**23CSE111**

**OBJECT-ORIENTED PROGRAMMING**

**LAB REPORT**



**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: T. Kushal**

**Verified By: Roll No: 24333**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Experiment Name** | **Page No.** | **Remarks** | **Signature** |
|  | **WEEK-1** |  |  |  |
| 1 | Installation Process of JDK |  |  |  |
| 2 | Simple Java Program for printing basic details of student |  |  |  |
|  | **WEEK-2** |  |  |  |
| 1 | Write a Java Program to find the factorial of a number |  |  |  |
| 2 | Write a Java Program to find the Fibonacci Series of given length |  |  |  |
| 3 | Write a Java Program to find the temperature from Celsius to Fahrenheit |  |  |  |
| 4 | Write a Java Program to find the Simple Interest |  |  |  |
| 5 | Write a Java Program to find the area of triangle using heron’s formula |  |  |  |
| 6 | Write a Java Program to find the area of rectangle |  |  |  |
|  | **WEEK-3** |  |  |  |
| 1 | Write a java program with the following instructions:  a. Create class with name car.  b. Create 4 attributes named car color, car brand, fuel type, mileage.  c. Create 3 methods named start(), stop(), service().  d. Create 3 objects C1, C2, C3.  e. Create a constructor with parameters with car color, car brand, fuel type, mileage. |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Create a class named bank account with methods deposit and withdraw. Where the deposit method should accepts a parameter and when this method is called the deposited amount should be added to current balance. In addition to that when a withdraw method is called it has to verify whether the withdraw amount is less than the current balance. If not display message saying that there are insufficient funds. Use the constructor to display the details of the customer (Name, Account number, IFSC code, Branch). Also create two customer objects C1, C2. |  |  |  |
|  | **WEEK-4** |  |  |  |
| 1 | Write a Java program with a class named book. The class should contain various attributes such as title, author, year of publication, and price. It should also contain a constructor with parameters that include title, author, year of publication, and price. Create a method that displays details of the book(Display the details of 3 books that is create 3 objects and display their details) |  |  |  |
| 2 | Create a java program with a Class named "my class" with a static variable 'count' of int type static and initialized to zero and a constant variable 'pi' of type double initialized to 3.1415 as attributes of that class now define a constructor of my class that increments the count variable each time an object of my class is created and finally prints the final values of count and pi variables. |  |  |  |
|  | **WEEK-5** |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output. |  |  |  |
| 2 | Vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes and they need a program to store details about each vehicle such as brand and speed.  i. Cars should have an additional property: number of doors, Seating capacity.  ii. Bikes should have a property indicating whether they have gears or not.  iii. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.  iv. Each class should have a constructor.  2. If the company decides to add a new type of vehicle ‘Truck’, how would you modify the program?  a. Truck should include and additional property capacity (in tons).  b. Create a showTruck() method to display the truck’s capacity.  c. Write a constructor for truck that initializes all properties.  3. 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 the details. |  |  |  |
|  | **WEEK-6** |  |  |  |
| 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, model, fuel type, and colour using the constructor |  |  |  |
| 2 | Create a Java program for the scenario.  A college is developing an automated admission system that verifies student eligibility for undergraduate (UG) and postgraduate(PG) programs. Each program has different eligibility criteria based on the student's percentage in their previous qualification.  i) UG admissions require a minimum of 60%  ii) PG admissions require a minimum of 70% |  |  |  |
| 3 | Write a Java Program to create a Calculator class with overloaded methods to perform addition: Take the integer values a and b from the user.  i) Add two integers  ii) Add two doubles  iii) Add three integer |  |  |  |
| 4 | Write a Java Program to 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. |  |  |  |

**WEEK-1**

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

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

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

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



**Checking of JDK Version:**

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



**2. Simple Java Program for printing the Name, Class, and Roll No, of a student**

**CODE:**

class Main

{

public static void main (String [] args)

{

System.out.println("Name: T.Kushal");

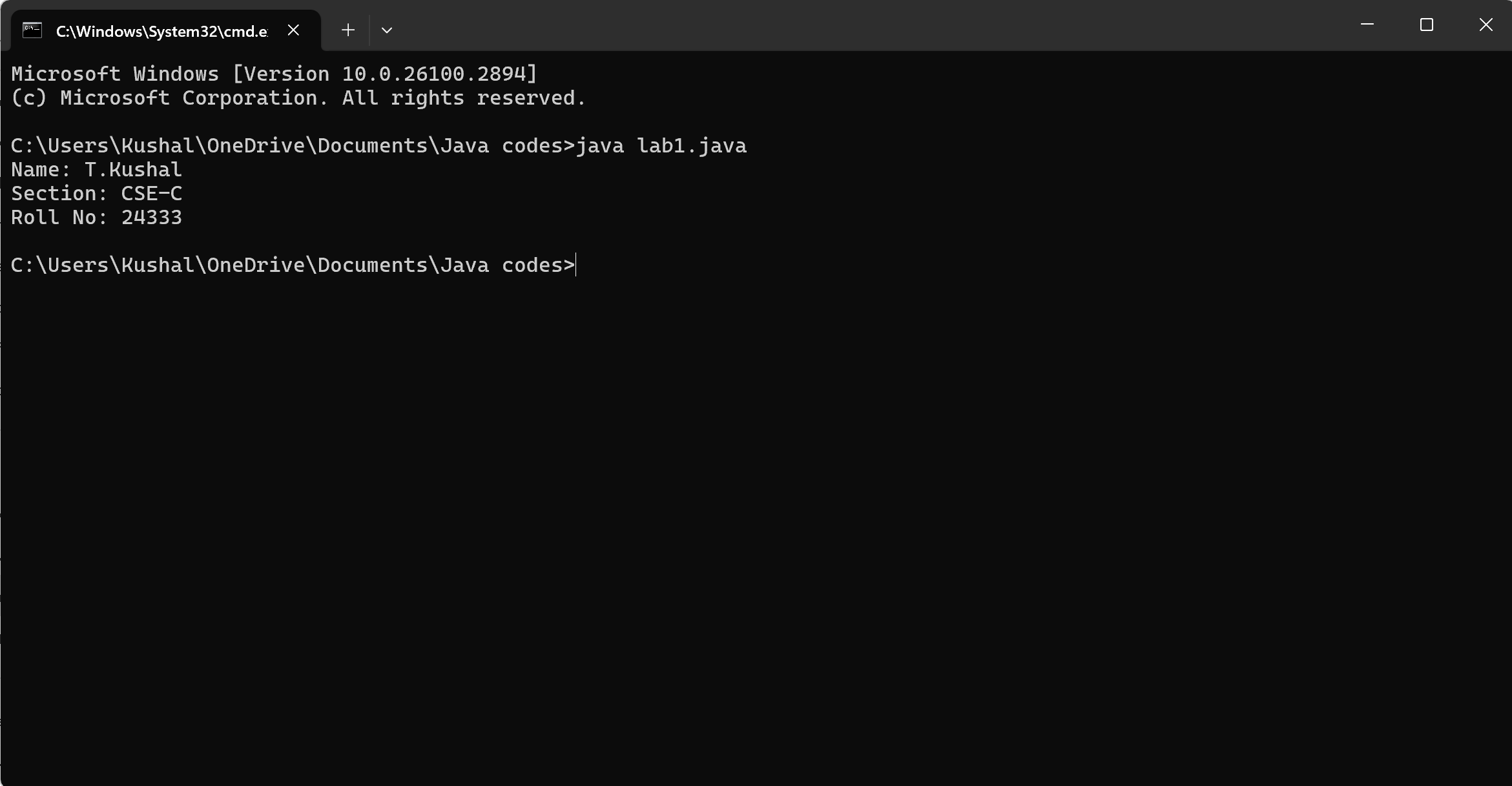
System.out.println("Section: CSE-C");

System.out.println("Roll No: 24333");

}

}

**Output:**



**WEEK-2**

1**. Write a Java program to find S.I.**

**Code:**

public class SimpleInterestCalculator {

     public static void main(String[] args) {

        try (java.util.Scanner scanner = new java.util.Scanner(System.in)) {

            System.out.print("Enter the principal amount (in rupees): ");

            double principal = scanner.nextDouble();

            System.out.print("Enter the annual interest rate (in %): ");

            double rate = scanner.nextDouble();

            System.out.print("Enter the time period (in years): ");

            int time = scanner.nextInt();

            double simpleInterest = (principal \* rate \* time) / 100;

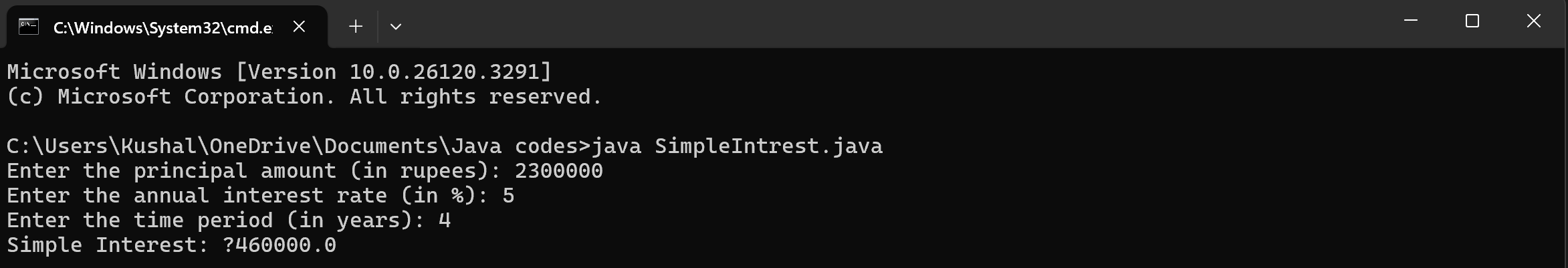
            System.out.println("Simple Interest: ₹" + simpleInterest);

        }

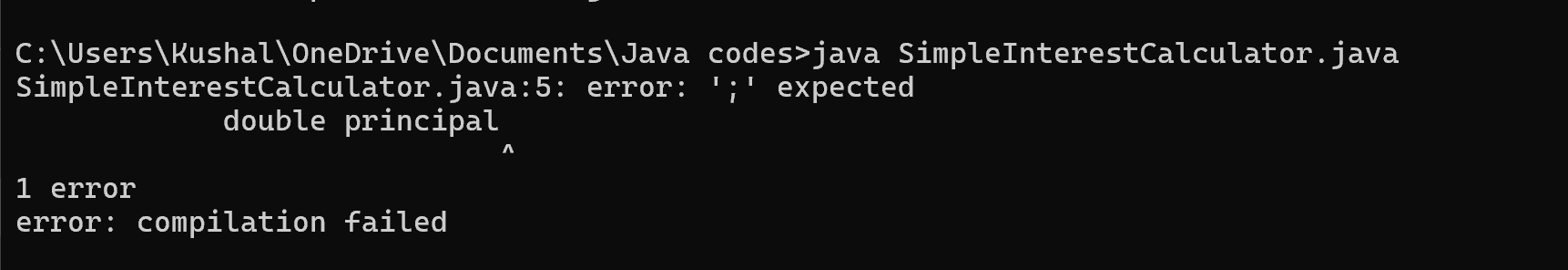
    }

}

**Positive Output:**

****

**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| **1**. | error: can't find primary (String []) method in class: SimpleInterestCalculator | Should close the string brackets [] |

**Important Points:**

* Used Scanner library to get input from user in run time.

Import java. Util. Scanner;”-step to import library.

* Scanner input= new Scanner (System.in);”-step to use the scanner. [case sensitive]
* Should give the ‘;’ symbol at the end of System.out.print("Enter the rate of interest (R) in percentage: ").

**2**. **Write a program on Java on the Factorial of a number with inputs taken.**

**Code:**

Import java.util.Scanner;

public class Factorialofanumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System. out.print("Enter the number: ");

int number = scanner.next();

long factorial = 1;

if (number < 0) {

System. out.println("No Factorial For Negative Numbers.");

} else {

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

factorial \*=i;

}

System.out.println("The factorial of " + number + " is " + factorial);

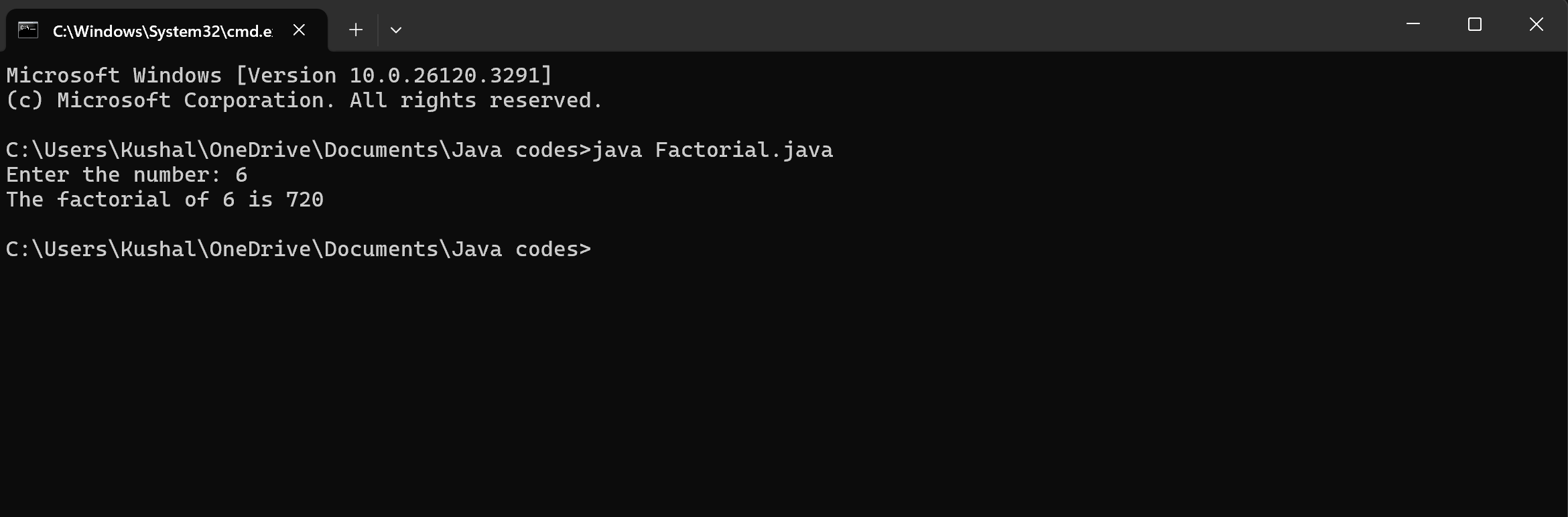
}

scanner.close();

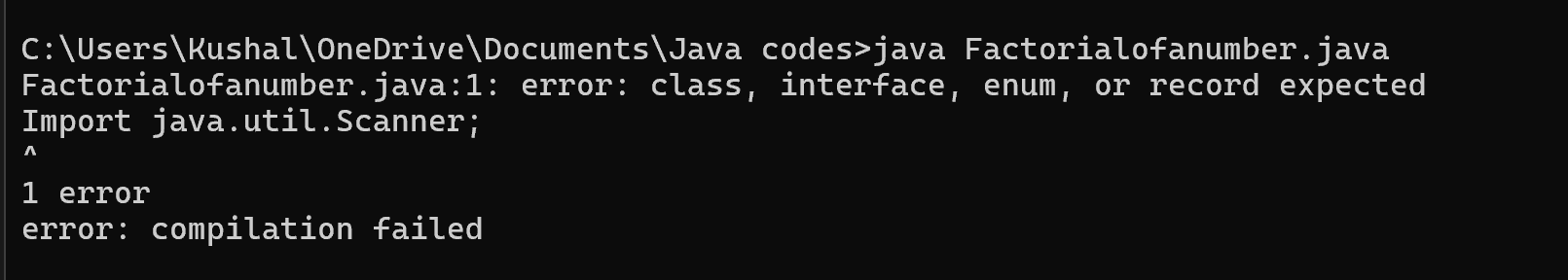
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | 1.While using for iteration, not giving the conditions correctly. | 1.We should give iterative statements correctly. |
| 2. | 2.Declaring the data type as double instead of int. | 2.We should give the data type as int for integers. |

**Important Points:**

* In the loop, the expression factorial\*i; is incorrect because the result of the multiplication is not assigned back to factorial. It should be factorial\*i; for proper multiplication and assignment.
* **Using a colon** after it will cause a syntax error.
* **Curly braces** {} are used to group the code inside the if block.

**3.** Write a Java program to convert the Temperature from Celsius to Fahrenheit & Fahrenheit to Celsius.

**CODE FOR CELCIUS TO FAHRENHEIT:**

Import java.util.Scanner;

public class CelsiustoFahrenheit {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System. out.print("Enter temperature in Celsius: ");

float celsius = scanner.nextFloat();

float Fahrenheit = (celsius \* 9 / 5) + 32;

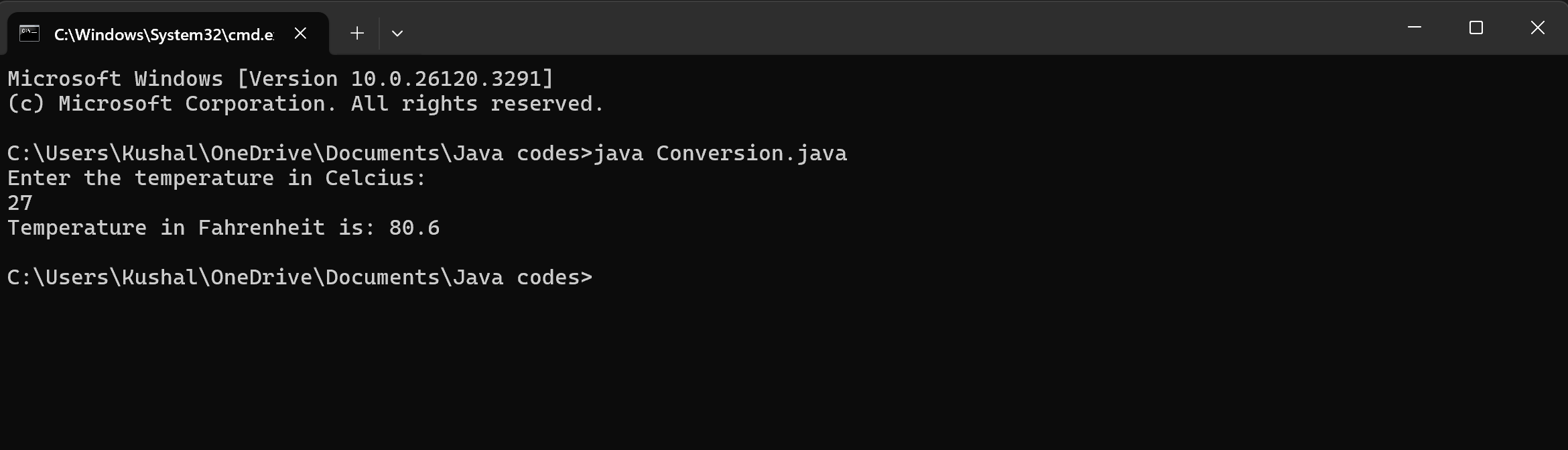
System.out.println(celsius + "°C is equal to " + Fahrenheit + "°F");

scanner.close();

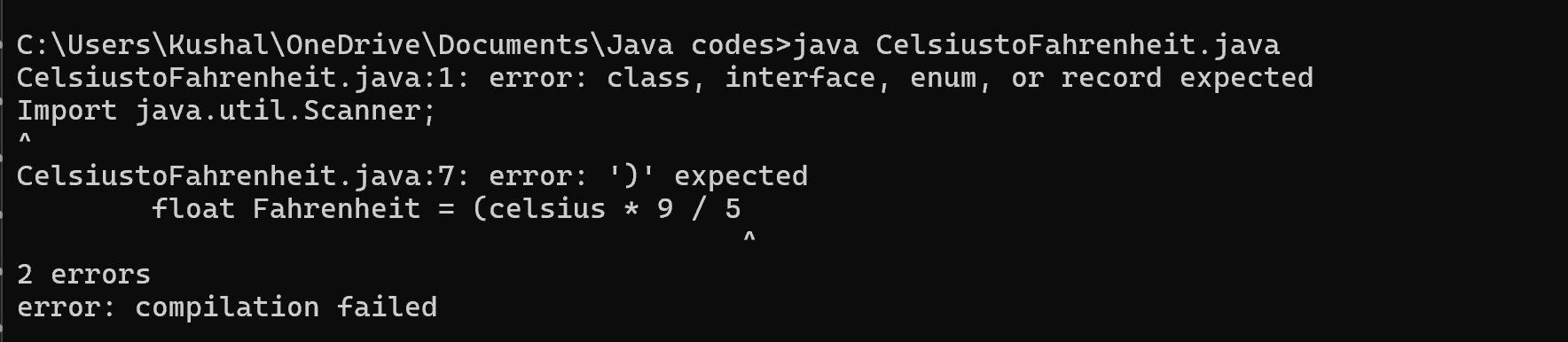
}

}

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | Duplicate close of scanner- scanner.close(); scanner.close() Reason: Use only one scanner.close(); | Duplicate close of scanner- scanner.close(); scanner.close() Reason: Use only one scanner.close(); |

**Important Points:**

* In the line float Fahrenheit=Celsius \* 9 / 5 + 32; the formula does not have proper parentheses for the multiplication and division. It should be float Fahrenheit = (Celsius \* 9 / 5) + 32; for the correct order of operations.
* In the System.out.println() statement, there is a missing semicolon at the end of the line.

**CODE FOR FAHRENHEIT TO CELCIUS:**

Import java.util.Scanner;

public class FahrenheittoCelsius {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System. out.print("Enter Fahrenheit temp: ");

float f = scanner.nextFloat();

float celsius = (f - 32) \* 5 / 9;

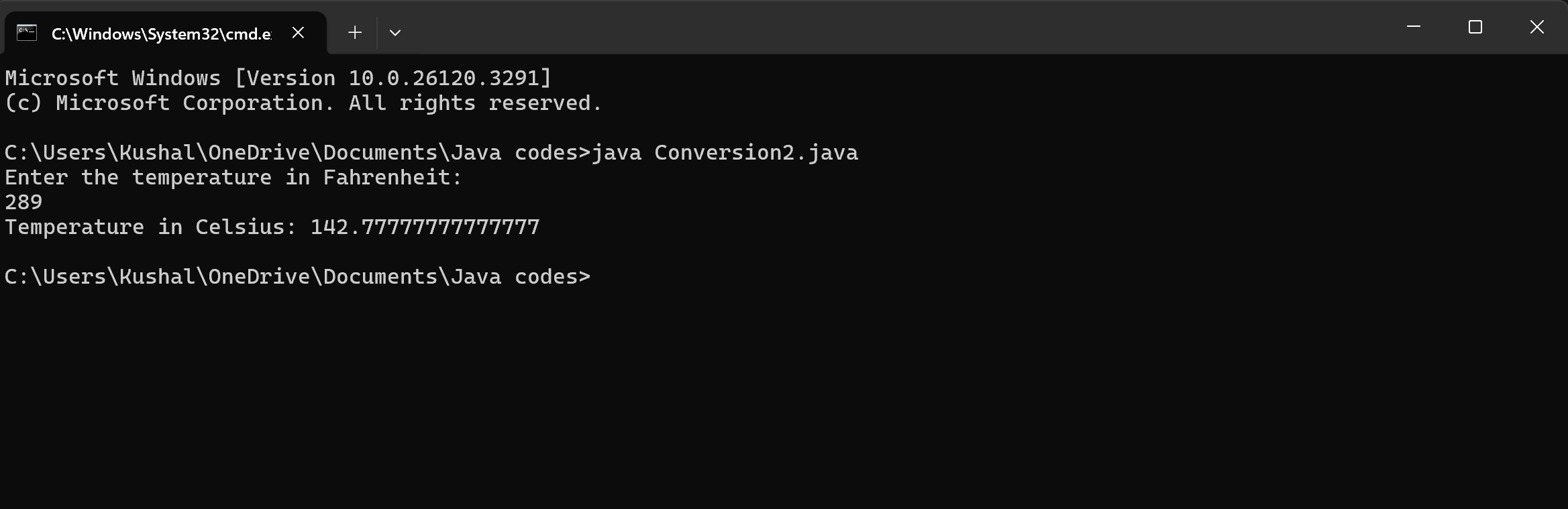
System.out.println(f + "°F is equal to " + celsius + "°C");

scanner.close();

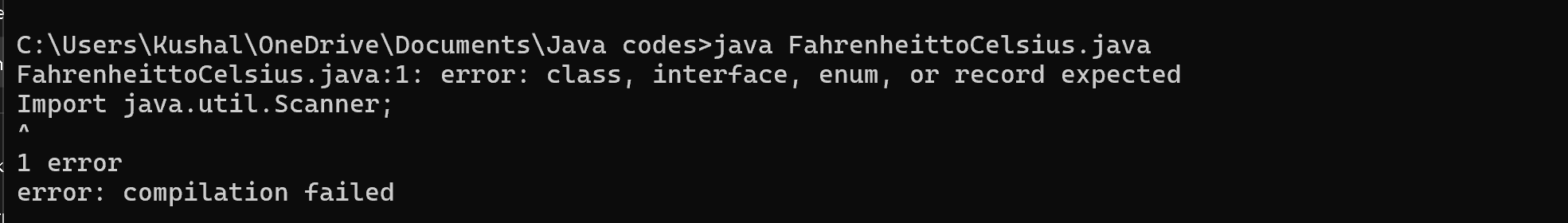
}

}

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | 1.While printing the variable not giving + sign. | 1.We should give correct indentation. |
| 2. | 2.Not closing the scanner. | 2.Closing the scanner is must. |

**Important points:**

* The System.out.println statement is using Celsius with a capital "C" which doesn't exist as a variable. Java is case-sensitive, so this will cause a compilation error. It should be lowercase Celsius.
* The scanner.close() method is called twice, which is unnecessary and could cause issues. It should only be called once.

**4. Write a Program on the Fibonacci sequence where input is taken from the user.**

**CODE:**

public class FibonacciSeries {

public static void main(String[] args) {

int n = 10;

int firstTerm = 0, secondTerm = 1;

System.out.println("Fibonacci Series up to " + n + " terms:");

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

System.out.print(firstTerm + ", ");

int nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

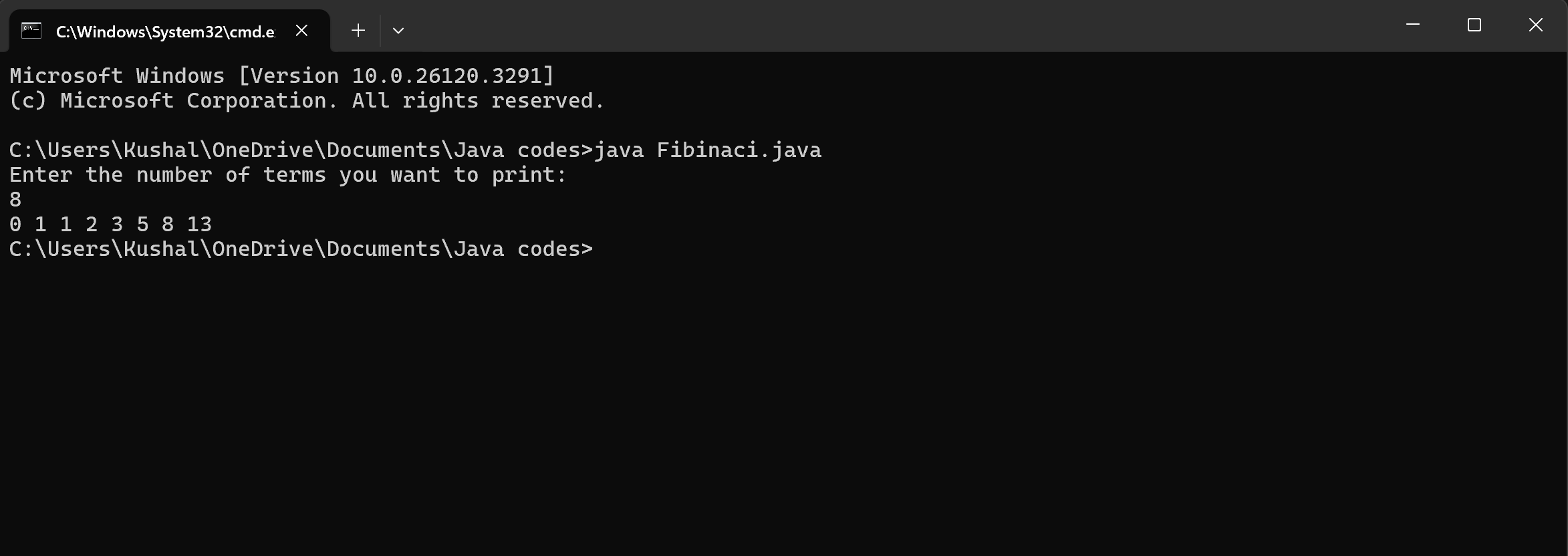
secondTerm = nextTerm;

}

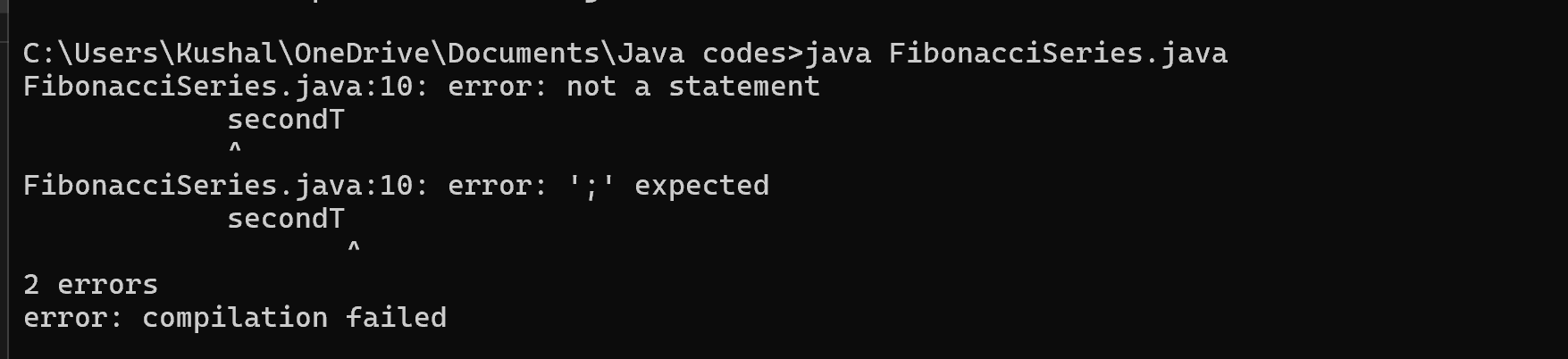
}

}

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | int[] fibonacci = new int[n]; System.out.println(fibonacci[n + 1]); invalid access replace: System.out.println("Fibonacci Series up to " + n + " terms:"); | int[] fibonacci = new int[n]; System.out.println(fibonacci[n + 1]); invalid access replace: System.out.println("Fibonacci Series up to " + n + " terms:"); |
| 2. | error: can't find primary (String []) method in class: | Should close the string brackets [] |

**IMPORTANT POINTS:**

* Variables should be initialized properly
* Accessing an array element outside its bounds.
* Loop condition should be properly defined. No syntax error should be there.

**5. Write a program with Java on How to calculate the Area of the Rectangle & Area of the Triangle.**

**CODE FOR AREA OF RECTANGLE:**

Import java.util.Scanner;

public class Rectangle area {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System. out.print("Enter the length of the rectangle: ");

double length = scanner.nextDouble();

System. out.print("Enter the width of the rectangle: ");

double width = scanner.nextDouble();

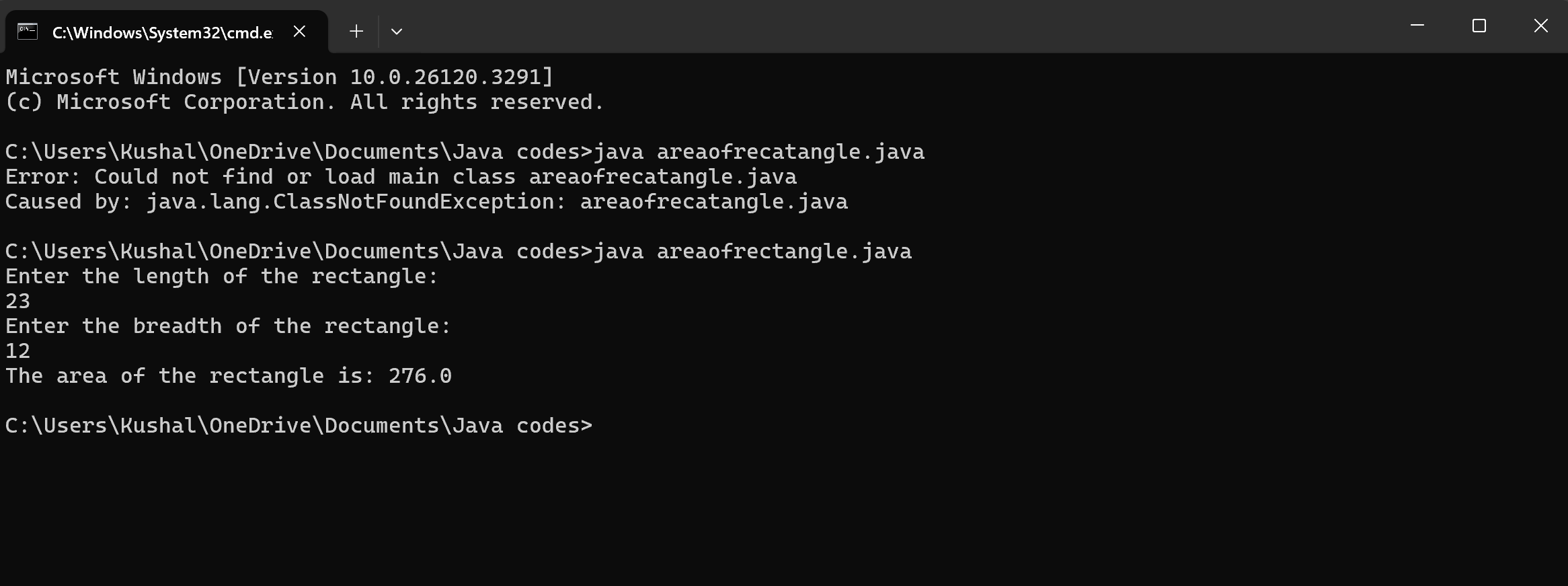
double area = length \* width;

System. out.println("The area of the rectangle is: " + area);

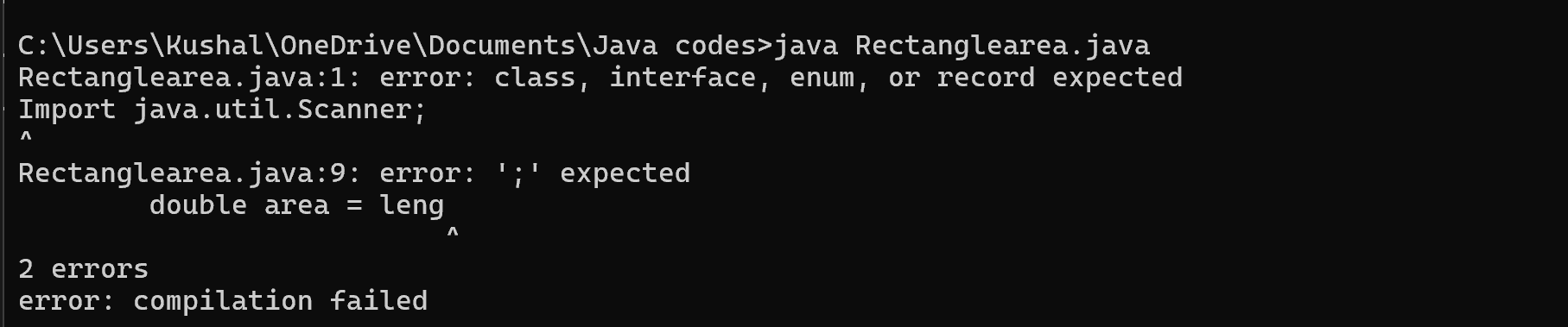
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | 1.While printing the variable not giving + sign. | 1.We should give correct indentation. |
| 2. | 2.Not closing the scanner. | 2.Closing the scanner is must. |

**Important points:**

* Trying to assign a double value to an int variable.
* Incorrect placement of braces, should be placed correctly.
* Should not give incorrect variables.

**CODE FOR AREA OF TRAINGLE:**

Import java.util.Scanner;

public class heronstriangle {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System. out.print("Enter the length of side a: ");

double a = scanner.nextDouble();

System. out.print("Enter the length of side b: ");

double b = scanner.nextDouble();

System. out.print("Enter the length of side c: ");

double c = scanner.nextDouble()

doubles = (a + b + c) / 2;

double area = Math.sqrt(s \* (s - a) \* (s - b) \* (s - c));

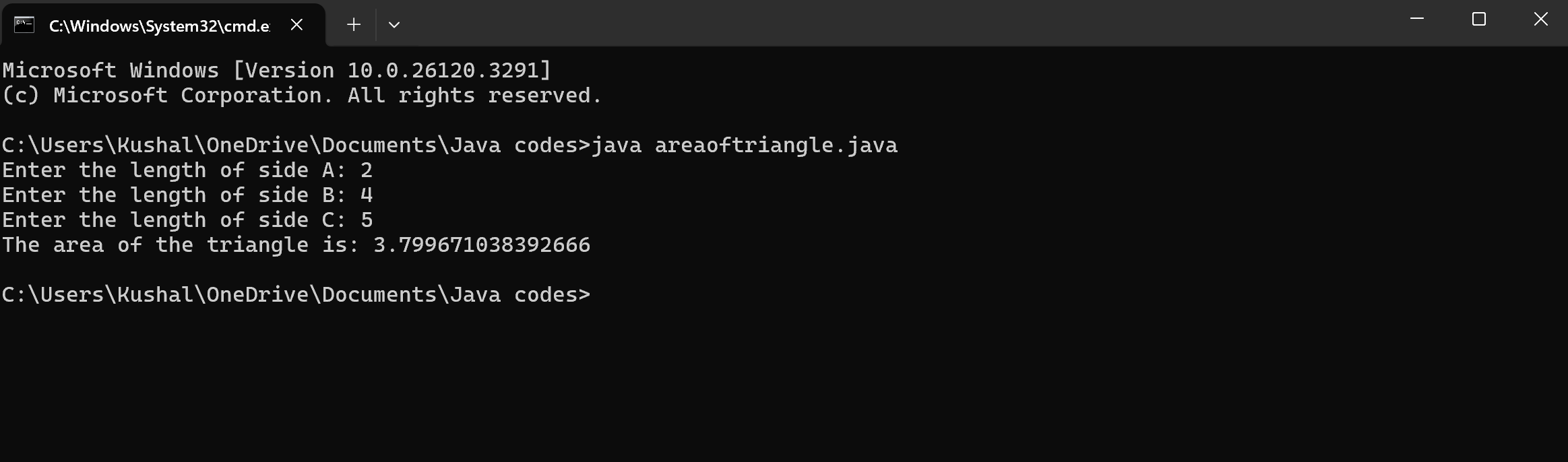
System. out.println("The area of the triangle is: " + area);

scanner.close();

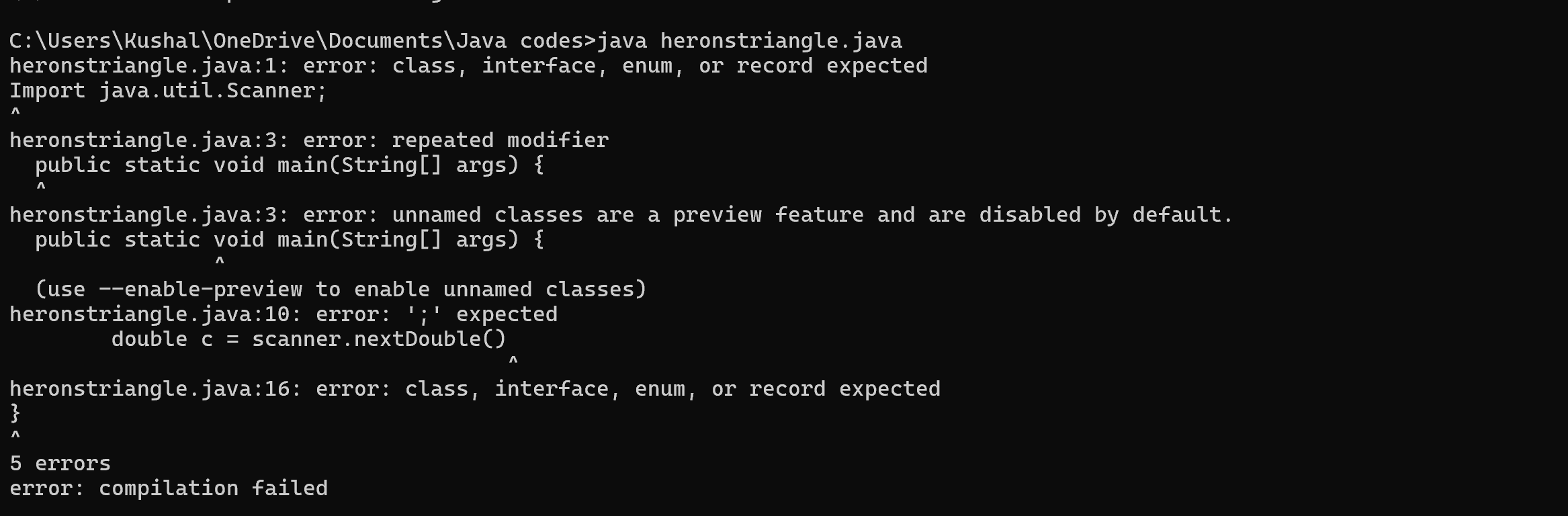
}

}

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | error: can't find primary (String []) method in class: | Should close the string brackets [] |

**Important points:**

* Verify that the input values can form a valid triangle (i.e., the sum of any two sides must be greater than the third side).
* Calculate the semi-perimeter ss using the formula: s=a+b+c/2.

**WEEK-3**

1. To create a Java program with the following:

a) Create a class with the name car

b) Create four attributes named Car\_colour, Car\_brand, Fuel\_type, mileage

c) Create three methods named Strat (), Stop (), Service ()

d) Create three objects named Car1, Car2, Car3.

**CLASS DIAGRAM-**

**+-------------------------------------+**

**| car |**

**+-------------------------------------+**

**| +car\_color : String |**

**| +car\_brand : String |**

**| +fuel\_type : String |**

**| +mileage : float |**

**+-------------------------------------+**

**| +start() : void |**

**| +stop() : void |**

**| +service() : void |**

**| +main(String[] args) : void |**

**+-------------------------------------+**

**CODE:**

class car

{

    public String car\_color;

    public String car\_brand;

    public String fuel\_type;

    public float mileage;

    public void start()

    {

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

    }

    public void stop()

    {

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

    }

    public void service()

    {

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

    }

    public static void main(String [] args){

            car car1= new car();

            car1.car\_color="Red";

            car1.car\_brand="BMW";

            car1.fuel\_type="Petrol";

            car1.mileage=56.7F;

            car1.start();

            car1.stop();

            car1.service();

            System.out.println("color of the car1 is "+car1.car\_color);

            System.out.println("brand of the car1 is "+car1.car\_brand);

            System.out.println("fuel type of the car1 is "+car1.fuel\_type);

            System.out.println("mileage of the car1 is"+car1.mileage);

            car car2= new car();

            car2.car\_color="Blue";

            car2.car\_brand="Mahindra";

            car2.fuel\_type="Hybrid";

            car2.mileage=60.5F;

            car2.start();

            car2.stop();

            car2.service();

            System.out.println("color of the car2 is "+car2.car\_color);

            System.out.println("brand of the car2 is "+car2.car\_brand);

            System.out.println("fuel type of the car2 is "+car2.fuel\_type);

            System.out.println("mileage of the car2 is"+car2.mileage);

            car car3= new car();

            car3.car\_color="Yellow";

            car3.car\_brand=" Mercedes";

            car3.fuel\_type="Diesel";

            car3.mileage=66.5F;

            car3.start();

            car3.stop();

            car3.service();

            System.out.println("color of the car3 is "+car3.car\_color);

            System.out.println("brand of the car3 is "+car3.car\_brand);

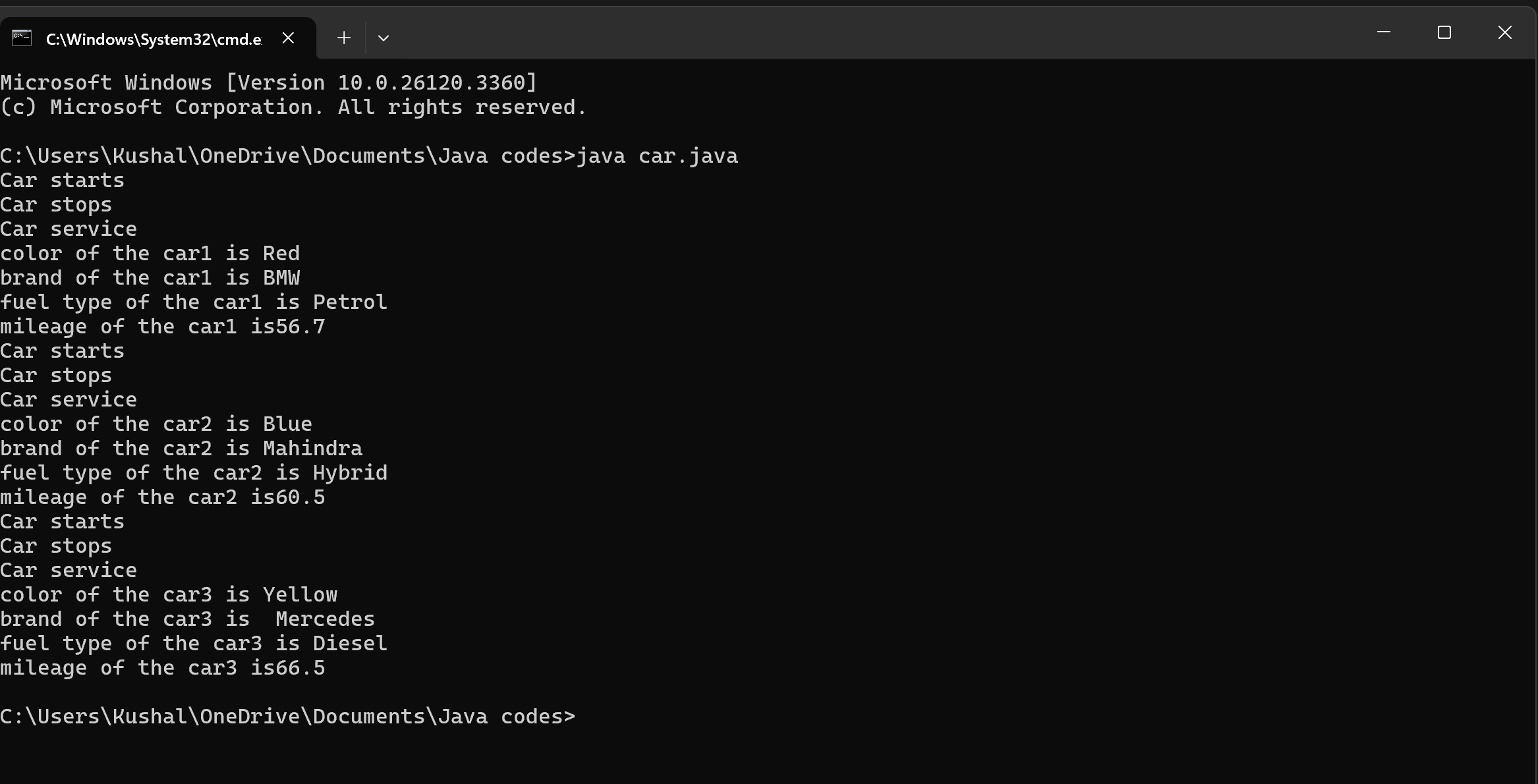
            System.out.println("fuel type of the car3 is "+car3.fuel\_type);

            System.out.println("mileage of the car3 is"+car3.mileage);

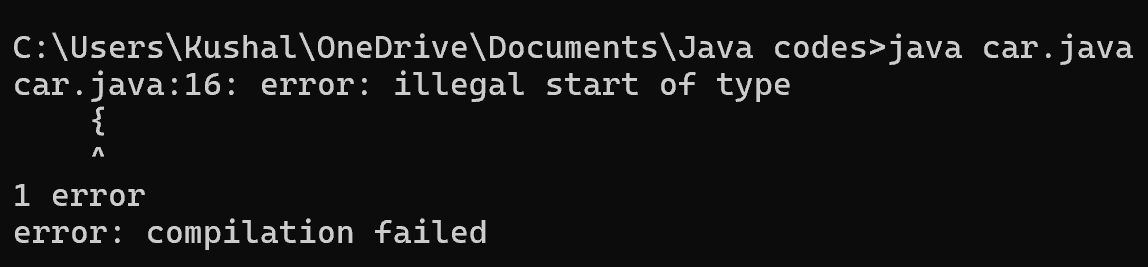
    }

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | error: can't find primary (String []) method in class: | Should close the string brackets [] |

**Important points:**

* Variable name mismatch: The variable car\_Color in the code should be car\_color
* Incorrect variable name: car1.car\_color is used when the actual variable is car1.car\_Color, which will cause an error due to case sensitivity.
* Missing Semicolon: Forgetting to add a semicolon at the end of a statement will cause a compilation error.

**1.write a java program to create a baseclass bankaccount with methods deposit &withdrawal .create two subclasses savingsaccount &checking acccount which overrides the withdrawal method in each subclass to impose different withdrawal limits &fees.**

**CLASS DIAGRAM-**

**+----------------------------------------------------------+**

**| BankAccount |**

**+----------------------------------------------------------+**

**| -accountHolder : String |**

**| -balance : double |**

**+----------------------------------------------------------+**

**| +BankAccount(accountHolder, balance) |**

**| +deposit(amount: double) : void |**

**| +withdraw(amount: double) : void |**

**| +displayBalance() : void |**

**+----------------------------------------------------------+**

**▲**

**-------------------------**

**| |**

**+------------------------+ +---------------------------+**

**| SavingsAccount | | CheckingAccount |**

**+--------------------------------------+ +---------------------------------------+**

**| -freeWithdrawals : int | -withdrawalFee : double |**

**| -withdrawalCount : int | |**

**| -withdrawalFee : double | |**

**+---------------------------------------+ +--------------------------------------+**

**| +withdraw(amount: double) : void (overridden) |**

**+---------------------------------------+ +---------------------------------------+**

**+----------------------------------------+**

**| BankDemo |**

**+----------------------------------------+**

**| +main(String[] args) : void |**

**+----------------------------------------+**

**CODE:**

class BankAccount {

protected double balance;

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: $" + amount + " | New Balance: $" + balance);

} else {

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

}

}

public void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: $" + amount + " | New Balance: $" + balance);

} else {

System.out.println("Invalid withdrawal amount.");

}

}

public double getBalance() {

return balance;

}

public static void main(String[] args) {

System.out.println("=== Savings Account ===");

SavingsAccount savings = new SavingsAccount(1000);

savings.deposit(200);

savings.withdraw(600);

savings.withdraw(300);

System.out.println("\n=== Checking Account ===");

CheckingAccount checking = new CheckingAccount(1000);

checking.deposit(100);

checking.withdraw(50);

checking.withdraw(1050);

}

}

class SavingsAccount extends BankAccount {

private static final double WITHDRAWAL\_LIMIT = 500.0;

public SavingsAccount(double initialBalance) {

super(initialBalance);

}

@Override

public void withdraw(double amount) {

if (amount > WITHDRAWAL\_LIMIT) {

System.out.println("Withdrawal denied! Savings account limit is $" + WITHDRAWAL\_LIMIT);

} else if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrawn from Savings: $" + amount + " | New Balance: $" + balance);

} else {

System.out.println("Invalid withdrawal amount.");

}

}

}

class CheckingAccount extends BankAccount {

private static final double TRANSACTION\_FEE = 2.0;

public CheckingAccount(double initialBalance) {

super(initialBalance);

}

@Override

public void withdraw(double amount) {

double totalAmount = amount + TRANSACTION\_FEE;

if (totalAmount > balance) {

System.out.println("Withdrawal denied! Not enough balance after fee.");

} else if (amount > 0) {

balance -= totalAmount;

System.out.println("Withdrawn from Checking: $" + amount + " | Fee: $" + TRANSACTION\_FEE + " | New Balance: $" + balance);

} else {

System.out.println("Invalid withdrawal amount.");

}

}

}

public class BankApplication {

public static void main(String[] args) {

System.out.println("=== Savings Account ===");

SavingsAccount savings = new SavingsAccount(1000);

savings.deposit(200);

savings.withdraw(600);

savings.withdraw(300);

System.out.println("\n=== Checking Account ===");

CheckingAccount checking = new CheckingAccount(1000);

checking.deposit(100);

checking.withdraw(50);

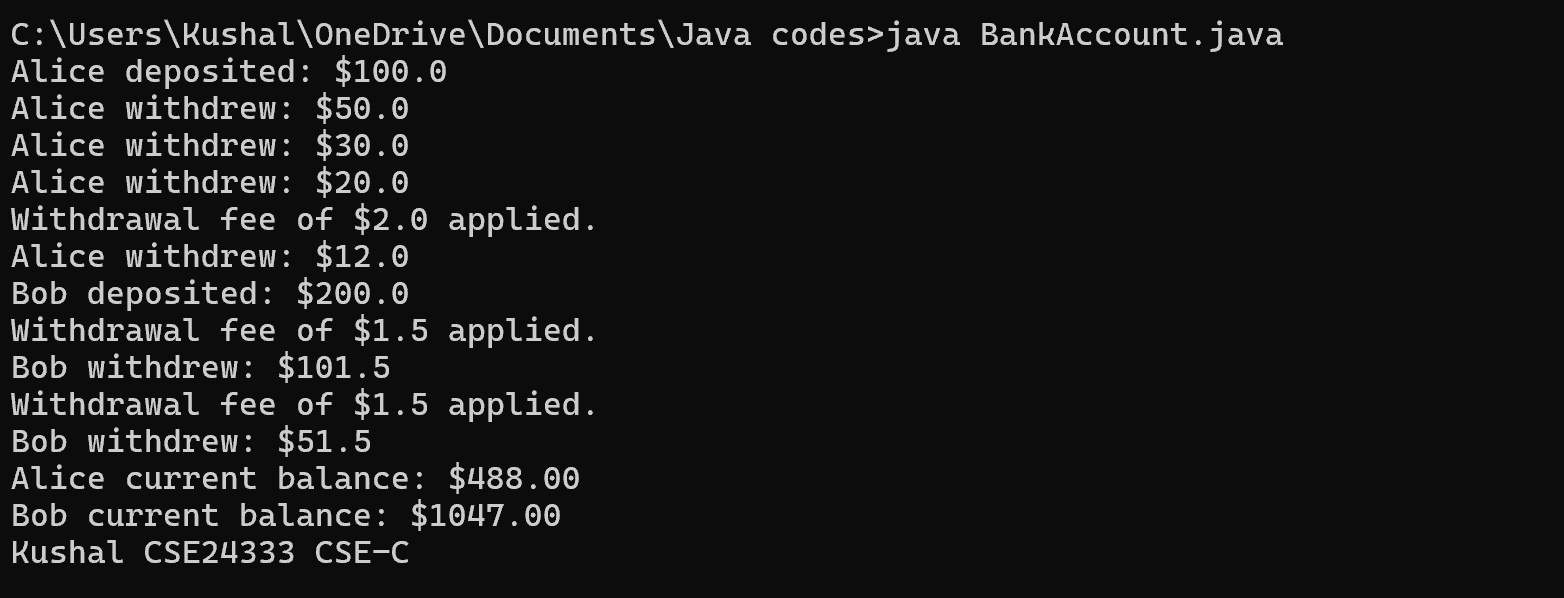
checking.withdraw(1050);

System.out.println("Kushal CSE24333 CSE-C");

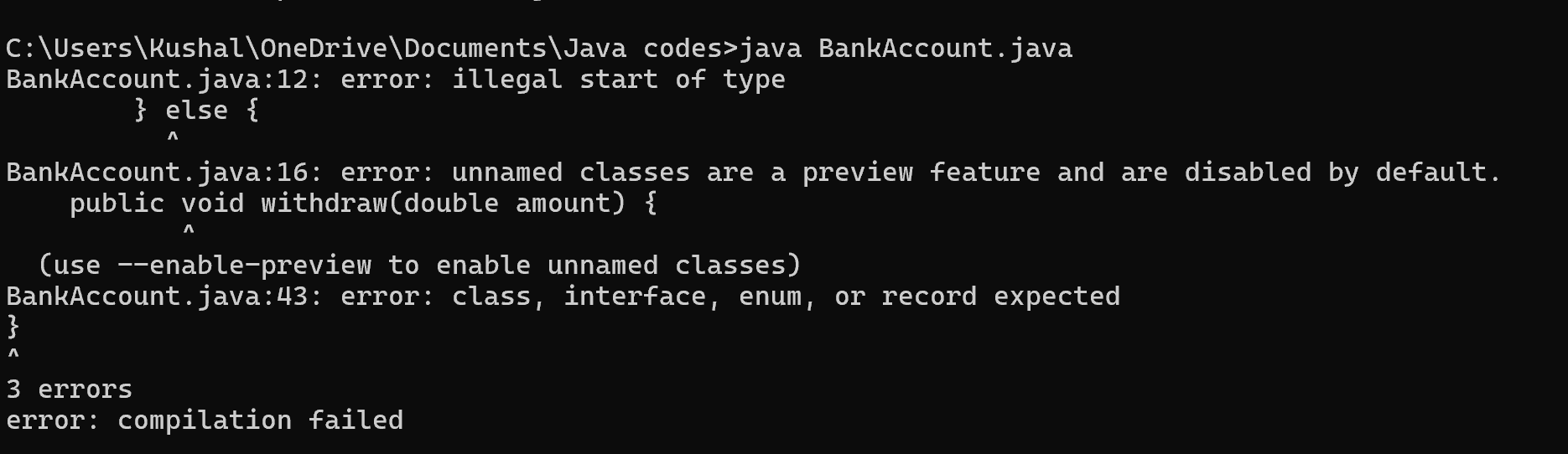
}

}

**Positive Output:**

****

**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | Existing>withdrawal | Existing=withdrawal |
| 2. | error: can't find primary (String []) method in class : | Should close the string brackets [] |

**Important points:**

1. Class Naming Conventions: Always follow Java naming conventions for readability and maintainability.
2. Resource Management: Always close resources like Scanner to avoid memory leaks. Using try-with-resources is a good practice.

**WEEK-4**

**1. Write a Java program with a class named Book**

**a) a class should contain various attributes such as title, author, and year of publication.**

**b) it should also contain a constructor with parameters which initialize the title, author, and year of publication.**

**c)create a method which displays the details of the book title, author, year of publication**

**Display the details of two books.**

CLASS DIAGRAM-

**+--------------------------------------------+**

**| Book |**

**+--------------------------------------------+**

**| +bookTitle : String |**

**| +bookAuthor : String |**

**| +bookYearOfPublication : int |**

**+--------------------------------------------+**

**| +title() : void |**

**| +author() : void |**

**| +main(String[] args) : void |**

**+--------------------------------------------+**

**CODE:**

class Book {

public String book title;

public String book author;

public int bookYearOfPublication;

public void title() {

System.out.println("Book Title");

}

public void author() {

System.out.println("Book Year of Publishing")

}

public static void main(String[] args) {

Book book1 = new Book();

book1.bookTitle = "Harry potter and the philosopher stone";

book1.bookAuthor = "J.K.Rowling";

book1.bookYearOfPublication = 1997;

book1.title();

book1.author();

System.out.println("Book title is: " + book1.bookTitle);

System.out.println("Book author is: " + book1.bookAuthor);

System.out.println("Book year of publication is: " + book1.bookYearOfPublication);

Book book2 = new Book();

book2.bookTitle = "Sherlock homes";

book2.bookAuthor = "Sir Arthur Conan Doyle";

book2.bookYearOfPublication = 1892;

book2.title();

book2.author();

System.out.println("Book title is: " + book2.bookTitle);

System.out.println("Book author is: " + book2.bookAuthor);

System.out.println("Book year of publication is: " + book2.bookYearOfPublication);

System.out.println("Kushal CSE24333 CSE-C");

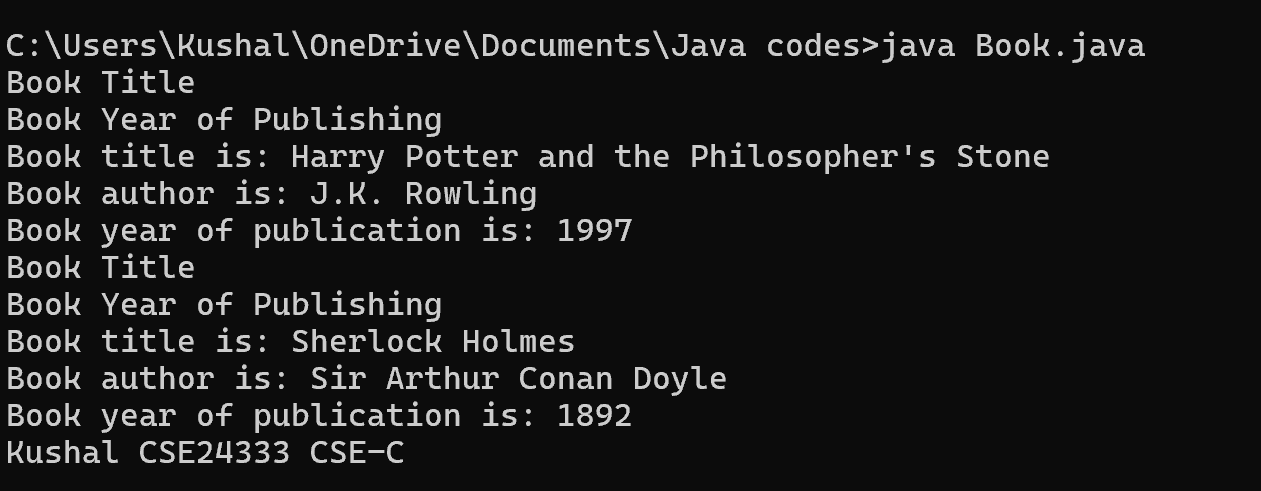
}

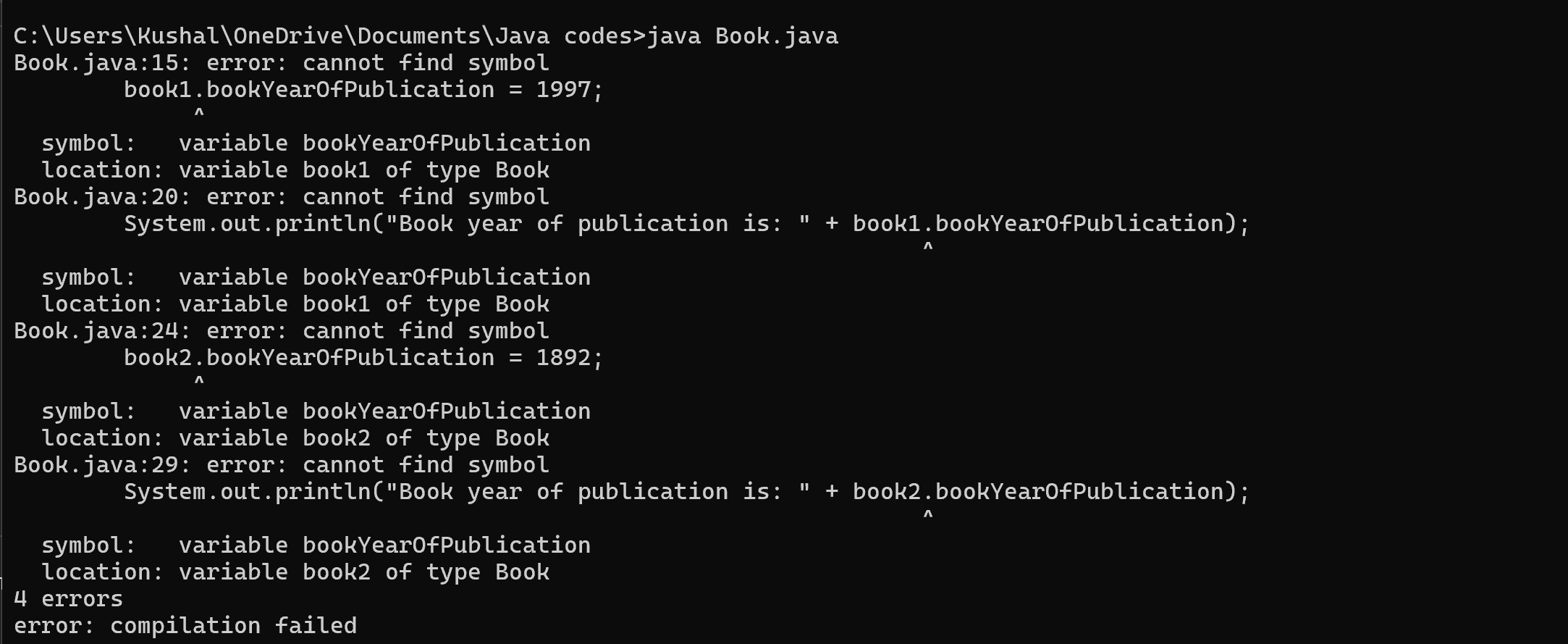
}

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | Not defining the function in a file. | To call the method we must define a function in a file. |
| 2. | Two public class files should not be saved in the same file | Two public class files should be saved in different files |

**Positive Output:**



**Negative Output:  
**

**Important points:**

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

**2. Create a Java program with a class named ‘MyClass’ with a static variable count of int type, initialized to zero and a constant variable ‘pi’ or 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 the count and pi variables. Create three objects and a constructor.**

**CLASS DIAGRAM-**

**+------------------------------------+**

**| MyClass |**

**+------------------------------------+**

**| -count : static int |**

**| -pi : final double = 3.14 |**

**+------------------------------------+**

**| +MyClass() |**

**| +main(String[] args) : void |**

**+------------------------------------+**

CODE:

class MyClass {

static int count = 0;

final double pi = 3.14;

public MyClass() {

count++;

}

public static void main(String[] args) {

MyClass object1 = new MyClass();

MyClass object2 = new MyClass();

MyClass object3 = new MyClass();

System. out.println("Final count of objects created: " + count);

System. out.println("Value of pi: " + object1.pi);

System.out.println("Kushal CSE24333 CSE-C");

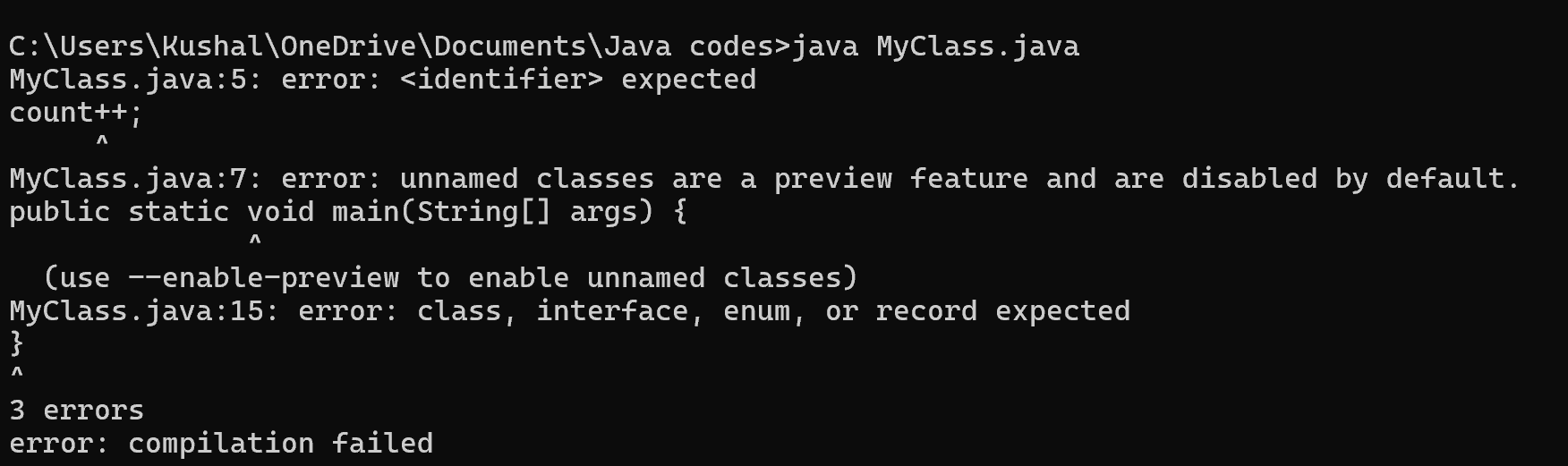
}

}

**Positive Output:**

****

**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | error: can't find primary (String []) method in class : | Should close the string brackets [] |

**Important points:**

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

**WEEK-5**

**1.Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output.**

**CLASS DIAGRAM-**

**+-------------------------------------------------+**

**| Addition |**

**+-------------------------------------------------+**

**| +add(a: double, b: double) : double |**

**+-------------------------------------------------+**

**▲**

**|**

**+------------------------------------------------------+**

**| Subtraction |**

**+------------------------------------------------------+**

**| +subtract(a: double, b: double) : double |**

**+------------------------------------------------------+**

**▲**

**|**

**+-------------------------------------------------------+**

**| Multiplication |**

**+-------------------------------------------------------+**

**| +multiply(a: double, b: double) : double |**

**+--------------------------------------------------------+**

**▲**

**|**

**+----------------------------------------------------+**

**| Division |**

**+----------------------------------------------------+**

**| +divide(a: double, b: double) : double |**

**+----------------------------------------------------+**

**+------------------------------------------+**

**| CalculatorDemo |**

**+------------------------------------------+**

**| +main(String[] args) : void |**

**+------------------------------------------+**

**CODE:**

class Addition {

public double add(double a, double b) {

return a + b;

}

public static void main(String[] args) {

Division calc = new Division();

double num1 = 20;

double num2 = 5;

System.out.println("Number 1: " + num1);

System.out.println("Number 2: " + num2);

System.out.println("Addition: " + calc.add(num1, num2));

System.out.println("Subtraction: " + calc.subtract(num1, num2));

System.out.println("Multiplication: " + calc.multiply(num1, num2));

System.out.println("Division: " + calc.divide(num1, num2));

}

}

class Subtraction extends Addition {

public double subtract(double a, double b) {

return a - b;

}

}

class Multiplication extends Subtraction {

public double multiply(double a, double b) {

return a \* b;

}

}

class Division extends Multiplication {

public double divide(double a, double b) {

return a / b;

}

}

public class CalculatorDemo {

public static void main(String[] args) {

Division calc = new Division();

double num1 = 20;

double num2 = 5;

System.out.println("Number 1: " + num1);

System.out.println("Number 2: " + num2);

System.out.println("Addition: " + calc.add(num1, num2));

System.out.println("Subtraction: " + calc.subtract(num1, num2));

System.out.println("Multiplication: " + calc.multiply(num1, num2));

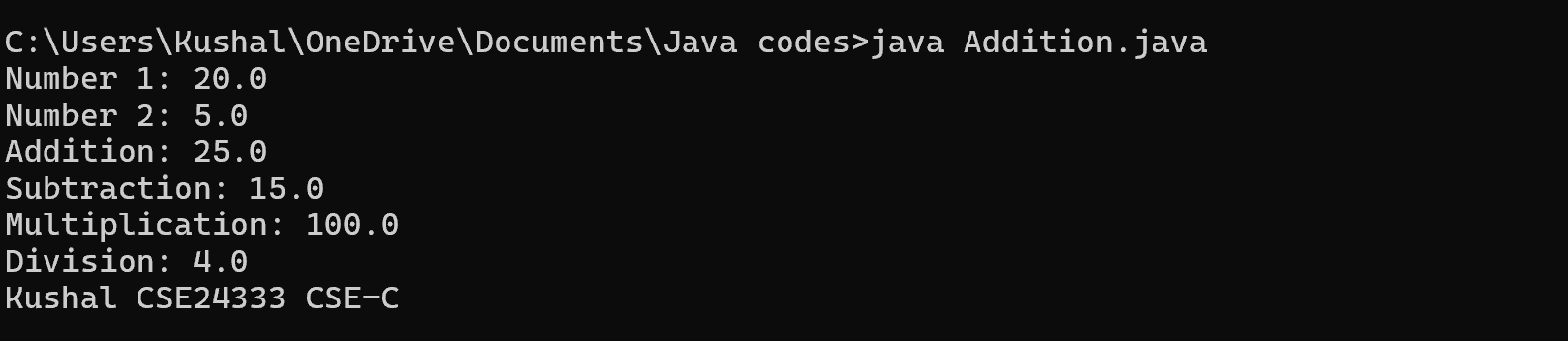
System.out.println("Division: " + calc.divide(num1, num2));

System.out.println("Kushal CSE24333 CSE-C");

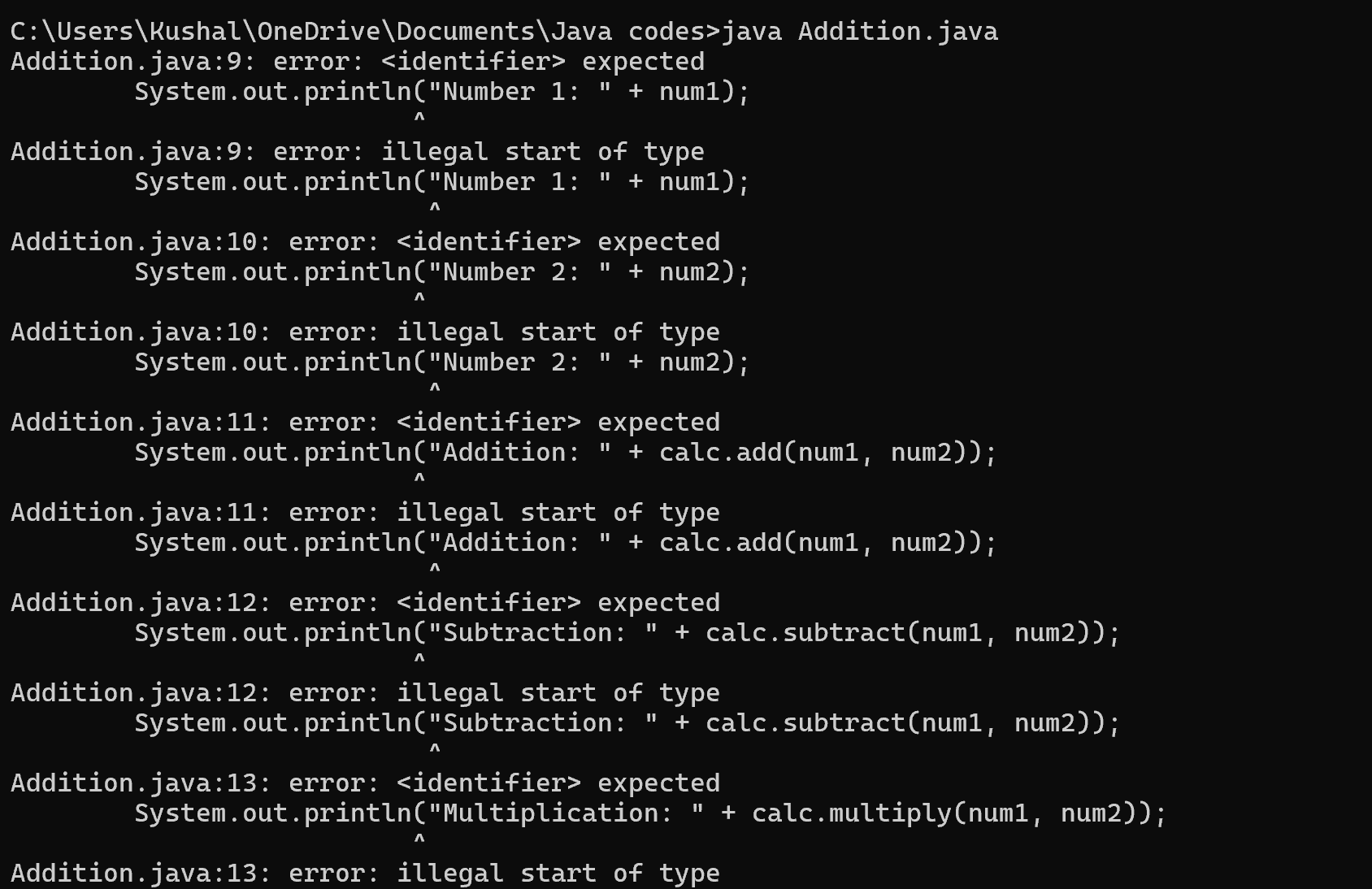
}

}

**Positive Output:**

****

**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | The Calculator class is an empty base class: | abstract class Calculator {  // Abstract method (optional)  public abstract int add(int a, int b);  } |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

1.Inheritance Structure:

Simple extends Calculator, Super extends Simple, and Advanced extends Super, which shows a good hierarchical structure for building more complex functionality on top of simpler functionality.

2.Use of Math.sqrt(a) in Super:

The squareRoot method in the Super class is good because it provides an additional operation not found in Simple.

**2**

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

**1. cars should have an additional property: no.of doors**

**2. Bikes should have a property indicating whether they have gears or not.**

**3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.**

**4. Every class should have a constructor**

**CLASS DIAGRAM-**

**+-----------------------------------------------------+**

**| Vehicle2 |**

**+-----------------------------------------------------+**

**| -brand : String |**

**| -speed : int |**

**+-----------------------------------------------------+**

**| +Vehicle2(brand: String, speed: int) |**

**| +start() : void |**

**| +displayDetails() : void |**

**+-----------------------------------------------------+**

**▲**

**-----------------------**

**| |**

**+-------------------+ +------------------+**

**| Car | | Bike |**

**+-------------------+ +-------------------+**

**| -noofDoors : int | -hasGears : boolean |**

**+-------------------------+ +------------------------+**

**| +Car(brand, speed, noofDoors) |**

**| +displayDetails() : void (overridden) |**

**+--------------------------+ +------------------------+**

**+--------------------------------------------------------+**

**| VehicleRentalSystem |**

**+--------------------------------------------------------+**

**| +main(String[] args) : void |**

**+--------------------------------------------------------+**

CODE:

class Vehicle2 {

protected String brand;

protected int speed;

public Vehicle2(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

public void start() {

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

}

public void displayDetails() {

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

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

}

public static void main(String[] args) {

Car car1 = new Car("Toyota", 180, 4);

Bike bike1 = new Bike("Yamaha", 120, true);

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

car1.displayDetails();

car1.start();

System.out.println("\nBike Details:");

bike1.displayDetails();

bike1.start();

}

}

class Car extends Vehicle2 {

private int noofDoors;

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

super(brand, speed);

this.noofDoors = noofDoors;

}

@Override

public void displayDetails() {

super.displayDetails();

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

}

}

class Bike extends Vehicle2 {

private boolean hasGears;

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

super(brand, speed);

this.hasGears = hasGears;

}

@Override

public void displayDetails() {

super.displayDetails();

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

}

}

public class VehicleRentalSystem {

public static void main(String[] args) {

Car car1 = new Car("Toyota", 180, 4);

Bike bike1 = new Bike("Yamaha", 120, true);

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

car1.displayDetails();

car1.start();

System.out.println("\nBike Details:");

bike1.displayDetails();

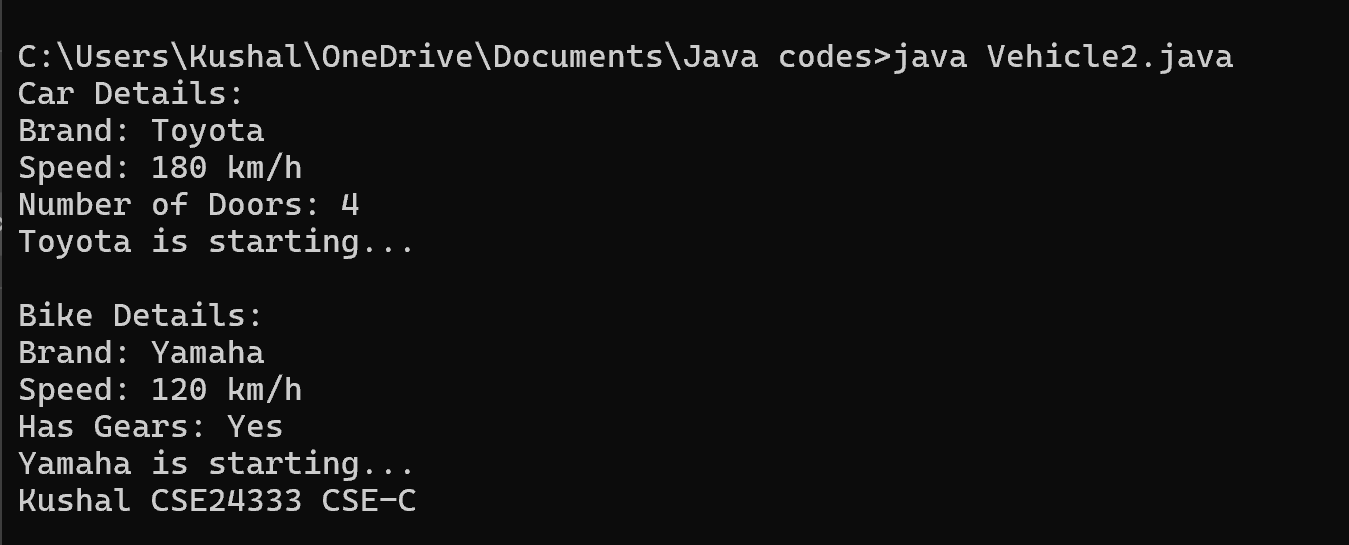
bike1.start();

System.out.println("Kushal CSE24333 CSE-C");

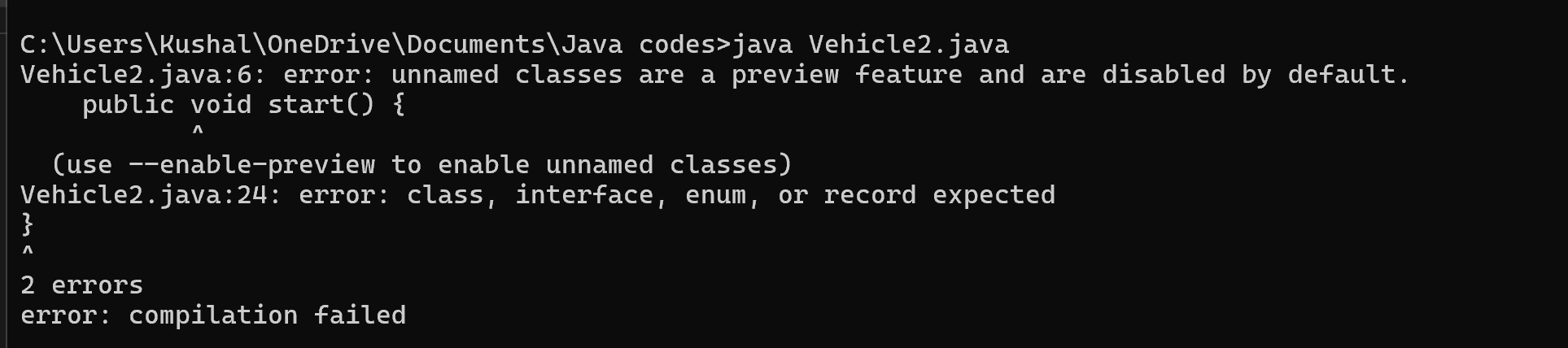
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | Java class names should start with an uppercase letter | Change CARS to Cars, Bikes to Bikes, and Trucks to Trucks |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

1. Encapsulation: Consider making the fields in the Vehicle class private and providing public getter methods to access them. This promotes encapsulation.
2. Code Reusability: If there are common functionalities across classes, consider abstract classes or interfaces to promote code reusability

**WEEK-6**

**1.Write a java program to create a Vehicle class with a method in the Car subclass to provide specific information about a car.**

CODE:

class Vehicle3 {

protected String brand;

protected int speed;

public Vehicle3(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

public void displayGeneralInfo() {

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

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

}

public static void main(String[] args) {

Car car = new Car("Honda", 160, 4, "Petrol");

car.displayCarInfo();

}

}

class Car extends Vehicle3 {

private int numberOfDoors;

private String fuelType;

public Car(String brand, int speed, int numberOfDoors, String fuelType) {

super(brand, speed);

this.numberOfDoors = numberOfDoors;

this.fuelType = fuelType;

}

public void displayCarInfo() {

displayGeneralInfo();

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

System.out.println("Fuel Type: " + fuelType);

}

}

public class VehicleInfo {

public static void main(String[] args) {

Car car = new Car("Honda", 160, 4, "Petrol");

car.displayCarInfo();

System.out.println("Kushal CSE24333 CSE-C");

}

}

**CLASS DIAGRAM-**

**+-------------------------------------------------+**

**| Vehicle3 |**

**+-------------------------------------------------+**

**| -brand : String |**

**| -speed : int |**

**+-------------------------------------------------+**

**| +Vehicle3(brand: String, speed: int) |**

**| +displayGeneralInfo() : void |**

**+-------------------------------------------------+**

**▲**

**|**

**+--------------------------------------------------------------------------------------------+**

**| Car |**

**+--------------------------------------------------------------------------------------------+**

**| -numberOfDoors : int |**

**| -fuelType : String |**

**+--------------------------------------------------------------------------------------------+**

**| +Car(brand: String, speed: int, numberOfDoors: int, fuelType: String) |**

**| +displayCarInfo() : void |**

**+--------------------------------------------------------------------------------------------+**

**+--------------------------------------+**

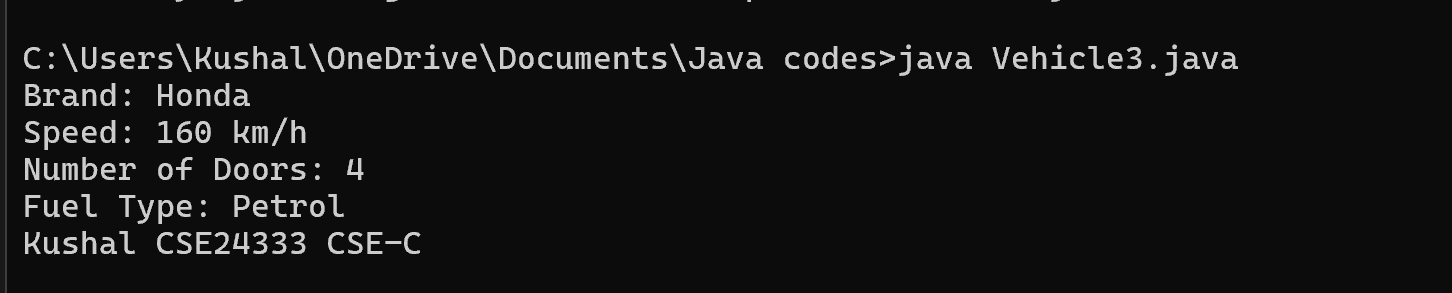
**| VehicleInfo |**

**+--------------------------------------+**

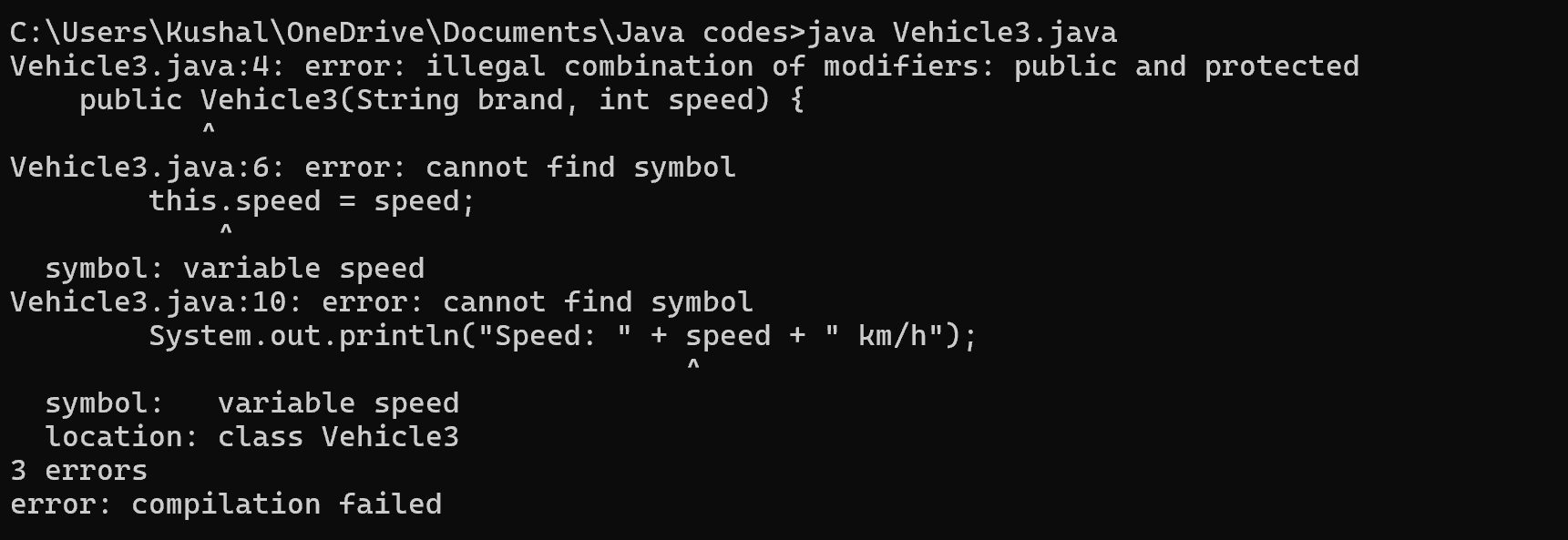
**| +main(String[] args) : void |**

**+--------------------------------------+**

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | Main method not found in class Vehicle, please define the main method as:  public static void main(String[] args) | **public static void main(String[] args);** |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

**1.**The @Override annotation is used to indicate that the method is meant to override a method in the superclass.

**2.**In order to run the java program which involves the overriding concept then we have to name the file with last class name.

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

* **UG admission require a min of 60%**
* **PG admission require a min of 70%**

**CLASS DIAGRAM-**

**+-------------------------------------------------------------+**

**| Student |**

**+-------------------------------------------------------------+**

**| -name : String |**

**| -percentage : double |**

**+-------------------------------------------------------------+**

**| +Student(name: String, percentage: double) |**

**| +displayInfo() : void |**

**+-------------------------------------------------------------+**

**▲**

**|**

**------------------------**

**| |**

**+------------------+ +------------------+**

**| UGStudent | | PGStudent |**

**+--------------------+ +--------------------+**

**| +UGStudent(name, percentage) |**

**| +checkEligibility() : void |**

**+--------------------+ +-------------------+**

**+-----------------------------------+**

**| AdmissionSystem |**

**+-----------------------------------+**

**| +main(String[] args) : void |**

**+------------------------------------+**

**CODE:**

class Student {

protected String name;

protected double percentage;

public Student(String name, double percentage) {

this.name = name;

this.percentage = percentage;

}

public void displayInfo() {

System.out.println("Student Name: " + name);

System.out.println("Percentage: " + percentage + "%");

}

public static void main(String[] args) {

UGStudent ug = new UGStudent("Alice", 65.5);

PGStudent pg = new PGStudent("Bob", 68.0);

System.out.println("UG Admission Check:");

ug.checkEligibility();

System.out.println("\nPG Admission Check:");

pg.checkEligibility();

}

}

class UGStudent extends Student {

public UGStudent(String name, double percentage) {

super(name, percentage);

}

public void checkEligibility() {

displayInfo();

if (percentage >= 60) {

System.out.println("Eligible for UG Admission.");

} else {

System.out.println("Not eligible for UG Admission.");

}

}

}

class PGStudent extends Student {

public PGStudent(String name, double percentage) {

super(name, percentage);

}

public void checkEligibility() {

displayInfo();

if (percentage >= 70) {

System.out.println("Eligible for PG Admission.");

} else {

System.out.println("Not eligible for PG Admission.");

}

}

}

public class AdmissionSystem {

public static void main(String[] args) {

UGStudent ug = new UGStudent("Alice", 65.5);

PGStudent pg = new PGStudent("Bob", 68.0);

System.out.println("UG Admission Check:");

ug.checkEligibility();

System.out.println("\nPG Admission Check:");

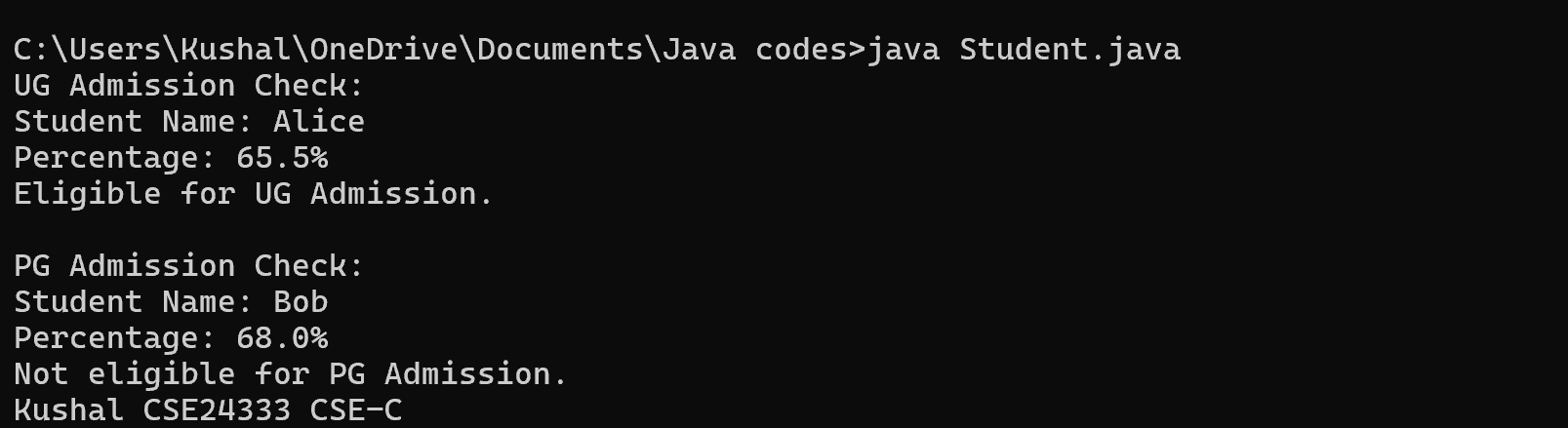
pg.checkEligibility();

System.out.println("Kushal CSE24333 CSE-C");

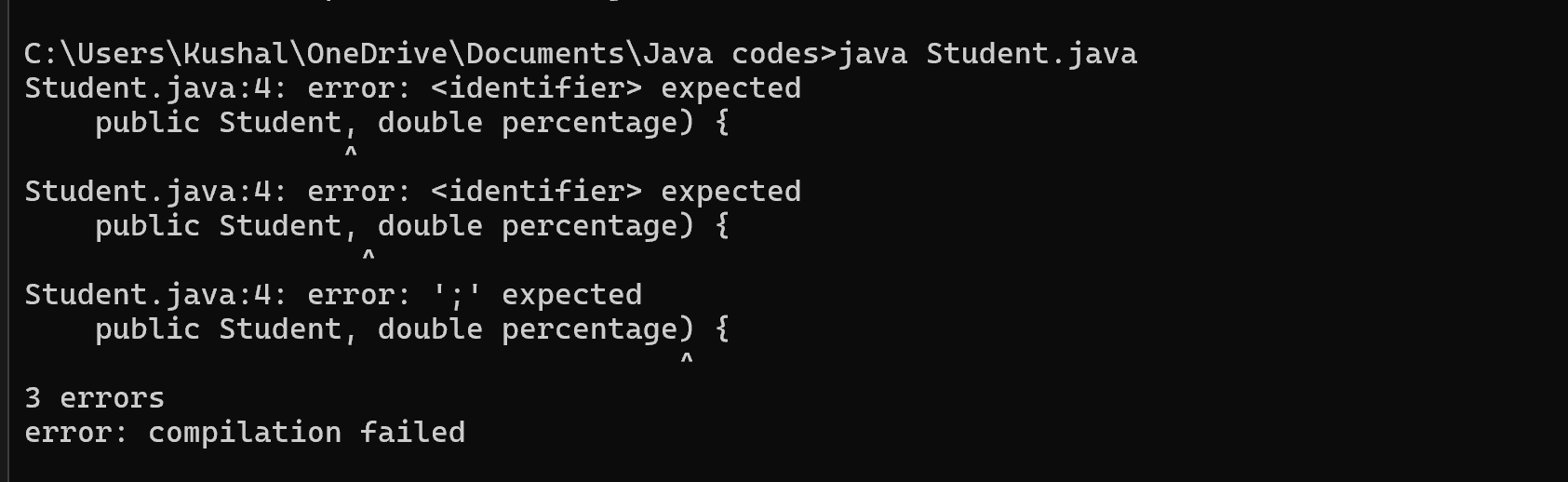
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | you are using the super(name, percentage) constructor in both UG and PG classes correctly, | Let's change the percentage field to be protected so that subclasses can access it directly, or alternatively, create a getPercentage() method. |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

Access Modifiers:

In this case, percentage was set to default visibility (which means it’s package-private). By changing it to protected, the subclasses UG and PG are able to access it directly.

Method Overriding:

You’ve correctly overridden the getEligibility method in the UG and PG classes. This allows you to provide specific behavior for these subclasses

**3**

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

1**.** Add two integers

2. Add two doubles

3. Add three integers

**CLASS DIAGRAM-**

**+-----------------------------------------------------+**

**| Calculator2 |**

**+-----------------------------------------------------+**

**| +add(int a, int b) : int |**

**| +add(double a, double b) : double |**

**| +add(int a, int b, int c) : int |**

**+----------------------------------------------------+**

**+-----------------------------------------------------+**

**| CalculatorDemo |**

**+-----------------------------------------------------+**

**| +main(String[] args) : void |**

**+-----------------------------------------------------+**

CODE:

class Calculator2 {

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;

}

public static void main(String[] args) {

Calculator2 calc = new Calculator2();

int sum1 = calc.add(10, 20);

double sum2 = calc.add(12.5, 7.3);

int sum3 = calc.add(5, 10, 15);

System.out.println("Sum of two integers: " + sum1);

System.out.println("Sum of two doubles: " + sum2);

System.out.println("Sum of three integers: " + sum3);

}

}

public class CalculatorDemo {

public static void main(String[] args) {

Calculator2 calc = new Calculator2();

int sum1 = calc.add(10, 20);

double sum2 = calc.add(12.5, 7.3);

int sum3 = calc.add(5, 10, 15);

System.out.println("Sum of two integers: " + sum1);

System.out.println("Sum of two doubles: " + sum2);

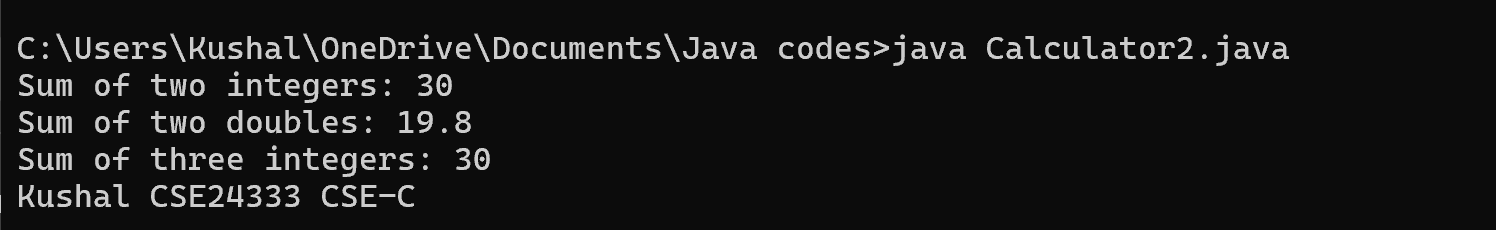
System.out.println("Sum of three integers: " + sum3);

System.out.println("Kushal CSE24333 CSE-C");

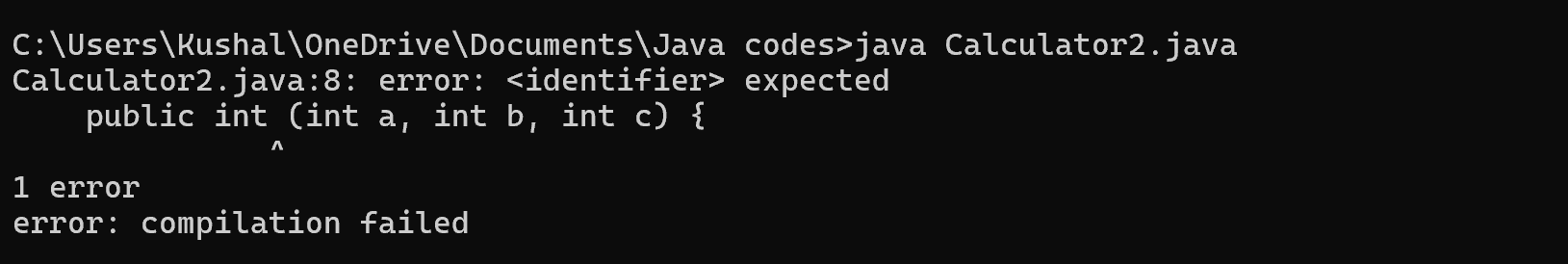
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | **Method Signature Ambiguity :**If you mix int and double types in certain combinations when overloading methods, | public void add(double a, int b) {  System.out.println("Sum of " + a + " and " + b + " is: " + (a + b));  } |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

Method Overloading:

Java allows overloading methods, which means you can define multiple methods with the same name but different parameter lists.

Avoid Ambiguity:

While method overloading is a great feature, it can sometimes lead to ambiguity when methods have similar signatures.

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

**CLASS DIAGRAM-**

**+--------------------------------------------------+**

**| Shape |**

**+--------------------------------------------------+**

**| +calculatearea(int l, int b) : void |**

**| +calculatearea(int x) : void |**

**+--------------------------------------------------+**

**▲**

**|**

**|**

**+---------------------------------------------------------+**

**| Circle |**

**+---------------------------------------------------------+**

**| +calculatearea(double pi, double r) : void |**

**+---------------------------------------------------------+**

**+----------------------------------------+**

**| Areas2 |**

**+----------------------------------------+**

**| +main(String[] args) : void |**

**+----------------------------------------+**

**CODE:**

class Shape {

public double calculateArea(double side) {

return side \* side;

}

public double calculateArea(double length, double width) {

return length \* width;

}

}

class Circle extends Shape {

private double radius;

public Circle(double radius) {

this.radius = radius;

}

@Override

public double calculateArea() {

return Math.PI \* radius \* radius;

}

}

public class ShapeDemo {

public static void main(String[] args) {

Shape shape = new Shape();

Circle circle = new Circle(5.0);

double squareArea = shape.calculateArea(4.0);

double rectangleArea = shape.calculateArea(5.0, 3.0);

double circleArea = circle.calculateArea();

System.out.println("Area of Square: " + squareArea);

System.out.println("Area of Rectangle: " + rectangleArea);

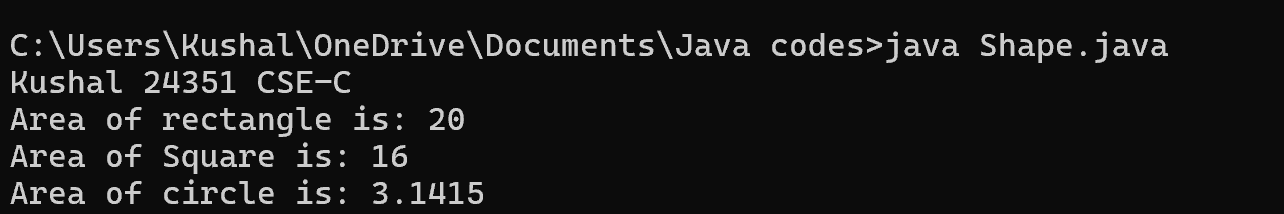
System.out.println("Area of Circle: " + circleArea);

System.out.println("Kushal CSE24333 CSE-C");

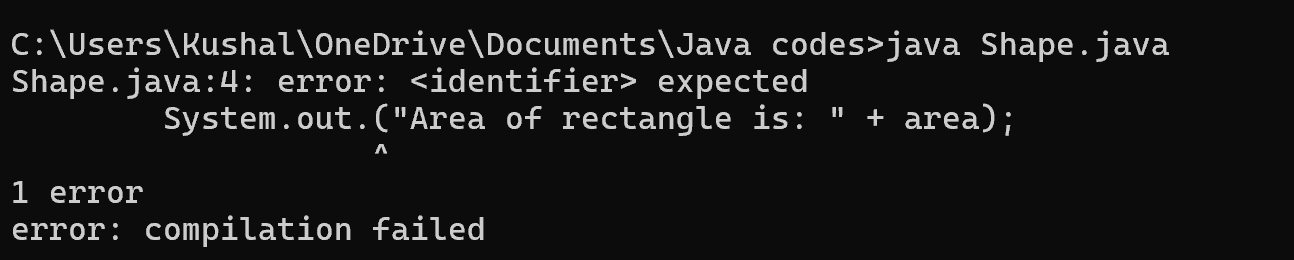
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | **Error in Circle class:**  **Method Signature Ambiguity :**If you mix int and double types in certain combinations when overloading methods,  The Circle class should override the calculateArea method properly. | public void add(double a, int b) {  System.out.println("Sum of " + a + " and " + b + " is: " + (a + b));  }  **Override the method in Circle** to calculate the area of a circle |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

Method Overloading:

Java allows overloading methods, which means you can define multiple methods with the same name but different parameter lists.

Avoid Ambiguity:

While method overloading is a great feature, it can sometimes lead to ambiguity when methods have similar signatures.

Inheritance:

In this case, Circle inherits from Shape, and the calculateArea method is overridden in the Circle class to calculate the area of a circle.

Method Overloading occurs when you define multiple methods with the same name but different parameter lists (for example, calculating area for squares, rectangles, and circles using different parameters).

**WEEK-7**

**1.Write a java program to create an abstract class Animal with an abstract method called sound().Create a subclass lion and tiger that extends the Animal class and implement the sound method to make a specific sound for each animal.**

CODE:

abstract class Animal {

abstract void makeSound();

public static void main(String[] args) {

Animal tiger = new Tiger();

Animal lion = new Lion();

tiger.makeSound();

lion.makeSound();

System.out.println("Kushal CSE24333 CSE-C");

}

}

class Tiger extends Animal {

@Override

void makeSound() {

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

}

}

class Lion extends {

@Override

void makeSound() {

System.out.println("Lion Roar");

}

}

public class AnimalTest {

public static void main(String[] args) {

Animal tiger = new Tiger();

Animal lion = new Lion();

tiger.makeSound();

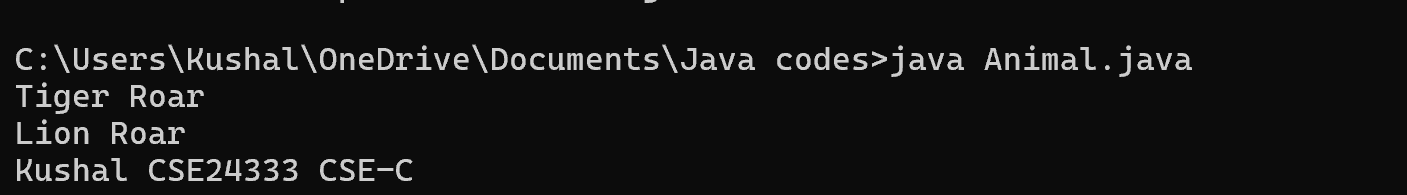
lion.makeSound();

System.out.println("Kushal CSE24333 CSE-C");

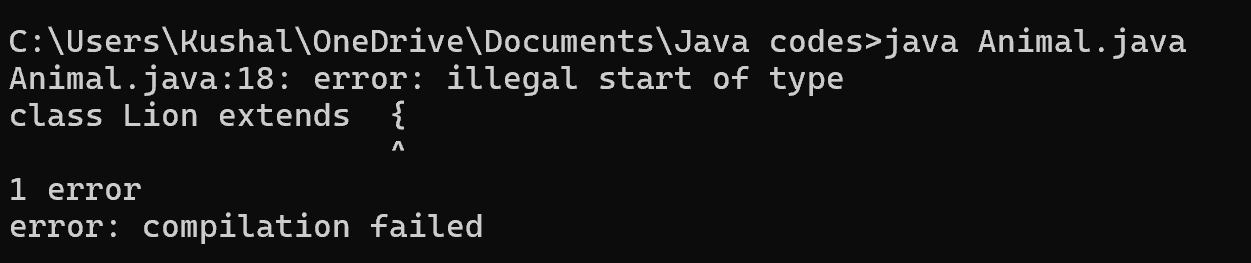
}

}

**Positive Output:**



**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | **Error in Circle class:**  **Method Signature Ambiguity :**If you mix int and double types in certain combinations when overloading methods,  The Circle class should override the calculateArea method properly. | public void add(double a, int b) {  System.out.println("Sum of " + a + " and " + b + " is: " + (a + b));  }  **Override the method in Circle** to calculate the area of a circle |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

Method Overloading:

Java allows overloading methods, which means you can define multiple methods with the same name but different parameter lists.

Avoid Ambiguity:

While method overloading is a great feature, it can sometimes lead to ambiguity when methods have similar signatures.

Inheritance:

In this case, Circle inherits from Shape, and the calculateArea method is overridden in the Circle class to calculate the area of a circle.

Method Overloading occurs when you define multiple methods with the same name but different parameter lists (for example, calculating area for squares, rectangles, and circles using different parameters).

**2. 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 calculateVolume();**

**abstract double calculateSurfaceArea();**

**}**

**class Sphere extends Shape3D {**

**private final double radius;**

**Sphere(double radius) {**

**this.radius = radius;**

**}**

**@Override**

**double calculateVolume() {**

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

**}**

**@Override**

**double calculateSurfaceArea() {**

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

**}**

**}**

**class Cube extends Shape3D {**

**private final double sideLength;**

**Cube(double sideLength) {**

**this.sideLength = sideLength;**

**}**

**@Override**

**double calculateVolume() {**

**return Math.pow(sideLength, 3);**

**}**

**@Override**

**double calculateSurfaceArea() {**

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

**}**

**}**

**public class Shape3DTest {**

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

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

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

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

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

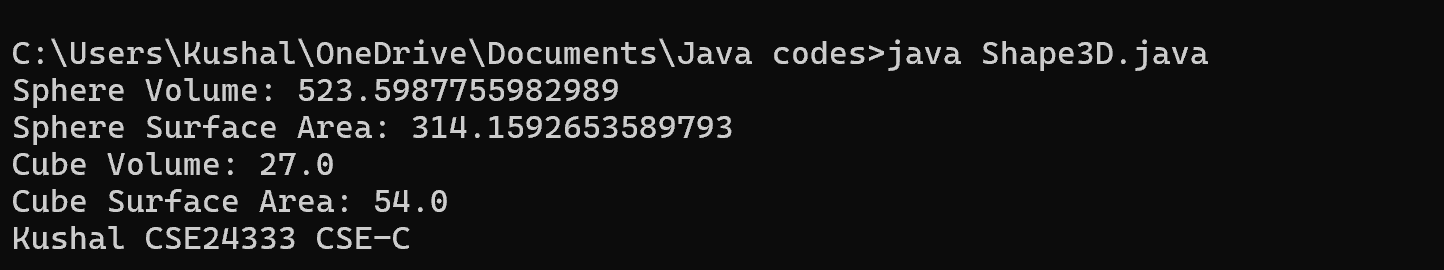
**System.out.println("Cube Volume: " + cube.calculateVolume());**

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

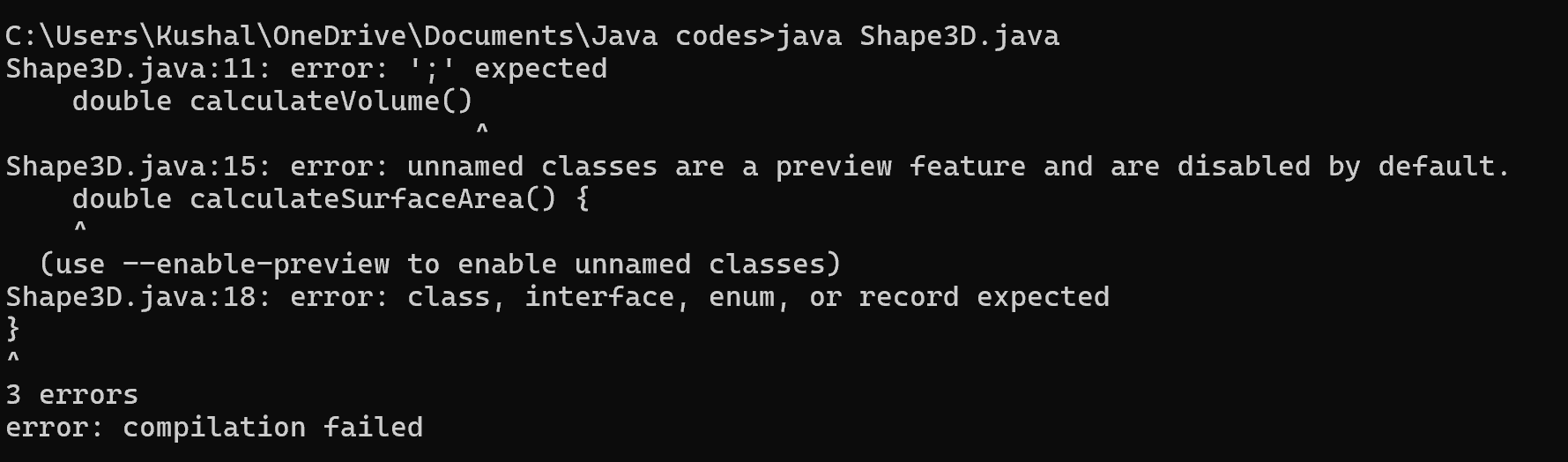
**}**

**}**

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | which has to be the first parameter, and cannot be a lambda parameter | Shape3D.java:17: error: ';' expected |
| 2. | Shape3D.java:17: error: <identifier> expected | this.radius = radius; |

**Important points:**

Inheritance:

In this case, Circle inherits from Shape, and the calculateArea method is overridden in the Circle class to calculate the area of a circle.

Method Overloading occurs when you define multiple methods with the same name but different parameter lists (for example, calculating area for squares, rectangles, and circles using different parameters).

1. **Write a Java program to create an abstract to define a method for pattern printing**

* **Create an abstract class named pattern printer with an abstract method (int n) and a concreate method to display the pattern title.**
* **Implement the subclasses-**

**2. Star Pattern-Prints a right angled triangle for stars**

**1.Number Pattern-Prints a right angled triiangle of increasing numbers**

* **In the main method create objects of both subclasses and print the pattern for a given number of rows**

CODE:

abstract class PatternPrinter {

abstract void printPattern(int n);

public static void main(String[] args) {

PatternPrinter starPattern = new StarPattern();

PatternPrinter numberPattern = new NumberPattern();

System.out.println("Star Pattern:");

starPattern.printPattern(5);

System.out.println("Number Pattern:");

numberPattern.printPattern(5);

System.out.println("Kushal CSE24333 CSE-C");

}

}

class StarPattern extends PatternPrinter {

@Override

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 {

@Override

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();

}

}

}

public class PatternPrinterTest {

public static void main(String[] args) {

PatternPrinter starPattern = new StarPattern();

PatternPrinter numberPattern = new NumberPattern();

System.out.println("Star Pattern:");

starPattern.printPattern(5);

System.out.println("Number Pattern:");

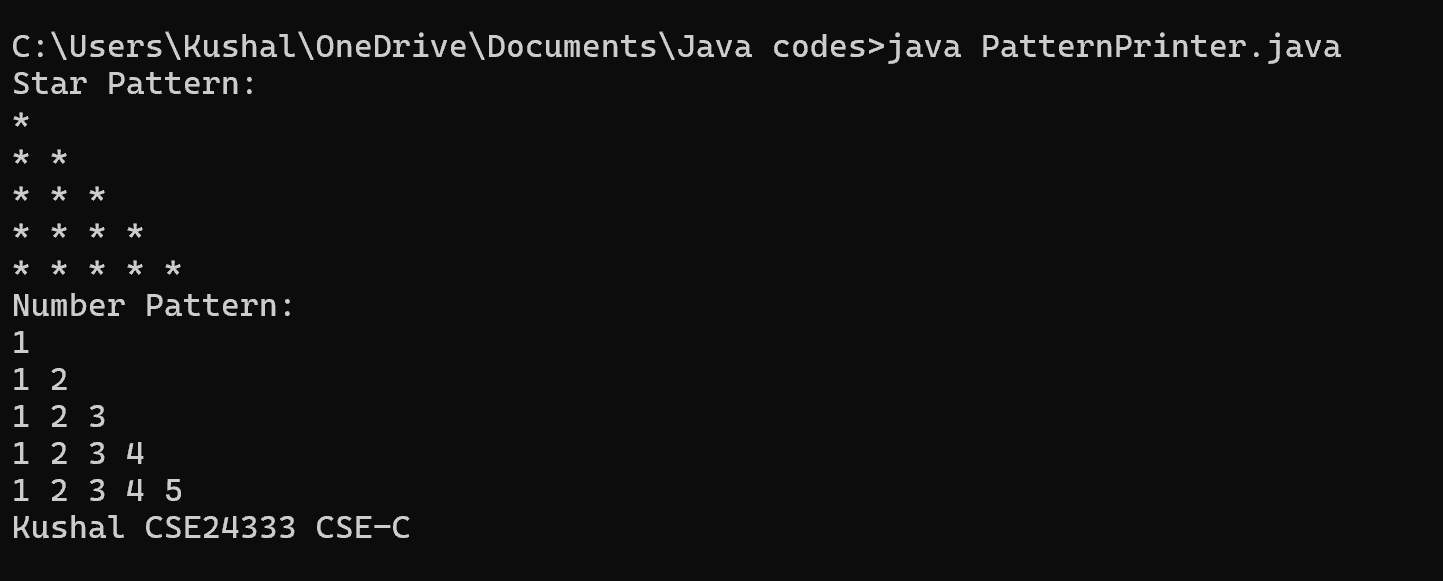
numberPattern.printPattern(5);

System.out.println("Kushal CSE24333 CSE-C");

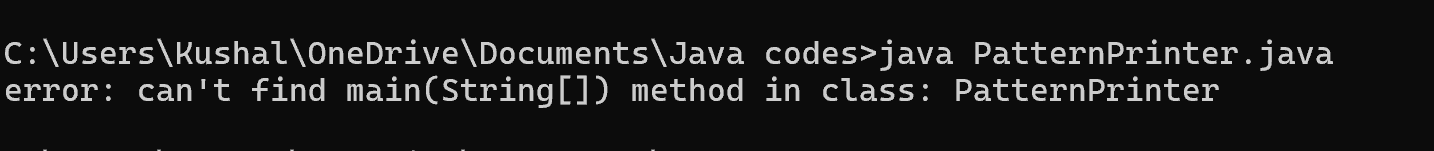
}

}

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | which has to be the first parameter, and cannot be a lambda parameter | Shape3D.java:17: error: ';' expected |
| 2. | Shape3D.java:17: error: <identifier> expected | this.radius = radius; |

**Important points:**

Inheritance:

In this case, Circle inherits from Shape, and the calculateArea method is overridden in the Circle class to calculate the area of a circle.

Method Overloading occurs when you define multiple methods with the same name but different parameter lists (for example, calculating area for squares, rectangles, and circles using different parameters).

**WEEK-8**

**1.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:

interface Shape2 {

    double getPerimeter();

    public static void main(String[] args) {

        Rectangle rectangle = Shape2Factory.createRectangle(5, 10);

        Circle circle = Shape2Factory.createCircle(7);

        Triangle triangle = Shape2Factory.createTriangle(3, 4, 5);

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

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

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

    }

}

class Rectangle implements Shape2 {

    private double width;

    private double height;

    public Rectangle(double width, double height) {

        this.width = width;

        this.height = height;

    }

    @Override

    public double getPerimeter() {

        return 2 \* (this.width + this.height);

    }

}

class Circle implements Shape2 {

    private double radius;

    public Circle(double radius) {

        this.radius = radius;

    }

    @Override

    public double getPerimeter() {

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

    }

}

class Triangle implements Shape2 {

    private double a, b, c;

    public Triangle(double a, double b, double c) {

        this.a = a;

        this.b = b;

        this.c = c;

    }

    @Override

    public double getPerimeter() {

        return this.a + this.b + this.c;

    }

}

class Shape2Factory {

    public static Rectangle createRectangle(double width, double height) {

        return new Rectangle(width, height);

    }

    public static Circle createCircle(double radius) {

        return new Circle(radius);

    }

    public static Triangle createTriangle(double a, double b, double c) {

        return new Triangle(a, b, c);

    }

}

public class Shape2 {

    public static double getPerimeter(Shape2 shape) {

        return shape.getPerimeter();

    }

    public static void main(String[] args) {

        Rectangle rectangle = Shape2Factory.createRectangle(5, 10);

        Circle circle = Shape2Factory.createCircle(7);

        Triangle triangle = Shape2Factory.createTriangle(3, 4, 5);

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

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

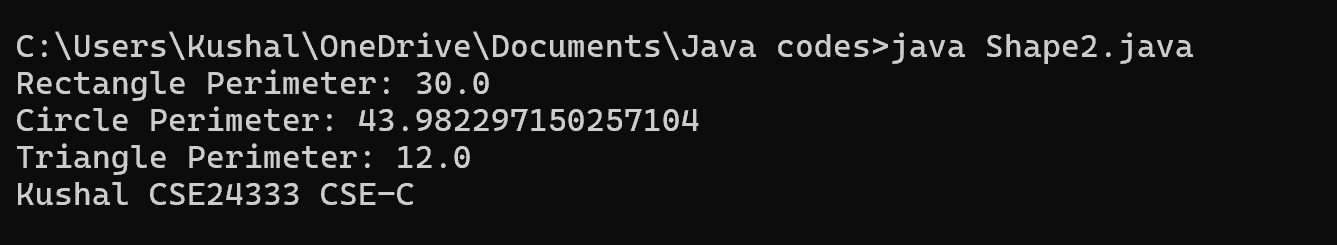
        System.out.println("Triangle Perimeter: " + triangle.getPerimeter());

        System.out.println("Kushal CSE24333 CSE-C");

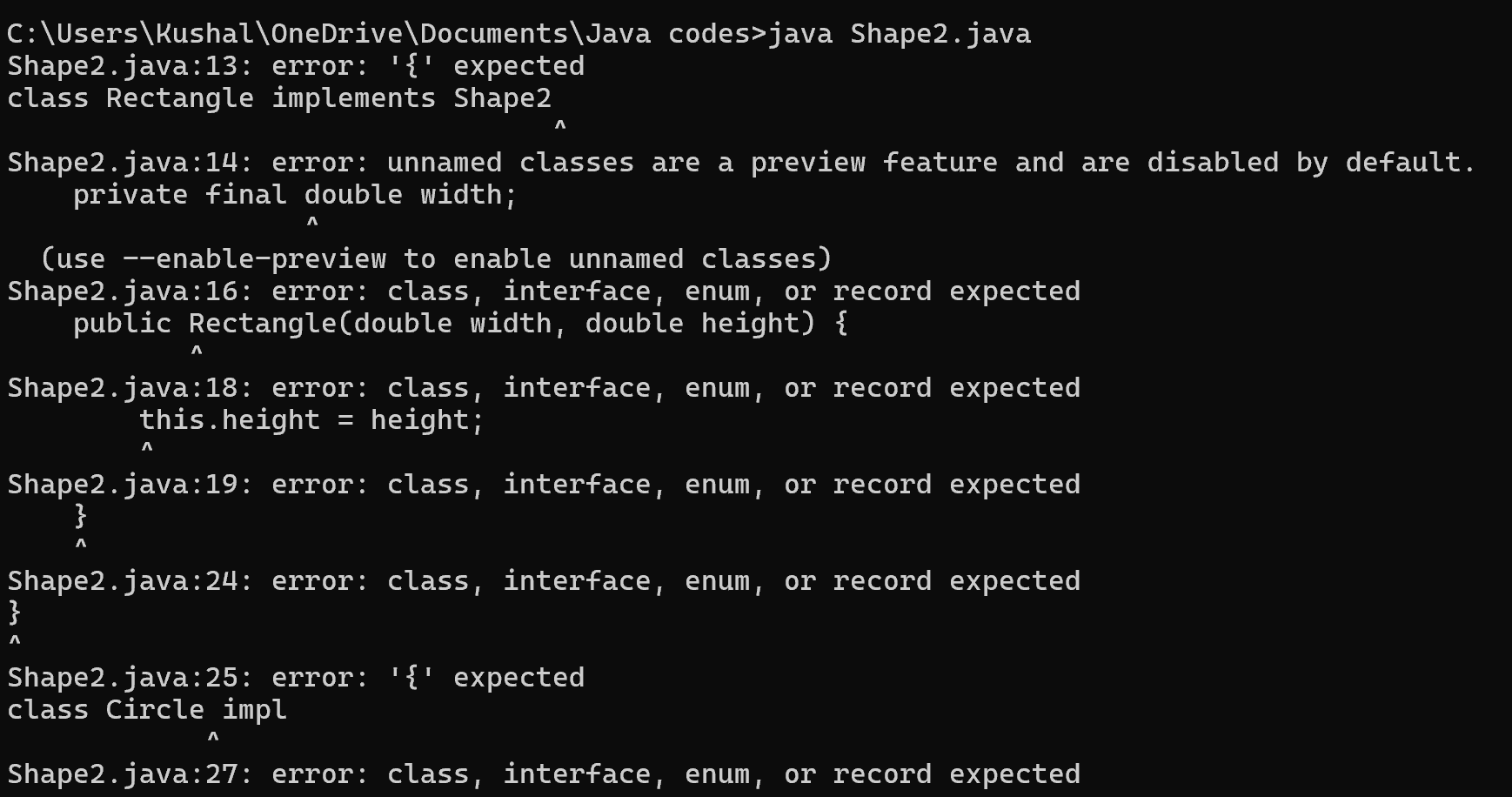
    }

}

**Positive Output:**

****

**Negative Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | PatternPrinter.java:22: error: class, interface, enum, or record expected | Condition needs to be verified |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

**1.**The @Override annotation is used to indicate that the method is meant to override a method in the superclass.

**2.**In order to run the java program which involves the overriding concept then we have to name the file with last class name.

**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 implement the Playable interface and override the play() method to play the respective sports.**

CODE:

interface Playable {

void play();

public static void main(String[] args) {

Playable football = new Football();

Playable volleyball = new Volleyball();

Playable basketball = new Basketball();

football.play();

volleyball.play();

basketball.play();

System.out.println("Kushal CSE24333 CSE-C");

}

}

class Football implements Playable {

@Override

public void play() {

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

}

}

class Volleyball implements Playable {

@Override

public void play() {

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

}

}

class Basketball implements Playable {

@Override

public void play() {

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

}

}

public class PlayableTest {

public static void main(String[] args) {

Playable football = new Football();

Playable volleyball = new Volleyball();

Playable basketball = new Basketball();

football.play();

volleyball.play();

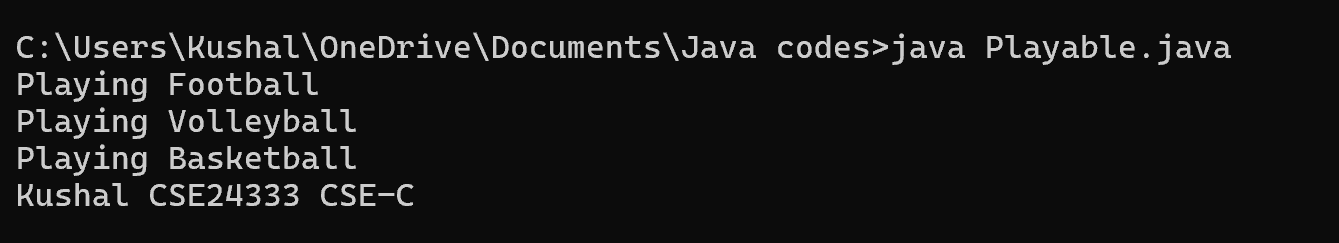
basketball.play();

System.out.println("Kushal CSE24333 CSE-C");

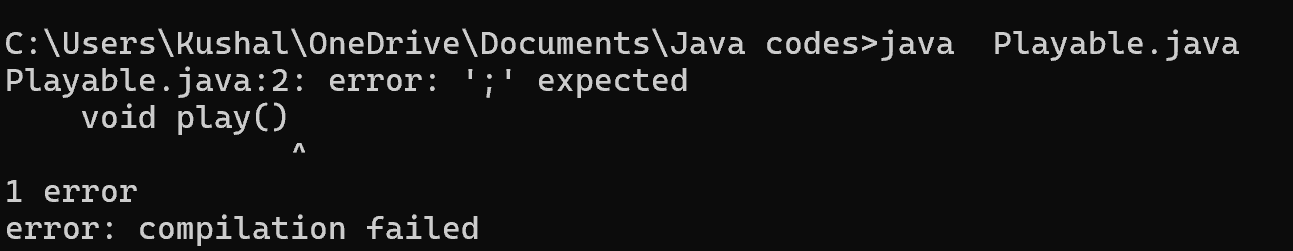
}

}

**Positive Output:**



**Negative Output:**



**Error:**

|  |  |  |
| --- | --- | --- |
| **SI.NO** | **ERROR MESSAGE** | **ERROR RECTIFICATION** |
| 1. | Playable.java:6: error: ';' expected | Keep ‘;’ after sentence |
| 2. | Class Name Error | Give the class name correctly |

**Important points:**

1. Class Naming Conventions: Always follow Java naming conventions for readability and maintainability.
2. Resource Management: Always close resources like Scanner to avoid memory leaks. Using try-with-resources is a good practice.