**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



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

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: N.chaitanya**

**manikanta**

**Roll No: 24233**

**INDEX**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | Create a class named bank account with methods deposit and withdraw. Where the deposit method should accepts a parameter and when this method is called the deposited amount should be added to current balance. In addition to that when a withdraw method is called it has to verify whether the withdraw amount is less than the current balance. If not display message saying that there are insufficient funds. Use the constructor to display the details of the customer (Name, Account number, IFSC code, Branch). Also create two customer objects C1, C2. |  |  |  |
|  | WEEK-4 |  |  |  |
| 1 | Write a Java program with a class named book. The class should contain various attributes such as title, author, year of publication, and price. It should also contain a constructor with parameters that include title, author, year of publication, and price. Create a method that displays details of the book(Display the details of 3 books that is create 3 objects and display their details) |  |  |  |
| 2 | Create a java program with a Class named "my class" with a static variable 'count' of int type static and initialized to zero and a constant variable 'pi' of type double initialized to 3.1415 as attributes of that class now define a constructor of my class that increments the count variable each time an object of my class is created and finally prints the final values of count and pi variables. |  |  |  |
|  | **WEEK-5** |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication and division using multilevel in heritance and display the desired output. |  |  |  |
| 2 | Vehicle rental company wants to develop a system that maintains information about different types of vehicles available for rent. The company rents out cars and bikes and they need a program to store details about each vehicle such as brand and speed.  i. Cars should have an additional property: number of doors, Seating capacity.  ii. Bikes should have a property indicating whether they have gears or not.  iii. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.  iv. Each class should have a constructor.  Questions:  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 and additional property capacity (in tons).  b. Create a showTruck() method to display the truck’s capacity.  c. Write a constructor for truck that initializes all properties.  3. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike subclasses. Finally display the details. |  |  |  |
|  | **WEEK-6** |  |  |  |
| 1 | Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information about a car, model, fuel type, and colour using the constructor |  |  |  |
| 2 | Create a Java program for the scenario.  A college is developing an automated admission system that verifies student eligibility for undergraduate (UG) and postgraduate(PG) programs. Each program has different eligibility criteria based on the student's percentage in their previous qualification.  i) UG admissions require a minimum of 60%  ii) PG admissions require a minimum of 70% |  |  |  |
| 3 | Write a Java Program to create a Calculator class with overloaded methods to perform addition: Take the integer values a and b from the user.  i) Add two integers  ii) Add two doubles  iii) Add three integer |  |  |  |
| 4 | Write a Java Program to create a shape class with a method calculateArea() that is overloaded for different shapes(e.g., Square, Rectangle ). Then create a subclass Circle that overrides the calculateArea() method for a circle. |  |  |  |
|  |  |  |  |  |
|  | **WEEK-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 print pattern (int n) and a concrete method to display the pattern tittle. Implement two sub  class :  1.star pattern -prints a right-angled triangle of stars(\*)  2.number pattern-prints a right angled triangle of increasing numbers. 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 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 play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements 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-8** |  |  |  |
| **1** | Write a java program to create an interface shape with 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 play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements 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**

1. AIM:

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

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

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

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

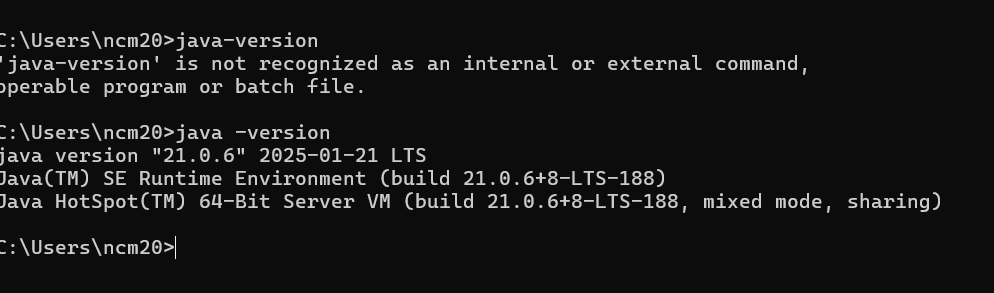
C:\Program Files\Java\jdk-21\bin)

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



**Checking JDK Version: -**

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



2)AIM:

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

Write your code in Notepad and execute it in cmd prompt

**CODE: -**

class Main

{

public static void main(String[] args)

{

System.out.println("Name:N.chaitanya manikanta");

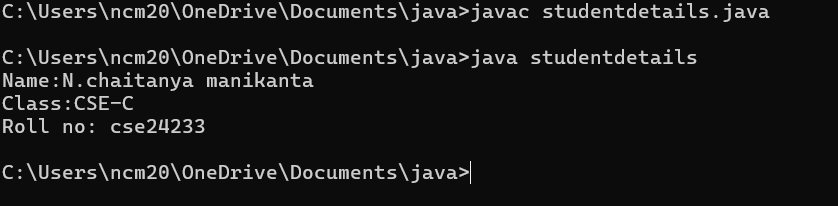
System.out.println("Class:CSE-C");

System.out.println("Roll No:24233");

}

}

**Output: -**



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

WEEK-2

1. AIM:

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

**User**

**Code:**

****

**Output**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1** | **Runtime error** | **Incorrect path** | **Copied correct path** |
| **2** | **Syntax error** | **{ missing** | **{ added** |
| **3** | **Logical error** | **Wrong formula** | **Formula rectified** |

**2 )AIM:**

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

**input from user**

**code:**

****

**Output:**

****

**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Undeclared variable error** | **Missing variable** | **Variable declared** |
| **2** | **Missing import statement** | **Not importing packages** | **Packages imported** |
| **3** | **Logical error** | **Wrong formula** | **Formula rectified** |

**3) AIM:Write a program to to calculate the fibonacii sequence and take the input from user**

**Code:**

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

**class fibo**

**{**

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

**{**

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

**int num;**

**int f3;**

**int f1 = 0;**

**int f2 = 1;**

**int i = 2;**

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

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

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

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

**while(i<num)**

**{**

**f3 = f1+f2;**

**f1 = f2;**

**f2 = f3;**

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

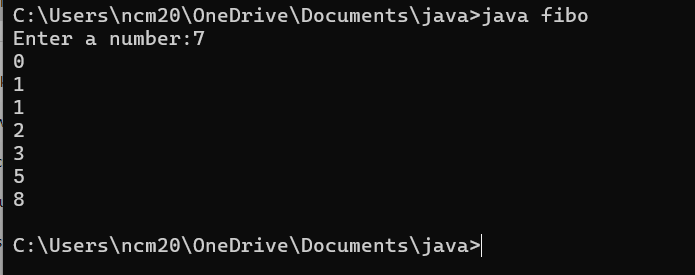
**i = i+1;**

**}**

**}**

**}**

**Output:**

****

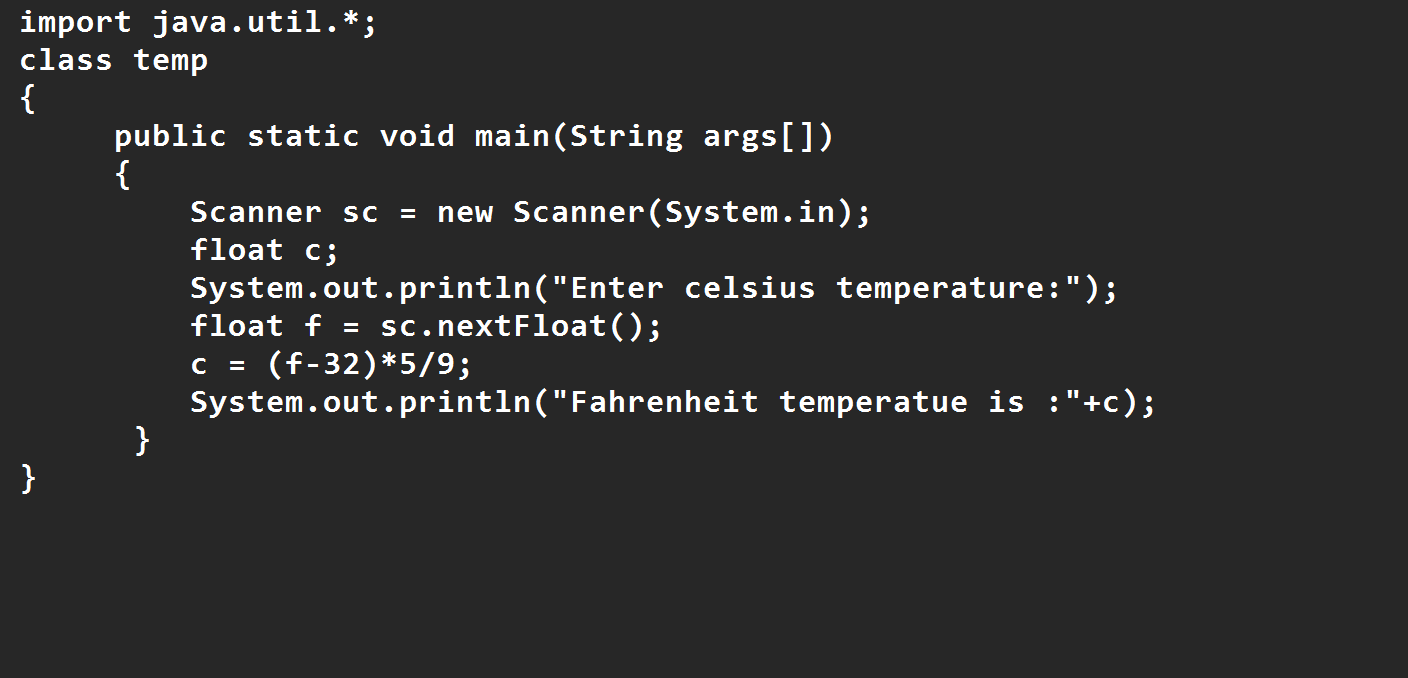
**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Logical error** | **Incorrect formula** | **Formula rectified** |
| **2** | **Run-time error** | **Incorrect path** | **Added correct path** |
| **3** |  |  |  |

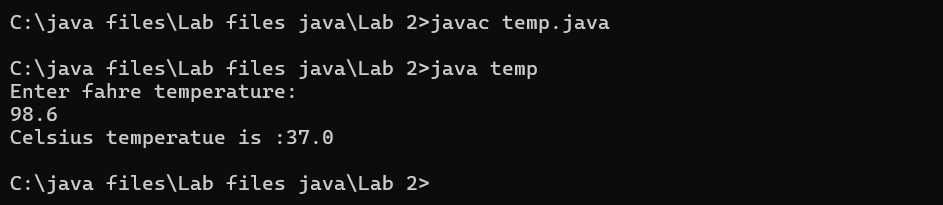
**5) AIM**

**Write a java program to convert temperature from Fahrenheit to celsius**

**Code:**

****

**Output:**

****

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

**5)AIM:**

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

**Code:**

****

**Output:**

****

**Error:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Runtime error** | **Incorrect path selection** | **Correct path added** |
| **2** | **Logical error** | **Incorrect logic** | **Correct logic** |
| **3** |  |  |  |

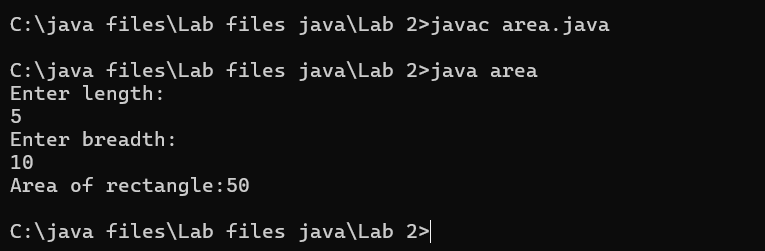
**6) AIM:**

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

**Code:**

****

**Output:**



|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **Rectification** |
| **1** | **Syntax error** | **Semi colon missing** | **Semi colon added** |
| **2** | **Missing import error** | **Import package missing** | **Import package added** |
| **3** |  |  |  |

7)AIM:

**Write a program to find the area of triangle by using heron’s formula take the input from the user**

**Code:**

****

**OUTPUT:**

****

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

**WEEK 3**

**Aim:**

**To create java program with following instructions**

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

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

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

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

**Code:**

import java.util.\*;

class car

{

public String Car\_color;

public String Car\_brand;

public String fuel\_type;

public int mileage;

public void start()

{

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

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

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

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

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

}

public void service()

{

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

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

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

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

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

}

public void stop()

{

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

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

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

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

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

}

public static void main(String args[])

{ System.out.println("\n ncm\n\n");

car car1 = new car();

car1.Car\_color = "Blue";

car1.Car\_brand = "BMW";

car1.fuel\_type = "Deisel";

car1.mileage = 10;

car1.start();

car car2 = new car();

car2.Car\_color = "Red";

car2.Car\_brand = "Tesla";

car2.fuel\_type = "EV";

car2.mileage = 300;

car2.stop();

car car3 = new car();

car3.Car\_color = "Yellow";

car3.Car\_brand = "MAHINDRA";

car3.fuel\_type = "Petrol";

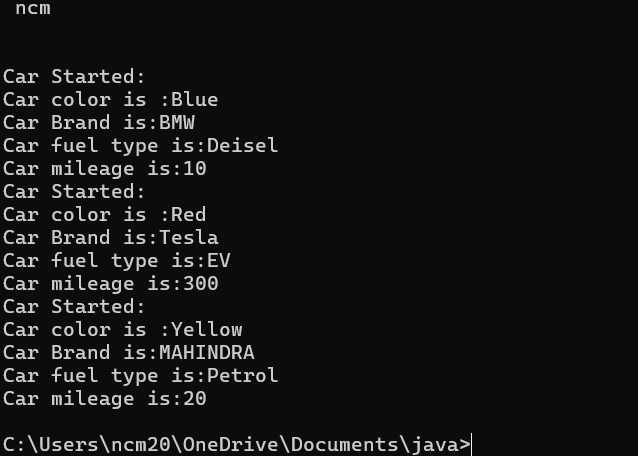
car3.mileage = 20;

car3.service();

}

}

**Output:**

****

**Class Diagram**

|  |
| --- |
| **Car** |
| **+ car\_color: String**  **+ car\_brand: String**  **+ fuel\_type: String**  **+ mileage: int** |
| **+ Car(): void**  **+ start(): void**  **+ service(): void**  **+ stop(): void** |

**2.AIM:**

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

**Code:**

class BankAccount

{

private double balance;

public BankAccount(double initialBalance)

{

if(initialBalance > 0)

{

this.balance = initialBalance;

}

else

{

this.balance = 0;

}

}

public void deposit(double amount)

{

if(amount>0)

{

balance = balance+amount;

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

}

else

{

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

}

}

public double getBalance()

{

return balance;

}

}

public class Main1

{

public static void main(String args[])

{

BankAccount account = new BankAccount(1000);

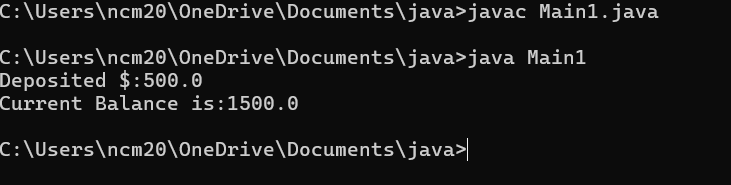
account.deposit(500);

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

}

}

**Output:**



**ERRORS:**

|  |  |  |
| --- | --- | --- |
| **Sno.** | **Error message** | **Error rectification** |
| 1. | error: ';' expected  cust1.withdraw(3050) | Add a “;”    cust1.withdraw(3050); |
| 2. | error: cannot find  symbol  thisCurrBal=CurrBal; | Add a “.”    this.CurrBal=CurrBal; |

WEEK-4

**1.AIM:**

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

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

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

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

**PROGRAM:**

public class Book {

public String title;

public String author;

public int year;

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

this.title = title;

this.author = author;

this.year = year;

}

public void displayDetails() {

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

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

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

}

public static void main(String[] args) {

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

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

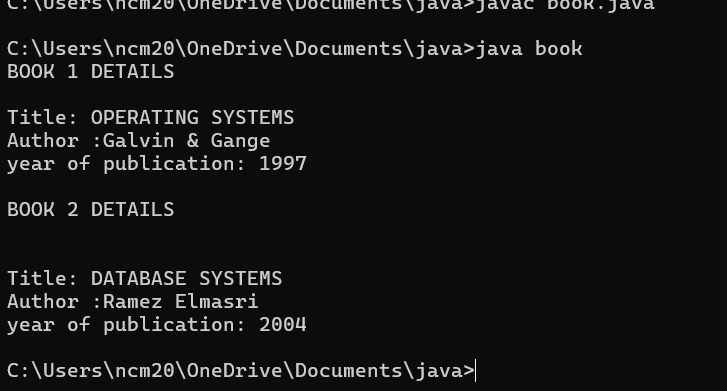
b1.displayDetails();

b2.displayDetails();

}

}

**Output:**

****

**Class diagram:**

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

**2.AIM:**

**WRITE A JAVA PROGRAM WITH CLASS NAMED “MyClass” WITH STATIC VARIABLE COUNT OF INT TYPE INTIALIZE IT TO ZERO AND CONSTANT “Pi” OF TYPE DOUBLE INITIALIZED TO “3.14” AS ATTRIBUTES OF THAT CLASS. NOW DEFINE A CONSTRUCTOR FOR “MyClass”, THAT INCREMENTS THE COUNT VARIABLE EACH TIME AN OBJECT OF “MyClass” IS CREATED. FINALLY, PRINT**

**FINAL VALUES OF ‘COUNT’ AND ‘PI’ VARIABLES AND CREATE 3 OBJECTS.**

**PROGRAM:**

public class MyClass {

static int count = 0;

static final double pi = 3.14;

MyClass() {

count++;

}

public static void main(String[] args) {

MyClass obj1 = new MyClass();

MyClass obj2 = new MyClass();

MyClass obj3 = new MyClass();

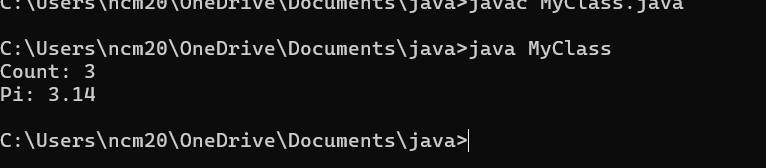
System.out.println("Count: " +count);

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

}

}

**OUTPUT:**

****

**CLASS DIAGRAM:**

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

**WEEK-5**

**1)Aim :** create a calculator using the operations including addition, subtraction, multiplication and division using multilevel inheritance and display the desired output .

**Program :**

public class calculator {

    int a = 18;

    int b = 1;

void addition(){

    System.out.println("Addition is :" + (a+b));

}

}

class multi1 extends calculator{

    void subtraction(){

        System.out.println("Subtraction is :" + (a-b));

    }

}

class multi2 extends calculator{

    void multiplication(){

        System.out.println("Multiplication is :" + (a\*b));

    }

}

class multi3 extends calculator{

    void division(){

        System.out.println("Division is :" + (a/b));

    }

}

class multipleinheritance {

    public static void main(String[] args) {

        multi1 m1 = new multi1();

        multi2 m2 = new multi2();

        multi3 m3 = new multi3();

        m1.addition();

        System.out.println("    ");

        m1.subtraction();

        System.out.println("    ");

        m2.multiplication();

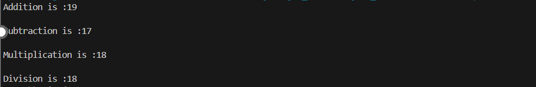
        System.out.println("    ");

        m3.division();

}

}

**OUTPUT :**



**ERROR :**

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

**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 variable such as brand and speed.

• Cars should have an additional property: numbers of doors, seating capacity.

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

• The system should also include a function to display details about each vehicle & indicates when a vehicle is starting.

• Every class should have 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)Aim: Implement the truck class and update the main method to create a truck object & also create an object for car & bike sub classed. Finally display its details

**Program :**

class Vehicle{

    String brand;

    int speed;

    Vehicle(String brand,int speed){

        this.brand=brand;

        this.speed=speed;

    }

    void Details(){

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

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

    }

}

class CARS extends Vehicle{

    int doors;

    int capacity;

    public CARS(String brand,int speed,int doors,int capacity){

        super(brand, speed);

        this.doors=doors;

        this.capacity=capacity;

    }

    void cardetails(){

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

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

    }

}

class Bikes extends Vehicle{

    Boolean gears;

    Bikes(String brand,int speed,Boolean gears){

        super(brand, speed);

        this.gears=gears;

    }

    void bikedetails(){

        if (gears==true)

        System.out.println("This bike has gears.");

        else

        System.out.println("This bike does not have gear system.");

    }

}

class Trucks extends Vehicle{

    int tons;

    Trucks(String brand,int speed,int tons){

        super(brand, speed);

        this.tons=tons;

    }

    void truckdetails(){

        System.out.println("The capacity of truck is: "+tons);

    }

}

class Rent{

    public static void main(String[] args){

        CARS c=new CARS("Tayota",120,5,5);

        c.cardetails();

        c.Details();

        Bikes b=new Bikes("KTM",80,true);

        b.bikedetails();

        b.Details();

        Trucks t=new Trucks("TATA",100,1);

        t.truckdetails();

        t.Details();

    }

}

**OUTPUT :**



**ERROR :**

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

**WEEK – 6**

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

**Program :**

class vehicle{

    String company;

    String model;

    String fuel;

    int capacity;

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

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

        this.company=company;

        this.model=model;

        this.fuel=fuel;

        this.capacity=capacity;

    }

}

class car extends vehicle{

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

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

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

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

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

    }

}

class poly1{

    public static void main(String[] args){

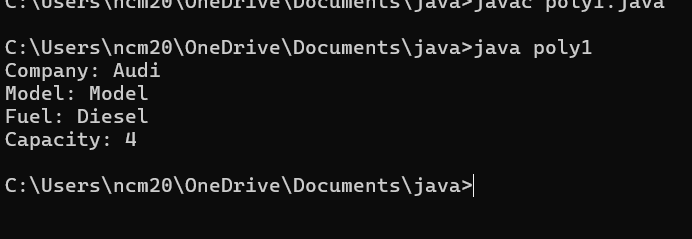
        car car1=new car();

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

    }

}

**OUTPUT :**

****

**ERRORS :**

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

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

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

(ii)PG admissions require a minimum of 70%

**Program :**

class College{

    String name;

    int percentage;

    void geteligibility(String name,int percentage){

        this.name=name;

        this.percentage=percentage;

    }

}

class UG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=60){

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

        }

        else{

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

        }

    }

}

class PG extends College{

    void geteligibility(String name,int percentage){

        if (percentage>=70){

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

        }

        else{

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

        }

    }

}

class poly2{

    public static void main(String[] args){

        UG ug=new UG();

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

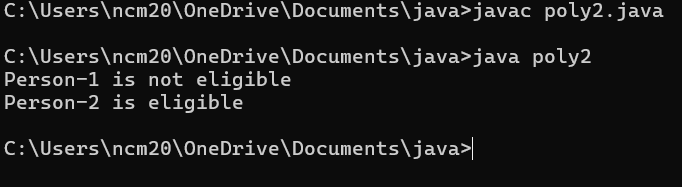
        PG pg=new PG();

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

    }

}

**OUPUT :**

****

**ERROR :**

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

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

(i) Add two integers.

(ii) Add two doubles.

(iii) Add three integers.

**Program :**

class Calcee{

    public int add(int a,int b){

        return a+b;

    }

    public double add(double a,double b){

        return a+b;

    }

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

        return a+b+c;

    }

}

class poly3{

    public static void main(String[] args){

        Calcee C1=new Calcee();

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

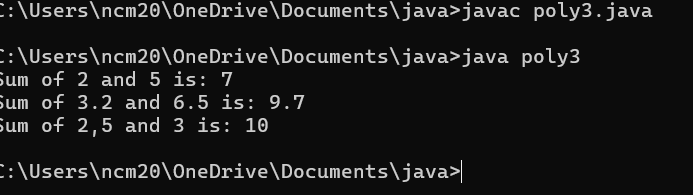
        System.out.println("Sum of 3.2 and 6.5 is: "+C1.add(3.2,6.5));

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

    }

}

**OUTPUT :**

****

**ERROR :**

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

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

**Program :**

class Shape { // class shape

    void calculateArea( int a) { // method 1

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

    }

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

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

    }

}

class circle extends Shape { // inheritance class

    void calculateArea(double a){ // method overloading

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

  } }

class main { // main program

    public static void main(String[] args) {

        // creating objects for class

        Shape s = new Shape();

        circle c = new circle();

        // calling methods

        s.calculateArea(4);

        System.out.println("    ");

        s.calculateArea(4, 5);

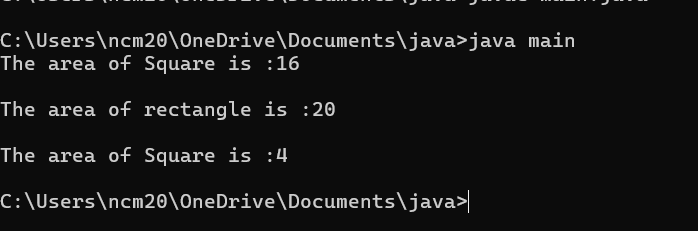
        System.out.println("    ");

        c.calculateArea(2);

    }

}

**OUTPUT :**

****

**ERRORS :**

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

**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.**

**Important points :**

1. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

2.An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

3. Here we used hierarchy inheritance concept means every sub class extends super class.

**Program:**

abstract class animal{

    abstract void sound();

}

class lion extends animal{

    void sound(){

        System.out.println("lion roars");

}

}

class tiger extends animal{

    void sound(){

        System.out.println("tiger growls");

    }

}

class ABanimal{

    public static void main(String[] args) {

        System.out.println("NAME : chaitanya");

        System.out.println("ROLL NO : 24233");

        System.out.println("SEC: CSE-c");

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

        lion l = new lion();

        l.sound();

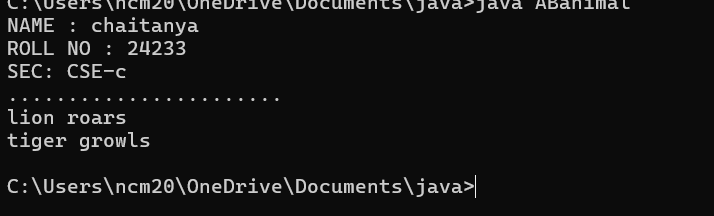
        tiger t = new tiger();

        t.sound();

    }

}

**Output:**

****

**Error table :**

|  |  |  |
| --- | --- | --- |
| S.NO |  |  |
| 1 | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| 2 | Method | We need to provide return type to the method. |
| 3 | Data type | As per need provide data type 4 Abstract method |
| 4 | Abstract method | Implementation in subclass only |

1. Aim : Write a Java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extend the Shape3D class and implement the respective methods to calculate the volume and surface area of each shape.

**Program:**

abstract class Shape3D {

    public abstract double calculateVolume();

    public abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

    private double radius;

    public Sphere(double radius) {

        this.radius = radius;

    }

    public double calculateVolume() {

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

    }

    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;

    }

    public double calculateVolume() {

        return Math.pow(side, 3);

    }

    public double calculateSurfaceArea() {

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

    }

}

public class  Shapees{

    public static void main(String[] args) {

        System.out.println("chaitanya");

        System.out.println( "24233");

        System.out.println( "Section C");

        Shape3D sphere = new Sphere(5);

        Shape3D cube = new Cube(4);

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

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

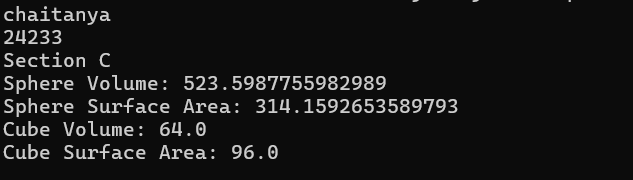
        System.out.println("Cube Volume: " + cube.calculateVolume());

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

    }

}

**Output:**



**Error table :**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Main Class | Better to create main class name same as the file you saved and first letter is capital. |
| 2 | Data type | As per need provide data type 3 Abstract method Implementation in subclass only |
| 3 | Abstract method | Implementation in subclass only |

**Important points :**

1. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.

2. An abstract method is a method declared in an abstract class that does not have an implementation, meaning it doesn't have a body within the abstract class.

3. Understanding the calling of constructor.

1. **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 print pattern (int n) and a concrete method to display the pattern tittle. Implement two sub**

**class :**

**1.star pattern -prints a right-angled triangle of stars(\*)**

**2.number pattern-prints a right angled triangle of increasing numbers. In the main () method,create objects of both subclasses and print the pattern for a given number of rows.**

**Program:**

abstract class PatternPrinter{

    public abstract void printPattern(int n);

    public void printTitle(String title) {

        System.out.println(title);

    }

}

class StarPattern extends PatternPrinter {

    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 {

    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 pattern {

    public static void main(String[] args) {

System.out.println("chaitanya");

System.out.println("24233");

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

        int rows=5;

        PatternPrinter starPattern = new StarPattern();

        PatternPrinter numberPattern = new NumberPattern();

        starPattern.printTitle("Star Pattern:");

        starPattern.printPattern(rows);

        System.out.println();

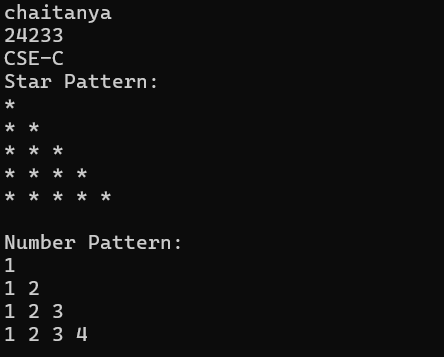
        numberPattern.printTitle("Number Pattern:");

        numberPattern.printPattern(rows);

    }

}

**Output:**

****

**Error table :**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | **Main Class** | **Better to create main class name same as the file you saved and first letter is capital.** |
| **2** | **Data type** | **As per need provide data type** |
| **3** | **Syntax** | **in for Initializing value and condition should be correct** |
| **4** | **overridding** | **Same method name** |

**Important points :**

**1. Here we used nested for loop concept the block of code is executed until the condition is false.**

**2. Here the logic very important .**

**3. Here we used abstract class concept it is a restricted class that cannot be instantiated (cannot have objects created directly) and is typically designed to be extended by subclasses.**

**WEEK-8**

**1.Write a java program to create an interface shape with getPerimeter() method. Create three classes Rectangle, Circle, and Triangle that implement the shape interface implement the getPerimeter() method for each of the three classes.**

**Program:**

interface Shapee {

    float pi = 3.14f;

    int getperimeter();

}

class Rectangle implements Shapee {

    int length;

    int breadth;

    public Rectangle(int length, int breadth) {

        this.length = length;

        this.breadth = breadth;

    }

    public int getperimeter() {

        return 2 \* (length + breadth);

    }

}

class Circle implements Shapee {

    private int radius;

    public Circle(int radius) {

    this.radius = radius;   }

    public int getperimeter() {

        return (int) (2 \* pi \* radius);

    }}

class Triangle implements Shapee {

    private int side1;

    private int side2;

    private int side3;

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

        this.side1 = side1;

        this.side2 = side2;

        this.side3 = side3;

    }

    public int getperimeter() {

        return (side1 + side2 + side3);

    }}

class InShape {

    public static void main(String[] args) {

    System.out.println("chaitanya");

    System.out.println("24233");

    System.out.println("CSE-C");

        Rectangle r = new Rectangle(5, 6);

        System.out.println("Rectangle perimeter: " + r.getperimeter());

        Circle c = new Circle(7);

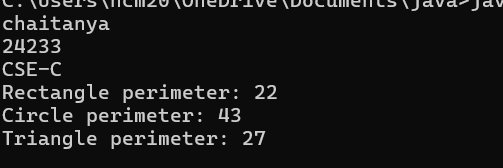
        System.out.println("Circle perimeter: " + c.getperimeter());

        Triangle t = new Triangle(8, 9, 10);

        System.out.println("Triangle perimeter: " + t.getperimeter());

    }}

**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Method not implemented | |  | | --- | |  |  |  | | --- | | Ensure all classes implement the getperimeter() method from the interface. | |
| 2 | |  | | --- | | Incorrect method name | | |  | | --- | | Use the exact method name getperimeter() as declared in the interface. | |
| 3 | Missing cast in Circle calculation | Cast the result to int: (int)(2 \* pi \* radius). |
| 4 | Access modifier confusion | |  | | --- | |  |  |  | | --- | | Clearly define public, private, or default where appropriate. | |

**Important Points:**

1. Interfaces can have constants: pi in Shapee is implicitly public static final.
2. All classes implement the same interface, so they must define the getperimeter() method.
3. Casting is done in Circle: (int)(2 \* pi \* radius) — this truncates the result to an integer.
4. Access Modifiers: Notice mixed use of public/default (e.g., length is package-private in Rectangle)

**2.Write a java program to create an interface playable with a method play that takes no arguments and returns play that takes no arguments and returns void. Create 3 classes Football, Volleyball, Basketball that implements the playable interface and override the play method to play the respective sports.**

**Program:**

public interface playable {

    void play();

}

class Football implements playable {

    public void play() {

        System.out.println("Football is being played...");

}}

class Vollyball implements playable {

    public void play() {

        System.out.println("Vollyball is being played...");

    }}

class Basketball implements playable {

    public void play() {

        System.out.println("Basketball is being played...");

    }}

class played1 {

    public static void main(String[] args) {

  System.out.println("My name is : chaitanya");

  System.out.println("My roll.no : 24233");

  System.out.println("My class-sec : CSE-C");

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

        Football fot = new Football();

        fot.play();

        Vollyball voly = new Vollyball();

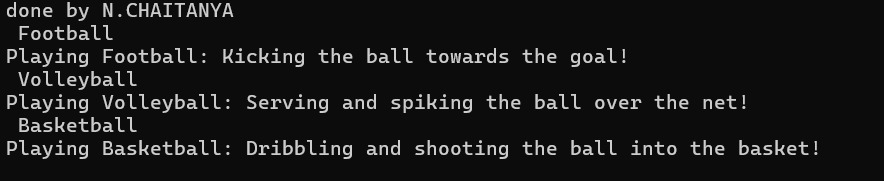
        voly.play();

        Basketball bask = new Basketball();

        bask.play();

}}

**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | |  | | --- | | **Interface not fully implemented** | | |  | | --- | | **Implement the play() method in all classes.** |   **.** |
| **2** | |  | | --- | | **Class/interface name mismatch** |  |  | | --- | |  | | **Make sure class and interface names match exactly (e.g., intplayable).** |
| **3** | |  | | --- | | **Method name spelling error** |  |  | | --- | |  | | **Use correct syntax: System.out.println("...");.** |
| **4** | |  | | --- | | **Object reference typo** |  |  | | --- | | **.** | | **Maintain consistent spelling in class names and object references** |

**Important Points:**

1. **Polymorphism is possible with intplayable but not used here (all objects are called separately).**
2. **Each class provides its own implementation of play().**
3. **Could be enhanced by using an array or list of intplayable objects and a loop.**

**3.Write a java program to implement a login system using interfaces**

**Program:**

interface loginSystem{

    public boolean  login(String username,String password);

}

class Universitylogin implements loginSystem {

    public boolean login(String username, String password) {

        if(username == "ashish1" && password == "cse1111") {

            System.out.println("Successfully logged in..");

            return true;

    }

    else {

        System.out.println("Invalid username or password");

        return false;

    }

}}

class login{

    public static void main(String[] args) {

System.out.println("chaitanya");

System.out.println("24233");

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

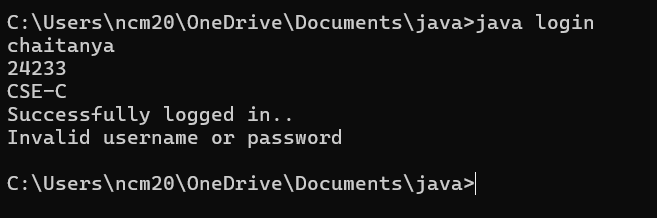
        Universitylogin ul = new Universitylogin();

        ul.login("ashish1", "cse1111");

        ul.login("ashih","ashish01");

    }}

**Output:**



**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | |  | | --- | | String comparison using == |  |  | | --- | | Use | | username.equals("ashish1") instead of username == "ashish1". |
| 2 | |  | | --- | | Missing return statement |  |  | | --- | |  | | Ensure login() returns true or false as required. |
| 3 | |  | | --- | | Poor class naming convention |  |  | | --- | |  | | Rename class login to Login (PascalCase) to follow Java conventions. |
| 4 | |  | | --- | | Incorrect method call in main |  |  | | --- | |  | | Call ul.login("username", "password") correctly with matching arguments. |

**Important Points:**

1. Use .equals() for string comparison in Java to compare values.
2. Interface method is correctly implemented by Universitylogin.
3. The method prints a success or failure message based on input.
4. Proper boolean return allows flexible control in further development.