattern(int): void  +display(String): void |

|  |
| --- |
| starpattern |
| +printpattern(int) |

|  |
| --- |
| numberpattern |
| +printpattern(int) |

**PROGRAM :**

abstract class PatternPrinter {

    abstract void printPattern(int n);

    void displayTitle(String title) {

        System.out.println(title);

    }

}

class StarPattern extends PatternPrinter {

    void printPattern(int n) {

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

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

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

            }

            System.out.println();

        }

    }

}

class NumberPattern extends PatternPrinter {

    void printPattern(int n) {

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

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

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

            }

            System.out.println();

        }

    }

}

 class PatternProgram {

    public static void main(String[] args) {

        StarPattern sp = new StarPattern();

        NumberPattern np = new NumberPattern();

        System.out.println("Name:K.Divya Teja  Roll no:AV.SC.U4CSE24138  Section:CSE-B");

        System.out.println("    ");

        sp.displayTitle("Star Pattern");

        sp.printPattern(5);

        System.out.println("    ");

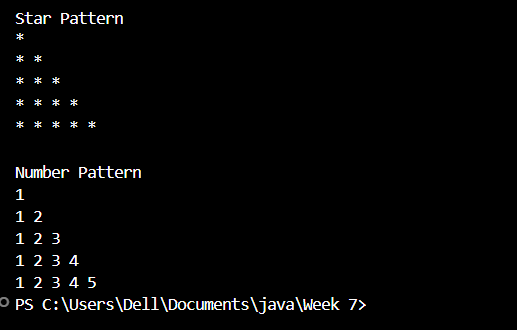
        np.displayTitle("Number Pattern");

        np.printPattern(5);

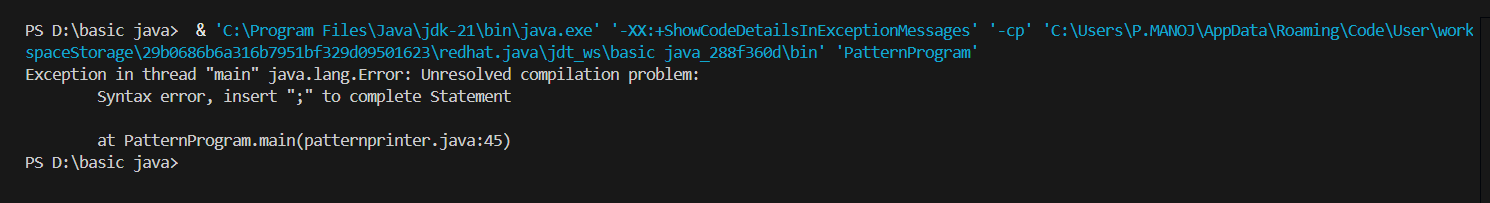
    }

}

**OUTPUT:**

****

**NEGATIVE CASE:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **For loop increment condition is missed in subclass method** | **Increment condition is added in subclass method** |
| **2** | **Logical error** | **Incorrect logic in subclass method** | **Corrected logic in subclass method** |

**WEEK – 8**

1. **Aim :**

**Write a Java program to create an interface Shape**

**with the getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.**

CLASS DIAGRAM :

|  |
| --- |
| <<interface>>  Shape |
| +getPerimeter() |

|  |
| --- |
| Rectangle |
| -l : int  - b: int |
| +getPerimeter() |

|  |
| --- |
| Triangle |
| - s1, s2, s3 |
| +getPerimeter() |

|  |
| --- |
| Circle |
| - r: int |
| +getPerimeter() |

**PROGRAM :**

**interface Shape{**

**double getPerimeter();**

**}**

**class Rectangle implements Shape{**

**private double length;**

**private double breadth;**

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

**this.length = length;**

**this.breadth = breadth;**

**}**

**public double getPerimeter(){**

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

**}**

**}**

**class Circle implements Shape{**

**private double radius;**

**Circle(double radius){**

**this.radius=radius;**

**}**

**public double getPerimeter(){**

**return 2\*Math.PI\*radius;**

**}**

**}**

**class Triangle implements Shape{**

**private double side1;**

**private double side2;**

**private double side3;**

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

**this.side1 = side1;**

**this.side2 = side2;**

**this.side3 = side3;**

**}**

**public double getPerimeter(){**

**return (side1+side2+side3);**

**}**

**}**

**public class Main{**

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

**Rectangle P1 = new Rectangle(9,8);**

**Circle    P2 = new Circle(8);**

**Triangle  P3 = new Triangle(5, 10, 15);**

**System.out.println(" Area of Rectangle is : " +P1.getPerimeter());**

**System.out.println(" Area of Circle is : " +P2.getPerimeter());**

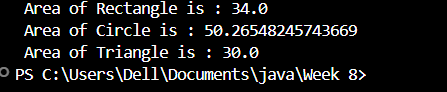
**System.out.println(" Area of Triangle is : " +P3.getPerimeter());**

**}**

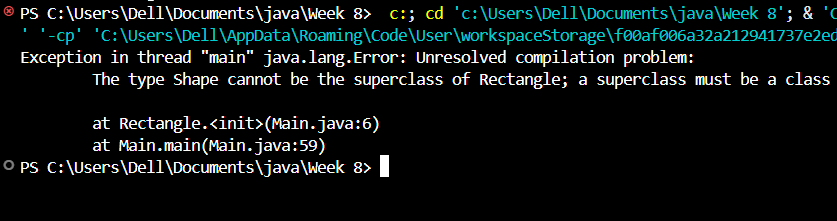
**}**

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERRORS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Used extends keyword instead of implements keyword in inheritance** | **Implements key word is added in inheritance** |

1. **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.

CLASS DIAGRAM :

|  |
| --- |
| <<interface>>  playable |
| +play(): void |

|  |
| --- |
| football |
| +play() |

|  |
| --- |
| Volleyball |
| +play() |

|  |
| --- |
| basketball |
| +play() |

**PROGRAM:**

**interface Playable{**

**public void play();**

**}**

**class Football implements Playable{**

**public void play(){**

**System.out.println("Play Football");**

**}**

**}**

**class Volleyball implements Playable{**

**public void play(){**

**System.out.println("Play Volleyball");**

**}**

**}**

**class Basketball implements Playable{**

**public void play(){**

**System.out.println("Play Basketball");**

**}**

**}**

**class Sportsplay{**

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

**Football f1 = new Football();**

**Volleyball v1 = new Volleyball();**

**Basketball b1 = new Basketball();**

**f1.play();**

**v1.play();**

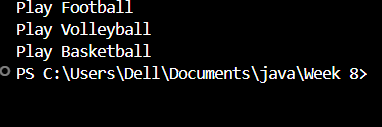
**b1.play();**

**}**

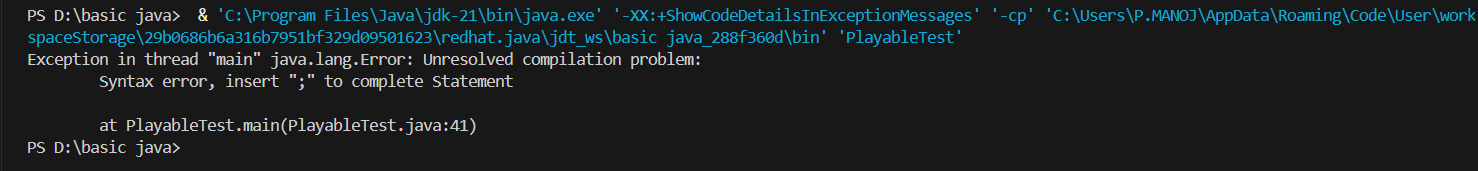
**}**

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Error in calling play method in football class because object is not created for football class** | **Object created for football class** |

**3)Aim:** write a java program to implement a login system using interfaces.

**OUTPUT:**

**interface LoginSystem {**

**boolean Login(String ID, int pass);**

**}**

**class CollegePortal implements LoginSystem {**

**public boolean Login(String ID, int pass) {**

**if ((ID=="P.KOMAL YASHWANTH") && (pass==24243)){**

**System.out.println("Login Successful..!");**

**return true;**

**}else {**

**System.out.println("Invalid ID or Password");**

**return false;**

**}**

**}**

**}**

**class LoginPortal {**

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

**CollegePortal CP = new CollegePortal();**

**System.out.println("    ");**

**CP.Login("P.KOMAL YASHWANTH" , 24243);**

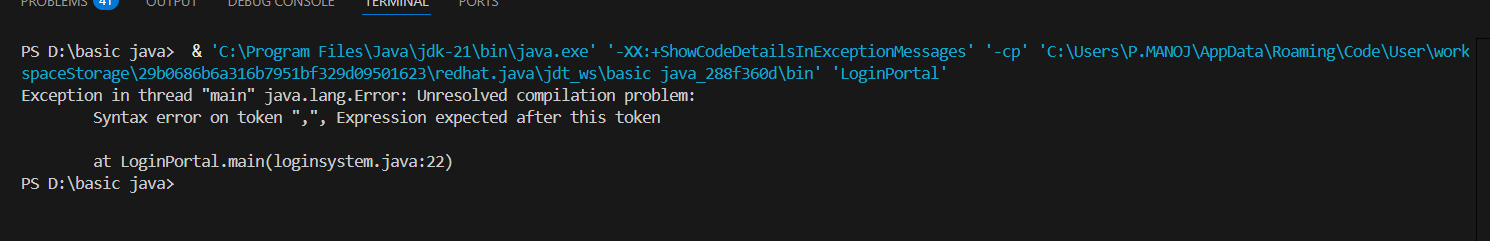
**}**

**}**

**OUTPUT:**

****

**NEGATIVE CASE:**

****

**ERROR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Syntax error | Error in If statement condition | If statement condition is corrected |