**23CSE111**

**OBJECT ORIENTED PROGRAMMING**

**LAB RECORD**



**Department of Computer Science and Engineering,**

**Amrita School of computing,**

**Amrita Vishwa Vidyapeetham, Amaravati Campus.**

|  |  |
| --- | --- |
| **Submitted by** | **Submitted to** |
| **Name: G Shivacharan Reddy** |  |
| **Roll no: 24123** |  |
| **Sec: CSE-B** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Task Name** | **Page No.** | **Remarks** | **Signature** |
| Week-1 | 1. Installation Process of JDK |  |  |  |
|  | 2.Basic java program |  |  |  |
|  | **3.**Simple Java Program for printing basic details of student |  |  |  |
| Week-2 | 1.Java program for simple intrest |  |  |  |
|  | 2.Java program for Area of rectangle |  |  |  |
|  | 3.Java program for area of triangle using herons formula |  |  |  |
|  | 4.(A) Java program for converting temp from celcius to farenheit |  |  |  |
|  | 4.(B) Java program for converting temp from farenheit to celcius |  |  |  |
|  | 5.Java program for factorial of a number |  |  |  |
|  | 6. Java program for Fibonacci series |  |  |  |

**WEEK-1**

**1.HOW TO DOWNLOAD AND INSTALL JDK-21?**

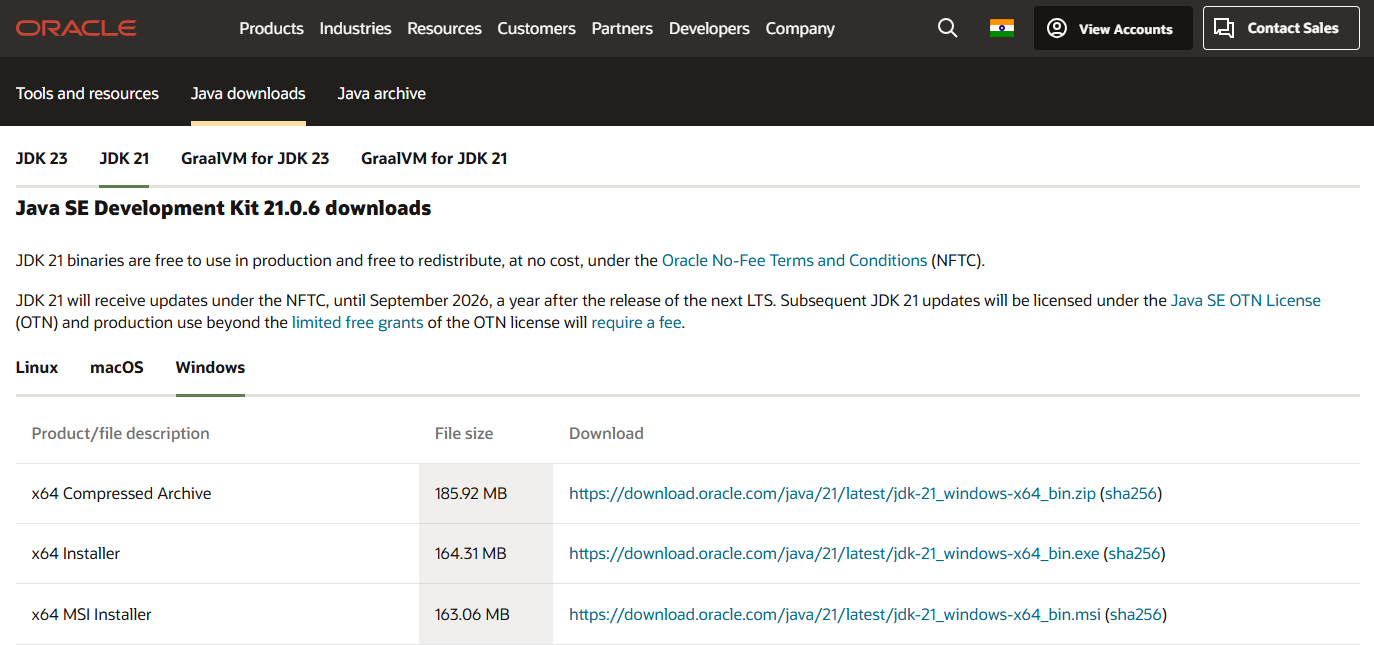
* **Steps for downloading**:

**Step 1)** Open any web browser and search for oracle Java, then go to java downloads page.

**Step 2)** Now we can see different versions of java software like JDK 23, JDK 21, GraalVM for JDK 23 and 21. Though JDK 23 is the latest version, instead of that we download JDK 21 as it is the Long Term Support (LTS) release of Java SE platform.

**Step 3)** After selecting JDK 21 we can see different OS options. We will be downloading it for windows operating system.

**Step 4)** In the windows we have three different files, we will be downloading x64installer.Below is the download link for JDK 21 version for windows OS. <https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe> ([sha256](https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe.sha256))

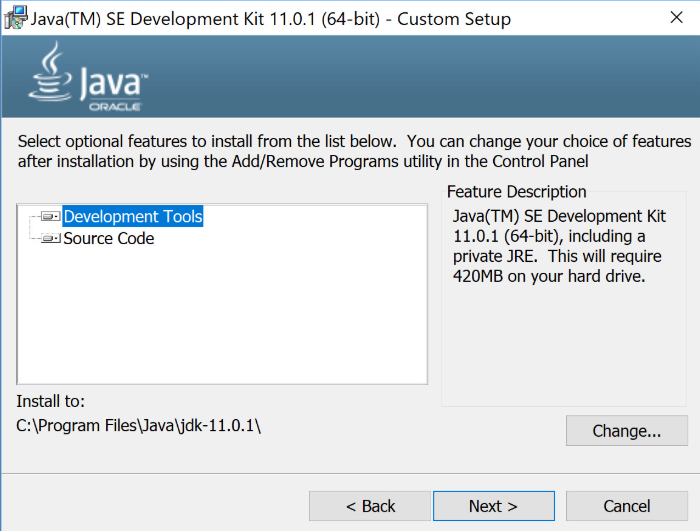
****

* **Steps for installing:**

**Step 5)** Once the download is complete, locate the downloaded file (usually in your Downloads folder).

**Step 6)** Double-click the installer file to run it.

**Step 7)** Click "Yes" if prompted by the User Account Control (UAC) to allow the installer to make changes to your device.



**Step 8)** In the installation wizard, click "Next" to continue.

**Step 9)** Choose the destination folder for the JDK installation (or leave the default path) and click "Next."

**Step 10)** Click "Install" to begin the installation process.

**Step 11)** Wait for the installation to complete and click "Finish" when done.

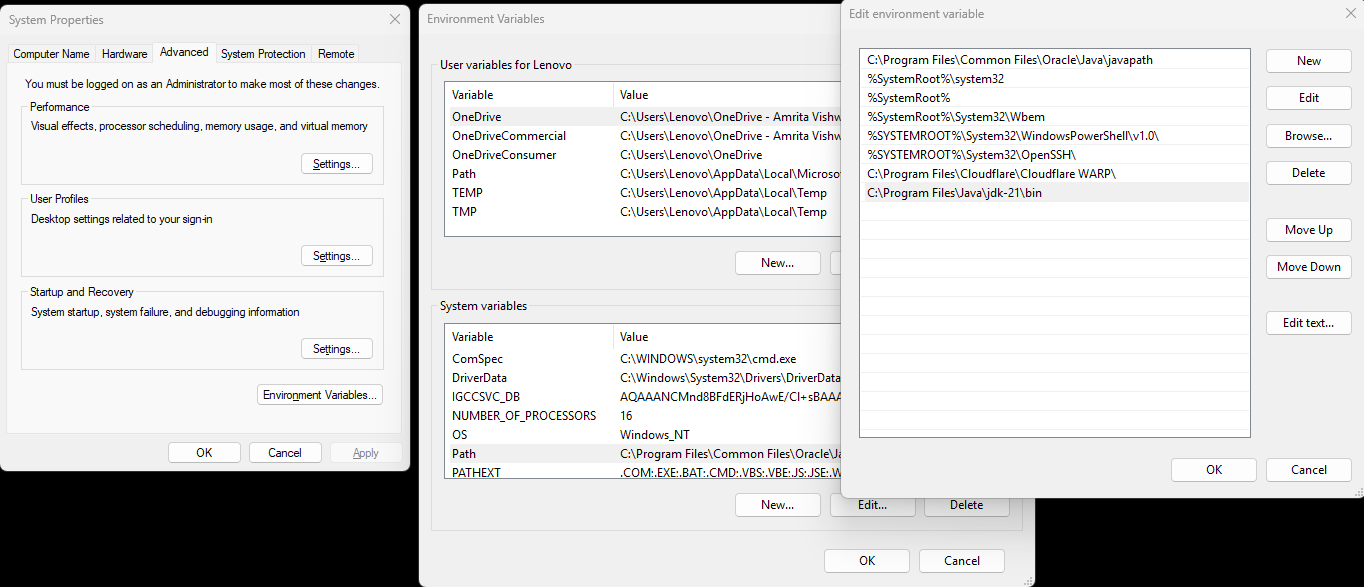
* **Set Environment Variables:**

**Step 12)** Open the Start Menu and search for "Environment Variables."

**Step 13)** In the System Properties window, click on the "Environment Variables" button.

**Step 14)** Under "System variables," click on path variable then click edit button.

**Step 15)** Now go to the file manager, then go to program files.



**Step 16)** In program files folder you can see java folder, open it, in that copy the path of bin file.

**Step 17)** In the edit environment variables paste the java bin file path that copied.

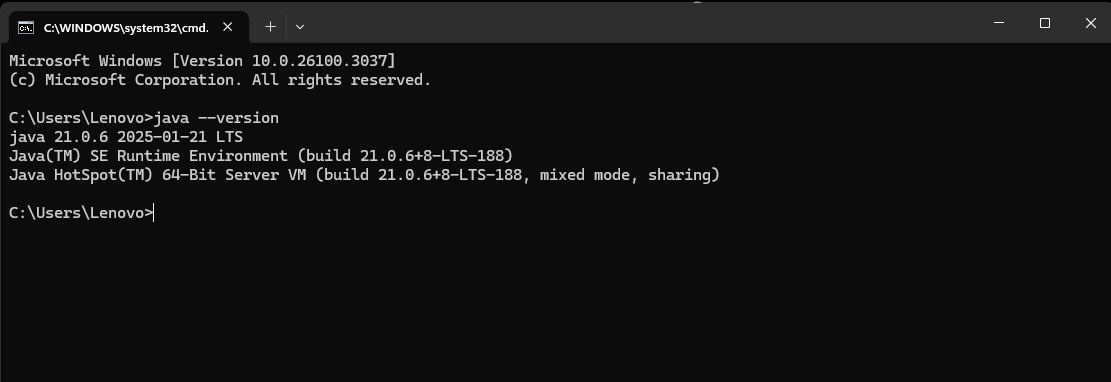
**Step 18)** Click "OK" to close all windows. Then the path is set.

* **Verify Installation:**

**Step 19)** Open Command Prompt by searching for cmd in the Start Menu and clicking on it.

**Step 20)** Type java --version and press Enter.

**Step 21)** If you see the version of JDK 21 then the installation is successful.



**2.Basic Java programs:**

**A) Printing “Hello, world!” statement:**

**PROGRAM:**

**Class bascis {**

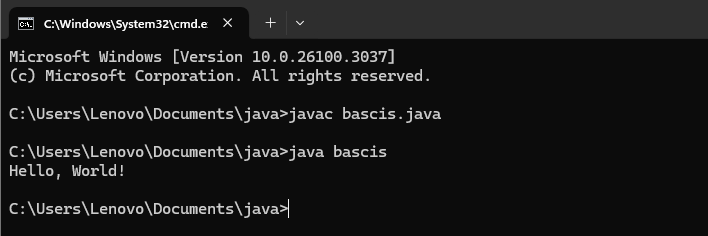
**public static void main(String[] args) {**

**System.out.println("Hello, World!");**

**}**

**}**

**OUTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| **1.** At first I did not save file name and class name as same, so I got an error. | **1.**we should class name same as file name. |

**B) Printing basic details of a Student:**

**Program:**

**class bascis{**

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

**System.out.println("NAME= G Shivacharan Reddy");**

**System.out.println("ROLL N0:24123");**

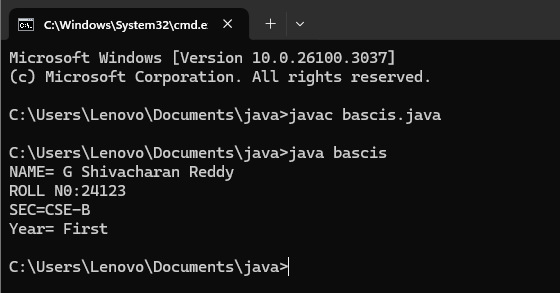
**System.out.println("SEC=CSE-B");**

**System.out.println("Year= First ");**

**}**

**}**

**OUTPUT:**



**ERROR:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| **1.** I used capital C instead of small c in the word “class” so, I got an error, as java language is case sensitive. | **1.** Use small c instead of capital C in the word “class”. |

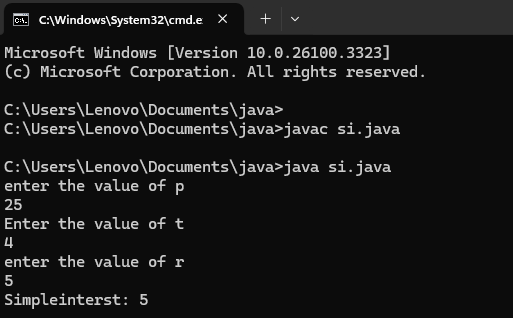
**WEEK-2:**

**PROGRAM-1:**

**Aim:Write a java program for SI**

******

**Output:**

******

**ERROR:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| Not giving parentheses after closing the input. | We must put parentheses after closing the input. |

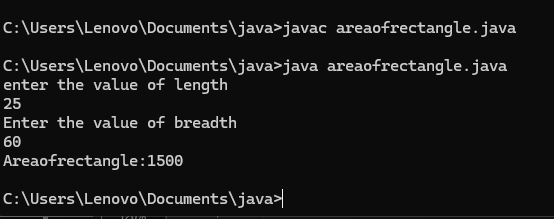
**PROGRAM-2:**

**Aim: Write a program in java for area of**

**rectangle.**

******

**Output:**

******

**ERROR TABLE:**

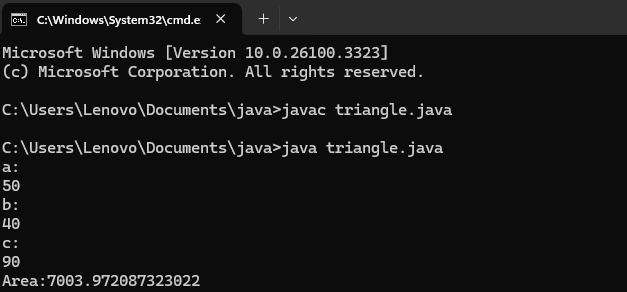
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly.  2.Declaring the data type as double instead of int. | 1.Give iterative statements correctly.  2.Give the data type as int for integers. |

**PROGRAM-3:**

**Aim: Write a program in java for area of triangle using heron’s formula.**

****

**Output:**

******

**ERROR:**

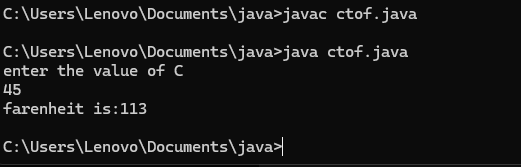
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While printing the variable not giving + sign.  2.Not closing the scanner. | 1.We should give correct indentation.  2.Closing the scanner is must. |

**PROGRAM-4(a):**

**Aim: Write a program in java for converting temperature from celcius to farenheit.**

****

**OUTPUT:**

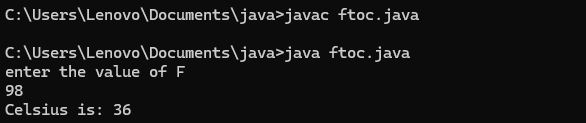
****

**PROGRAM-4(b):**

**Aim: Write a program in java for converting temperature from farenheit to celcius.**

******

**Output:**

******

**ERROR TABLE:**

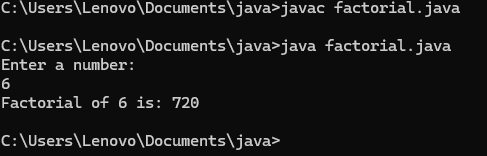
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While printing the variable not giving + sign.  2.Not closing the scanner. | 1.We should give correct indentation.  2.Closing the scanner is must. |

**PROGRAM-5:**

**Aim: Write a program in java for factorial of a number.**

******

**OUTPUT:**

****

**ERROR :**

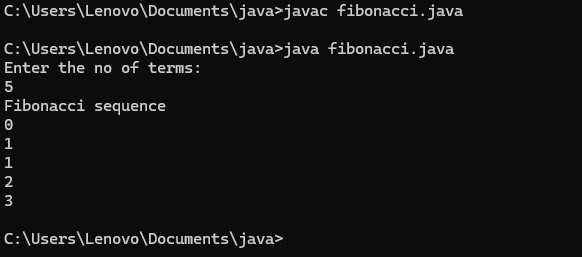
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly.  2.Declaring the data type as double instead of int. | 1.We should give iterative statements correctly.  2.We should give the data type as int for integers. |

**PROGRAM-6:**

**Aim: Write a program in java for fibonacci series.**

****

**OUTPUT:**

****

**ERRORs :**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not giving parenthesis after closing the input. | 1.Should not give space between next and Double.  2.We must put parenthesis after closing the input. |

**Week 3**

**Program 1:**

**Aim :** Write a java program with the following instructions

1. Create a class with name “Car”
2. Create 4 attributes named CarColor,CarBrand,Fueltype,Miliage
3. Create 3 methods named Start,Stop,Service
4. Create 3 objects named c1,c2,c3
5. Create a constructor with parameters CarColor,CarBrand,Fueltype,Miliage

**Code:**

class Car{

String CarColor;

String CarBrand;

String Fueltype;

double Miliage;

Car(String CarColor,String CarBrand,String Fueltype,double Miliage){

this.CarColor = CarColor;

this.CarBrand = CarBrand;

this.Fueltype = Fueltype;

this.Miliage = Miliage;

System.out.println(CarColor+" "+CarBrand+" "+Fueltype); System.out.println(Miliage);

}

void Start(){

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

}

void Stop(){

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

}

void Service(){

System.out.println("Your Car is in service");

}

public static void main(String[] args){

Car c1 = new Car("Blue", "BMW","Petrol",10.5); c1.Start();

c1.Stop(); c1.Service();

Car c2 = new Car("Black","Jaguar","Petrol",12); c2.Start();

c2.Stop();

c2.Service();

Car c3 = new Car("Yellow","Defender","Diesel", 9.5); c3.Start();

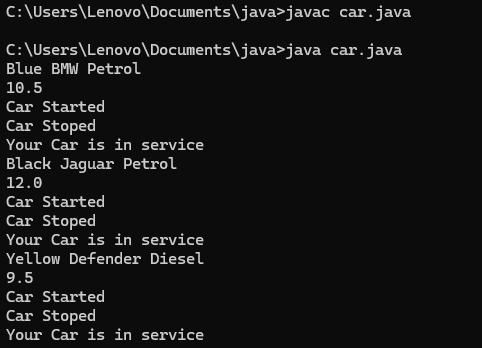
c3.Stop(); c3.Service();

}

**Class diagram :**

|  |
| --- |
| **car** |
| Color : string Brand : string Fuel\_type : string Milage : double |
| Car(color: String, brand: String, fuel\_type: String, milage: double)  start(): void stop(): void service(): void  main(args: String[]): void |

**Output :**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.I have kept ‘,’ between variables in print statement | 1.Insted of ‘,’ we should use ‘+’ between  variables in print statement |

**Program 2:**

Aim: Create a class named bank account with methods Deposit,withdraw,were the deposit method should accepts a parameter and when this method is called the deposited amount should added to current balance, in addition to that when a withdraw method is called it has to verify where current balance, if current balance is less then withdraw amount , then “There are insufficient funds” message should display.

Use the constructor to display the details of the customer,(Name,AccountNumber,IFSE,Branch)

Also Create two customer objects

**Program:**

**import java.util.Scanner;**

**class Bankaccount{**

**long current\_balance;**

**String name;**

**String account\_number;**

**String IFSE;**

**String branch;**

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

**public Bankaccount(long current\_balance,String name,String account\_number,String IFSE,String branch){**

**this.current\_balance=current\_balance;**

**this.name=name;**

**this.account\_number=account\_number;**

**this.IFSE=IFSE;**

**this.branch=branch;**

**System.out.println("User name:"+name+" account\_number:"+account\_number+" IFSE details:"+IFSE+" branch number:"+branch);**

**}**

**public void deposit(){**

**System.out.println("enter the depositing amount: ");**

**long deposit\_amount=input.nextLong();**

**long sum=current\_balance+deposit\_amount;**

**System.out.println("the current blance after depoisting is "+ sum);**

**}**

**public void withdraw(){**

**System.out.println("enter the withdrawing amount: ");**

**long withdraw\_amount=input.nextLong();**

**long dum=current\_balance-withdraw\_amount;**

**if(dum>0){**

**System.out.println("the current blance after withdrawal is "+ dum);}**

**else{**

**System.out.println("the current blance is insufficent ");**

**}**

**}**

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

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

**System.out.println("enter the Balance amount: ");**

**long amount=input.nextLong();**

**Bankaccount a1=new Bankaccount(amount,"Shiva","123456789","SBI2345","15000");**

**a1.deposit();**

**System.out.println("enter the Balance amount: ");**

**long amount1=input.nextLong();**

**Bankaccount a2=new Bankaccount(amount1,"Charan","961","HDFC4563","50000");**

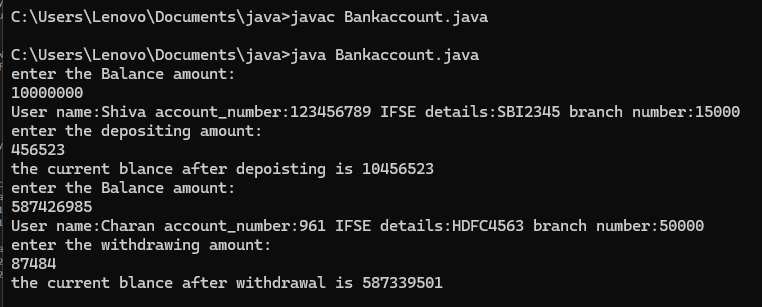
**a2.withdraw();}**

**}**

**Class diagram :**

|  |
| --- |
| **Bank account** |
| **name : string account no : int**  **current amount : float** |
| **bankaccount(string,int,float) deposit()**  **withdrawal()** |

**Output:**

****

**ERROR TABLE:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| **1. The condition checking in the withdrawal amount should be <=** | **1. Change the condition to correct form** |

**WEEK-4**

1) **Aim :** 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).

**Important points :**

1. Understand the calling of a Constructor

2. Giving class name correctly

3. Give the parameters Correctly

4. Method name

**Program :**

class book{ String tittle; String author; int yop;

book(String tittle,String author,int yop){ this.tittle=tittle;

this.author=author; this.yop=yop;

}

void display(){ System.out.println("Title: "+tittle); System.out.println("author:"+author); System.out.println("yop:"+yop);

}

}

class details{

public static void main(String[] args) {

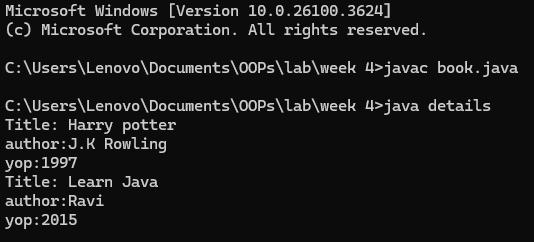
book b1=new book("java","Rahul",2010); book b2=new book("python","Rajesh",2015); b1.display();

b2.display();

}

}

**Output:**

****

**Class diagram:**

|  |
| --- |
| **Book** |
| title: String author: String  year\_of\_publication: int price: double |
| Book(title: String, author: String, year\_of\_publication: int, price: double) details\_of\_book(): void  main(args: String[]): void |

**Error table :**

|  |  |  |
| --- | --- | --- |
| **s.no** | **Error** | **rectification** |
| **1** | **Class name error** | **Give class name corrcetly** |
| **2** | **Syntax/compilation error** | **Absence of semicolon** |
| **3** | **Constructor calling** | **Give same name as class name** |

2. **Aim** : 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.

**Important points :**

* 1. Understand the calling of a Constructor
  2. Giving class name correctly
  3. Give the parameters Correctly
  4. Method name
  5. Final step of calling the value of pi

**Program :**

**class myclass{**

**static int count=0;**

**final double pi=3.1415;**

**void myclass(){**

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

**count=count+1;**

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

**}**

**}**

**class details{**

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

**myclass obj1=new myclass();**

**myclass obj2=new myclass();**

**myclass obj3=new myclass();**

**myclass obj4=new myclass();**

**myclass obj5=new myclass();**

**obj1.myclass();**

**obj2.myclass();**

**obj3.myclass();**

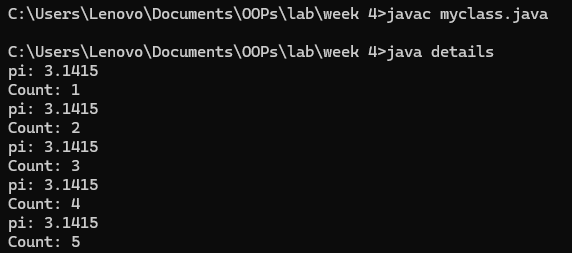
**obj4.myclass();**

**obj5.myclass();**

**}**

**}**

**Output :**

****

**Class diagram :**

|  |
| --- |
| my class |
| static count: int final Pi: double |
| myclass()  main(args: String[]): void |

**Error table :**

|  |  |  |
| --- | --- | --- |
| **s.no** | **error** | **rectification** |
| **1** | **Closing brackets** | **Give closing brackets accordingly** |
| **2** | **Indentation error** | **Give correct spacing** |
| **3** | **Case letter** | **Give case letters in needed places** |

**WEEK 5**

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

**Important Points:**

* + 1. Understand the calling of a Constructor
    2. Giving class name correctly
    3. Give the parameters Correctly

**Program:**

**class bcalculator {**

**int a, b;**

**int sum, diff;**

**bcalculator(int a, int b) {**

**this.a = a;**

**this.b = b;**

**}**

**public void add(){**

**diff = a - b;**

**sum = a + b;**

**System.out.println("Difference: " + diff);**

**System.out.println("Sum: " + sum);**

**}**

**}**

**class acalcululator extends bcalculator {**

**int mul;**

**acalcululator(int a, int b) {**

**super(a, b);**

**}**

**public void mult() {**

**mul = a \* b;**

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

**}**

**}**

**class ucalculator extends acalcululator {**

**float div;**

**ucalculator(int a, int b) {**

**super(a, b);**

**}**

**public void divi()**

**{**

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

**div = (float) a / b;**

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

**}**

**else {**

**System.out.println("Division by zero error!");**

**}**

**}**

**}**

**class ocalculator {**

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

**ucalculator c = new ucalculator(15, 3);**

**c.divi();**

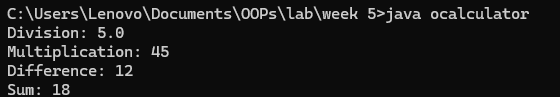
**c.mult();**

**c.add();**

**}**

**}**

OUTPUT:



**Errors:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |

**2.Aim:** 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.

1. Cars should have an additional property: number of doors, Seating capacity.
2. Bikes should have a property indicating whether they have gears or not.
3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.
4. 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?
     1. Truck should include and additional property capacity (in tons).
     2. Create a showTruck() method to display the truck’s capacity.
     3. Write a constructor for truck that initializes all properties.
  3. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike subclasses. Finally display the details.

**Important Points:**

1. Understand the calling of a Constructor
2. Giving class name correctly
3. Give the parameters Correctly

**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 details{**

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

**CARS c=new CARS("BMW",220,5,5);**

**c.cardetails();**

**c.Details();**

**Bikes b=new Bikes("HERO",80,true);**

**b.bikedetails();**

**b.Details();**

**Trucks t=new Trucks("TATA",100,5);**

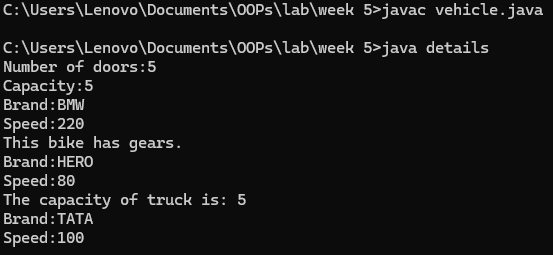
**t.truckdetails();**

**t.Details();**

**}**

**}**

**OUTPUT:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | **Syntax/ Compilation Error** | **Absence of Semicolon** |
| **2** | **Closing Brackets** | **Need to Close the brackets** |
| **3** | **Class Name Error** | **Give the class name correctly** |
| **4** | **Constructor Calling** | **Call the constructor correctly** |

**WEEK 6**

1. **Aim:** Write a Java program to create a vehicle class with a method displayInfo(). Override this method in the car subclass to provide specific information about a car, model, fuel type, and color using the constructor .

**Important Points:**

1. Understand the calling of a Constructor
2. Giving class name correctly
3. Give the parameters Correctly

**PROGRAM:**

**public class vehicles {**

**String Brand;**

**String model;**

**String fuel;**

**String color;**

**int seating;**

**vehicles (String Brand, String model, String fuel, int seating, String color) {**

**this.Brand = Brand;**

**this.model = model;**

**this.fuel = fuel;**

**this.seating = seating;**

**this.color = color;**

**}**

**void displayInfo(String Brand, String model, String fuel, int seating, String color) {**

**System.out.println("vehicle Details: ");**

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

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

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

**System.out.println("seating: " + seating);**

**System.out.println("Color: " + color);**

**}**

**}**

**class Car extends vehicles {**

**Car(String Brand, String model, String fuel, int seating, String color) {**

**super(Brand, model, fuel, seating, color);**

**}**

**void displayInfo() {**

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

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

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

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

**System.out.println("seating: " + seating);**

**System.out.println("Color: " + color);**

**}**

**}**

**class details {**

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

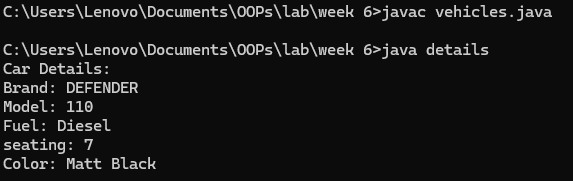
**Car car1 = new Car("DEFENDER", "110", "Diesel", 7, "Matt Black");**

**car1.displayInfo();**

**}**

**}**

**OUTPUT:**

****

**Errors:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | **Syntax/ Compilation Error** | **Absence of Semicolon** |
| **2** | **Closing Brackets** | **Need to Close the brackets** |
| **3** | **Class Name Error** | **Give the class name correctly** |
| **4** | **Constructor Calling** | **Call the constructor correctly** |

**2.Aim: 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.

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

**Important Points:**

* + - 1. Understand the calling of a Constructor
      2. Giving class name correctly
      3. Give the parameters Correctly

**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 checks{**

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

**UG ug=new UG();**

**ug.geteligibility("praveen",51);**

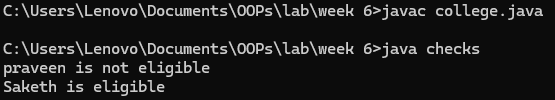
**PG pg=new PG();**

**pg.geteligibility("Saketh",89);**

**}**

**}**

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |
| 4 | Constructor Calling | Call the constructor correctly |

**3.Aim**: 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.

1. Add two integers

2.Add two doubles

3.Add three integers

**Important Points:**

Understand the calling of a Constructor

Giving class name correctly

Give the parameters Correctly

**PROGRAM:**

**class calculator{**

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

**return a+b;**

**}**

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

**return a+b;**

**}**

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

**return a+b+c;**

**}**

**}**

**class results{**

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

**calculator C1=new calculator();**

**System.out.println("Sum of 10 and 45 is: "+C1.add(10,45));**

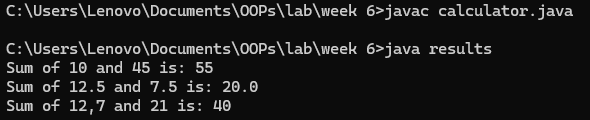
**System.out.println("Sum of 12.5 and 7.5 is: "+C1.add(12.5,7.5));**

**System.out.println("Sum of 12,7 and 21 is: "+C1.add(12,7,21));**

**}**

**}**

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Syntax/ Compilation Error | Absence of Semicolon |
| 2 | Closing Brackets | Need to Close the brackets |
| 3 | Class Name Error | Give the class name correctly |
| 4 | Constructor Calling | Call the constructor correctly |

**4.Aim:** 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.

**Important Points:**

1.Understand the calling of a Constructor

2.Giving class name correctly

3.Give the parameters Correctly

**Program:**

**class Shape {**

**double calculateArea(double side) {**

**return side \* side;**

**}**

**double calculateArea(double width, double height) {**

**return width \* height;**

**}**

**}**

**class Circle extends Shape {**

**double calculateArea(double radius) {**

**return 3.14 \* radius \* radius;**

**}**

**}**

**class results {**

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

**Shape S1 = new Shape();**

**System.out.println("Area of square: " + S1.calculateArea(13));**

**System.out.println("Area of rectangle: " + S1.calculateArea(5, 12));**

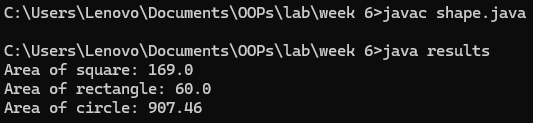
**Circle C1 = new Circle();**

**System.out.println("Area of circle: " + C1.calculateArea(17));**

**}**

**}**

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | **Syntax/ Compilation Error** | **Absence of Semicolon** |
| **2** | **Closing Brackets** | **Need to Close the brackets** |
| **3** | **Class Name Error** | **Give the class name correctly** |
| **4** | **Constructor Calling** | **Call the constructor correctly** |

**WEEK 7**

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

**Program:**

abstract class animal{

    public abstract void sound();{

    }

}

    class lion extends animal{

        @Override

        public void sound(){

            System.out.println("Animal loin roar's");

    }

    }

    class tiger extends animal{

        @Override

        public void sound(){

            System.out.println("Animal Tiger growl's");

        }

    }

    public class Q1{

        public static void main(String[] args) {

            System.out.println("Name:G Shivacharan Reddy; Roll no: 24123; section: CSE-B");

            tiger t = new tiger();

            t.sound();

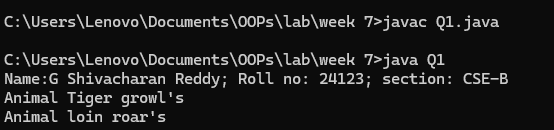
            lion l = new lion();

            l.sound();

    }

}

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Error Name** | **Error Rectification** |
| **1** | Closing brackets | Start/end the brackets correctly |
| **2** | Class name error | Give same class name as file name |

**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.

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

**Program :**

abstract class Shape3D {

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D {

    int radius;

    Sphere(int radius) {

        this.radius = radius;

    }

    double calculateVolume() {

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

    }

    double calculateSurfaceArea() {

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

    }

}

class Cube extends Shape3D {

    int edge;

    Cube(int edge) {

        this.edge = edge;

    }

    double calculateVolume() {

        return Math.pow(edge, 3);

    }

    double calculateSurfaceArea() {

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

    }

}

public class Q2 {

    public static void main(String[] args) {

        Shape3D s = new Sphere(5);

        System.out.println("Name:G Shivacharan Reddy; Roll no: 24123; section: CSE-B");

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

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

        Shape3D c = new Cube(6);

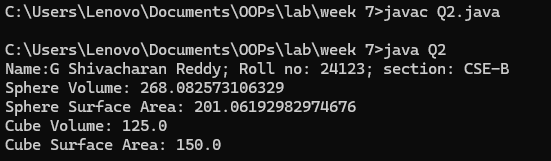
        System.out.println("Cube Volume: " + c.calculateVolume());

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

    }

}

**Output :**

****

**Error table :**

|  |  |  |
| --- | --- | --- |
| S.NO | Error Name | Error Rectification |
| 1 | Abstract method | Implement in only subclass |
| 2 | Data type | Give correct data type |

**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.

**3)Aim :** write a java program using an abstract class to define a method for pattern printing

Create an abstract class named pattern printer with an abstract method 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 {

@Override

public void printPattern(int n){

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

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

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

}

System.out.println();

}

}

}

class NumberPattern extends PatternPrinter {

@Override

public void printPattern(int n) {

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

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

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

}

System.out.println();

}

}

}

public class Q3 {

public static void main(String[] args) {

int rows=5;

        System.out.println("Name:G Shivacharan Reddy; Roll no: 24123; section: CSE-B");

PatternPrinter starPattern = new StarPattern();

PatternPrinter numberPattern = new NumberPattern();

starPattern.printTitle("Star Pattern:");

starPattern.printPattern(rows);

System.out.println();

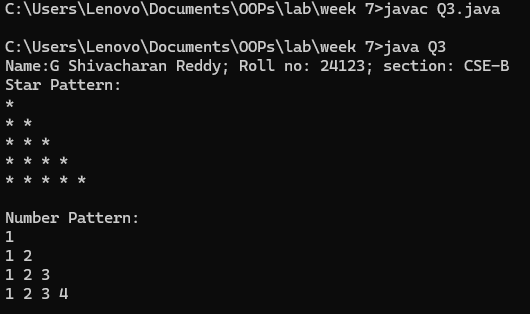
numberPattern.printTitle("Number Pattern:");

numberPattern.printPattern(rows);

}

}

**Output :**



**Error table :**

|  |  |  |
| --- | --- | --- |
| **S no** | **Error name** | **Error rectification** |
| **1** | Error in for loop | Condition should be correct |
| **2** | Main class name | Save same as file name |

**Important points :**

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

2.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) AIM: write a java program creating an interface Shape with the get perimeter method create 3 classes rectangle,triangleand circle that implements the shape interface ,implement the getperimeter method for each of the three classes.

**PROGRAM:**

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

**interface shape {**

**int getperimeter();**

**}**

**class circle implements shape {**

**int radius;**

**circle(int radius) {**

**this.radius = radius;**

**}**

**@Override**

**public int getperimeter() {**

**return (int)(2 \* 3.14 \* radius);**

**}**

**}**

**class rectangle implements shape {**

**int length;**

**int width;**

**rectangle(int length, int width) {**

**this.length = length;**

**this.width = width;**

**}**

**@Override**

**public int getperimeter() {**

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

**}**

**}**

**class triangle implements shape {**

**int side1;**

**int side2;**

**int side3;**

**triangle(int side1, int side2, int side3) {**

**this.side1 = side1;**

**this.side2 = side2;**

**this.side3 = side3;**

**}**

**@Override**

**public int getperimeter() {**

**return (side1 + side2 + side3);**

**}**

**}**

**public class Q1{**

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

**System.out.println("Name: G Shivacharan Reddy; Rollno: 24123; Section: CSE-B");**

**System.out.println("Perimeter of circle");**

**circle c = new circle(7);**

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

**System.out.println("Perimeter of Rectangle");**

**rectangle r = new rectangle(5, 6);**

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

**System.out.println("Perimeter of Triangle");**

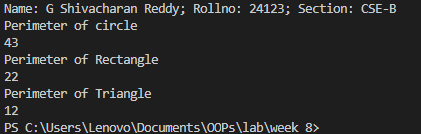
**triangle t = new triangle(3, 4, 5);**

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

**}**

**}**

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **SNO:** | **ERROR** | **RECTIFICATION** |
| 1 | Interface return type mismactched. | It should be public same as main class. |
| 2 | Incorrect use of parameters | If we have already instance variables there is no need to pass parameters again in getperimeter method. |

**IMPORTANT POINTS:**

1. Declares a method getperimeter() that all shapes must implement.

2. Ensures the method getperimeter() correctly overrides the one from the interface.

3.**circle**: Implements getperimeter() using the formula 2πr.

**rectangle**: Uses 2 × (length + width) for perimeter.

**triangle**: Adds all three sides.

**2)write a java program to create an interface playable with a method play() that takes no arguments and returns void create three classes football,volleyball and basketball that implements the playable and override the play method to play the respective sports**

**PROGRAM:**

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

**interface playable{**

**void play();**

**}**

**class football implements playable{**

**@Override**

**public void play(){**

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

**}**

**}**

**class volleyball implements playable{**

**@Override**

**public void play(){**

**System.out.println("Volleyball is being played");**

**}**

**}**

**class basketball implements playable{**

**@Override**

**public void play(){**

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

**}**

**}**

**public class Q2 {**

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

**System.out.println("Name: G Shivacharan Reddy; Rollno: 24123; Section: CSE-B");**

**football f = new football();**

**volleyball v = new volleyball();**

**basketball b = new basketball();**

**f.play();**

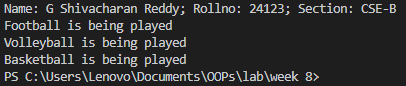
**v.play();**

**b.play();**

**}**

**}**

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **ERROR** | **RECTIFICATION** |
| **1** | Override ‘o’ is small | Use capital ‘O’ |

**IMPORTANT POINTS:**

1. Contains a method play() that must be implemented by all game classes.

2. **football**, **volleyball**, and **basketball** all implement playable. Each class overrides play() to display a message indicating the sport is being played.

**3)AIM:**write a java program to implement a login system using interfaces

**PROGRAM:**

interface LoginSystem {

    boolean login(String id, String password);

}

class UniversityPortal implements LoginSystem {

    @Override

    public boolean login(String id, String password) {

        if (id.equals("shiva\_charan") && password.equals("123456")) {

            return true;

        } else {

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

            return false;

        }

    }

}

public class Q3 {

    public static void main(String[] args) {

        UniversityPortal portal = new UniversityPortal();

        System.out.println("Name: G Shivacharan Reddy; Rollno: 24123; Section: CSE-B");

        boolean loginSuccess1 = portal.login("shiva\_charan", "123456");

        System.out.println("Login successful: " + loginSuccess1);

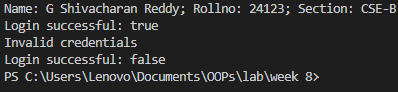
        boolean loginSuccess2 = portal.login("shiva\_charan", "wrong pass");

        System.out.println("Login successful: " + loginSuccess2);

    }

}

**OUTPUT:**

****

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **SNO** | **ERROR** | **RECTIFICATION** |
| **1** | Closing brackets of class | Close brackets properly |

**IMPORTANT POINTS:**

1. Declares a method login(String id, String password) to be implemented by any login system.

2. class (universityportal) Implements the LoginSystem interface. Validates user credentials. Returns true if credentials match, otherwise prints "Invalid credentials" and returns false.