**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB REPORT**



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

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**Name: M.JYOTHEESWAR**

**Roll No: 24230**

**Verified By :**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **s.no** | **Week** | **Experiment** | **Page.no** |
| **1.** | **Week-1** | **1.Installation of jdk21**  **2. Write a simple java program for printing Name, Class, Roll No, of a Student** |  |
| **2.** | **Week-2** | **1.Write a java program to find simple interest where all inputs are taken from user**  **2.Write a java program to calculate factorial of a number , read the input from user**  **3.Write a java program to calculate the Fibonacci Sequence of a input taken from user**  **4.a: Write a java program to convert temperature from Celsius to Fahrenheit**  **B: Write a java program to convert temperature from Fahrenheit to Celsius**  **5. Write a java program to calculate the area of rectangle**  **6. Write a java program to calculate the area of triangle by using heron’s formula** |  |
| **3.** | **Week-3** | **Create a 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**  **b) Create a class bankAccount with elements deposit() and Withdrawl** |  |
| **4.** | **Week-4** | **a. Write a java program with class named book , the class should contain various attributes such as title , author , year of publication , it should also contain a constructor parameters which initializes title , author , year of publication**  **Create a method which displays details of the book**  **Display details of two books**  **b. Create a java program with class name myclass with static variable count of int type ; initialized to zero and a constant variable “pi’**  **Of type double initialized to 3.14 as attribute of that class , now define a constructor for “ my class “ that increment the count variable each time an object of my class is created count = 0 ,finally print the final values of “count” ans pi variable ,create 3 objects** |  |
| **5.** | **Week-5** | **a)create a calculator using the operation including adding, subtraction,multiplication,and division using multilevel inheritance and display desired output**  **every class one method (int a ,int b)**  **pass values as parameter during ,method call** |  |
| **6.** | **Week-6** | **a: 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**  **b: A college is developing an automated admission system that verifies student**  **eligibility for UG and 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%**  **C: AIM : Create a Shape class with a method calculateArea() that is overloaded forDifferent shapes. Then, create a subclass circle that overrides the calculateArea() method for a circle**  **D: . Create a calculator class with overloaded methods to perform addition**   1. **Add two integers 2. Add two doubles 3. Add three integers** |  |
| **7.** |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 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**:

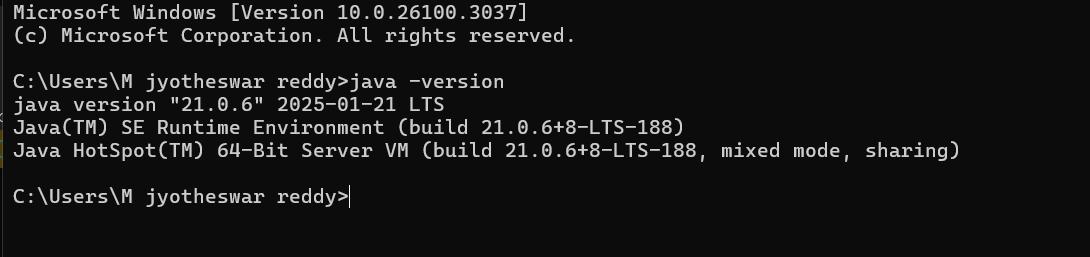
**Variable value:** The folder address where JDK is installed (like

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:P.MANOJ");

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

System.out.println("Roll No:24241");

}

}

**Output: -**

A computer screen with white text

AI-generated content may be incorrect.

ERROR TABLE:

|  |  |  |
| --- | --- | --- |
| 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:**

**import java.util.Scanner;**

**class Main {**

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

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

**System.out.print("Enter the principal: ");**

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

**System.out.print("Enter the rate: ");**

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

**System.out.print("Enter the time: ");**

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

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

**System.out.println("Principal: " + principal);**

**System.out.println("Interest Rate: " + rate);**

**System.out.println("Time Duration: " + time);**

**System.out.println("Simple Interest: " + interest);**

**input.close();**

**}**

**}**

**Output:**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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:**

**class Test {**

**static int factorial(int n)**

**{**

**int res = 1, i;**

**for (i = 2; i <= n; i++)**

**res \*= i;**

**return res;**

**}**

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

**{**

**int num = 5;**

**System.out.println("Factorial of " + num + " is "**

**+ factorial(5));**

**}**

**}**

**Output**

****

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 TABLE:**

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

**4)AIM:**

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

**Code:**

**class fah {**

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

**{**

**double f = 50.0, c = 0.0;**

**c = (f - 32) / 1.8;**

**System.out.println(**

**"value of temperature in celsius:" + c);**

**}**

**}**

**Output:**

**A black screen with white text

AI-generated content may be incorrect.**

**ERROR TABLE:**

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

**5)AIM:**

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

**Code:**

**class celsiustofahrenheit {**

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

**{**

**double celsius = 10.0, fahrenheit = 0.0;**

**fahrenheit = (celsius \* 1.8) + 32;**

**System.out.println(**

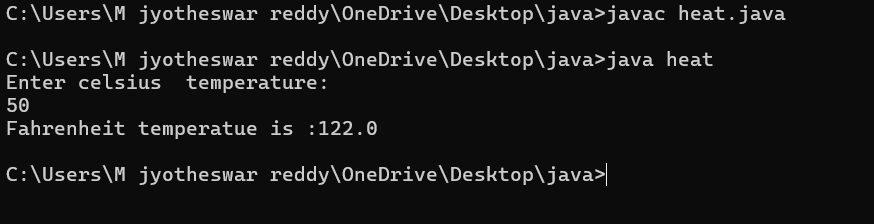
**" value of temperature in fahrenheit:"**

**+ fahrenheit);**

**}**

**}**

**Output:**

****

**ERROR TABLE:**

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

**6)AIM:**

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

**Code:**

**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 computer screen shot of a black screen

AI-generated content may be incorrect.

ERROR TABLE:

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

**7)AIM:**

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

**take the input from the user**

**Code:**

**port java.io.\*;**

**class GFG {**

**static double area(double h, double b)**

**{**

**return (h \* b) / 2;**

**}**

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

**{**

**double h = 10;**

**double b = 5;**

**System.out.println("Area of the triangle: "**

**+ area(h, b));**

**}**

**}**

**Output:**

****

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

ERROR TABLE:

**WEEK 3**

**1)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**

Class Diagram:

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

**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 manoj\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:**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**Concepts to be known:**

1. (String car\_color,String car\_brand,String fuel\_type,int mileage){ } – It is a constructor (method with name same as class), which requires parameters such as car\_color (String data-type) and so on.
2. this.car\_color=car\_color; - “this” is a default method, which is used to point to the instance variables.
3. public void start(){} – used to declare a method, which will return nothing(void) in public accessibility.
4. Car car1=new Car("Red","Maruti","Diesel",20); - used to create a object in class Car, with object name as car1.

**2)AIM:**

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

**CLASS DIAGRAM:**

|  |
| --- |
| **CLASS = BankAccount** |
| **ATTRIBUTES:**  **- Acchname: String**  **- Accnumber: int**  **- curramount: double** |
| **METHODS:**  **+ BankAccount(Acchname: String, Accnumber: int, curramount: double)**  **+ Withdrawl:Void**  **+Deposit :Void** |
|  |

**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:**

A black screen with white text

AI-generated content may be incorrect.

**ERROR TABLE:**

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

**Concepts to be known:**

1. private String name; - Used to declare a variable named name, with data type as String with private accessibility.
2. BankAccount(String name,int Accno,int CurrBal){ } – It is a constructor (method with name same as class), which requires parameters such as name (String data-type) and so on.
3. this.CurrBal=CurrBal; - “this” is a default method, which is used to point to the instance variables.
4. public void withdraw(int WAmt){ } – used to declare a method, which will return nothing(void) in public accessibility, which requires a parameter WAmt(integer data type).
5. public int deposit(int DAmt){} - used to declare a method, which will return integer data type in public accessibility, which requires a parameter DAmt(integer data type).
6. BankAccount cust1=new BankAccount("Ram",5587,20000); - used to create a object in class BankAccount, with object name as cust1.
7. cust1.withdraw(50000); - Calling a method, under object cust1, by passing a parameter.

System.out.println("Your current balance after depositing money is:"+cust1.deposit(25000)); - Deposit method will return the value, which will be directly printed.

WEEK-4

**1.AIM:**

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

and create a method which displays the details of 2 books.

**CLASS DIAGRAM:**

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

**Code:**

**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();**

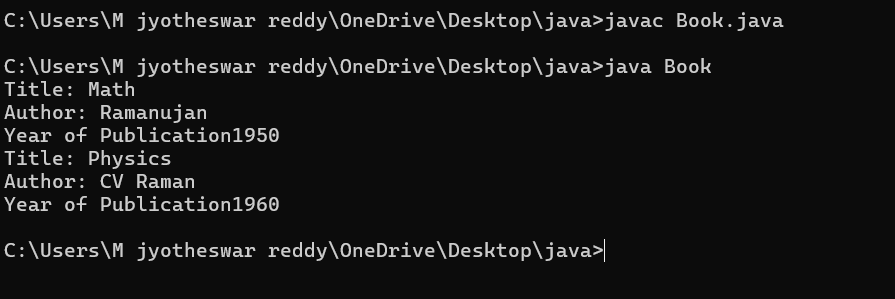
**b2.displayDetails();**

**}**

**}**

**Output:**

**1)positive case:**

****

**2)negative case:**

**ERROR TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **ERROR TYPE** | **Reason for error** | **Rectification** |
| **1.** | Syntax error | No semicolon | Semicolon added |
| **2.** | Runtime error | Incorrect path | Copied correct path |

**IMPORTANT POINTS:**

1. **Constructor**:

* The constructor Book(String, String, int) is used to initialize the object when it is created.
* The keyword **this** is used to differentiate between class attributes and constructor parameters.

2.**Method**:

* The method displayDetails() is used to display the book details.
* The **System.out.println()** method prints the details to the console.

3. **Object Creation**:

* Two objects b1 and b2 are created using the constructor.

**2)AIM:**

write a java program with class named “myclass” with a static variable

count of int type. intialize it to zero and a constant variable “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 the final values of ‘count’ and ‘pi’ variables**

**and create 3 objects.**

**CLASS DIAGRAM:**

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

**code:**

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:**

**1)positive case:**

**A screen shot of a computer

AI-generated content may be incorrect.**

1. **negative case:**

**ERROR TABLE:**

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

**Week -5**

**1) Aim:**

**create a calculator using the operation including adding, subtraction,multiplication,and division using multilevel inheritance and display desired output every class one method (int a ,int b)pass**

**values as parameter during method call.**

**Classs diagram:**

|  |
| --- |
| **Class : addition** |
| **+ add( int a ,int b)** |
|  |

|  |
| --- |
| **Class : subtraction** |
| **+ sub( int a ,int b)** |

|  |
| --- |
| **Class : multiplication** |
| **+ multi( int a ,int b)** |

|  |
| --- |
| **Class : divission** |
| **+ div( int a ,int b)** |

**code:**

**class addition{**

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

**System.out.println("addition is done in calculator below:");**

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

**int addition = a+b;**

**return addition;**

**}**

**}**

**class subtraction extends addition {**

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

**System.out.println("subtraction is done in calculator below:");**

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

**int subtraction = a-b;**

**return subtraction;**

**}**

**}**

**class multiplication extends subtraction**

**{**

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

**System.out.println("multiplication is done in calculator below:");**

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

**int multiplication = a\*b;**

**return multiplication;**

**}**

**}**

**class division extends multiplication{**

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

**System.out.println("divission is done in calculator below:");**

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

**int division = a/b;**

**return division;**

**}**

**}**

**class calculator**

**{**

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

**{**

**division obj = new division();**

**addition img = new addition();**

**subtraction del = new subtraction();**

**multiplication zym = new multiplication();**

**System.out.println("Addition is:"+ img.add(99,101));**

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

**System.out.println ("Subtraction is:"+del.sub(200,100));**

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

**System.out.println("Multiplication is:"+zym.mult(50,4));**

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

**System.out.println("Division is:"+obj.div(1000,100));**

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

**}**

**}**

**Output:**

**1.positive case:**

A screenshot of a computer program

AI-generated content may be incorrect.

**2.negative case:**

**A computer screen with white text

AI-generated content may be incorrect.**

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**important points :**

1. **used contructor and ssigned values as int a and int b.**
2. **used multilevel unheritance .**
3. **used this keyword to assign values .**

**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, bikes and truck and they need a program to Store details about each vehicle, such as brand and speed Cars should have an additional property: number of doors Bikes should have a property indicating whether they have gears or not The system should also include a function to display details about each vehicle And indicate when a vehicle is starting**

Class diagram:

|  |
| --- |
| Class vehicle |
| + brand:string  + speed :int |
| Vehicle(String brand ,int speed) |

|  |
| --- |
| Class car |
| +noofdoors:int |
| car(String brand ,int speed ,noofdoors) |

|  |
| --- |
| Class bike |
| + gear : bool |
| bike(String brand ,int speed,Boolean grars) |

|  |
| --- |
| Class truck |
| + weight:int |
| truck(String brand ,int speed) |

code:

class vehicle{

String brand;

int speed;

public vehicle(String brand,int speed){

this.brand=brand;

this.speed=speed;

}

public static void main(String[] args) {

car obj1=new car("maruti",140,70);

bike obj2=new bike("unicorn",80,true);

truck obj3=new truck("zimban",120,20);

}

}

class car extends vehicle{

int noofdoors;

public car(String brand, int speed,int noofdoors) {

super(brand, speed);

this.noofdoors=noofdoors;

System.out.println(brand);

System.out.println(speed);

System.out.println(noofdoors);

}

}

class bike extends vehicle{

boolean gears;

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

super(brand, speed);

this.gears=gears;

System.out.println(brand);

System.out.println(speed);

System.out.println(gears);

}

}

class truck extends vehicle{

int weight;

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

super(brand,speed);

this.weight=weight;

System.out.println(brand);

System.out.println(speed);

System.out.println(weight);

}

public static void main(String[] args) {

car obj1=new car("maruti",143,70);

bike obj2=new bike("unicorn",80,true);

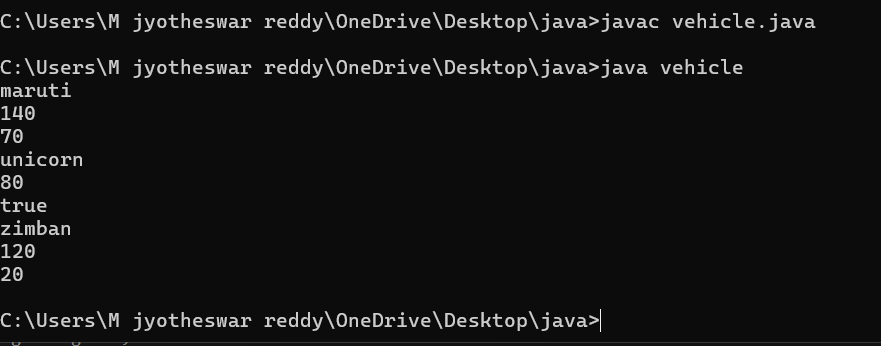
truck obj3=new truck("zimban",120,20);

}

}

**Out put:**

1. **positive case :**

****

1. **negative case:**

**A computer screen with white text

AI-generated content may be incorrect.**

Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**Important points:**

1.Inheritance:

Obtaining the qualities from parent class.

2.hirechial inheritance: different child class obtaing qualities from one parent.

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 a about car**

CLASS DIAGRAM:

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

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

**Code:**

class vehicle {

String brand;

int speed;

int capacity;

Vehicle(String brand, int speed, int capacity) {

this.brand = brand;

this.speed = speed;

this.capacity = capacity;

}

public void displayInfo() {

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

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

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

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

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

}

}

class Car extends Vehicle {

Car(String brand, int speed, int capacity) {

super(brand, speed, capacity);

}

public void displayInfo() {

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

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

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

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

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

}

}

public class world {

public static void main(String[] args) {

Vehicle vehicle = new Vehicle("bmw", 150, 5);

vehicle.displayInfo();

System.out.println();

Car car = new Car("honda", 200, 7);

System.out.println("car details are given below :");

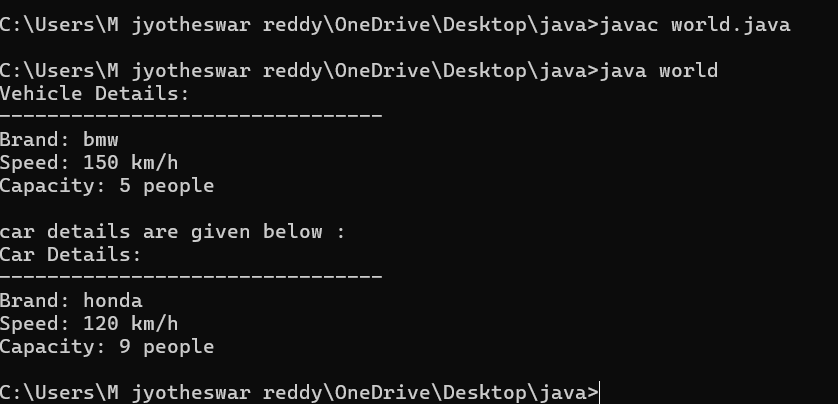
car.displayInfo();

}

}

OUT PUT :

1.POSITIVE CASE :



2.NEGATIVE CASE:

A computer screen shot of a black screen

AI-generated content may be incorrect.

Important points:

**1.Inheritance** allows the Car class to reuse the Vehicle class

functionality without repeating code.

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

3. **Polymorphism** makes the code flexible, allowing different classes

(e.g., Car, Bike) to provide customized behavior for displayInfo().

2.AIM:

A college is developing an automated admission system that verifies student eligibility for UG and 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 :

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

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

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

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

Code:

class University {

String name;

int percentile;

University(String name, int percentile) {

this.name = name;

this.percentile = percentile;

}

public void office() {

System.out.println("Admission criteria are mentioned below.");

}

}

class PG extends University {

PG(String name, int percentile) {

super(name, percentile);

}

public void office() {

if (percentile >= 70) {

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

} else {

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

}

}

}

class UG extends University {

UG(String name, int percentile) {

super(name, percentile);

}

public void office() {

if (percentile >= 60) {

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

} else {

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

}

}

}

public class admissions {

public static void main(String[] args) {

UG ugStudent = new UG("uday", 95);

PG pgStudent = new PG("jyotheeswar", 96);

ugStudent.office();

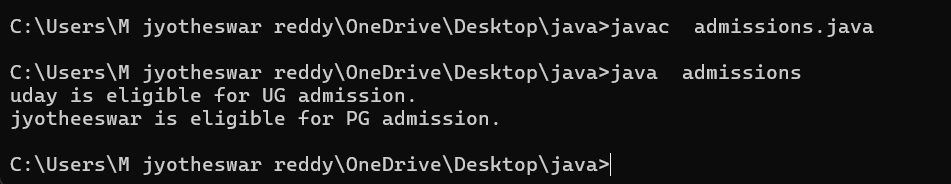
pgStudent.office();

}

}

Output :

1.positive case :



2.negative case:

A black screen with white text

AI-generated content may be incorrect.

Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

IMPORTANT POINTS:

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

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

3. **Polymorphism** enables the same office() method to behave

differently for PG and UG objects based on their typ

3)AIM :

Create a Shape class with a method calculateArea() that is overloaded forDifferent shapes. Then, create a subclass circle that overrides the calculateArea() method for a circle

**Class diagram :**

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

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

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

code:

class Shape {

public void calculateArea(int length, int width) {

int area = length \* width;

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

}

public void calculateArea(double base, double height) {

double area = 0.5 \* base \* height;

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

}

public void calculateArea(int side) {

int area = side \* side;

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

}

}

class Circle extends Shape {

public void calculateArea(double radius) {

double area = 3.14 \* radius \* radius;

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

}

}

public class area {

public static void main(String[] args) {

Shape shape = new Shape();

shape.calculateArea(5, 60);

shape.calculateArea(7.0, 34.0);

shape.calculateArea(9);

Circle circle = new Circle();

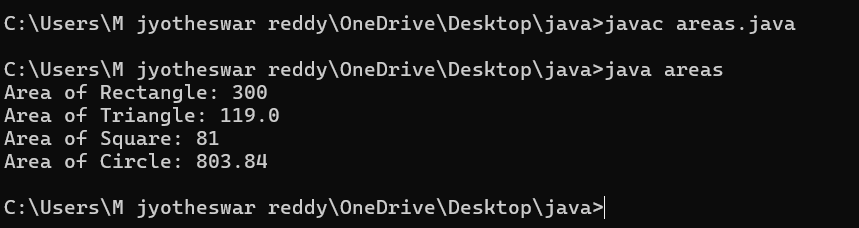
circle.calculateArea(16.0);

}

}

Out put:

1.positive case :



2.negative case:

A black screen with white text

AI-generated content may be incorrect.

Error table:

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
|  |  |  |  |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

IMPORTANT POINTS :

**1.Method Overloading** allows the Shape class to calculate areas for different shapes (rectangle, triangle, square) using multiple calculateArea() methods with different parameters.

**2.Method Overriding** in the Circle class provides a custom implementation of calculateArea() for calculating the area of a circle.

**3.Polymorphism** enables objects of different types (e.g., Shape, Circle)

to call the appropriate version of calculateArea() based on the object type and input parameters.

4)**AIM:**

**Create a calculator class with overloaded methods to perform addition**

**1.Add two integers**

**2.Add two doubles**

**3.Add three integers**

**CLASS DIAGRAM:**

|  |
| --- |
| CLASS CALCULATORMAN |
| +add(int, int): int  +add(double, double): double + +add(int, int, int): int |
|  |

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

**CODE:**

**class calculatorman**

**{**

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

**{**

**System.out.println(" addition of in type: ");**

**return a+b;**

**}**

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

**{ System.out.println(" addition of double type: ");**

**return a+b;**

**}**

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

**{**

**System.out.println(" addition of int type 3 variables: ");**

**return a+b+c;**

**}**

**}**

**class HOSTEL**

**{**

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

**calculatorman kaala = new calculatorman();**

**System.out.println("Addition of 100 and 100 is:"+kaala.add(100,100));**

**System.out.println("Addition of 99.3 and 0.7 is:"+kaala.add(99.3 , 0.7));**

**System.out.println("Addition of 7 7 7 is:"+kaala.add(7,7,7));**

**System.out.println(" did by : M.jyotheeswar");**

**}**

**}**

**OUTPUT:**

**1.POSITIVE CASE :**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**2.NEGATIVE CASE:**

**A computer screen with white text

AI-generated content may be incorrect.**

**ERROR TABLE :**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Reason for error** | **rectification** |
| **1.** | **Logical error** | **Incorrect formula** | **Correcting formula** |
| **2.** | **Runtime errror** | **Incorrect path** | **Using correct path** |
| **3.** | **Syntax error** | **Incorrect symbols** | **Using correct symbols** |

**IMPORTANT POINTS :**

**1. This program demonstrates method overloading, where multiple add() methods have the same name but different parameter types or counts.**

**2. Each method performs addition and prints the type of addition**

**being done (int, double, or three integers).**

**3. In the main() method, the correct overloaded method is called**

**based on the arguments passed, showcasing polymorphism.**

**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 extends the animal

Class and implements the sound() method to make a specific sound

for each animal

CLASS DIAGRAM:

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

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

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

**PROGRAM:**

abstract class Animal {

    public abstract void sound();

}

class Lion extends Animal {

    public void sound() {

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

    }}

class Tiger extends Animal {

    public void sound() {

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

    }}

public class forest17 {

    public static void main(String[] args) {

System.out.println("jyotheeswar cse-c 24230");

         Lion  l = new Lion();

         Tiger t  = new Tiger();

        l.sound();

        t.sound();

    }}

OUTPUT:

A black screen with white text

AI-generated content may be incorrect.

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Error type** | **Cause** | **Rectification** |
| **1** | **Static reference to non-static method** | **Calling non-static method from static context** | **Create object correctly** |
| **2** | **Instantiation of abstract class(OOP error)** | **Trying to create an object of an abstract class** | **Use subclass or interface implementation** |
| **3** | **Syntax error** | **Missing parenthesis** | **Adding parenthesis** |

**IMPORTANT POINTS:**

Use of Abstract Class for Generalization

The Animal class is abstract and defines a common method sound() for all

animals, which ensures that all subclasses provide their own specific

implementation.

Method Implementation in Subclasses (Lion & Tiger)

Subclasses Lion and Tiger override the abstract method sound() to give

specific behavior, showing how abstract classes enforce method

implementation in child classes.

Polymorphism in Action

By using Animal references to call sound() on Lion and Tiger objects, the

program demonstrates runtime polymorphism, allowing the program to

decide which method to call at execution time.

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

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

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

**PROGRAM:**

abstract class Shape3D {

abstract double calculateVolume();

abstract double surfaceArea();

}

class Sphere extends Shape3D {

int r;

public Sphere(int r) {

this.r = r;

}

public double calculateVolume() {

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

}

public double surfaceArea() {

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

}}

class Cube extends Shape3D {

int a;

public Cube(int a) {

this.a = a;

}

public double calculateVolume() {

return Math.pow(a, 3);

}

public double surfaceArea() {

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

}}

class shap {

public static void main(String[] args) {

System.out.println(" jyotheeswar cse-c cse24230");

Sphere obj = new Sphere(6);

Cube obj1 = new Cube(8);

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

System.out.println("Sphere Surface Area: " + obj.surfaceArea());

System.out.println("Cube Volume: " + obj1.calculateVolume());

System.out.println("Cube Surface Area: " + obj1.surfaceArea());

}}

**OUTPUT:**

A computer screen shot of a computer code

AI-generated content may be incorrect.

**Error Table:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Error Type** | **Cause** | **Rectification** |
| **1** | **Return type missing** | **Method doesn’t specify return type** | **Adding the correct return type** |
| **2** | **Instantiation of abstract class** | **Trying to create an object of abstract class without any subclass or interface** | **Use subclass or interface implementation** |
| **3** | **Static reference to Non-static Method** | **Calling non-static method from static context** | **Create object or make method static** |

**IMPORTANT POINTS:**

**Abstract Class Enforcing Common Structure**

The abstract class Shape3D defines the blueprint with the abstract methods calculateVolume() and calculateSurfaceArea(), ensuring that all subclasses

(like Sphere and Cube) must implement these methods.

**Implementation of Shape-Specific Calculations**

Each subclass (Sphere and Cube) provides its own **specific implementation**

of the methods to calculate the volume and surface area using the formulas

appropriate for each 3D shape.

**Polymorphism for Flexible Calculations**

By using Shape3D references to store both Sphere and Cube objects, we demonstrate **runtime polymorphism**, allowing us to call calculateVolume() and calculateSurfaceArea() on the base class type, but the correct method is executed based on the actual object type.

3.**AIM:**

Write a java program using an abstract class to define a method for pattern printing.

Create an abstract class named PatternPrinter with anAbstract method printpattern(int n)

and a concrete method to display the pattern title.

Implement two subclasses

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

2.Number pattern- prints a right angled triangle of increasing numbers.

In the main() method create objects of both subclasses and print the patterns

For a given number of rows.

CLASS DIAGRAM :

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

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

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

**PROGRAM :**

abstract class PatternPrinter{

abstract void printpattern(int n);

public void display(String title)

{

System.out.println("\n=== "+ title+"===");

} }

class starpattern extends PatternPrinter

{

void printpattern(int n)

{

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

{

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

{

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

System.out.println();

}}}

class numberpattern extends PatternPrinter

{

void printpattern(int n)

{

int num=1;

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

{

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

{

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

num++;

}

System.out.println();

}}}

class Main2

{

public static void main(String args[])

{

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

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

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

int rows=5;

PatternPrinter star = new starpattern();

star.display("Number pattern");

star.printpattern(rows);

PatternPrinter number = new numberpattern();

number.display("Number pattern");

number.printpattern(rows);

}}

OUTPUT :

A computer screen shot of a black screen

AI-generated content may be incorrect.

**IMPORTANT POINTS:**

**Abstraction is used to define a common method printpattern() in the abstract**

**class PatternPrinter, allowing different patterns to be printed through subclasses.**

**The starpattern and numberpattern classes implement their own version printpattern(), demonstrating polymorphism.**

**The display() method in the abstract class allows for a consistent header before printing any pattern, making the output structured and easy to manage.**

**WEEK- 8**

1. **AIM:**

Write a Java program to create an interface Shape with the getPerimeter method. Create three classes Rectangle, Circle, and Triangle that implement the Shape interface. Implement the getPerimeter() method for each of the three classes.

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

CLASS DIAGRAM :

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

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

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

**PROGRAM:**

interface Shape {

getPerimeter();

}

class Rectangle implements Shape {

int l, b;

public Rectangle(int l, int b) {

this.l = l;

this.b = b;

}

public void getPerimeter() {

int perimeter = 2 \* (l + b);

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

}

}

class Triangle implements Shape {

int s1, s2, s3;

public Triangle(int s1, int s2, int s3) {

this.s1 = s1;

this.s2 = s2;

this.s3 = s3;

}

public void getPerimeter() {

int perimeter = s1 + s2 + s3;

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

}}

class Circle implements Shape {

int r;

public Circle(int r) {

this.r = r;

}

public void getPerimeter() {

double perimeter = 2 \* Math.PI \* r;

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

}}

class perimeter20 {

public static void main(String[] args) {

System.out.println("jyotheeswar Reddy 24230 sec-C");

Shape obj1 = new Rectangle(12,6);

Shape obj2 = new Triangle(5, 7, 6);

Shape obj3 = new Circle(5);

obj1.getPerimeter();

obj2.getPerimeter();

obj3.getPerimeter();

}}

**OUTPUT:**

A black screen with white text

AI-generated content may be incorrect.

**NEGATIVE CASE:**

A black screen with white text

AI-generated content may be incorrect.

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1 | Syntax error | Rectified by adding void |

**IMPORTANT POINTS:**

Interface Usage for Abstraction:

* The Shape interface defines a contract (getPerimeter() method) that all shape classes must follow.
* This allows for polymorphism — you can handle different shapes uniformly, which is very useful in large applications.

Polymorphism in Action:

* In the main() method, we use the Shape reference to refer to different types of objects (Rectangle, Circle, Triangle).
* This demonstrates runtime polymorphism, where the correct getPerimeter() method is called depending on the object type.

Encapsulation and Implementation:

* Each class (Rectangle, Circle, Triangle) encapsulates its own data (like length, radius, sides) and provides its specific formula for calculating the perimeter.
* This keeps the code clean, modular, and easy to maintain.

**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 :

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

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

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

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

**PROGRAM:**

interface Playable {

void play();

}

class Football implements Playable {

public void play() {

System.out.println("Playing Football: Kicking the ball towards the goal");

}

}

class Volleyball implements Playable {

public void play() {

System.out.println("Playing Volleyball: Bumping, setting, and spiking the ball");

}

}

class Basketball implements Playable {

public void play() {

System.out.println("Playing Basketball: Dribbling and shooting the ball");

}

}

public class TestSports25 {

public static void main(String[] args) {

System.out.println("jyotheeswar Reddy CSE-C 24230");

Playable football = new Football();

Playable volleyball = new Volleyball();

Playable basketball = new Basketball();

football.play();

volleyball.play();

basketball.play();

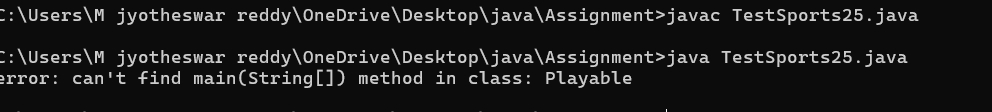
}}

**OUTPUT:**

A screen shot of a computer

AI-generated content may be incorrect.

**NEGATIVE CASE:**



**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.NO | ERROR | RECTIFICATION |
| 1 | Syntax error | Give correct class name |

**IMPORTANT POINTS:**

**Easy Extensibility:**

* Adding new sports is simple: just create a new class that implements Playable and define the play() method.
* This design follows good **object-oriented principles** and makes the program scalable.