**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB MANUAL**

****

**Department of Computer Science and Engineering Amrita School of computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Verified by:** NAME: M. BINDHU SREE

ROLLNO: AV.SC.U4CSE24226

INDEX

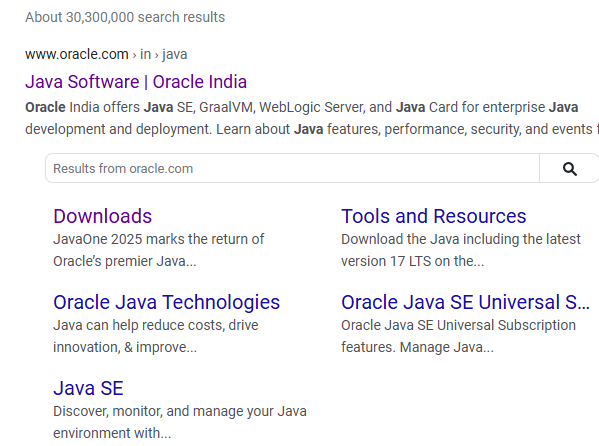
|  |  |  |
| --- | --- | --- |
|  | EXPERIMENTS | REMARKS |
| WEEK-1 |  |  |
| 1) | Downloading and installation of java |  |
| 2) | Write a java program that prints name, roll no, section of a student. |  |
| WEEK-2 |  |  |
| 1) | Write a java program to find the simple interest where all the inputs are taken from the user |  |
| 2) | Write a java program to find the fibonacci sequence of a given number |  |
| 3) | Write a java program to find the area of rectangle |  |
| 4) | Write a java program to find the area of triangle |  |
| 5) | Write a java code to convert the temperature from celsius to fahrenheit and from fahrenheit to celsius. |  |
| 6) | Write a java code to convert the temperature from celsius to fahrenheit and from fahrenheit to celsius. |  |
| WEEK-3 |  |  |
| 1) | To create java program with following instructions   1. create a class with name car. 2. create four attributes named car\_color, car\_brand, fuel\_type, mileage. 3. create three methods named start(), stop(), service(). 4. create three methods named car1, car2, car3. |  |
| 2) | To create a class bankaccount with methods deposit() and withdrawl |  |
| WEEK-4 |  |  |
| 1) | Write a java program with class named “book”. the class should contain various attributes such as “title of the book”, “author”, year of publications”, it should also contain a constructor with parameter which initializes “title of the book”, “author”, “year of publication”. create a method which displays the details of the book. title of the book (), author (), year of publication ().display the details of two book, by creating 2 objects |  |
| 2) | To create a java program with class named “myclass” with a static variable count of “int” type, in sized to “zero” and a constant variable “pi” of type “double” initialized to 3.1415 as attributes of that class. define a contractor for “myclass” is created finally print the final values “count” and “pi” variables. |  |
| WEEK-5 |  |  |
| 1) | Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output.Hint: collect required variables using super class, Create each class for a parameter and each class must contain a method. |  |
| 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  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?  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  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 |  |
| 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 a About car. |  |
| 2) | A college is developing an automated admissions systems 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 qualification.  1. UG admission require min of 60%  2. PG admission require min of 70% |  |
| 3) | To create a Java Program with class named “my class” with a Static Variable Count int type and initialize to 0 and A Constant Variable "pi" of type double initialized to 3.1415 has attributes of that class. Now defi a Constructor for my class that increments the Count Variable each time an object of my class is created. Finaly Print the final values of count. |  |
| 4) | Write a Java Program and create a Shape class with a method calcArea(). That is overloaded for different shapes like square and rectangle. Create a sub class circle that overrides the calcArea() for a circle. |  |
| WEEK-7 |  |  |
| 1) | Write a java program to create an abstract class animal with an abstract method called sound(). Create subclasses Lion 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 PatternPrinter with an abstract method printPattern(int n) and a concrete method to display the pattern title.  Implement two subclasses:  1.StarPattern- Prints a right angled triangle stars  2.NumberPateern-Prints a right angled triangle increasing number  IN the main() method create objects of both subclasses and print the pattern for a given number of rows. |  |
| 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. Create three 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. |  |

**WEEK-1**

1. AIM:How to download and install java software in your computer.

**1.Steps for downloading.**

Step 1: Java is developed by oracle. so, open your web browser and search for java oracle.



Step 2: Then go to the website <https://www.oracle.com/in/java/> .

Step 3: And navigate to the java downloads. Then some earlier versions are available like

* [**JDK 23**](https://www.oracle.com/in/java/technologies/downloads/#java23)
* [**JDK 21**](https://www.oracle.com/in/java/technologies/downloads/#java21)
* [**GraalVM for JDK 23**](https://www.oracle.com/in/java/technologies/downloads/#graalvmjava23)
* [**GraalVM for JDK 21**](https://www.oracle.com/in/java/technologies/downloads/#graalvmjava21)

Step 3: JDK 21 is the latest *Long-Term Support (LTS)* release of the Java SE Platform. So, we use this version.

Step 4: According to your device operating system choose the product/file description.

Step 5: Then the version will be automatically downloaded.

**2.Steps for installation.**

Step 1: Go to the folder where it was downloaded.

Step 2: Then open and accept all terms and conditions.

Step 3: And install it.

**3.Setting environmental variables.**

Step 1: Open file explorer, then right click on This PC next select on properties then it will take you to the settings app then click on advanced system settings and then click on **Environment Variables**.

Step 2: 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)

Step 3: Find Path under **System Variables**, click **Edit**, and add the path of the jdk-21(D:\UNI+\EDU\JAVA) **A screenshot of a computer

AI-generated content may be incorrect.**

**4.Checking for jdk(java development kit) version.**

Step 1: Open command prompt.

Step 2: Enter javac --version for version of jdk installed.

Step 3: Enter java –version for all details like when downloaded and path of environment variables

A screenshot of a computer program

AI-generated content may be incorrect.

1)AIM: write a simple java program to print your name, roll no,sec etc.

**Program :**

class Demo

{

public static void main(String[] args)

{

System.out.println("NAME= M. BINDHU");

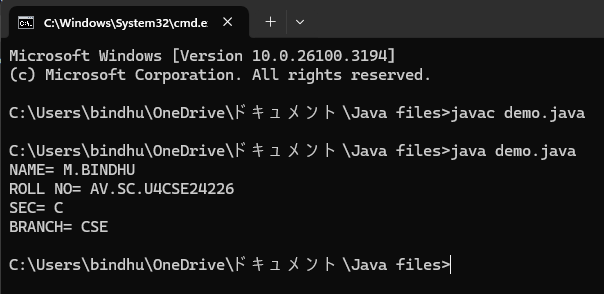
System.out.println("ROLL NO= AV.SC.U4CSE24226");

System.out.println("SEC= C");

System.out.println("BRANCH= CSE");

}

}



**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Trial.java:6: error: ';' expected  System.out.println("NAME= M.BINDHU")  ^  1 error |

**WEEK -2**

**SIMPLE JAVA programs**

**1)AIM:WRITE A JAVA PROGRAM TO FIND AREA OF RECTANGLE**

**Program:**

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

**class area**

**{**

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

**{**

**int area;**

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

**System.out.println("Enter Length:");**

**int l = sc.nextInt();**

**System.out.println("Enter Breadth:");**

**int b = sc.nextInt();**

**area = l\*b;**

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

**}**

**}**

**Output:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Error:

|  |  |  |
| --- | --- | --- |
| 1 | Syntax error | area.java:7: error: ';' expected  Scanner sc = new Scanner(System.in)  ^  1 error |

Important Points:

1. The program defines a class named `area` with a `main` method that serves as the entry point for execution.

2. It uses the `Scanner` class from `java.util` to take user input for the length and breadth of a rectangle.

3. The area is calculated using the formula `area = length \* breadth` and stored in an integer variable.

4. The result is displayed to the user using `System.out.println()`.

5. It is recommended to close the `Scanner` object to prevent resource leaks by calling `sc.close()` at the end.

**2)AIM: WRITE A JAVA PROGRAM TO CALCULATE SIMPLE INTREST**

**Program:**

import java.util.\*;

class Simpleinterest{

public static void main(String[] args)

{

System.out.println(" taking input");

Scanner sc = new Scanner(System.in);

System.out.println("enter INTa number:");

float p = sc.nextFloat();

System.out.println("enter a number:");

float t = sc.nextFloat();

System.out.println("enter a num:");

float r = sc.nextFloat();

float s = (p\*t\*r)/100;

System.out.println("simple intrest is"+s);

}

}

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

Error:

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1** | **Runtime error** | **error: file not found: Simpleinterest.java**  **Usage: javac <options> <source files>**  **use --help for a list of possible options** |

Important Points:

1. The program defines a class named `Simpleinterest` with a `main` method that serves as the entry point for execution.

2. It uses the `Scanner` class from `java.util` to take user input for principal amount (`p`), time period (`t`), and rate of interest (`r`).

3. The simple interest is calculated using the formula `s = (p \* t \* r) / 100` and stored in a float variable.

4. The result is displayed to the user using `System.out.println()`, showing the calculated simple interest.

5. It is advisable to close the `Scanner` object with `sc.close()` at the end to prevent resource leaks.

**3)AIM:WRITE A JAVA PROGRAM TO FIND FIBNOACCI SERIES.**

**Program:**

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

**class fibo**

**{**

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

**{**

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

**int num;**

**int f3;**

**int f1 = 0;**

**int f2 = 1;**

**int i = 2;**

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

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

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

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

**while(i<num)**

**{**

**f3 = f1+f2;**

**f1 = f2;**

**f2 = f3;**

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

**i = i+1;**

**}**

**}**

**}**

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1** | **Runtime error** | **error: file not found: Fibonacci.java**  **Usage: javac <options> <source files>**  **use --help for a list of possible options** |

**Important Points:**

**1. The program defines a class named `fibo` with a `main` method that calculates and prints the Fibonacci sequence up to a user-specified number of terms.**

**2. It uses the `Scanner` class from `java.util` to take input for the number of terms (`num`) to be generated in the Fibonacci sequence.**

**3. The first two Fibonacci numbers (0 and 1) are initialized, and a loop is used to calculate subsequent numbers by summing the last two numbers in the sequence.**

**4. The calculated Fibonacci numbers are printed to the console in each iteration of the loop.**

**5. The program does not close the `Scanner` object, which is a good practice to prevent resource leaks; it should include `sc.close()` at the end.**

**4)AIM: WRITE A JAVA PROGRAM TO CALCULATING TEMPERATURE IN CELSIUS.**

**Program:**

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

**class temp**

**{**

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

**{**

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

**float c;**

**System.out.println("Enter Fahrenheit Temperature:");**

**float f = sc.nextFloat();**

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

**System.out.println("celcius Temperature is :"+c);**

**}**

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Syntax** | temp.java:7: error: ';' expected  Scanner sc = new Scanner(System.in) |

**Important Points:**

**1. The program defines a class named `temp` with a `main` method that converts a temperature from Fahrenheit to Celsius.**

**2. It uses the `Scanner` class from `java.util` to take user input for the temperature in Fahrenheit (`f`).**

**3. The conversion formula `c = (f - 32) \* 5 / 9` is used to calculate the equivalent Celsius temperature.**

**4. The calculated Celsius temperature is printed to the console using `System.out.println()`.**

**5. It is advisable to close the `Scanner` object with `sc.close()` at the end to prevent resource leaks.**

**5)AIM:WRITE A JAVA PROGRAM TO FIND FACTORIAL OF A GIVEN NUMBER**

**Program:**

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

**class Fact**

**{**

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

**{**

**int number;**

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

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

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

**System.out.println("Enter time:");**

**int answer = factorial(number);**

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

**} static int factorial(int n){**

**if(n==1)**

**{**

**return 1;**

**}**

**return n \* factorial(n-1);**

**}**

**}**

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
|  | **Logical error** | **Incorrect input** | **Correcting input** |
|  | **Runtime error** | **Incorrect path** | **Using correct path** |

**Important Points:**

**1. The program defines a class named `Fact` with a `main` method that calculates the factorial of a user-provided number using recursion.**

**2. It uses the `Scanner` class from `java.util` to take input for the number whose factorial is to be calculated.**

**3. The `factorial` method is defined as a static method that recursively computes the factorial of the given integer `n`.**

**4. The result is printed to the console, displaying the factorial of the entered number.**

**5. The program does not close the `Scanner` object, which is a good practice to prevent resource leaks; it should include `sc.close()` at the end.**

**6)AIM: WRITE A JAVA PROGRAM TO CONVERTE CELSIUS TO FAHRENHEIT:**

**Program:**

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

**class heat**

**{**

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

**{**

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

**float f;**

**System.out.println("Enter Celsius Temperature:");**

**float c = sc.nextFloat();**

**f = (c\*9/5)+32;**

**System.out.println("Fahrenheit Temperature is :"+f);**

**}**

**}**

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect input** | **Correcting input** |

**Important Points:**

**1. The program defines a class named `heat` with a `main` method that converts a temperature from Celsius to Fahrenheit.**

**2. It uses the `Scanner` class from `java.util` to take user input for the temperature in Celsius (`c`).**

**3. The conversion formula `f = (c \* 9 / 5) + 32` is used to calculate the equivalent Fahrenheit temperature.**

**4. The calculated Fahrenheit temperature is printed to the console using `System.out.println()`.**

**5. It is advisable to close the `Scanner` object with `sc.close()` at the end to prevent resource leaks.**

**7)AIM:WRITE A JAVA PROGRAM TO CALCULATING AREA OF TRIANGLE USING HERONS’ FORMULA:**

**Program:**

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

**import java.lang.Math;**

**class Triangle{**

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

**double s, c, a, b, p;**

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

**System.out.println("Enter the value of a, b, and c:");**

**a = sc.nextDouble();**

**b = sc.nextDouble();**

**c = sc.nextDouble();**

**s = (a + b + c) / 2;**

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

**System.out.println("Area of triangle by Heron's formula is = " + p);**

**}**

**}**

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect input** | **Correcting input** |

**Important Points:**

**1. The program defines a class named `Triangle` with a `main` method that calculates the area of a triangle using Heron's formula.**

**2. It uses the `Scanner` class from `java.util` to take user input for the lengths of the triangle's sides (`a`, `b`, and `c`).**

**3. The semi-perimeter `s` is calculated as half the sum of the side lengths.**

**4. The area is computed using Heron's formula: `p = Math.sqrt(s \* (s - a) \* (s - b) \* (s - c))`.**

**5. The calculated area of the triangle is printed to the console, and it is advisable to close the `Scanner` object with `sc.close()` at the end to prevent resource leaks.**

**WEEK-3**

**1)AIM: WRITE A JAVA PROGRAM TO CREATE A CLASS WITH CAR NAME.**

**Class Diagram:**

**Main**

**- car\_color: String**

**- car\_brand: String**

**- fuel\_type: String**

**- mileage: int**

**+ Main(car\_color: String, car\_brand: String, fuel\_type: String, mileage: int)**

**+ color(): String**

**+ brand(): String**

**+ fuel(): String**

**+ mile(): int**

**PROGRAM:**

**class Main {**

**String car\_color;**

**String car\_brand;**

**String fuel\_type;**

**int mileage;**

**// Constructor**

**public Main(String car\_color, String car\_brand, String fuel\_type, int mileage) {**

**this.car\_color = car\_color;**

**this.car\_brand = car\_brand;**

**this.fuel\_type = fuel\_type;**

**this.mileage = mileage; // Fixed the spelling mistake here**

**}**

**// Getter methods**

**public String color() {**

**return car\_color;**

**}**

**public String brand() {**

**return car\_brand;**

**}**

**public String fuel() {**

**return fuel\_type;**

**}**

**public int mile() {**

**return mileage;**

**}**

**// Main method**

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

**Main car1 = new Main("red", "Hyundai", "diesel", 19);**

**System.out.println("Car 1 - Color: " + car1.color() + ", Brand: " + car1.brand() +**

**", Fuel Type: " + car1.fuel() + ", Mileage: " + car1.mile());**

**Main car2 = new Main("blue", "Toyota", "petrol", 17);**

**System.out.println("Car 2 - Color: " + car2.color() + ", Brand: " + car2.brand() +**

**", Fuel Type: " + car2.fuel() + ", Mileage: " + car2.mile());**

**Main car3 = new Main("black", "Tata", "diesel", 19);**

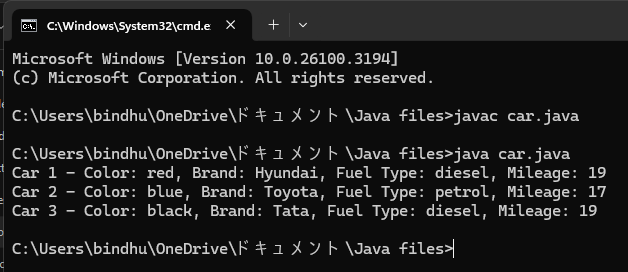
**System.out.println("Car 3 - Color: " + car3.color() + ", Brand: " + car3.brand() +**

**", Fuel Type: " + car3.fuel() + ", Mileage: " + car3.mile());**

**}**

**}**

**Output:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Identifier error** | **Main.java:3: error: <identifier> expected**  **car\_brand;** |

**Important Points:**

**1. The program defines a class named `Main` that represents a car with attributes such as color, brand, fuel type, and mileage.**

**2. A constructor initializes these attributes when a new car object is created.**

**3. Getter methods are provided to access the car's attributes: `color()`, `brand()`, `fuel()`, and `mile()`.**

**4. The `main` method creates three car objects with different attributes and prints their details to the console.**

**5. The program demonstrates the use of object-oriented programming principles, including encapsulation and constructor usage.**

**2)AIM: WRITE A JAVA PROGRAM TO CREATE A CLASS BANK ACCOUNT WITH METHODS DEPOSIT AND WITHDRAWAL.**

**Class diagram:**

**Bank\_Account**

**- accname: String**

**- acno: int**

**- balance:**

**+ Bank\_Account(accname: String, acno: int, balance: float)**

**+ withdraw(amount: int): void**

**+ deposit(amount: int): void**

**+ displayDetails(): void**

**PROGRAM:**

**class Bank\_Account {**

**private String accname;**

**private int acno;**

**private float balance;**

**// Constructor**

**public Bank\_Account(String accname, int acno, float balance) {**

**this.accname = accname;**

**this.acno = acno;**

**this.balance = balance;**

**}**

**// Withdraw method**

**public void withdraw(int amount) {**

**if (amount <= balance) {**

**balance -= amount;**

**System.out.println("Withdrawal of " + amount + " successful. Remaining balance: " + balance);**

**} else {**

**System.out.println("Insufficient balance for " + accname);**

**}**

**}**

**// Deposit method**

**public void deposit(int amount) {**

**balance += amount;**

**System.out.println("Deposit of " + amount + " successful. Updated balance: " + balance);**

**}**

**// Method to display account details**

**public void displayDetails() {**

**System.out.println("Account Name: " + accname);**

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

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

**}**

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

**Bank\_Account s = new Bank\_Account("janu", 34571, 25000);**

**s.withdraw(20000);**

**s.deposit(2000);**

**s.displayDetails();**

**System.out.println(); // Just for better readability**

**Bank\_Account s1 = new Bank\_Account("bindhu", 333226, 25000);**

**s1.withdraw(2500);**

**s1.deposit(250000);**

**s1.displayDetails();**

**}**

**}**

**OUTPUT:**

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Syntax** | Bank\_Account.java:7: error: ';' expected  Scanner sc = new Scanner(System.in) |

**Important Point:**

**1. The program defines a class named `Bank\_Account` that encapsulates the properties and behaviors of a bank account, including account name, account number, and balance.**

**2. A constructor initializes the account details when a new `Bank\_Account` object is created.**

**3. The class includes methods for withdrawing (`withdraw`) and depositing (`deposit`) money, with checks for sufficient balance during withdrawals.**

**4. A method named `displayDetails` prints the account's information, including the account name, number, and current balance.**

**5. The `main` method demonstrates the functionality by creating two bank account objects, performing transactions, and displaying their details.**

**WEEK-4**

1. **AIM:Create a java program with class named “Myclass” with a static variable count of “int” type ,insized to “zero” and a constant variable “pi” of type “double” initialized to 3.1415 as attributes of that class . define a contructor for “Myclass” is created finally print the final values “count” and “pi” variables.**

**Class Diagram:**

**MyClass**

**- count: int**

**- pi: double**

**+ MyClass()**

**+ display(): void**

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

**PROGRAM:**

**class MyClass {**

**static int count = 0; //**

**final double pi = 3.1415;**

**MyClass() {**

**count = count + 1;**

**}**

**public void display() {**

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

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

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

**}**

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

**MyClass Asec = new MyClass();**

**Asec.display();**

**MyClass Bsec = new MyClass();**

**Bsec.display();**

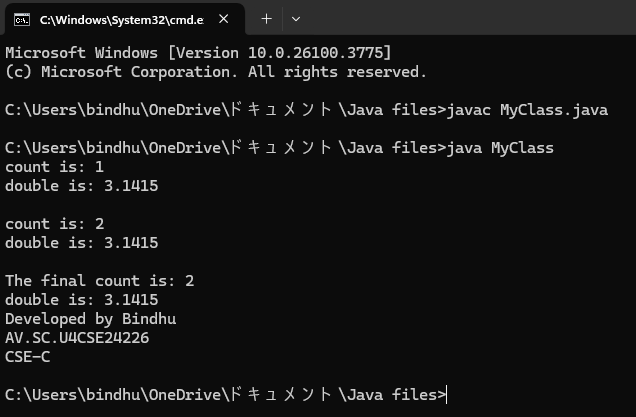
**System.out.println("The final count is: " + count);**

**System.out.println("double is: " + Bsec.pi);**

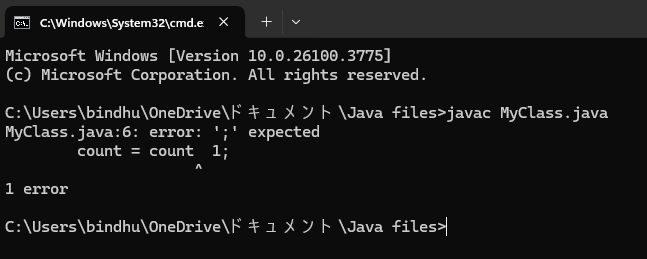
**}**

**}**

**OUTPUT:**

****

**Negative Case:**

****

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Syntax** | Bsec.java:23:error:illegal character:’\u00a0 |

**Important Points:**

**1. The program defines a class named `MyClass` with a static variable `count` that tracks the number of instances created and a final variable `pi` representing the value of π.**

**2. The constructor increments the `count` variable each time a new instance of `MyClass` is created.**

**3. The `display` method prints the current count of instances and the value of `pi`.**

**4. The `main` method creates two instances of `MyClass`, calls the `display` method for each, and prints the final count and the value of `pi` from the second instance.**

**5. The use of a static variable allows the count to be shared across all instances, while the final variable `pi` remains constant for all instances.**

2)AIM: **Create a java program with class named” book”. The class should contain various attributes such as “Title of the book”, “Author”, Year of publications”, it should also contain a constructor with parameter which initializes “Title of the book”, “Author”, “Year of publication”. Create a method which displays the details of the book. Title of the book (), Author (), Year of publication ().Display the details of two book, by creating 2 objects.**

**Class Diagram:**

**Book**

**- title: String**

**- author: String**

**- yearOfPublication: int**

**+ Book(String, String, int)**

**+ getBookDetails(): void**

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

**PROGRAM:**

**class Book {**

**// Declaring attributes**

**String title;**

**String author;**

**int yearOfPublication;**

**// Constructor to initialize values**

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

**this.title = title;**

**this.author = author;**

**this.yearOfPublication = yearOfPublication;**

**System.out.println("Your book: " + this.title);**

**}**

**// Creating a method**

**public void getBookDetails() {**

**System.out.println("Title of the book: " + this.title);**

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

**System.out.println("Year of publication: " + this.yearOfPublication);**

**}**

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

**// Creating objects for class Book**

**Book book1 = new Book("The Great Gatsby", "F. Scott Fitzgerald", 1925);**

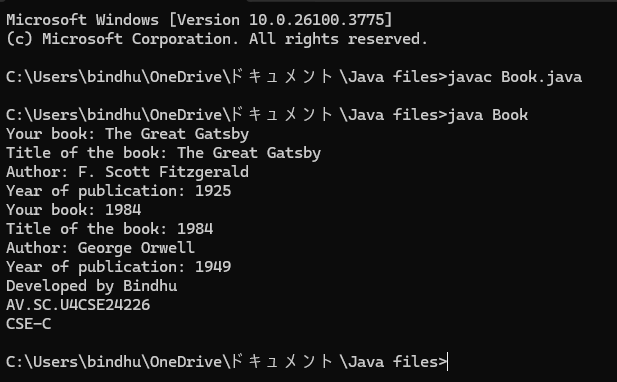
**book1.getBookDetails();**

**Book book2 = new Book("1984", "George Orwell", 1949);**

**book2.getBookDetails();**

**}**

**OUTPUT:**

**Positive Case**

**Negative Case:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**ERROR:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | In the statement at the end ; is not mentioned | Rectified by keeping ; at the end of the ststement |

**Important Points:**

**1. The program defines a class named `Book` with attributes for the title, author, and year of publication of a book.**

**2. A constructor initializes these attributes and prints a message indicating the book's title when a new `Book` object is created.**

**3. The `getBookDetails` method displays the book's details, including its title, author, and year of publication.**

**4. The `main` method creates two instances of the `Book` class, each representing a different book, and calls the `getBookDetails` method for each.**

**5. Note that the `Book` class is defined twice in the provided code, which would lead to a compilation error; only one definition is needed.**

**WEEK -5**

**1)AIM:** **Create a calculator using the operations including addition, subtraction, multiplication and division using Multilevel Inheritance and display the desired output.**

**Class Diagram:**

**Calculator**

**- a: double**

**- b: double**

**+ Calculator(double, double)**

**+ add(): double**

**+ subtract(): double**

**+ multiply(): double**

**+ divide(): double**

**+ displayResults(): void**

**Program:**

**import java.util.Scanner;**

**class BasicCalculator {**

**protected double a, b;**

**public BasicCalculator(double a, double b) {**

**this.a = a;**

**this.b = b;**

**}**

**public double add() {**

**return a + b;**

**}**

**public double subtract() {**

**return a - b;**

**}**

**}**

**class AdvancedCalculator extends BasicCalculator {**

**public AdvancedCalculator(double a, double b) {**

**super(a, b);**

**}**

**public double multiply() {**

**return a \* b;**

**}**

**public double divide() {**

**if (b == 0) {**

**throw new ArithmeticException("Error: Division by zero is not allowed.");**

**}**

**return a / b;**

**}**

**}**

**class ScientificCalculator extends AdvancedCalculator {**

**public ScientificCalculator(double a, double b) {**

**super(a, b);**

**}**

**public double power() {**

**return Math.pow(a, b);**

**}**

**public void displayResults() {**

**System.out.println("Addition: " + add());**

**System.out.println("Subtraction: " + subtract());**

**System.out.println("Multiplication: " + multiply());**

**try {**

**System.out.println("Division: " + divide());**

**} catch (ArithmeticException e) {**

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

**}**

**System.out.println("Power (a^b): " + power());**

**}**

**}**

**public class AllCalculator {**

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

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

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

**double a = input.nextDouble();**

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

**double b = input.nextDouble();**

**ScientificCalculator calc = new ScientificCalculator(a, b);**

**calc.displayResults();**

**System.out.println("Developed by bindhu.");**

**input.close();**

**}**

**}**

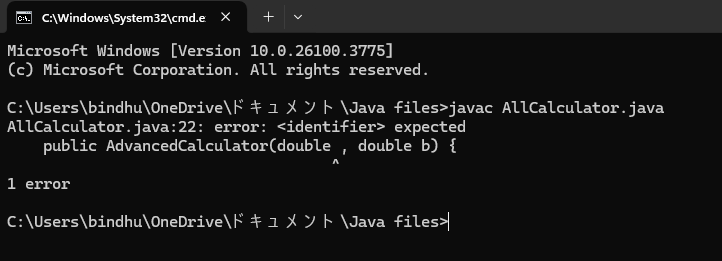
**Output:**

**Positive Case:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Negative Case:**

****

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Syntax** | Bsec.java:23:error:illegal character:’\u00a0 |

**Important Points:**

**1. The program defines a hierarchy of calculator classes: `BasicCalculator`, `AdvancedCalculator`, and `ScientificCalculator`, each extending the previous one to add more functionality.**

**2. The `BasicCalculator` class provides methods for addition and subtraction, while the `AdvancedCalculator` class adds multiplication and division capabilities.**

**3. The `ScientificCalculator` class further extends the functionality by including a method for exponentiation (power) and a method to display all results.**

**4. The `main` method in the `AllCalculator` class prompts the user for two numbers, creates an instance of `ScientificCalculator`, and displays the results of various calculations.**

**5. The program includes error handling for division by zero and ensures proper resource management by closing the `Scanner` object at the end.**

**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**

**• Cars should have an additional property: number of doors**

**• Bike should have a property indicating whether they have gears or not**

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

**• Every class should have a constructor**

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

**3. Implement the truck class and update the main method to create a truck object, also create an object for car and bike subclassed. Finally display its details.**

**Class Diagram:**

**Vehicle**

**- brand: String**

**- speed: int**

**+ Vehicle(String, int)**

**+ start(): void**

**+ showDetails(): void**

**Program:**

**import java.util.Scanner;**

**class Vehicle {**

**private String brand;**

**private 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;**

**}**

**@Override**

**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;**

**}**

**@Override**

**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;**

**}**

**@Override**

**public void showDetails() {**

**super.showDetails();**

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

**}**

**}**

**public class Rent {**

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

**// Creating objects**

**Car car = new Car("Toyota", 150, 4);**

**Bike bike = new Bike("Yamaha", 120, true);**

**Truck truck = new Truck("Volvo", 90, 10);**

**// Displaying details**

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

**car.start();**

**car.showDetails();**

**System.out.println("\n===== Bike Details =====");**

**bike.start();**

**bike.showDetails();**

**System.out.println("\n===== Truck Details =====");**

**truck.start();**

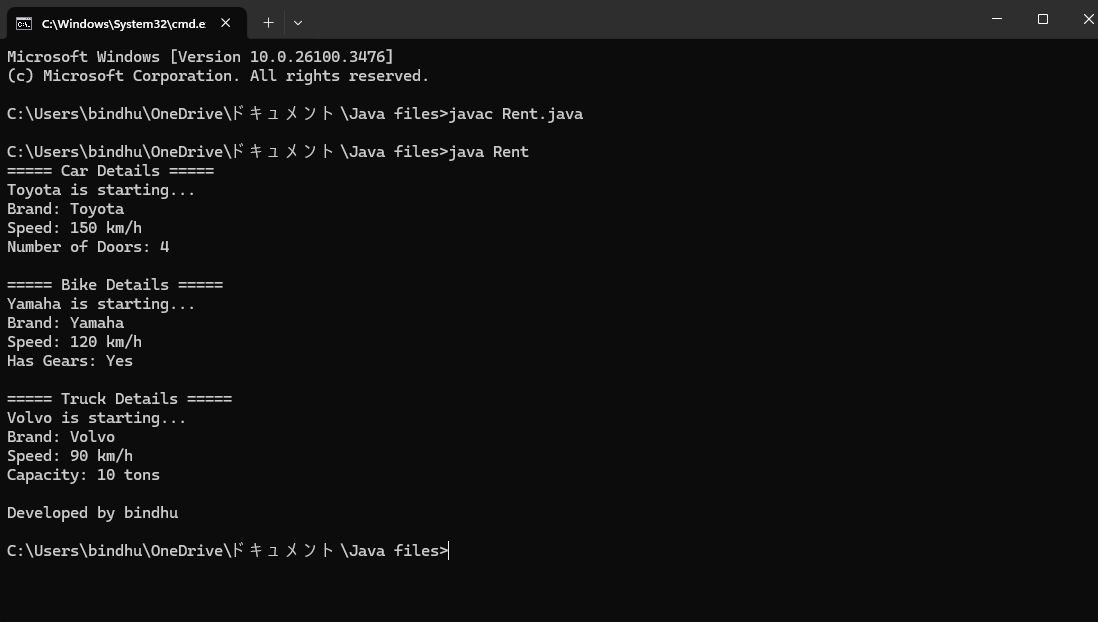
**truck.showDetails();**

**System.out.println("\nDeveloped by bindhu");**

**}**

**}**

**Output:**

**Positive Case:**

**Negative Case:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Syntax** | Give wrong class name Vehicle instead of Rent |

**Important Points:**

**1. The program defines a class hierarchy for vehicles, including a base class `Vehicle` and derived classes `Car`, `Bike`, and `Truck`, each with specific attributes and behaviors.**

**2. The `Vehicle` class has attributes for brand and speed, along with methods to start the vehicle and display its details.**

**3. Each derived class (`Car`, `Bike`, `Truck`) extends `Vehicle` and overrides the `showDetails` method to include additional attributes specific to each vehicle type.**

**4. The `main` method in the `Rent` class creates instances of `Car`, `Bike`, and `Truck`, and displays their details by calling the respective methods.**

**5. The program demonstrates object-oriented programming principles such as inheritance and method overriding, while also providing a clear output format for vehicle details.**

**WEEK- 6**

**1)AIM: Write a java program to create a Vehicle class with a method displayInfo(). Override this method in the Car subclass to provide specific information about a car.**

**Class Diagram:**

**Vehicle**

**+ displayInfo(): void**

**Car**

**- make: String**

**- model: String**

**- year: int**

**+ Car(String, String**

**Program:**

**class Vehicle {**

**public void displayInfo() {**

**System.out.println("This is a vehicle.");**

**}**

**}**

**class Car extends Vehicle {**

**private String make;**

**private String model;**

**private int year;**

**public Car(String make, String model, int year) {**

**this.make = make;**

**this.model = model;**

**this.year = year;**

**}**

**@Override**

**public void displayInfo() {**

**System.out.println("This is a car.");**

**System.out.println("Make: " + make);**

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

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

**}**

**}**

**public class Main {**

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

**// Create a Vehicle object**

**Vehicle vehicle = new Vehicle();**

**vehicle.displayInfo();**

**// Create a Car object**

**Car car = new Car("Toyota", "Camry", 2022);**

**car.displayInfo();**

**System.out.println("\nDeveloped by bindhu");**

**}**

**}**

**Output:**

**Positive Case:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Negative Case:**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**Errors:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect input** | **Correcting input** |

**Important Points:**

**1. The program defines a base class `Vehicle` with a method `displayInfo` that outputs a general message about vehicles.**

**2. The `Car` class extends `Vehicle` and includes additional attributes for make, model, and year, along with an overridden `displayInfo` method to provide specific details about the car.**

**3. In the `main` method of the `Main` class, an instance of `Vehicle` is created and its information is displayed.**

**4. An instance of `Car` is also created with specific attributes, and its `displayInfo` method is called to show detailed information about the car.**

**5. The program demonstrates inheritance and method overriding in object-oriented programming, allowing for specialized behavior in derived classes.**

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

**UG Admissions require a minimum of 60%.**

**PG Admissions require a minimum of 70%.**

**Class Diagram:**

**Student**

**- name: String**

**- percentage: double**

**+ Student(String, double)**

**+ getName(): String**

**+ getPercentage(): double**

**AdmissionSystem**

**+ isEligibleForUG(Student): boolean**

**+ isEligibleForPG(Student): boolean**

**AdmissionSystemDemo**

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

**Program:**

**import java.util.Scanner;**

**class Student {**

**private String name;**

**private double percentage;**

**public Student(String name, double percentage) {**

**this.name = name;**

**this.percentage = percentage;**

**}**

**public String getName() {**

**return name;**

**}**

**public double getPercentage() {**

**return percentage;**

**}**

**}**

**class AdmissionSystem {**

**// Method to check UG eligibility**

**public boolean isEligibleForUG(Student student) {**

**return student.getPercentage() >= 60.0;**

**}**

**// Method to check PG eligibility**

**public boolean isEligibleForPG(Student student) {**

**return student.getPercentage() >= 70.0;**

**}**

**}**

**public class AdmissionSystemDemo {**

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

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

**System.out.print("Enter student's name: ");**

**String name = scanner.nextLine();**

**System.out.print("Enter student's percentage: ");**

**double percentage = scanner.nextDouble();**

**Student student = new Student(name, percentage);**

**AdmissionSystem admissionSystem = new AdmissionSystem();**

**boolean eligibleForUG = admissionSystem.isEligibleForUG(student);**

**boolean eligibleForPG = admissionSystem.isEligibleForPG(student);**

**System.out.println("\nEligibility Results for " + student.getName() + ":");**

**if (eligibleForUG) {**

**System.out.println("Eligible for Undergraduate (UG) program.");**

**} else {**

**System.out.println("Not eligible for Undergraduate (UG) program.");**

**}**

**if (eligibleForPG) {**

**System.out.println("Eligible for Postgraduate (PG) program.");**

**} else {**

**System.out.println("Not eligible for Postgraduate (PG) program.");**

**}**

**System.out.println("\nDeveloped by bindhu");**

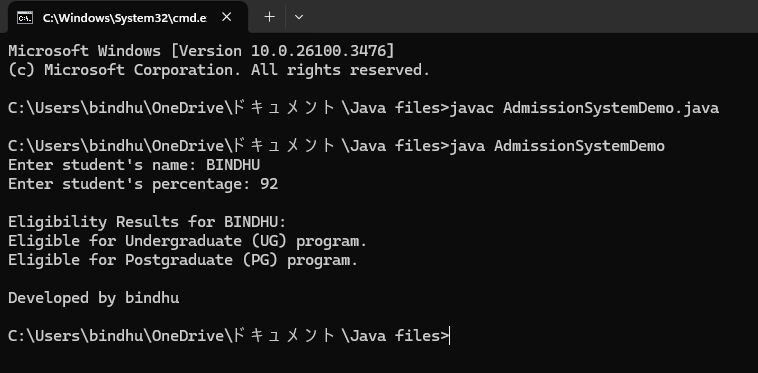
**scanner.close();**

**}**

**}**

**Output:**

**Positive Case:**

****

**Negative Case:**

**A computer screen with white text

AI-generated content may be incorrect.**

**Error:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** |
| **1.** | **Syntax** | Bsec.java:23:error:illegal character:’\u00a0 |

**Important Points:**

**1. The program defines a `Student` class with attributes for the student's name and percentage, along with getter methods to access these attributes.**

**2. The `AdmissionSystem` class contains methods to check eligibility for undergraduate (UG) and postgraduate (PG) programs based on the student's percentage.**

**3. In the `AdmissionSystemDemo` class, user input is collected for the student's name and percentage, and a `Student` object is created with this data.**

**4. The program checks the student's eligibility for UG and PG programs using the `AdmissionSystem` methods and displays the results accordingly.**

**5. The program demonstrates the use of encapsulation, object-oriented design, and user interaction through console input and output.**

**3) AIM:** **Create a Calculator class with overloaded methos to perform addition:**

**i) Add two integers**

**ii) Add two doubles**

**iii) Add three integers**

**Class Diagram:**

**Calculator**

**+ add(int, int): int**

**+ add(double, double): double**

**+ add(int, int, int): int**

**CalculatorDemo**

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

**Program:**

**class Calculator {**

**public int add(int a, int b) {**

**return a + b;**

**}**

**public double add(double a, double b) {**

**return a + b;**

**}**

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

**return a + b + c;**

**}**

**}**

**public class CalculatorDemo {**

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

**Calculator calculator = new Calculator();**

**int sumInt = calculator.add(5, 10);**

**System.out.println("Sum of two integers (5 + 10): " + sumInt);**

**double sumDouble = calculator.add(5.5, 10.5);**

**System.out.println("Sum of two doubles (5.5 + 10.5): " + sumDouble);**

**int sumThreeInts = calculator.add(1, 2, 3);**

**System.out.println("Sum of three integers (1 + 2 + 3): " + sumThreeInts);**

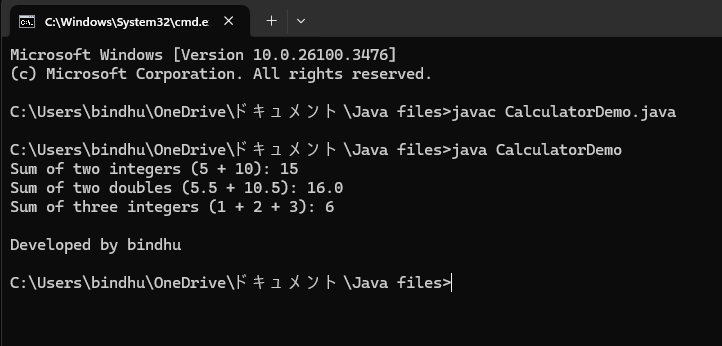
**System.out.println("\nDeveloped by bindhu");**

**}**

**}**

**Output:**

**Positive Case:**

****

**Negative Case:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Error:**

**No error**

**Important Points:**

**1. The program defines a `Calculator` class that includes overloaded `add` methods to handle different types and numbers of parameters: two integers, two doubles, and three integers.**

**2. Each `add` method performs addition and returns the result, demonstrating method overloading based on parameter types and counts.**

**3. In the `CalculatorDemo` class, an instance of `Calculator` is created to perform various addition operations.**

**4. The results of adding two integers, two doubles, and three integers are printed to the console, showcasing the functionality of the overloaded methods.**

**5. The program illustrates the concept of method overloading in object-oriented programming, allowing for flexible and intuitive use of the `add` method.**

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

**Class Diagram:**

**Shape**

**+ calculateArea(double): double**

**+ calculateArea(double, double): double**

**Circle**

**+ calculateArea(double): double**

**ShapeDemo**

**+ main(String[]):**

**Program:**

**class Shape {**

**// Method to calculate area of a square**

**public double calculateArea(double side) {**

**return side \* side; // Area of square = side^2**

**}**

**public double calculateArea(double length, double width) {**

**return length \* width; // Area of rectangle = length \* width**

**}**

**}**

**class Circle extends Shape {**

**@Override**

**public double calculateArea(double radius) {**

**return Math.PI \* radius \* radius; // Area of circle = π \* radius^2**

**}**

**}**

**public class ShapeDemo {**

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

**Shape shape = new Shape();**

**Circle circle = new Circle();**

**double squareArea = shape.calculateArea(4.0);**

**System.out.println("Area of square (side = 4.0): " + squareArea);**

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

**System.out.println("Area of rectangle (length = 5.0, width = 3.0): " + rectangleArea);**

**double circleArea = circle.calculateArea(3.0);**

**System.out.println("Area of circle (radius = 3.0): " + circleArea);**

**System.out.println("\nDeveloped by bindhu");**

**}**

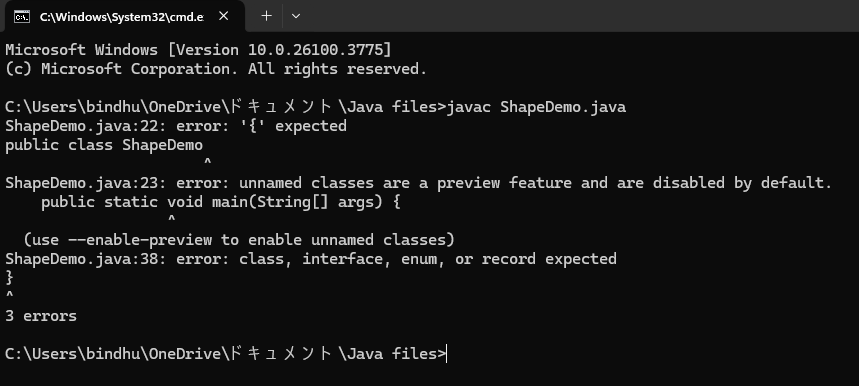
**}**

**Output:**

**Positive Case:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Negative Case:**

**Error:**

**No error**

**Important Points:**

**1. The program defines a `Shape` class with overloaded `calculateArea` methods to compute the area of a square and a rectangle based on the provided dimensions.**

**2. The `Circle` class extends `Shape` and overrides the `calculateArea` method to calculate the area of a circle using the formula ( pi (radius)^2 ).**

**3. In the `ShapeDemo` class, instances of `Shape` and `Circle` are created to demonstrate the area calculations for a square, rectangle, and circle.**

**4. The program outputs the calculated areas for a square with a side of 4.0, a rectangle with dimensions 5.0 by 3.0, and a circle with a radius of 3.0.**

**5. This implementation showcases method overloading and overriding in object-oriented programming, allowing for flexible area calculations for different shapes.**

**WEEK -7**

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

Class diagram:

<<abstract>>

Animal

+ sound(): void

Tiger Lion

+ sound(): void + sound(): void

**Program:**

**abstract class Animal {**

**public abstract void sound();**

**}**

**class Tiger extends Animal {**

**@Override**

**public void sound() {**

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

**}**

**}**

**class Lion extends Animal {**

**@Override**

**public void sound() {**

**System.out.println("Lion roars");**

**}**

**}**

**public class Main1 {**

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

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

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

**tiger.sound();**

**lion.sound();**

**System.out.println("Developed by Bindhu");**

**System.out.println("AV.SC.U4CSE24226");**

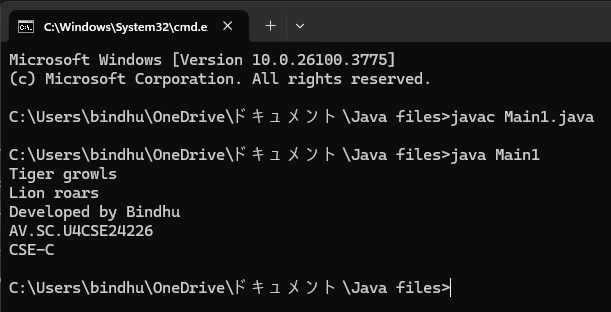
**System.out.println("CSE-C");**

**}**

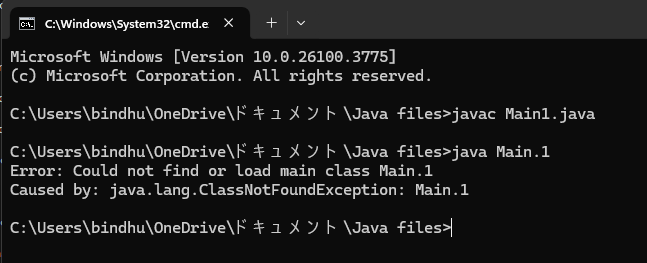
**}**

**Output:**

**Positive Case:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S NO:** | **ERROR** | **RECTIFICATION** |
| **1** | **error: ‘;’** | **Kept it.** |

**Important Points:**

**1. Abstract Class: `Animal` is an abstract class with an abstract method `sound()`, which must be implemented by subclasses.**

**2. Inheritance: `Tiger` and `Lion` are concrete classes that extend the `Animal` class, providing specific implementations of the `sound()` method.**

**3. Method Implementation: `Tiger` implements `sound()` to print "Tiger growls", while `Lion` implements it to print "Lion roars".**

**4. Polymorphism: The code demonstrates polymorphism by allowing `Animal` references to point to `Tiger` and `Lion` objects.**

**5. Encapsulation: The classes encapsulate behavior related to different animal sounds, promoting code reusability and organization.**

**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.**

**Class Diagram:**

**<<abstract>>**

**Shape3D**

**+ calculateVolume(): double**

**+ calculateSurfaceArea(): double**

**Sphere Cube**

**- radius: double - side: double**

**+ calculateVolume(): + calculateVolume():**

**double double**

**+ calculateSurfaceArea(): + calculateSurfaceArea():**

**double double**

**Program:**

**abstract class Shape3D {**

**public abstract double calculateVolume();**

**public abstract double calculateSurfaceArea();**

**}**

**class Sphere extends Shape3D {**

**private double radius;**

**public Sphere(double radius) {**

**this.radius = radius;**

**}**

**@Override**

**public double calculateVolume() {**

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

**}**

**@Override**

**public double calculateSurfaceArea() {**

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

**}**

**}**

**class Cube extends Shape3D {**

**private double side;**

**public Cube(double side) {**

**this.side = side;**

**}**

**@Override**

**public double calculateVolume() {**

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

**}**

**@Override**

**public double calculateSurfaceArea() {**

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

**}**

**}**

**public class Main2 {**

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

**// Create a Sphere with radius 5**

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

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

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

**// Create a Cube with side length 3**

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

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

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

**System.out.println("Developed by Bindhu");**

**System.out.println("AV.SC.U4CSE24226");**

**System.out.println("CSE-C");**

**}**

**}**

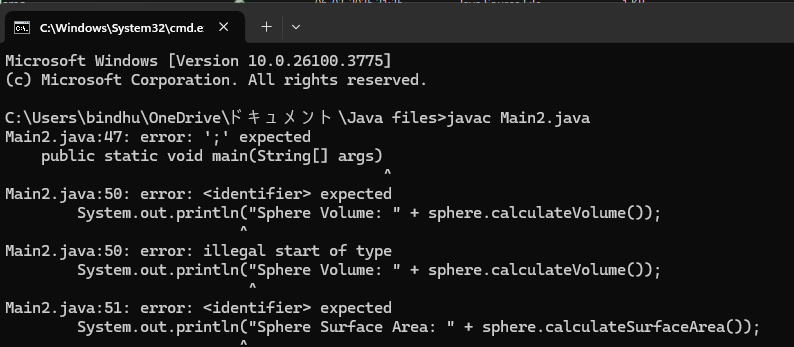
**Output:**

**Positive Case:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**Negative Case:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S NO** | **ERROR** | **RECTIFICATION** |
| **1** | **public static void main(String[] args)** | **public static void main(String[] args){** |

**Important Points:**

**1. Abstract Class: `Shape3D` is an abstract class that defines two abstract methods: `calculateVolume()` and `calculateSurfaceArea()`, which must be implemented by subclasses.**

**2. Concrete Classes: `Sphere` and `Cube` are concrete classes that extend `Shape3D`, each providing specific implementations for calculating volume and surface area.**

**3. Sphere Implementation: The `Sphere` class has a private attribute `radius` and implements the volume formula (4/3 pi r^3) and surface area formula (4 pi r^2).**

**4. Cube Implementation: The `Cube` class has a private attribute `side` and implements the volume formula (side^3) and surface area formula ( 6\*side^2).**

**5. Polymorphism: The code demonstrates polymorphism by allowing `Shape3D` references to point to `Sphere` and `Cube` objects, enabling dynamic method invocation for volume and surface area calculations.**

**3)AIM:** **Write a Java program using an abstract class to define a method for pattern printing.**

**Create an abstract class named PatternPrinter with an abstract method printPattern(int n) and a concrete method to display the pattern title.**

**Implement two subclasses:**

**1.StarPattern- Prints a right angled triangle stars**

**2.NumberPateern-Prints a right angled triangle increasing number**

**IN the main() method create objects of both subclasses and print the pattern for a given number of rows.**

**Class Diagram:**

**<<abstract>>**

**PatternPrinter**

**+ displayTitle(title: String): void**

**+ printPattern(n: int): void [abstract]**

**StarPattern NumberPattern**

**+ printPattern(n: int): void + printPattern(n: int): void**

**Program:**

**abstract class PatternPrinter {**

**public void displayTitle(String title) {**

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

**}**

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

**}**

**class StarPattern extends PatternPrinter {**

**@Override**

**public 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**

**public 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 Main3 {**

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

**int rows = 5; // Number of rows for the patterns**

**PatternPrinter starPattern = new StarPattern();**

**starPattern.displayTitle("Star Pattern:");**

**starPattern.printPattern(rows);**

**PatternPrinter numberPattern = new NumberPattern();**

**numberPattern.displayTitle("Number Pattern:");**

**numberPattern.printPattern(rows);**

**System.out.println("Developed by Bindhu");**

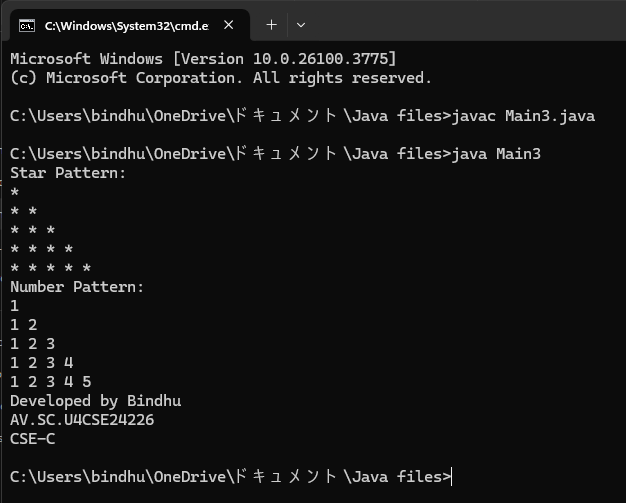
**System.out.println("AV.SC.U4CSE24226");**

**System.out.println("CSE-C");**

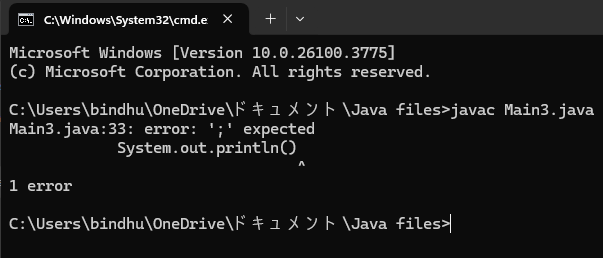
**}**

**Output:**

**Positive Case:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S NO** | **ERROR** | **RECTIFICATION** |
| **1** | **System.out.println()** | **System.out.println();** |

**Important Points:**

**1. Abstract Class: `PatternPrinter` defines an abstract method `printPattern(int n)` for printing patterns and a concrete method `displayTitle(String title)` for displaying the pattern title.**

**2. StarPattern Class: This subclass implements `printPattern(int n)` to print a right-angled triangle of stars based on the specified number of rows.**

**3. NumberPattern Class: This subclass implements `printPattern(int n)` to print a right-angled triangle of increasing numbers, also based on the specified number of rows.**

**4. Main Method: The `main()` method creates instances of `StarPattern` and `NumberPattern`, calls `displayTitle()` to print titles, and invokes `printPattern()` to display the respective patterns.**

**5. Output: The program outputs a star pattern and a number pattern for a given number of rows, demonstrating the use of abstract classes and method overriding in Java.**

**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. Create three getPerimeter() method for each of the three classes.

**Class Diagram:**

**Shape**

**+ getPerimeter(): double**

**Rectangle Circle Triangle**

**-width -radius -side1**

**-height -side2**

**-side3**

**Program:**

**interface Shape {**

**double getPerimeter();**

**}**

**class Rectangle implements Shape {**

**private double width;**

**private double height;**

**public Rectangle(double width, double height) {**

**this.width = width;**

**this.height = height;**

**}**

**@Override**

**public double getPerimeter() {**

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

**}**

**}**

**class Circle implements Shape {**

**private double radius;**

**public Circle(double radius) {**

**this.radius = radius;**

**}**

**@Override**

**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;**

**}**

**@Override**

**public double getPerimeter() {**

**return side1 + side2 + side3;**

**}**

**}**

**public class Main14 {**

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

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

**Shape circle = new Circle(7);**

**Shape triangle = new Triangle(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("Developed by Bindhu");**

**System.out.println("AV.SC.U4CSE24226");**

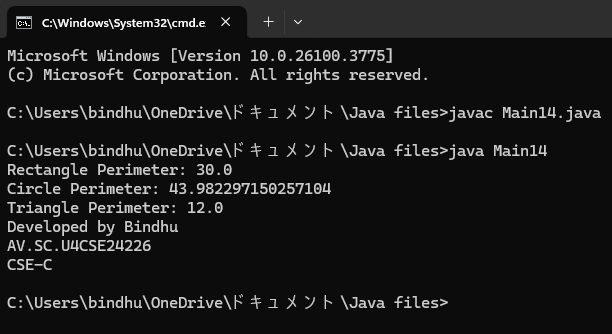
**System.out.println("CSE-C");**

**}**

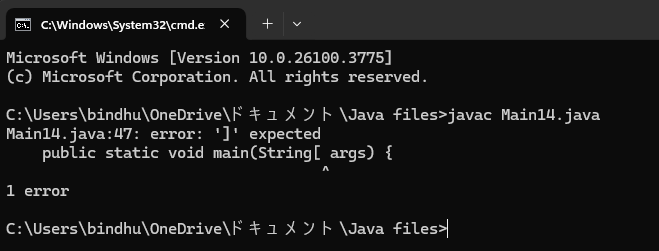
**}**

**Output:**

**Positive Case:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **ERROR** | **RECTIFICATION** |
| **1** | **Negative dimensions for shapes** | **Added validation in constructors** |

**Important Points:**

1. **The Shape interface defines a contract for all shapes to implement the getPerimeter() method.**
2. **Each shape class provides its own implementation of the perimeter calculation based on its specific properties.**
3. **The program demonstrates polymorphism by using the Shape interface to refer to different shape objects.**

**2)AIM:** Write a JAVA program to create an interface Playable with a method play() that takes no arguments and returns void. Create three classes Football, Volleyball and Basketball that implement the Playable interface and override the play() method to play the respective sports.

**Class Diagram:**

**Playable**

**+ play (): void**

**Football Volleyball Basketball**

**+play() +play() +play()**

**Program:**

**interface Playable {**

**void play();**

**}**

**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 Main15 {**

**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("Developed by Bindhu");**

**System.out.println("AV.SC.U4CSE24226");**

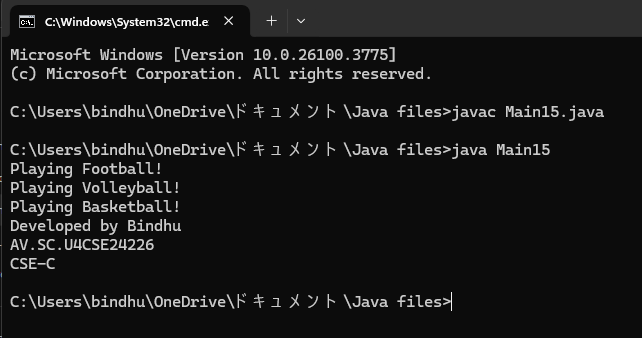
**System.out.println("CSE-C");**

**}**

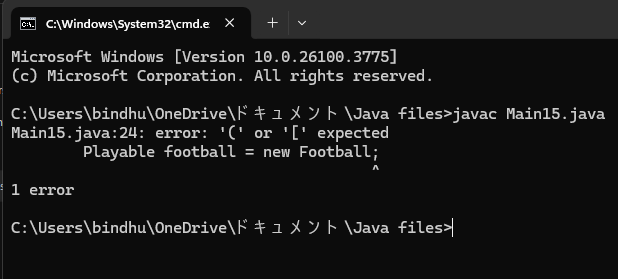
**}**

**Output:**

**Positive Case:**

****

**Negative Case:**

****

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **ERROR** | **RECTIFICATION** |
| **1** | **Did not keep ()** | **Kept ()** |

**Important Points:**

1. **The Playable interface defines a contract for any class that wants to implement the play() method.**
2. **Each sport class (Football, Volleyball, Basketball) provides its own implementation of the play() method, demonstrating polymorphism.**
3. **The program showcases how to use interfaces in Java to achieve abstraction and code reusability.**
4. **This design allows for easy extension; new sports can be added by simply creating new classes that implement the Playable interface**