### **23CSE111**

**OBJECT ORIENTED PROGRAMMING LAB REPORT**

****

**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: N.Manikanta Gowtham**

**Roll No: AV.SC.U4CSE24231**

**Verified By:**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S NO** | **TITLE** | **DATE** | **PAGE NO** | **SIGNATURE** |
| **LAB 1** | **ALGORITHMS ON SEQUENCE**  **STATEMENTS** | **24-09-**  **2024** |  |  |
| 1 | Write an algorithm for adding of two numbers |  |  |  |
| 2 | Write an algorithm to calculate simple interest |  |  |  |
| 3 | Write an algorithm for average of two numbers |  |  |  |
| 4 | Write an algorithm for swapping of two numbers |  |  |  |
| 5 | Write an algorithm to calculate  Fahrenheit to Celsius |  |  |  |
| **LAB 2** | **ALGORITHMS ON SELECTION**  **STATEMENTS** | **1-10-**  **2024** |  |  |
| 1 | Write an algorithm to calculate  even or odd number |  |  |  |
| 2 | Write an algorithm to calculate  leap year |  |  |  |
| 3 | Write an algorithm to calculate  eligibility to vote |  |  |  |
| 4 | Write an algorithm to calculate  biggest of 3 numbers |  |  |  |
| 5 | Write an algorithm to calculate  grade of a student |  |  |  |
| **LAB 3** | **ALGORITHMS ON LOOPS**  **STATEMENTS** | **8-10-**  **2024** |  |  |
| 1 | Write an algorithm to calculate  factorial of a number |  |  |  |
| 2 | Write an algorithm to calculate Fibonacci series |  |  |  |
| 3 | Write an algorithm to calculate reverse of a number |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | Write an algorithm to calculate  sum of given digits |  |  |  |
| 5 | Write an algorithm to calculate  palindrome |  |  |  |
| **LAB 4** |  |  |  |  |
| 1 | Write a java program with following instructions |  |  |  |
| 2 | Write a java program with following instructions |  |  |  |
| **LAB 5** |  |  |  |  |
| **1** | create a calculator using the operations including addition, subtraction, multiplication and division using multilevel inheritance and display the desired output . |  |  |  |
| **2** | A vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes and they need a program to store details about each variable such as brand and speed.  1. Cars should have an additional property: numbers of doors, seating capacity.  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 & indicates when a vehicle is starting.  4. Every class should have constructor.  QUESTION:  1. Which OOP concept is used in the above program? Explain why it is useful in this scenario.  2. If the company decides to add a new type of vehicle: Truck, how would you modify the program?  a.Truck should include an additional property capacity(in tons)  b.Create a ShowTruckDetails() method to display the truck’s capacity.  c.Write a constructor for Truck that initializes all properties. |  |  |  |
| **LAB 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.** |  |  |  |
| **2** | 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% |  |  |  |
| **3** | Create a Calculator class with overloaded methods to perform addition:  (i) Add two integers.  (ii) Add two doubles.  (iii) Add three integers |  |  |  |
| **4** | Create a Calculator class with overloaded methods to perform addition:  (i) Add two integers.  (ii) Add two doubles.  (iii) Add three integers |  |  |  |
| **LAB 7** |  |  |  |  |
| **1** | **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.** |  |  |  |
| **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** |  |  |  |
| **3** | **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** |  |  |  |
| **LAB 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.** |  |  |  |
| **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.** |  |  |  |
| **3** | **write a java program to implement a login system using interfaces** |  |  |  |

# **WEEK -1**

#### **Program -1:**

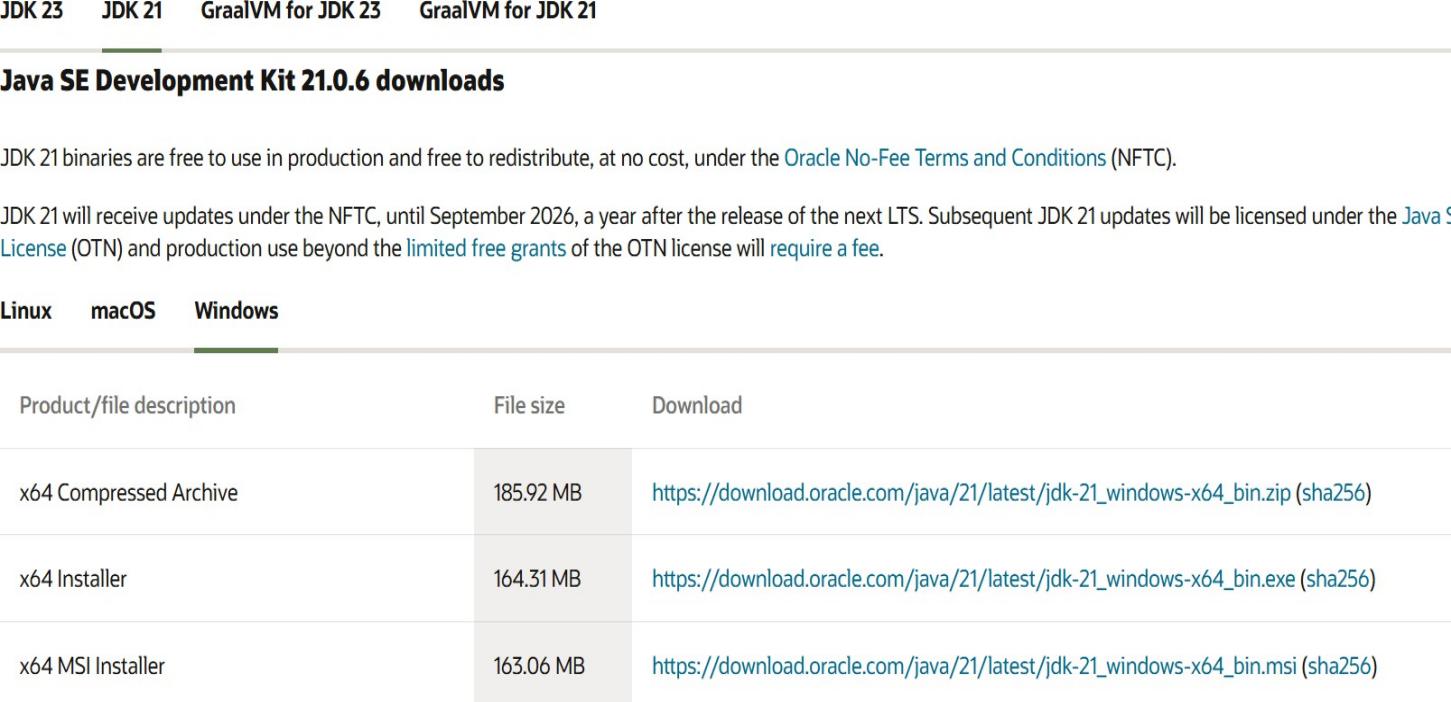
#### **Aim:**

Download and Install Java Software

#### **Procedure:**

**Step-1: Download JDK-21**

* Go to web browser and type Oracle JDK Download.
* Now click on the official website.
* Scroll down to the Java SE Development Kit 21.0.6 downloads section.
* Choose the operating system (macOS, Windows, Linux).
* Click on Download, then wait for the download to complete.



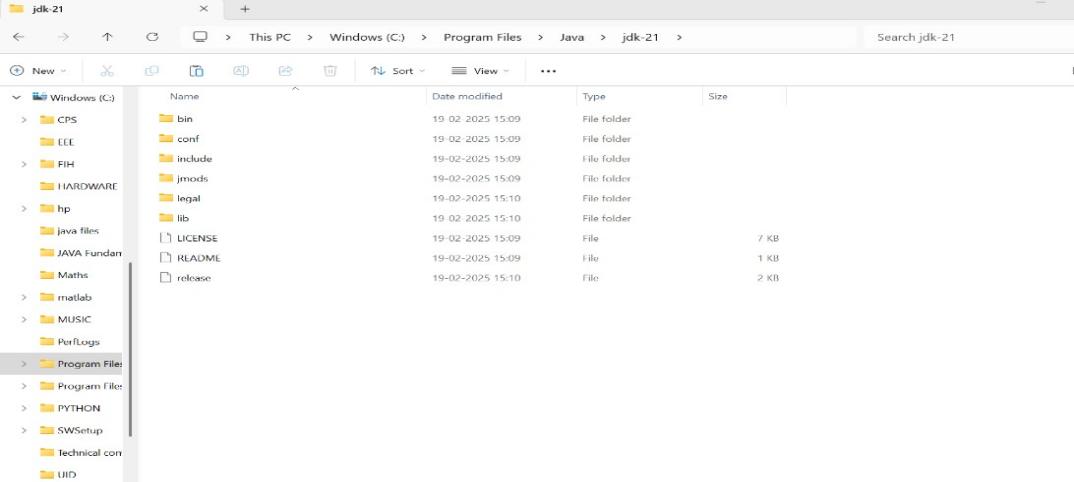
#### **Step 2: Install JDK 21**

* Locate the downloaded jdk-21\_windows-x64\_bin.exe file.
* Double-click to launch the installer.
* Click Next on the setup wizard.
* Choose the installation path (default is C:\Program Files\Java\jdk-21).
* Click Next, then click Install, wait for the installation to complete.
* Click Close once the installation is finished.



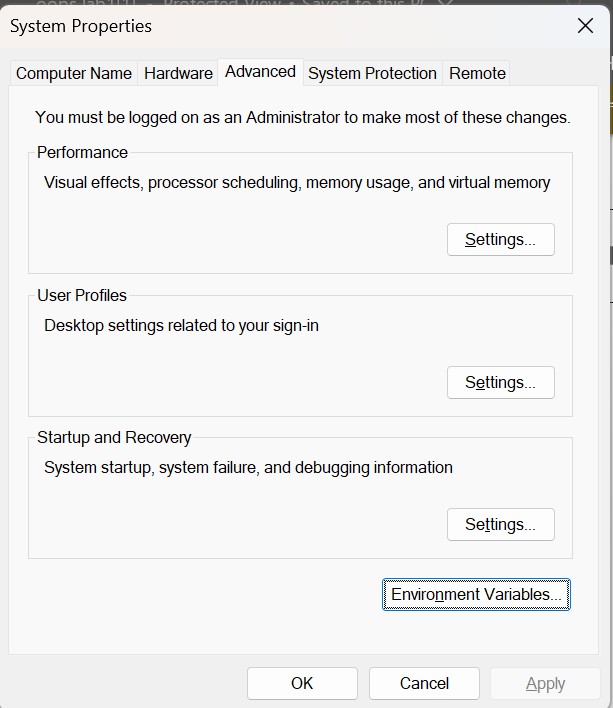
#### **Step 3: Setting up the path**

* Go to “Windows C” Drive on Desktop.
* Choose Program Files, select Java, then JDK 21, then select Bin.
* Select and copy the path at the address bar.



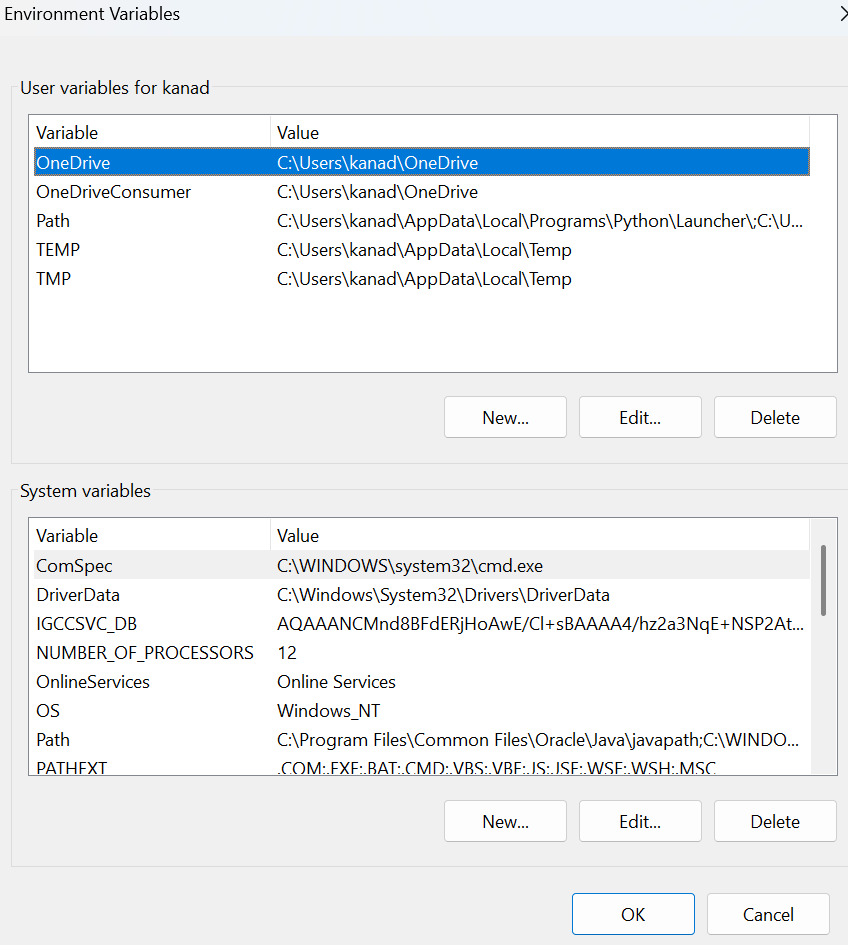
#### **Step 4: Open System Properties**

* + Open file explorer, then right click on This PC.
  + Next select on properties then it will take you to the settings app.
  + Click on Advanced tab.
  + Click on Environment Variables at the bottom.



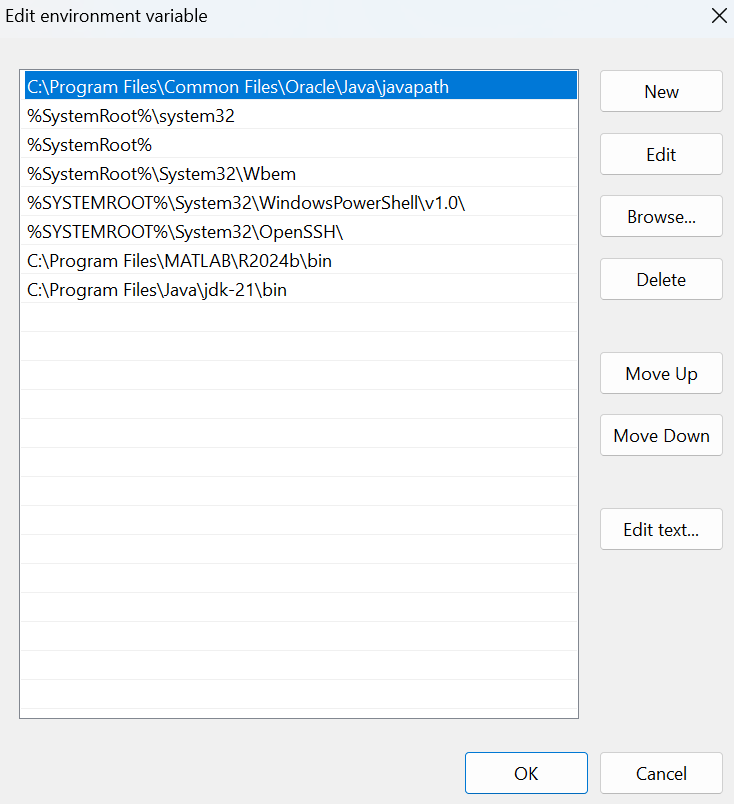
#### **Step 5: Set JAVA\_HOME**

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



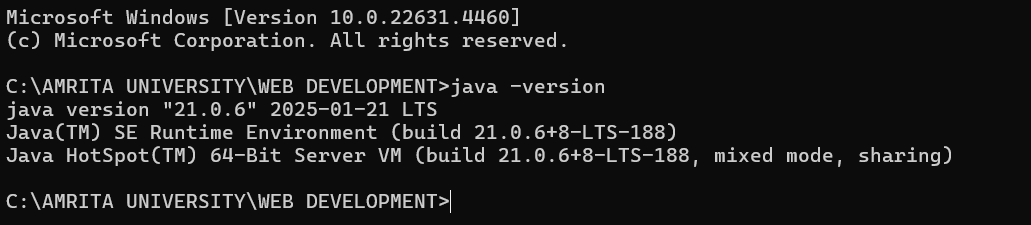
#### **Step 6: Update PATH Variable**

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

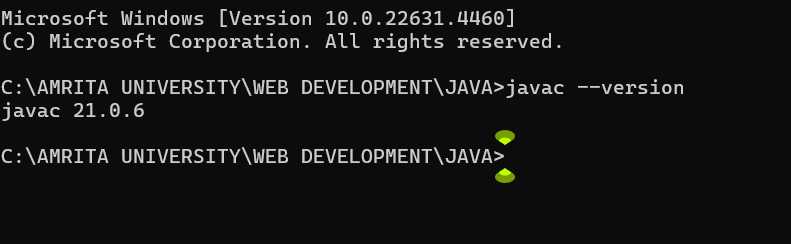


#### **Step 7: Verify Installation**

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



* To check the java compiler type: **javac --version.**

****

**Program-2:**

#### **Aim:**

Write a program that prints Helloworld.

#### **Code:**

class Helloworld {

public static void main(String[] args) { System.out.print("Hello World");

}

}

#### **OUTPUT:**

#### **POSITIVE CASE:**

#### 

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| None Found | None Rectified |

**Program-3:**

#### **Aim:**

Write a program that prints Student details.

#### **Code:**

class Main

{

public static void main(String[] args)

{

System.out.println("Name:N.Manikanta Gowtham");

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

System.out.println("Roll No:24231);

}

}

**OUTPUT:**



|  |  |
| --- | --- |
| Error Found | Error Rectified |
| The class name should start with capital letter. | I had changed the first letter from small to capital. |

### **WEEK-2**

#### **Program-1:**

#### **Aim:**

Write a program to calculate area of rectangle.

#### **Code:**

import java.util.Scanner; class Area1 {

public static void main(String[] args) { Scanner input = new Scanner(System.in); System.out.print("Enter length: ");

float length = input.nextFloat(); System.out.print("Enter width: "); float breadth = input.nextFloat(); input.close();

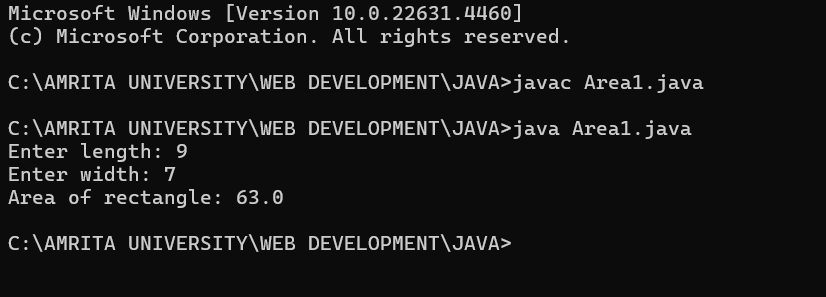
float area = length \* breadth; System.out.println("Area of rectangle: " + area);

}

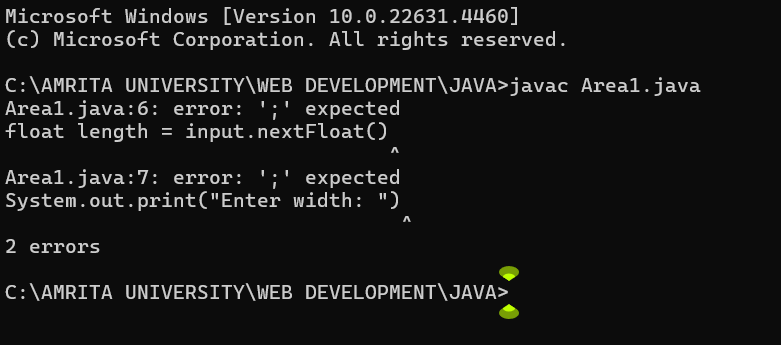
}

**OUTPUT:**

**POSITIVE CASE:**

****

## **NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Error: ‘;’ expected | Need to put ‘;’ at the end |

### **IMPORTANT POINTS :**

The program uses java.util.Scanner to take input from the user for: Length of the rectangle.

Breadth of the rectangle.

**Program-2:**

#### **Aim:**

Write a program to convert temperature from fahrenheit to Celsius and vice versa.

#### **Code:**

import java.util.Scanner; class Temperature1 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in); System.out.print("Enter Temperature in Fahrenheit: "); float F = input.nextFloat();

input.close();

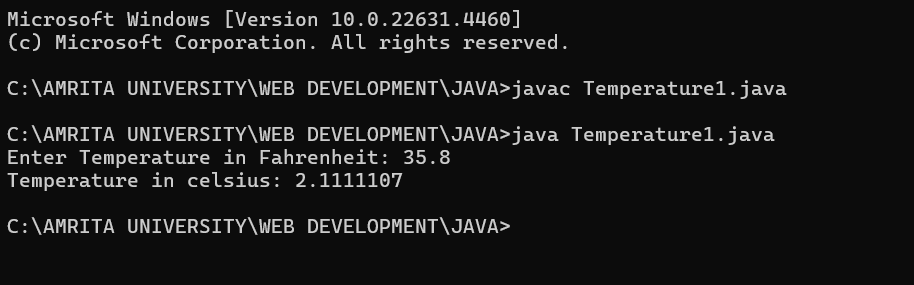
float C = (F - 32)\*5/9; System.out.println("Temperature in celsius: " + C);

}

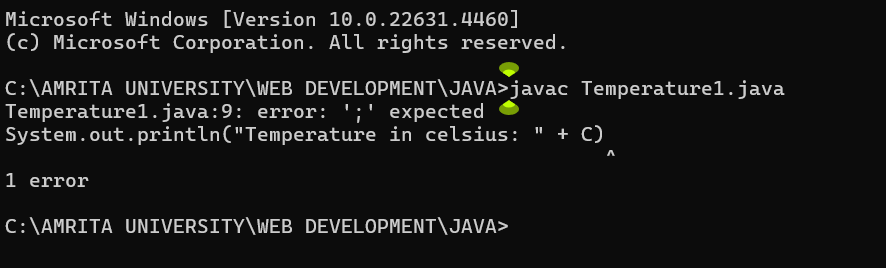
}

**OUTPUT:**

**POSITIVE CASE:**

****

## **NEGATIVE CASE:**

****

#### **Code:**

import java.util.Scanner; class Temperature2 {

public static void main(String[] args) { Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Celsius: "); float C = input.nextFloat();

input.close();

float F = (C \* 9/5) + 32; System.out.println("Temperature in Fahrenheit: " + F);

}

}

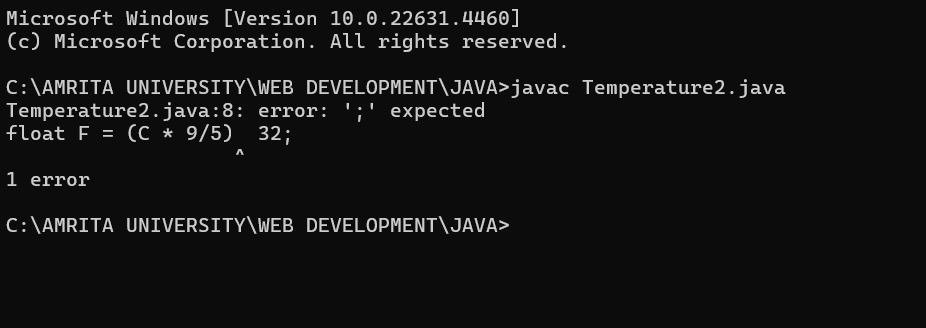
**:**

**OUTPUT:**

**POSITIVE CASE:**

****

## **NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| While printing the variable not giving + sign. | We should give correct indentation. |

### **IMPORTANT POINTS:**

The line “Scanner input = new Scanner(System.in),” tends to create a new Scanner object named “input” that reads input from the standard input stream (System.in), like keyboard.

**Program-3:**

#### **Aim:**

Write a program to calculate simple interest.

#### **Code:**

import java.util.Scanner; class Simpleinterest1 {

public static void main(String[] args) { Scanner input = new Scanner(System.in); System.out.print("Enter principle: "); float p = input.nextFloat(); System.out.print("Enter time: ");

float t = input.nextFloat(); System.out.print("Enter rate: "); float r = input.nextFloat(); input.close();

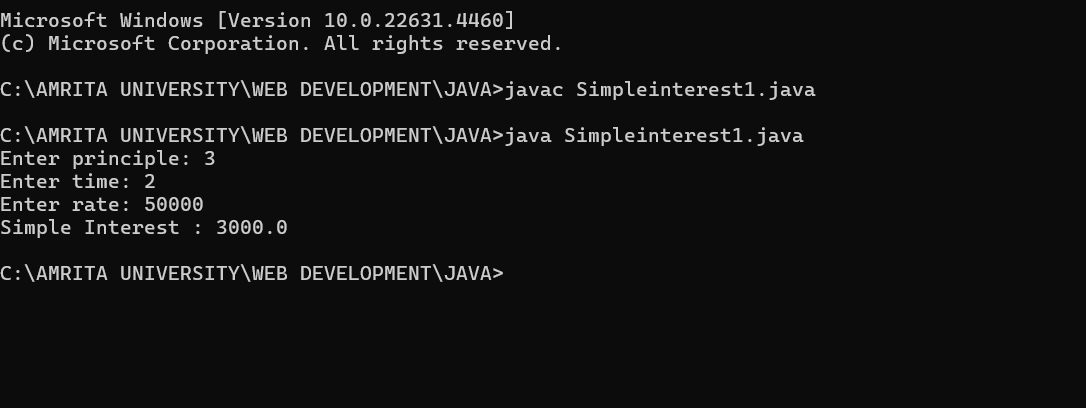
float SI = (p\*t\*r)/100; System.out.println("Simple Interest : " +SI );

}

}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

### **ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| The class name should start with capital letter  Syntax error; | I had changed the first letter from small to capital.  I was changed syntax |

### **IMPORTANT POINTS:**

line “import java.util.Scanner” indicates:

Import: tells the java compiler that we want to use a specific class in code.

Java.util : It contains utility classes for Java programming.

Scanner: It is the class that allows you to read input from the keyboard.

**Program-4:**

#### **Aim:**

Write a program to calculate Area of triangle.

#### **Code:**

public class Areaoftriangle1{

public static void main(String[] args) { double s1, s2, s3;

double area, resArea; s1 = 25.0;

s2 = 30.0;

s3 = 5.0;

area = (s1+s2+s3)/2.0d;

resArea = Math.sqrt(area\* (area - s1) \* (area - s2) \* (area -

s3));

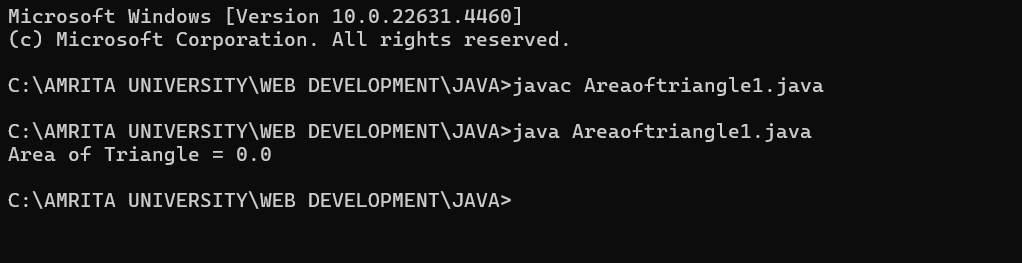
}

}

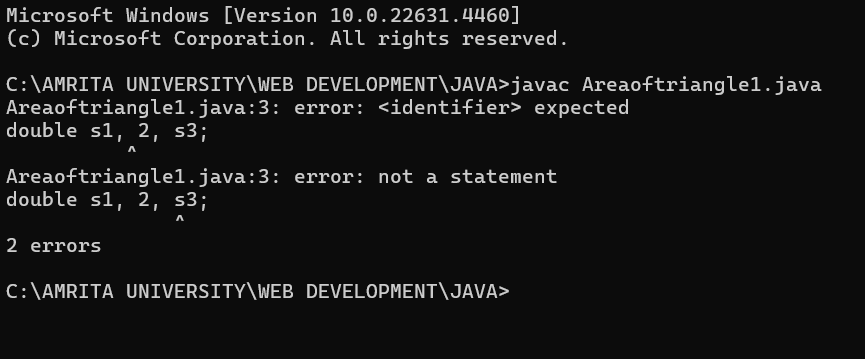
System.out.println("Area of Triangle = " + resArea);

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Error: Double s1, s2, s3;  ^ | Rectified: double s1,s2,s3; |

### **IMPORTANT POINTS:**

we’re finding the area of a triangle using heron’s formula. S is the semi-parameter of the triangle

**Program-5:**

#### **Aim:**

Write a program to calculate Factorial of a number.

#### **Code:**

import java.util.Scanner; public class Factorial1{

public static void main(String[] args) { Scanner input = new Scanner(System.in); System.out.print("Enter number: ");

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

long factorial = 1;

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

}

System.out.println("Factorial of " + n + " is: " + factorial);

}

}

**OUTPUT:**

**POSITIVE CASE: **

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| System.out.print(Enter number: ): “ ” ,’;’expected;  Syntax error | System.out.print(“Enter number:” );  Syntax rectified |

### **IMPORTANT POINTS:**

The factorial of n is calculated using a for loop.

we are using the data type “int” just to calculate the integer values and it doesn’t support floating points.

**Program-6:**

#### **Aim:**

Write a program to calculate Fibonacci series.

#### **Code:**

import java.util.Scanner; public class Fibonacci1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); System.out.print("Enter the number of terms: "); int terms = scanner.nextInt();

long firstTerm = 0, secondTerm = 1;

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

for (int i = 1; i <= terms; ++i) { System.out.print(firstTerm + " ");

long nextTerm = firstTerm + secondTerm; firstTerm = secondTerm;

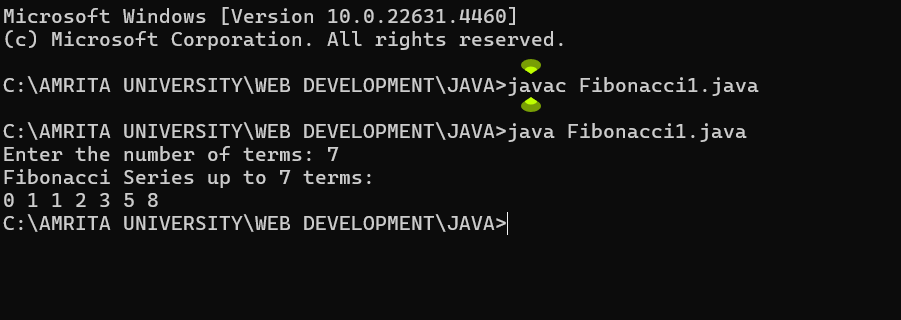
secondTerm = nextTerm;

}

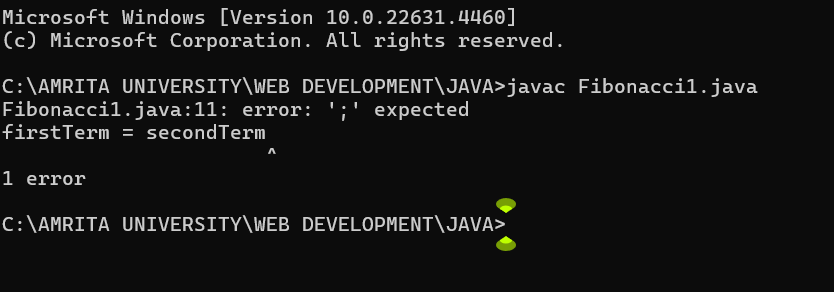
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Fibonacci.java:12: error: ';' expected | At the end firstTerm = secondTerm should have; |

# **IMPORTANT POINTS:**

We use while loop in this program.

The process in this program is repeated certain number of times until the conditions meet.

# **WEEK-3**

### **1. AIM:**

**To create java program with following instructions:**

Create a class with a name car.

Create four attributes named car\_color, car\_brand, fuel\_type, mileage. Create three methods named start(), stop(), service().

Create the objects named car1, car2, car3.

**CLASS DIAGRAM :**

|  |
| --- |
| **CAR** |
| car\_brand : String  car\_colour : String  fuel\_type : String  mileage : int |
| + car (String, String, String,int)  +start() : void  +stop() : void  +service() : void  + car\_details : void |

#### **Code:**

class car {

String car\_brand;

String car\_colour;

String fuel\_type;

int mileage;

public car(String car\_brand,String car\_colour,String fuel\_type,int mileage){

this.car\_brand = car\_brand;

this.car\_colour = car\_colour;

this.fuel\_type = fuel\_type;

this.mileage = mileage;

}

public void start() {

System.out.println("CAR IS STARTED");

}

public void stop() {

System.out.println("CAR IS STOPPED");

}

public void service() {

System.out.println("CAR IS I SERVICE");

}

public void car\_details() {

System.out.println("CAR BRAND IS :" + car\_brand);

System.out.println("CAR COLOUR IS :" + car\_colour);

System.out.println("CAR FUEL TYPE IS :" + fuel\_type);

System.out.println("CAR MILEAGE IS :" + mileage);

}

public static void main(String[] args) {

car c1 = new car("MCLAREN","GREEN","PETROL",12);

car c2 = new car("FERRARI","RED","PETROL",18);

car c3 = new car("LAMBORGHINI","ORANGE","PETROL",20);

c1.car\_details();

System.out.println(" ");

c2.car\_details();

System.out.println(" ");

c3.car\_details();

}

}class car {

String car\_brand;

String car\_colour;

String fuel\_type;

int mileage;

public car(String car\_brand,String car\_colour,String fuel\_type,int mileage){

this.car\_brand = car\_brand;

this.car\_colour = car\_colour;

this.fuel\_type = fuel\_type;

this.mileage = mileage;

}

public void start() {

System.out.println("CAR IS STARTED");

}

public void stop() {

System.out.println("CAR IS STOPPED");

}

public void service() {

System.out.println("CAR IS I SERVICE");

}

public void car\_details() {

System.out.println("CAR BRAND IS :" + car\_brand);

System.out.println("CAR COLOUR IS :" + car\_colour);

System.out.println("CAR FUEL TYPE IS :" + fuel\_type);

System.out.println("CAR MILEAGE IS :" + mileage);

}

public static void main(String[] args) {

car c1 = new car("MCLAREN","GREEN","PETROL",12);

car c2 = new car("FERRARI","RED","PETROL",18);

car c3 = new car("LAMBORGHINI","ORANGE","PETROL",20);

c1.car\_details();

System.out.println(" ");

c2.car\_details();

System.out.println(" ");

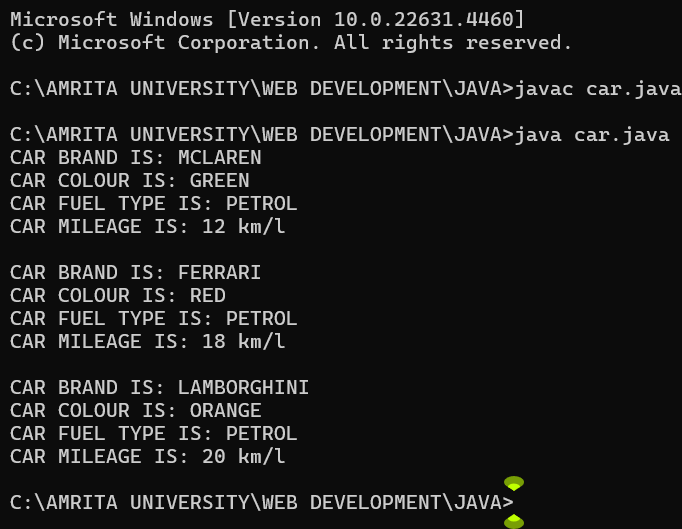
c3.car\_details();

}

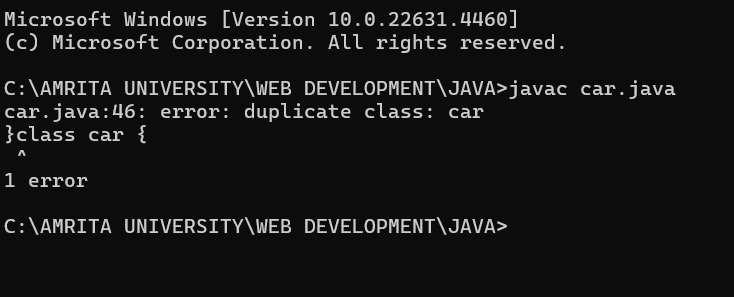
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | Runtime error | Incorrect symbol in main program | { symbol is added |
| 2 | Logical error | Incorrect logic | Correct logic |

### **IMPORTANT POINTS:**

The program defines a car class that represents a car's attributes and behaviors.

The class contains four attributes. Three methods define car behaviors.

Three car objects (car1,car2,car3) are created with different attributes.

### **2)AIM:**

To create a class bankaccount with methods deposit() and withdrawl

**CLASS DIAGRAM :**

|  |
| --- |
| bank |
| name:String  number:String  IFSC:String  branch:String  balance:int |
| bank(name:String , number:String, IFSC:String, +branch:String, balance:int)  +bank\_details():void  +deposit(amount:int):void  +withdrawl(amount:int):void |

#### **Code:**

class Bank {

String name;

String number;

String IFSC;

String branch;

int balance;

public Bank(String name, String number, String IFSC, String branch, int balance) {

this.name = name;

this.number = number;

this.IFSC = IFSC;

this.branch = branch;

this.balance = balance;

}

public void bankDetails() {

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

System.out.println("Account Number: " + number);

System.out.println("IFSC Code: " + IFSC);

System.out.println("Branch: " + branch);

System.out.println("Current Balance: " + balance);

}

public void deposit(int amount) {

if (amount > 0) {

balance += amount;

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

System.out.println("Total Balance: " + balance);

} else {

System.out.println("ERROR: Invalid deposit amount.");

}

}

public void withdrawal(int amount) {

if (amount <= balance) {

balance -= amount;

System.out.println("Withdrawn Amount: " + amount);

System.out.println("Updated Balance: " + balance);

} else {

System.out.println("INSUFFICIENT BALANCE");

}

}

public static void main(String[] args) {

Bank c1 = new Bank("Rahul", "1234567890", "SBI1997", "GUNTUR", 0);

Bank c2 = new Bank("Iyer", "0987654321", "SBI1996", "GUNTUR", 0);

System.out.println("CUSTOMER 1 DETAILS\n");

c1.bankDetails();

System.out.println("\nCUSTOMER 1 DEPOSIT");

c1.deposit(1000);

System.out.println("\nCUSTOMER 1 WITHDRAW");

c1.withdrawal(500);

System.out.println("\n-----------------------------\n");

System.out.println("CUSTOMER 2 DETAILS\n");

c2.bankDetails();

System.out.println("\nCUSTOMER 2 DEPOSIT");

c2.deposit(5000);

System.out.println("\nCUSTOMER 2 WITHDRAW");

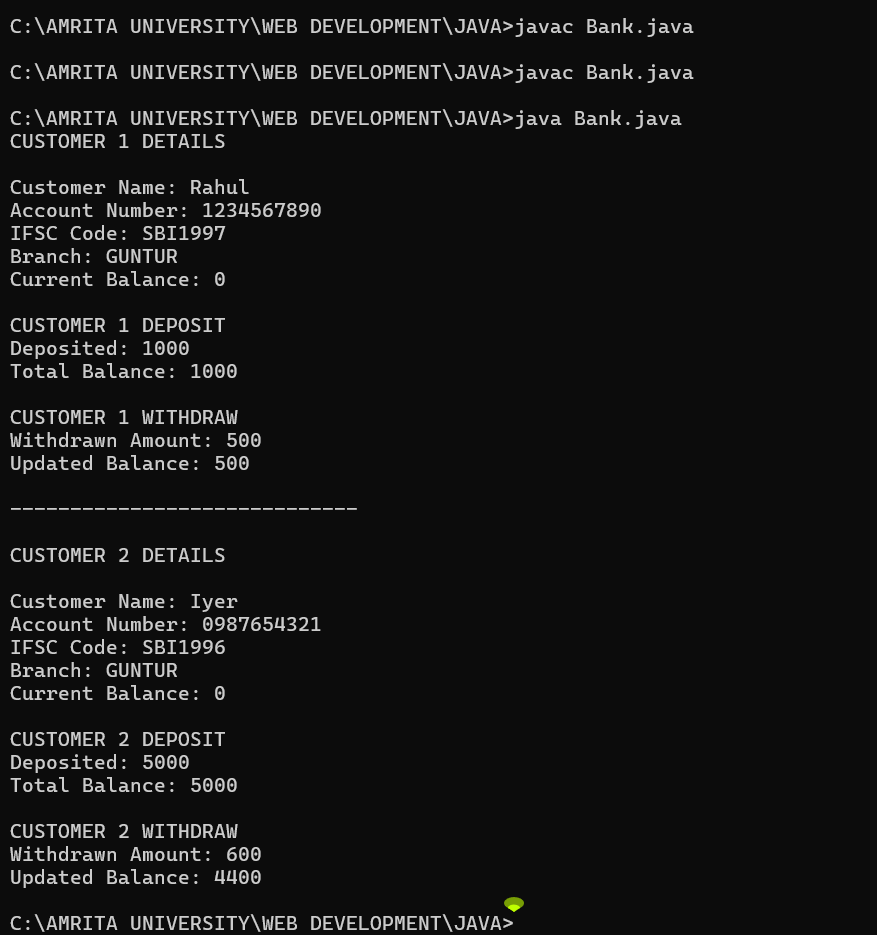
c2.withdrawal(600);

}

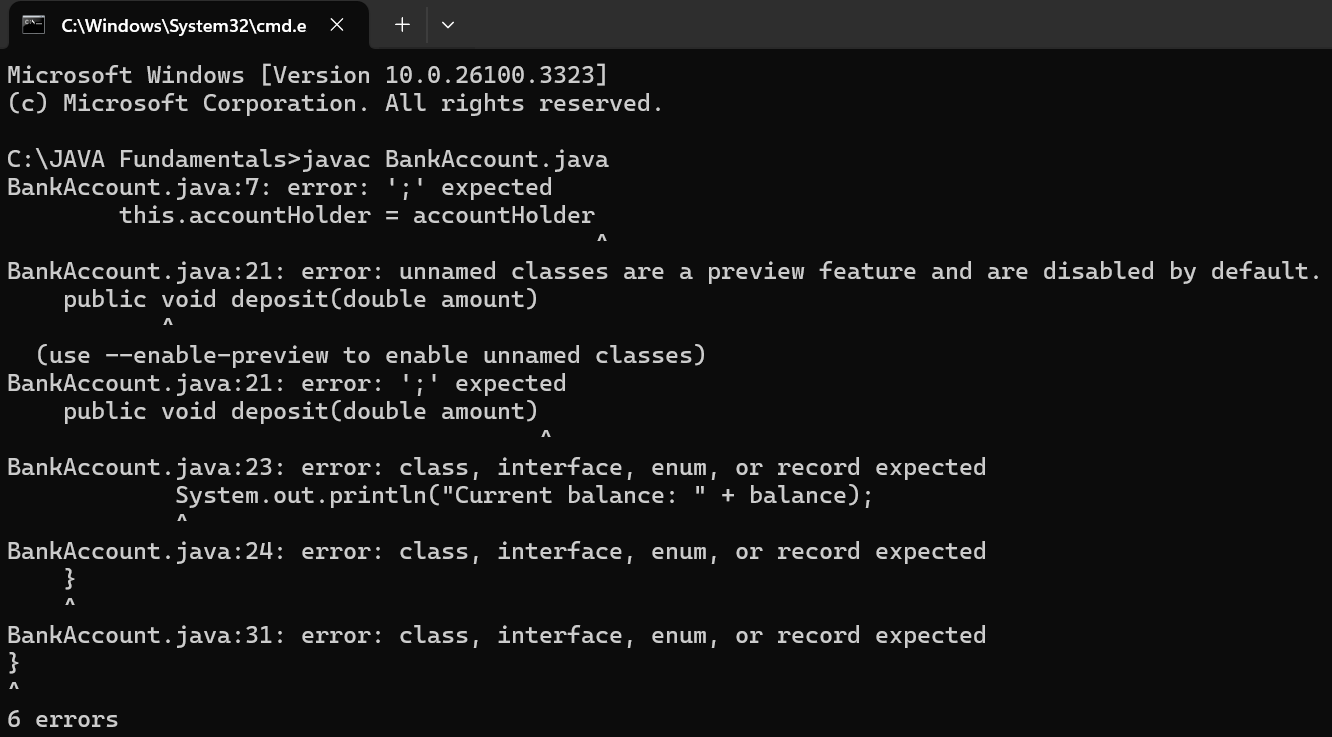
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| 1 | syntax error | Forgot to keep main word in main program | Main is added |
| 2 | Logical error | Incorrect logic | Correct logic |

### **IMPORTANT POINTS:**

The condition inside the if statement must be correct.

It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amoun

### **WEEK-4**

#### **Program-1:**

#### **Aim:**

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

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

#### **CLASS DIAGRAM:**

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

#### **Code:**

class Book {

public String bookTitle; public String bookAuthor;

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

book1.bookYearOfPublication = 1997-2007; 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 = "Think And Grow Rich"; book2.bookAuthor = "Napoleon Hill"; book2.bookYearOfPublication = 1937; 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: " +

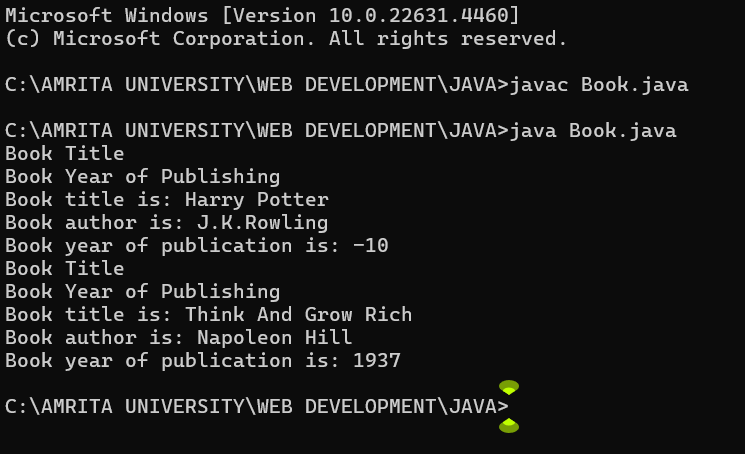
book2.bookYearOfPublication);

}

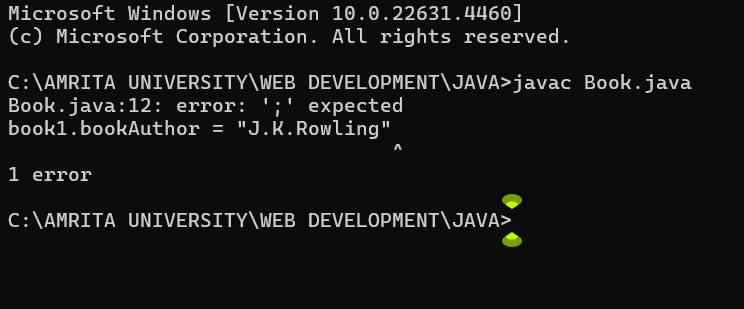
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Book.java:12: error: ';' expected public static void main(String[] args)  ^ | public static void main(String[] args) { |

### **IMPORTANT POINTS**:

While defining a method we should also define a function to call that method.

**Program – 2**:

**AIM**:

Create a java Program with class named myclass with static variable count of int type, initialized to zero and a constant variable “pi” of type double initialized to 3.14 as attributes of the class, ow define a constructor for “myclass” that increments the count variable each time an object of my class is created (count++), finally print the final values of count and pi variables create three objects.

**Class Diagram:**

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

#### **Code:**

public class Myclass { static int count = 0; final double pi = 3.14; public Myclass() { count++;

}

public static void main(String[] args) { Myclass obj1 = new Myclass(); Myclass obj2 = new Myclass(); Myclass obj3 = new Myclass(); System.out.println("count: " + count);

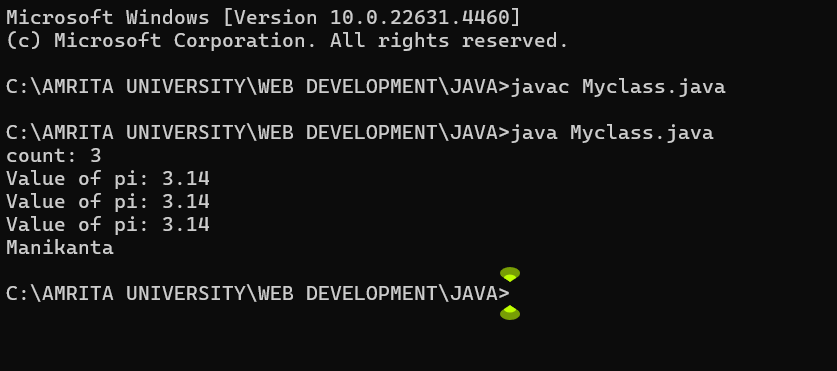
System.out.println("Value of pi: " + obj1.pi); System.out.println("Value of pi: "+ obj2.pi); System.out.println("Value of pi: " + obj3.pi); System.out.println("Manikanta");

}

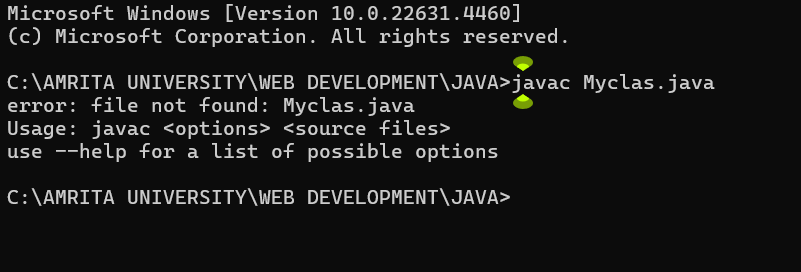
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Not giving the indentation properly. | All the indentation must be correct to run the code correctly. |

### **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 increasin

### **WEEK-5**

#### **Program – 1:**

#### **AIM:**

Create a calculator using the operations including addition, subtraction, multiplication, inheritance, and display the desired output.

#### **CLASS DIAGRAM:**

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

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

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

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

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

**Code:**

import java.util.Scanner; class Calculator {

protected double a, b;

public Calculator(double a, double b) { this.a = a;

this.b = b;

}

public double add() { return a + b;

}

public double subtract() { return a - b;

}

public double multiply() { return a \* b;

}

public double divide() { if (b != 0) {

return a / b;

} else {

System.out.println("Error: Division by zero is not allowed.");

return Double.NaN;

}

}

public void displayResults() { System.out.println("Addition: " + add()); System.out.println("Subtraction: " + subtract()); System.out.println("Multiplication: " + multiply()); System.out.println("Division: " + divide());

}

}

public class AllCalculator {

public static void main(String[] args) { Scanner input = new Scanner(System.in); System.out.print("Enter a number: "); double a = input.nextDouble(); System.out.print("Enter b number: "); double b = input.nextDouble();

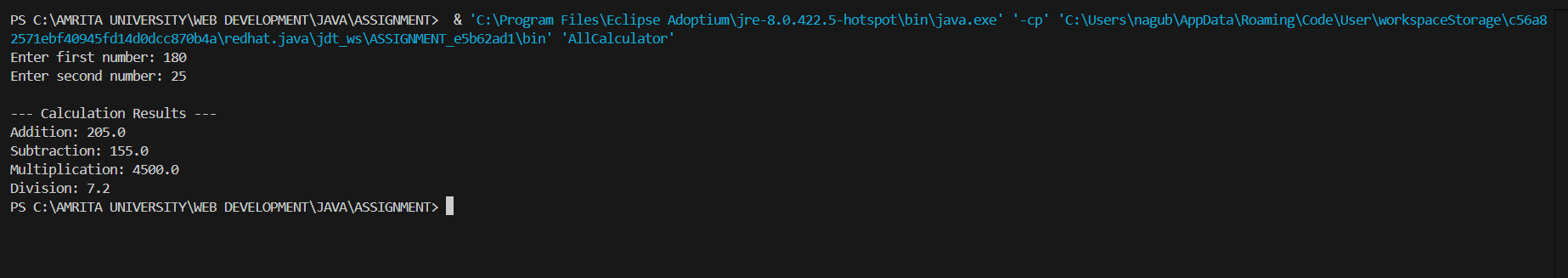
Calculator calc = new Calculator(a, b); calc.displayResults(); System.out.println("Manikanta"); input.close();

}

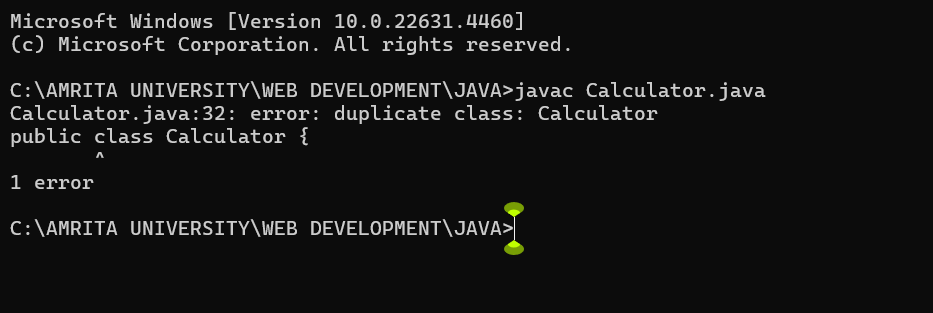
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Not mentioning super to obtain the super class constructor. | To obtain the super class we need to mention super. |

### **IMPORTANT POINTS:**

1. To get the inputs from the user we use import java.util.Scanner; this is a package.
2. Scanner class is used to get the user input.
3. In java.util.Scanner, the java.util is a package while Scanner is a class of the java.util package.

**Program – 2:**

### **AIM:**

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

1. cars should have an additional property: no of doors
2. Bikes should have a property indicating whether they have gears or not.
3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.
4. Every class should have a constructor Question:
   1. Which oops concept is used in the above program
   2. If the company decides to add a new type of vehicle, Truck, how would you modify the program?
      1. Truck should include an additional property capacity (in tons)
      2. Create a show Truck details () method to display the truck’s capacity.
      3. 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 sub classes Finally, display the details.

**CLASS DIAGRAM:**

|  |
| --- |
| Bike |
| -has gears: bool |
| +int (brand, speed, has gears)  +displaydetails() |

|  |
| --- |
| Car |
| - no. of doors: int |
| +int (brand, speed displaydetails(), no. of doors)  +displaydetails() |

|  |
| --- |
| Truck |
| -capacity: float |
| +displaydetails()  +showtruckdetails() |

#### **Code:**

public class Rent {

public static void Main(String[] args) { Car car = new Car("Toyota", 150, 4);

Bike bike = new Bike("Yamaha", 120, true); Truck truck = new Truck("Volvo", 90, 10); System.out.println("Car Details:"); car.start();

car.showDetails(); System.out.println("Bike Details:"); bike.start();

bike.showDetails(); System.out.println("Truck Details:"); truck.start();

truck.showDetails();

}

}

class Vehicle {

public String brand;

public int speed;

public Vehicle(String brand, int speed) { this.brand = brand;

this.speed = speed;

}

public void start() { System.out.println(brand + " is starting");

}

public void showDetails() { System.out.println("Brand: " + brand); System.out.println("Speed: " + speed + " km/h");

}

}

class Car extends Vehicle { private int noOfDoors;

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

this.noOfDoors = noOfDoors;

}

public void showDetails() { super.showDetails();

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

}

}

class Bike extends Vehicle {

private boolean hasGears;

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

this.hasGears = hasGears;

}

public void showDetails() { super.showDetails();

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

}

}

class Truck extends Vehicle { private int capacity;

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

this.capacity = capacity;

}

public void showDetails() { super.showDetails();

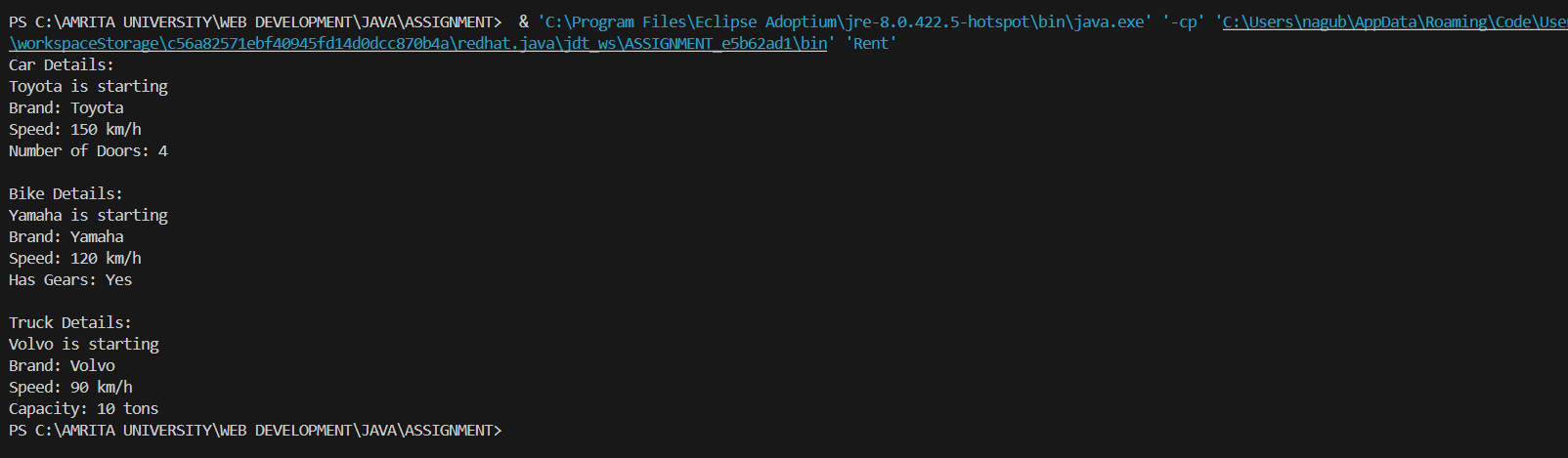
System.out.println("Capacity: " + capacity + " tons");

}

}

**OUTPUT:**

**POSITIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Declaring two super classes inside the same file. | Make two separate files to save the two super classes. |

### **IMPORTANT POINTS:**

1. a constructor helps in initializing an object that doesn't exist.
2. a method performs functions on pre-constructed or already developed objects.
3. The void keyword in java is used to specify that a method does not return any value. it is a return type that indicates the method performs a function and doesn't produce a result.

#### **Answers:**

1. The oops concepts used in the above program are:

Inheritance, encapsulation, polymorphism, abstraction.

1. To add a new vehicle type truck we need to create a truck class that will:
   1. Include an additional property capacity (in tons).
   2. Implement a showTruckdetials() method to display the truck's capacity.
   3. Implement a constructor for the truck class to initialize all its properties.

### **WEEK-6**

#### **Program – 1:**

#### **AIM:**

Write a java program to create a vehicle class with a method displayinfo(). Override this method in the car subclass to provide specific information about a car.

**CLASS DIAGRAM:**

|  |
| --- |
| CLASS VEHICLE |
| ATTRIBUTES :  + SPEED  + BRAND  + CAPACITY |
| +DISPLAYINFO():void  + VEHICLE (INT ,SPEED INT CAPACITY ,STRING BRAND) |

|  |
| --- |
| class car |
| car(int speed ,int capacity ,string brand)  displayinfo() : void |

#### **Code:**

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();

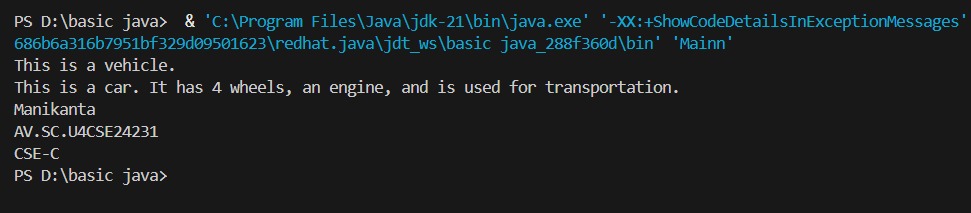
car1.displayInfo("Audi","Model","Diesel",4);

}

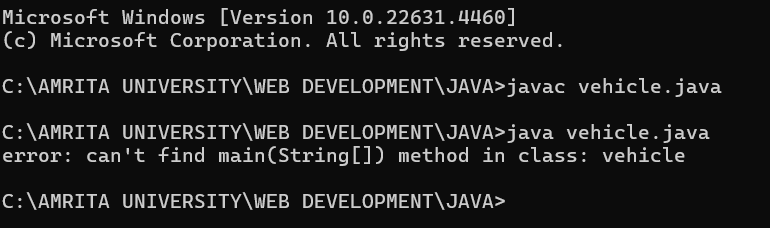
}

**OUTPUT:**

**POSITIVE CASE:**



**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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().

**Program – 2:**

### **AIM:**

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 qualifications.

Qualifications:

* UG admissions require a minimum of 60%.
* PG admissions require a minimum of 70%.

**CLASS DIAGRAM:**

|  |
| --- |
| University |
| - name: String  - percentile: int |
| + University(String, int)  + office(): void |

|  |
| --- |
| PG |
| + PG(String, int)  + office(): void |

|  |
| --- |
| UG |
| + UG(String, int)  + office(): void |

|  |
| --- |
| admissions |
| + main(String[]): void |

#### **Code**:

import java.util.Scanner; public class Admission {

public static void main(String[]args) { Scanner input = new Scanner(System.in);

System.out.println("Manikanta\nAV.SC.U4CSE24231\nCSE-C ");

System.out.println("Enter student name: "); String name = input.nextLine();

System.out.println("Enter qualification percentage: "); double percentage = input.nextDouble(); input.nextLine();

System.out.println("Enter program: ");

String program = input.next().toUpperCase(); if (program.equals("UG")) {

if (percentage>=60) {

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

} else {

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

}

} else if (program.equals("PG")) { if (percentage>= 70) {

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

} else {

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

}

} else {

System.out.println("Invalid course entered.");

}

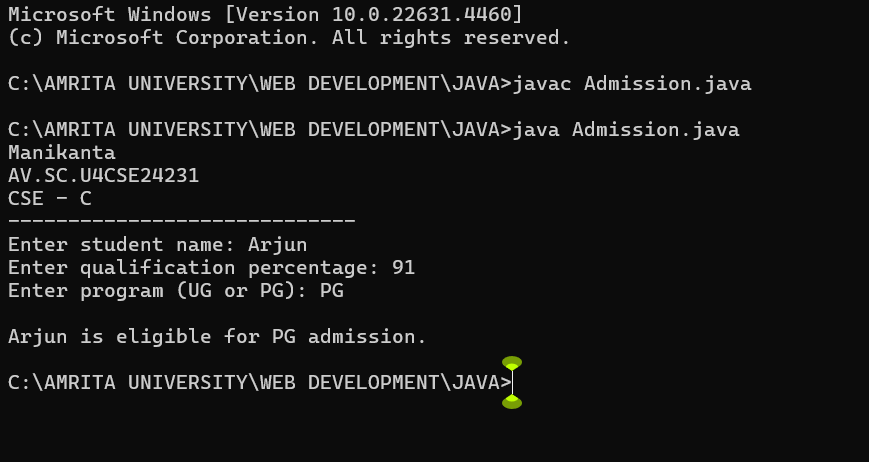
input.close();

}

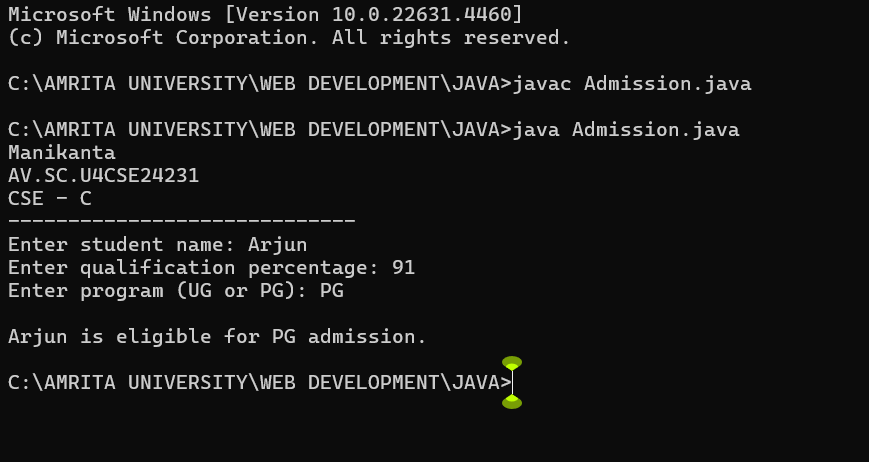
}

#### **OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| No errors | No errors |

**IMPORTANT POINTS:**

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

1. **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.

**Program – 3:**

### **AIM:**

Create a calculator class with overloaded methods to perform addition.

* Add two integers.
* Add two doubles.
* Add three integers.

#### **Code:**

class Calculator {

public void addition(int a, int b) { System.out.println("Addition of two integers: " + (a + b));

}

}

class Calci2 extends Calculator {

public void addition(double a, double b) { System.out.println("Addition of two doubles: " + (a + b));

}

}

class Calci3 extends Calci2 {

public void addition(int a, int b, int c) { System.out.println("Addition of three integers: " + (a + b + c));

}

}

public class Cal {

public static void main(String[] args) {

Calci3 c = new Calci3(); c.addition(8.5, 9.5);

c.addition(3, 4);

c.addition(1, 2, 3);

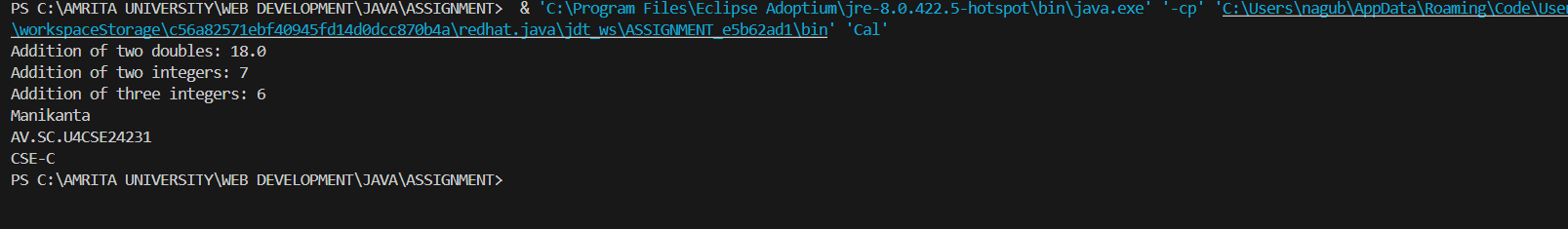
System.out.println("Manikanta\n AV.SC.U4CSE24231\n CSE-C");

}

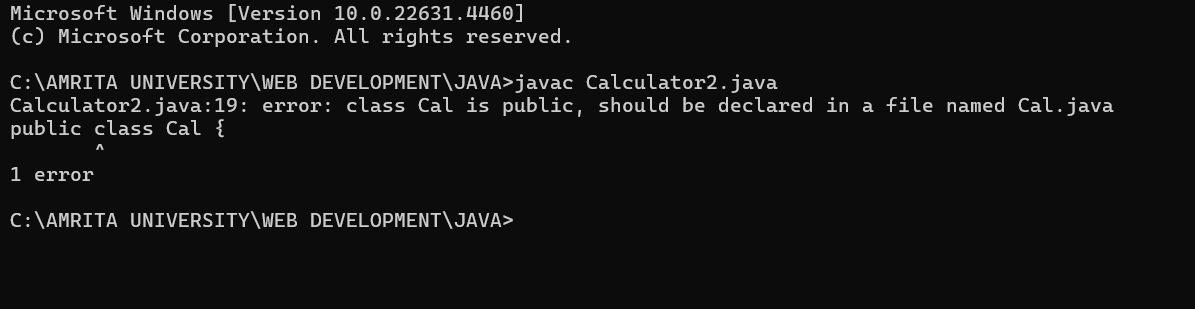
}

#### **OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| Cal.java:19: error: ')' or ',' expected  c.addition( 1, 2, c:); | Assign a value to c. |

**Program – 4:**

### **AIM:**

Create a shape class with a method calculatorArea() that is over loaded for different shapes (eg : square, rectangle) then, create a subclass circle that overrides the calculateArea()method for a circle.

**Claas diagram :**

|  |
| --- |
| **Class shapes** |
| **+ calculateArea(int, int): void**  **+ calculateArea(double, double): void**  **+ calculateArea(int): void** |

|  |
| --- |
| Class circle |
| + calculateArea(double): void |

|  |
| --- |
| Area |
| + main(String[]): void |

#### **Code:**

class Shape {

void calculateArea( int a) {

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

}

void calculateArea(int a , int b) {

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

}

}

class Circle extends Shape { void calculateArea(double a){

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

} }

public class ShapeTest {

public static void main(String[] args) { Shape s = new Shape();

Circle c = new Circle(); s.calculateArea(6); System.out.println(" ");

s.calculateArea(10,5);

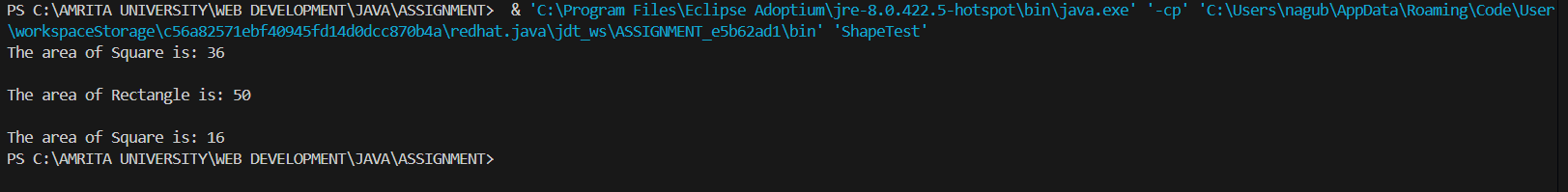
System.out.println(" "); c.calculateArea(4);

}

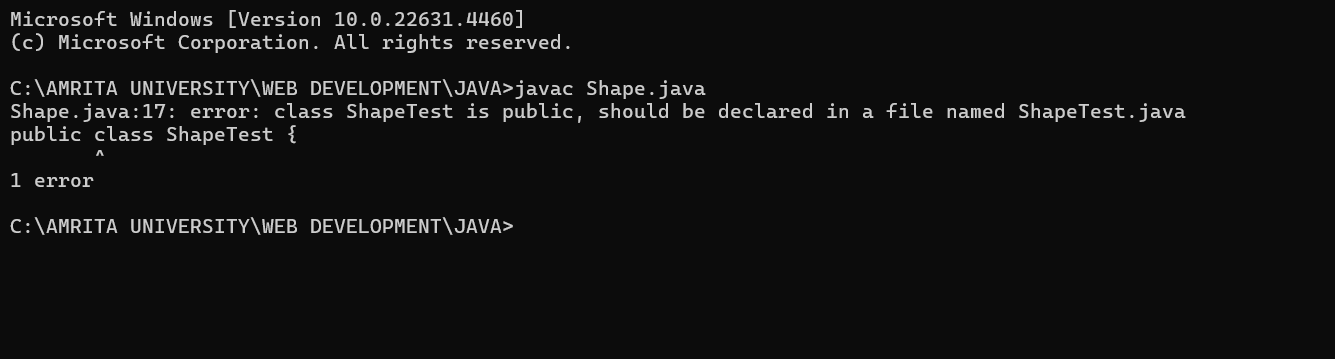
}

**OUTPUT:**

**Positive Case:**

****

**Negative Case:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| Error Found | Error Rectified |
| ShapeTest.java:17: error:  public class ShapeTest {  1 error | Rectified: class ShapeTest{ |

**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.

**WEEK-7**

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

**CLASS DIAGRAM:**

|  |
| --- |
| < ABSTACT CLASS >  Animal |
| + sound() : void |

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

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

**PROGRAM:**

abstract class Animal {

    abstract void sound();

}

class Lion extends Animal {

    void sound() {

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

    }

}

class Tiger extends Animal {

    void sound() {

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

    }

}

class Sound {

    public static void main(String[] args) {

        Lion l = new Lion();

        Tiger t = new Tiger();

        System.out.println("Name : N.Manikanta Gowtham"  + "Roll No : AV.SC.U4CSE24231" + "Section : CSE-C");

        System.out.println("    ");

        l.sound();

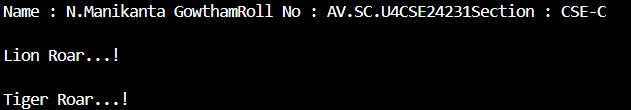
        System.out.println("    ");

        t.sound();

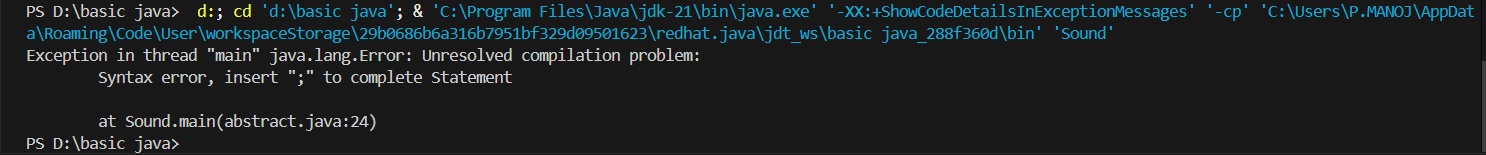
    }

}

**OUTPUT:**

****

**NEGATIVE OUTPUT:**

****

**ERROR TABLE:**

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

**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 :**

import java.math.\*;

abstract class Shapes3D {

    abstract void CalculateVolume();

    abstract void CalculateSurfaceArea();

}

class Sphere extends Shapes3D {

    int radius;

    Sphere(int radius) {

        this.radius = radius;

    }

    void CalculateVolume() {

        System.out.println("Volume of sphere is: " + (4.0 / 3.0) \* Math.PI \* radius \* radius \* radius);

    }

    void CalculateSurfaceArea() {

        System.out.println("Surface area of sphere is: " + 4 \* Math.PI \* radius \* radius);

    }

}

class Cube extends Shapes3D {

    int side;

    Cube(int side) {

        this.side = side;

    }

    void CalculateVolume() {

        System.out.println("Volume of cube is: " + side \* side \* side);

    }

    void CalculateSurfaceArea() {

        System.out.println("Surface area of cube is: " + 6 \* side \* side);

    }

}

public class Shapes18 {

    public static void main(String[] args) {

        Sphere sp = new Sphere(5);

        Cube c = new Cube(4);

        System.out.println("Name :N.Manikanta Gowtham  Section : CSE-C  Roll no : AV.SC.U4CSE24231");

        System.out.println("    ");

        sp.CalculateSurfaceArea();

        System.out.println("    ");

        sp.CalculateVolume();

        System.out.println("    ");

        c.CalculateSurfaceArea();

        System.out.println("    ");

        c.CalculateVolume();

    }

}

CLASS DIAGRAM

|  |
| --- |
| <abstract>>  Shape3D |
| +calculateVolume(): double +calculateSurfaceArea(): double |

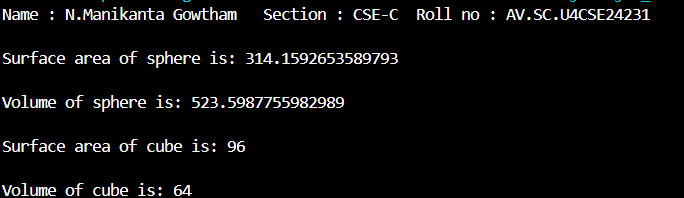
|  |
| --- |
|  |

|  |
| --- |
| CUBE |
| - a: int |
| +calculateVolume()  +calculateSurfaceArea() |

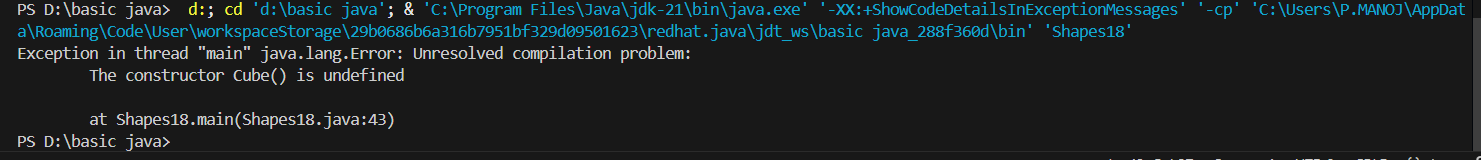
|  |
| --- |
| Sphere |
| - radius: int |
| +calculateVolume() +calculateSurfaceArea() |

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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:N.Manikanta Gowtham Roll no:AV.SC.U4CSE24231  Section:CSE-C");

        System.out.println("    ");

        sp.displayTitle("Star Pattern");

        sp.printPattern(5);

        System.out.println("    ");

        np.displayTitle("Number Pattern");

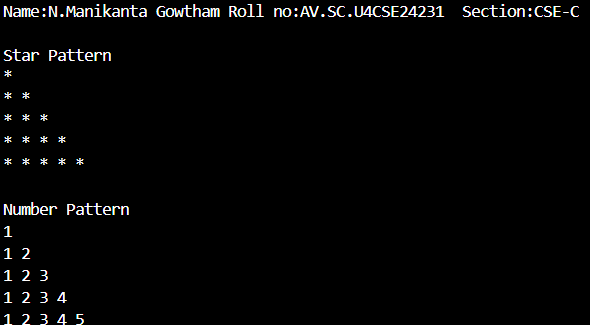
        np.printPattern(5);

    }

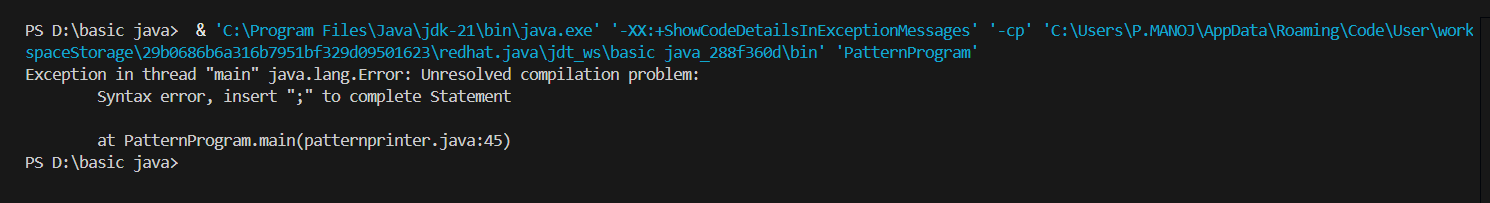
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

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

    public Rectangle(double length, double width) {

        this.length = length;

        this.width = width;

    }

    public double getPerimeter() {

        return 2 \* (length + width);

    }

}

class Circle implements Shape {

    private double radius;

    public 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;

    public 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 Perimeter {

    public static void main(String[] args) {

        Shape rectangle = new Rectangle(5, 3);

        Shape circle = new Circle(4);

        Shape triangle = new Triangle(3, 4, 5);

        System.out.println("Name:N.Manikanta Gowtham Roll no:AV.SC.U4CSE24231  Section:CSE-C");

        System.out.println("    ");

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

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

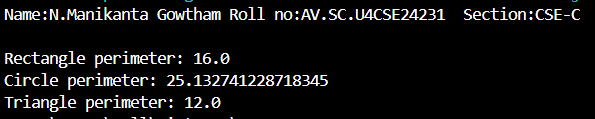
        System.out.println("Triangle perimeter: " + triangle.getPerimeter());

    }

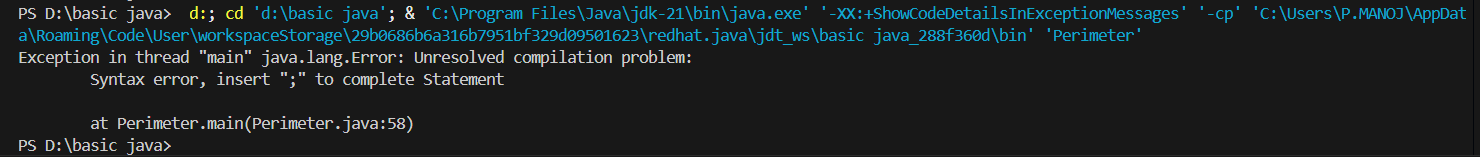
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR** **TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 {

    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 {

    public void play() {

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

    }

}

class Volleyball implements Playable {

    public void play() {

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

    }

}

class Basketball implements Playable {

    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();

        System.out.println("Name: N.Manikanta Gowtham Roll no: AV.SC.U4CSE24231  Section:CSE-C");

        System.out.println("    ");

        football.play();

        System.out.println("    ");

        volleyball.play();

        System.out.println("    ");

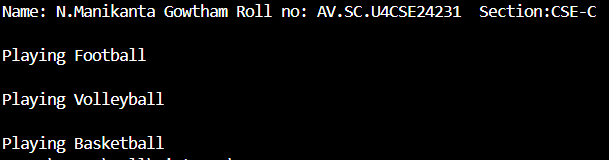
        basketball.play();

    }

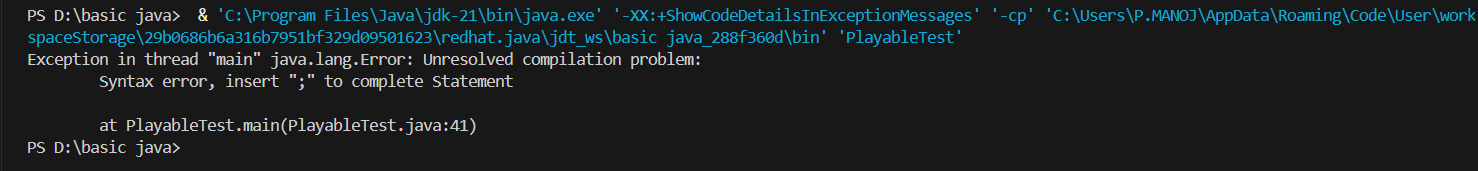
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |

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

**PROGRAM:**

interface LoginSystem {

    boolean Login(String ID, int pass);

}

class CollegePortal implements LoginSystem {

    public boolean Login(String ID, int pass) {

        if ((ID=="Manikanta") && (pass==24231)){

            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("Name: N.Manikanta Gowtham Roll no: AV.SC.U4CSE24231  Section:CSE-C");

        System.out.println("    ");

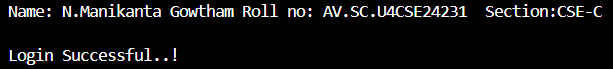
        CP.Login("Manikanta", 24231);

    }

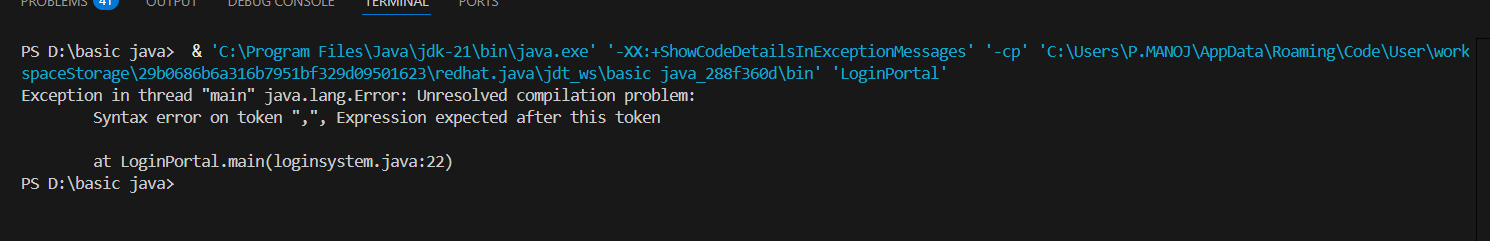
}

**OUTPUT:**

**POSITIVE CASE:**

****

**NEGATIVE CASE:**

****

**ERROR TABLE:**

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