**23CSE111**

**OBJECT-ORIENTED PROGRAMMING**

**LAB REPORT**

****

**Department of Computer Science Engineering**

**Amrita School of Computing**

**Amrita Vishwa Vidyapeetham, Amaravati Campus**

**VERIFIED BY: NAME: K. MANASA**

**ROLL NO: AV.SC.U4CSE24206**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNo | Title | Date | Page No | Signature |
| WEEK 1 |  |  |  |  |
| 1 | Download and Install Java Software |  |  |  |
| 2 | Write a Java program to print the message “Welcome to Java Programming.” |  |  |  |
| WEEK 2 |  |  |  |  |
| 1 | Write a Java program to calculate area of rectangle. |  |  |  |
| 2 | Write a java program to calculate the area of the triangle. |  |  |  |
| 3 | Write a Java program to convert temperature from Fahrenheit to Celsius |  |  |  |
| 4 | Write a Java program to convert temperature from Celsius to Fahrenheit. |  |  |  |
| 5 | Write a Java program to calculate factorial of a number. |  |  |  |
| 6 | Write a Java program to calculate fibonacci of a number. |  |  |  |
| 7 | Write a Java program to calculate Simple interest. |  |  |  |
| WEEK 3 |  |  |  |  |
| 1 | Write a java program with  1. A class with name Car.  2.Create 4 attributes named car\_color , car\_brand ,fuel\_type ,mileage. 3.Create 3 methods named start() ,stop() ,service(). 4.Create 3 objects named car1 ,car2 ,car3. |  |  |  |
| 2 | Write a java program with 1.Create a class named Bankaccount. 2.Create a constructor. 3.Create 2 methods which are withdrawl() and deposit(). |  |  |  |
| WEEK 4 |  |  |  |  |
| 1 | Write a java program with class named book. This class should contain various attributes such as title, author ,year of publication. It should also contain a constructor with parameters which initializers title ,author ,year of publication. Create a method which displays the details of the book .Display the details of 2 books. |  |  |  |
| 2 | Create a java program with class named myclass with a static variable count of int type ,initial value to zero and a constant variable "pi" of type double initialize 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 values of count and pi variables . Create 3 objects. |  |  |  |
| WEEK 5 |  |  |  |  |
| 1 | Create a calculator using the operations including addition, subtraction, multiplication, and division using multi-level inheritance and display the desired output. |  |  |  |
| 2 | A vehicle rental company wants to develop a system that maintains information about different types of vechicles available for rent the company rents out cars and bikes, and they need a program to store details about each vehicle, such as brand and speed( should be in super class)  1.cars should have an additional property: no.of doors 2.Bikes should have a property indicating whether they have gears or not. 3.The system should also include a function to display details about each vehicle and indicate when a vehicle is starting. 4.Every class should have a constructor. Question:  1.Which oops concept is used in the above program  2.If the company decides to add a new type of vehicle, Truck, how would you modify the program?  a.Truck should include an additional property capacity (in tons)  b.Create a showTruckdetails() method to display the truck’s capacity.  c.Write a constructor for Truck that initializes all properties Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details. |  |  |  |
| WEEK 6 |  |  |  |  |
| 1 | Write a java program to create a vehicle class with a method displayinfo(). Override this method in the car subclass to provide specific information about a car. |  |  |  |
| 2 | A college is developing an automated admissions systems that verifies students eligibility for undergraduate(UG) and postgraduate(PG) programs. Each program has different eligibility. Criteria based on the students percentage in their previous qualification.  1. UG admission require min of 60%  2. PG admission require min of 70% |  |  |  |
| 3 | Create a calculator class with overloaded methods to perform addition.  A. Add two integers  B. Add two double  C. Add three integer |  |  |  |
| 4 | 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 01**

**PROGRAM-1:**

**AIM:** Download and Install Java Software

**PROCEDURE:**

**Step 1: Download JDK 21**

1. Open your web browser and go to the Oracle JDK Downloads page
2. Scroll down to the Java SE Development Kit 21 section.
3. Choose the Windows x64 Installer version.
4. Click on Download, then Wait for the download to complete**.**



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

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



**Step 3: Setting up the path**

1) Go to “Windows C” Drive on Desktop

2) Choose Program Files, select Java, then JDK 21, then select Bin.

 3) Select and copy the path at the address bar.

**Step 4: Open System Properties**

1. Press Windows + R, type sysdm.cpl, and click Ok-
2. The System Properties window will open.
3. Navigate to the Advanced tab.
4. Click on Environment Variables at the bottom.

**Step 5: Set JAVA\_HOME**

1)Under System Variables, click New.

2)Set the Variable name as JAVA\_HOME.

3)Set Variable value as C:\Program Files\Java\jdk-21 (or your installation path).

4)Click OK.



**Step 6: Update PATH Variable**

1)In System Variables, find Path and click Edit.

2)Click New and add: C:\Program Files\Java\jdk-21\bin

3)Click OK to save.



**Step 7:Verify Installation**

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



**PROGRAM-2:**

**AIM:** Write a Java program to print the message Hello World

”

**CODE:**

Public class helloworld {

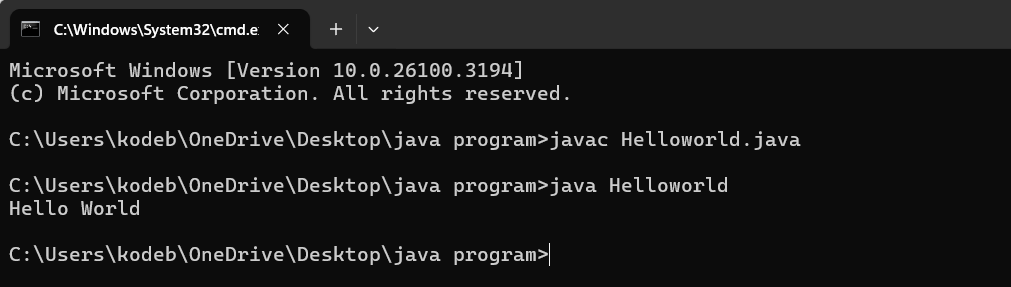
public static void main (String [] args) {

System. out. println ("Helloworld");

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| The class name should be in capital letter | Mention the first letter of the class name by capital letter |

**PROGRAM-3:**

**AIM:** Write a Java Program that prints Name, Roll No, Section of a student.

**CODE:**

class studentdetails{

public static void main (String [] args) {

System.out.println("Name: K.Manasa");

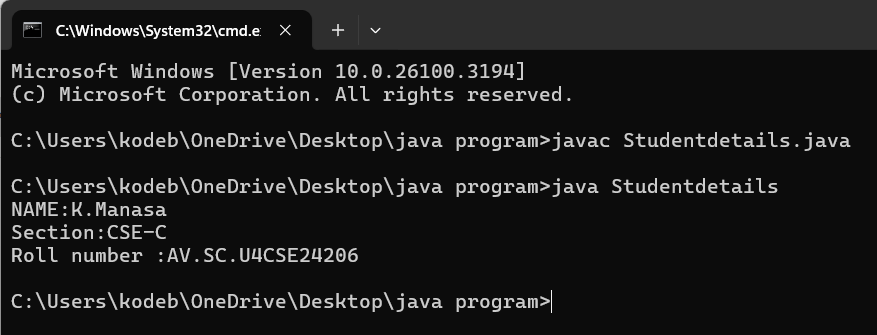
System.out.println("Section: CSE-C");

System.out.println("Roll no:AV.SC.U4CSE24206");

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| cannot find symbol  public static void main(string args[]) {  ^ | public static void main (String [] args) { |

**WEEK 02**

**PROGRAM-1:**

**AIM:** Write a Java program to calculate area of rectangle and area of triangle using herons formula.

**CODE:**

import java.util.Scanner;

class Area {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter length: ");

float l = input.nextFloat();

System.out.print("Enter width: ");

float b = input.nextFloat();

input.close();

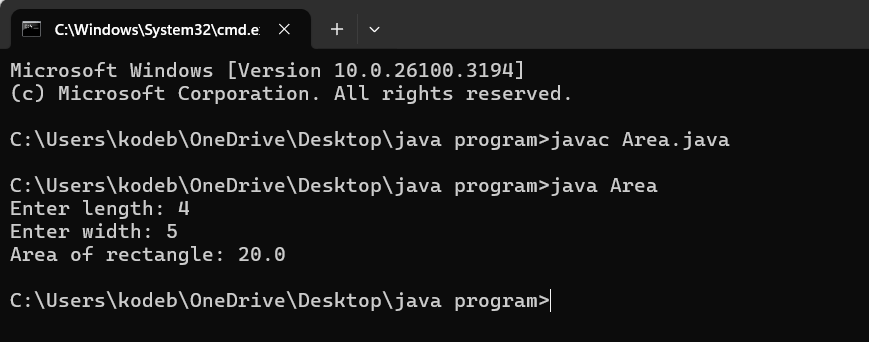
float area = l \* b;

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

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| Area.java:7: error: cannot find symbol  float l = inputnextFloat();  ^  symbol: method inputnextFloat()  location: class Area | float l = input.nextFloat(); |

**CODE:**

public class Areat {

public static void main(String[] args) {

double s1, s2, s3;

double area, resArea;

s1 = 35.0;

s2 = 8.0;

s3 = 38.0;

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

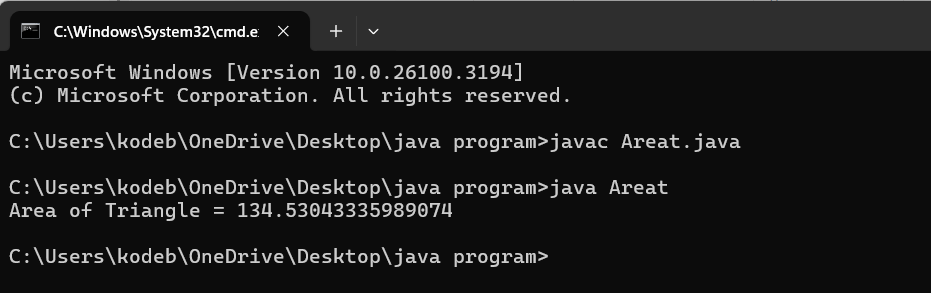
resArea = Math.sqrt(area\* (area - s1) \* (area - s2) \* (area - s3));

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

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| Areat.java:5: error: ';' expected  double area resArea;  ^  Areat.java:5: error: not a statement  double area resArea;  ^ | double area, resArea; |

**PROGRAM-2:**

**AIM:** Write a Java program to convert temperature from Fahrenheit to Celsius and vice versa.

**CODE:**

import java.util.Scanner;

class Temperature {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Fahrenheit: ");

float F = input.nextFloat();

input.close();

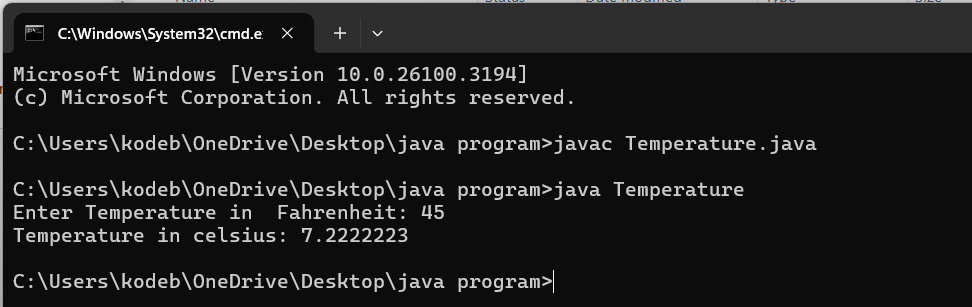
float C = (F - 32)\*5/9;

System.out.println("Temperature in celsius: " + C);

}

}

**OUTPUT:**



**ERRORS:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| S should be capital in System.out.print | System.out.println("Temperature in celsius: " + C); |

**CODE:**

import java.util.Scanner;

class temperature2 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter Temperature in Celsius: ");

float C = input.nextFloat();

input.close();

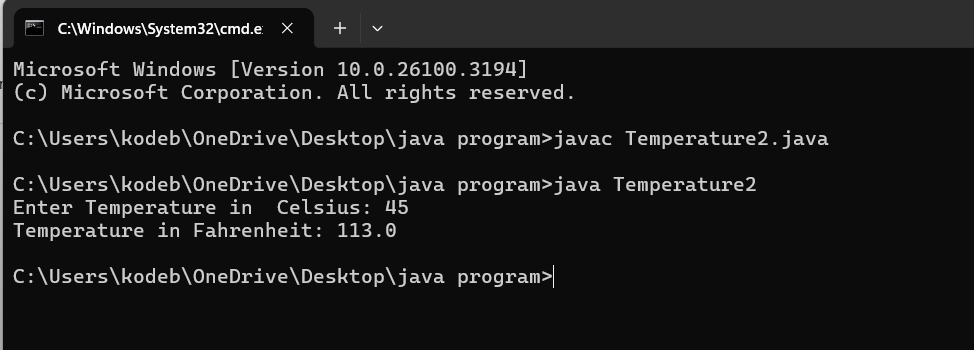
float F = (C \* 9/5) + 32;

System.out.println("Temperature in Fahrenheit: " + F);

}

}

**OUTPUT:**

****

|  |  |
| --- | --- |
| Error found | Error rectified |
| Temperature2.java:13: error: package system does not exist  system.out.println("Temperature in Fahrenheit: " + F); | System.out.println("Temperature in Fahrenheit: " + F); |

**PROGRAM-3:**

**AIM:** Write a Java program to calculate simple intrest.

**CODE:**

import java.util.Scanner;

class Simpleinterest {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Enter principle: ");

float P = input.nextFloat();

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

float T = input.nextFloat();

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

float R = input.nextFloat();

input.close();

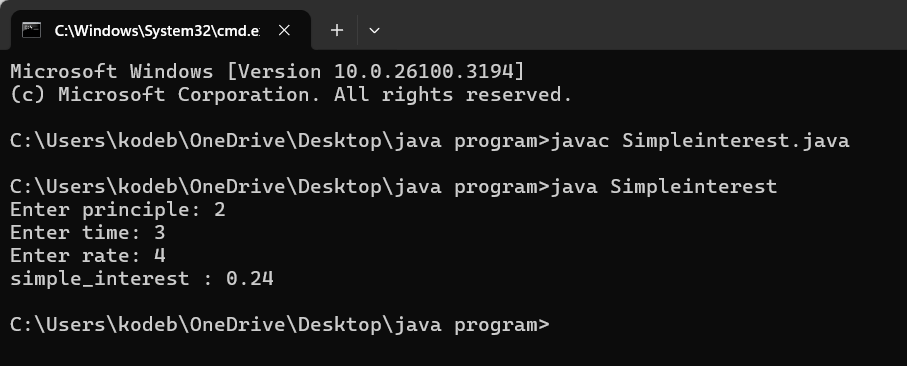
float SI = (P\*T\*R)/100;

System.out.println("simple\_interest : " + SI );

}

}

**OUTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| unclosed string literal  System.out.print("Enter time: );  ^ | System.out.print("Enter time: "); |

**PROGRAM-4:**

**AIM:** Write a Java program to calculate factorial of a number.

**CODE:**

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

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

int n = input.nextInt();

input.close();

long factorial = 1;

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

factorial \*= i;

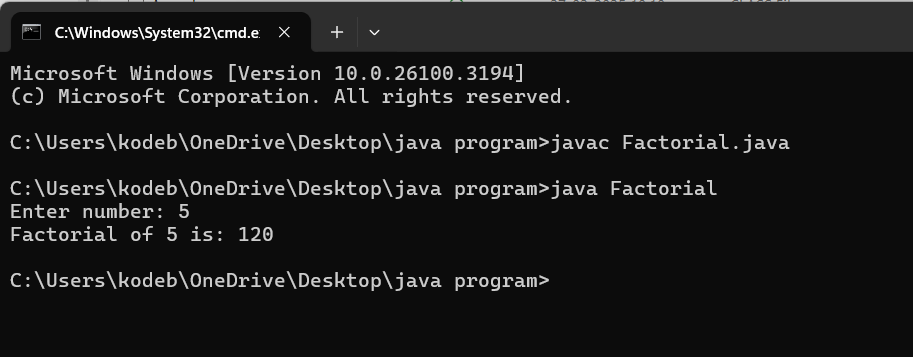
}

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

}

}

**OUTPUT:**



**ERROR:**

|  |  |
| --- | --- |
| Error Found | Error rectified |
| Factorial.java:2: error: class, interface, enum, or record expected  Public class Factorial {  ^ | public class Factorial { |

**PROGRAM-5**

**AIM:** To Write a java program on Fibonacci series

**Code:**

import java.util.Scanner;

public class Fibonacci {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of terms: ");

int terms = scanner.nextInt();

long firstTerm = 0, secondTerm = 1;

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

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

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

long nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

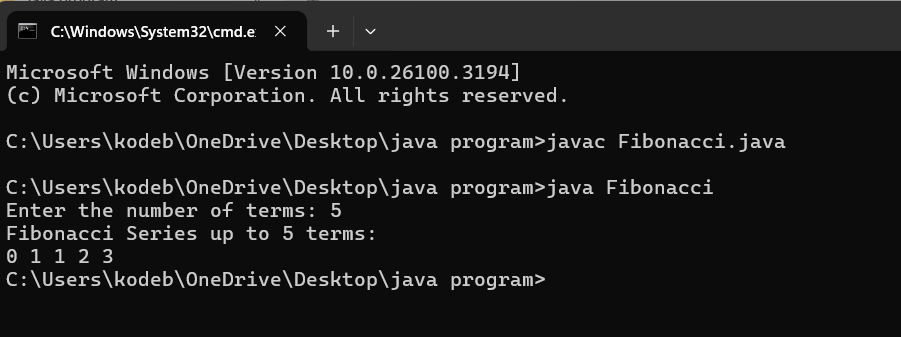
secondTerm = nextTerm;

}

}

}

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| Error found | Error rectified |
| Fibonacci.java:7: error: ';' expected  int terms = scanner.nextInt()  ^ | At the end int terms = scanner.nextInt() this line we should keep ; |

**WEEK 03**

**PROGRAM-1:**

**AIM:** Write a java program with  
 1. A class with name Car.  
 2.Create 4 attributes named car\_color , car\_brand ,fuel\_type ,mileage.

3.Create 3 methods named start() ,stop() ,service().  
 4.Create 3 objects named car1 ,car2 ,car3.  
**CODE:**class Car {  
 private String car\_color;  
 private String car\_brand;  
 private String fuel\_type;  
 private double mileage;  
 public Car(String car\_color,String car\_brand,String fuel\_type,double mileage){  
 this.car\_color = car\_color;  
 this.car\_brand = car\_brand;  
 this.fuel\_type = fuel\_type;  
 this.mileage = mileage;  
 System.out.println("the car is described as "+car\_color+","+car\_brand+","+fuel\_type+","+mileage+" miles");   
 }  
 public void start(String car\_brand){  
 System.out.println(car\_brand +" is starting");  
 }  
 public void stop(String car\_brand){  
 System.out.println(car\_brand +" is stoping");  
 }  
 public void service(String car\_brand){  
 System.out.println(car\_brand +" needs service");  
 }  
 public static void main(String[] args){  
 // object 1 is taken as car1  
 Car car1 = new Car("White","Benz","Diesel",47.9);  
 car1.start("Benz");  
 car1.stop("Benz");  
 car1.service("Benz");  
 // object 2 is taken as car 2  
 Car car2 = new Car("Red","Ferrari","Petrol",14.6);  
 car2.start("Ferrari");  
 car2.stop("Ferrari");  
 car2.service("Ferrari");  
 // object 3 is taken as car 3  
 Car car3 = new Car("Grey","Lamborghini","Gasoline",22);  
 car3.start("Lamborghini");  
 car3.stop("Lamborghini");  
 car3.service("Lamborghini");  
 }  
}

**CLASS DIAGRAM:**

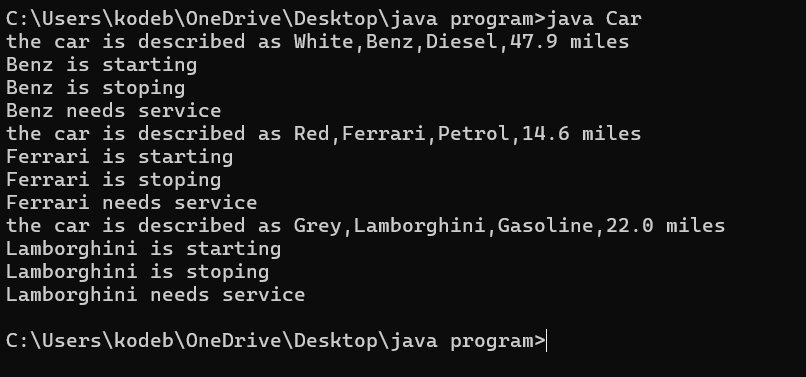
Class Car

- car\_color : String  
- car\_brand : String  
- fuel\_type : String  
- mileage : Double

+ start() : void  
+ stop() : void  
+ service(): void

**IMPORTANT POINT :** 1. The Car class defines attributes such as car\_color, car\_brand, fuel\_type, and mileage, all of which are private fields**.** 2. The constructor public Car(String car\_color, String car\_brand, String fuel\_type, double mileage) initializes the attributes of the Car object and prints a description of the car. A constructor means having the same name as class.3. The Car class includes three methods: start(String car\_brand), stop(String car\_brand), and service(String car\_brand).

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| Errors Found | Errors Rectification |
| I typed private car\_color; | I rectified it as private String car\_color; |
| I wrote it as System.out.println(+car" is starting"); | I rectified it as System.out.println(car\_brand+"is starting"); |

**PROGRAM 2 :  
AIM :** Write a java program with  
 1.Create a class named Bankaccount.  
 2.Create a constructor.  
 3.Create 2 methods which are withdrawl() and deposit().  
**CODE:**class Bankaccount {  
 private String AccName;  
 private int AccNo;  
 private float currentbalance;  
 public Bankaccount(String AccName,int AccNo,float currentbalance){  
 this.AccName=AccName;  
 this.AccNo=AccNo;  
 this.currentbalance=currentbalance;  
 System.out.println("the customer details are "+AccName+","+AccNo+","+currentbalance);  
 }  
 public void withdraw(int amount){  
 if(amount< currentbalance){  
 currentbalance=currentbalance-amount; System.out.println("the currentbalance is "+currentbalance);   
 }  
 }  
 public void deposit(int amount){ currentbalance=currentbalance+amount; System.out.println("the currentbalance is "+currentbalance);  
 }   
 public static void main(String[] args){  
 Bankaccount p1 = new Bankaccount("Manasa",12072007,10000000); p1.withdraw(700000);  
 p1.deposit(500000);  
 }  
}

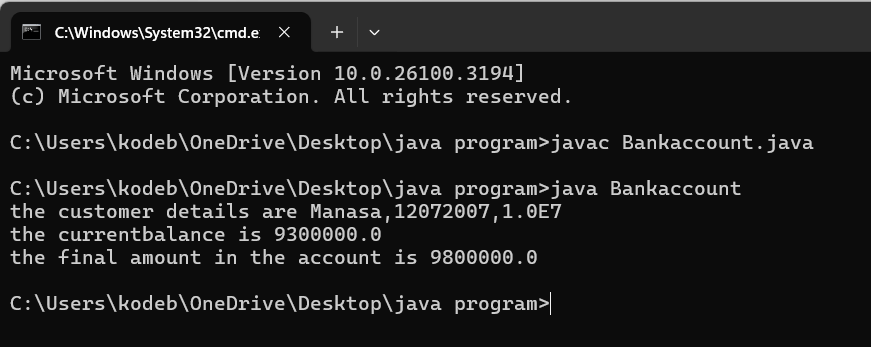
**CLASS DIAGRAM:**

Class Bankaccount

- AccName : String  
- AccNo : int  
- currentbalance : float

+ withdraw() : void  
+ deposit() : void

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| Errors Found | Errors Rectification |
| I typed the code as This.AccName = AccName ; | I rectified it as this.AccName = AccName; |
| I wrote the print statement as  System.out.println("the customer details are "+AccName+","+AccNo+","+currentbalance"); | I rectified the print statement as  System.out.println("the customer details are "+AccName+","+AccNo+","+currentbalance); |

**WEEK-4**

**PROGRAM-1 :**

**AIM:**Write a Java program with class named book the class should contain various attributes such as title,author,year of publication,It should also contain aconstructor with parameters which initializes title,author,yr of publication.Create a method which displays the details od three books.

**CODE:**

class Book{

private String author;

private String title;

private int year;

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

this.author=author;

this.title=title;

this.year=year;

}

public void author() {

System.out.println("The author of the book is"+author);

}

public void title() {

System.out.println("the title of the book is"+title);

}

public void year() {

System.out.println("the year of publication of a book is"+year);

}

public static void main(String[] args){

Book Book1 =new Book("George Orwell","Animal Farm",1945);

Book Book2 =new Book("Jane Austen","Pride And Prejudice",1813);

Book Book3 =new Book("Lewis Carroll","Alice’s Adventures In Wonderland",1865);

Book1.author();

Book1.title();

Book1.year();

Book2.author();

Book2.title();

Book2.year();

Book3.author();

Book3.title();

Book3.year();

}

}

**CLASS DIAGRAM:**

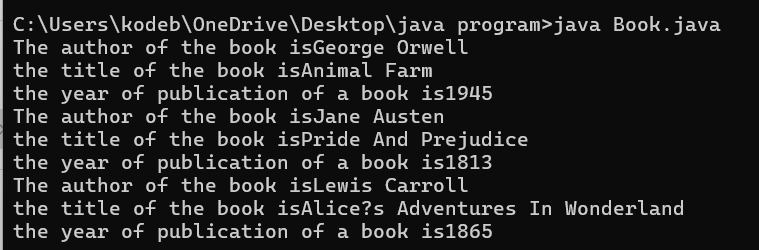
Class Book

- Title: String  
- Author : String  
- Year of Publication : int

+ Title() : void  
+ Author() : void

+Year of publication() :void

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| Errors Found | Errors Rectification |
| error: ';' expected  Book2.title() | Rectified by keeping ; at the end of Book2.title() |

**PROGRAM:2**

**AIM:**Create a Java program program with class name Myclass with a static variable count of int type initialised to zero and the constant variable “pi” of type double initialsed to 3.14 as attributes of the class .Now define a constructor for “Myclass” that increments the count variable each time an object my class in created .Finally print the final values of count &pi varables.Create 3 objects.

**CODE:**

class Myclass {

static int Count=0;

final static double pi = 3.14;

public Myclass(){

Count++;

}

public int Count(int objCount){

System.out.println("the count value of obj"+Count+" is "+Count);

System.out.println("the pi value is "+pi);

return Count;

}

public static void main(String[] args){

Myclass obj1 = new Myclass();

obj1.Count(0);

Myclass obj2 = new Myclass();

obj2.Count(1);

Myclass obj3 = new Myclass();

obj3.Count(2);

}

}

**CLASS DIAGRAM:**

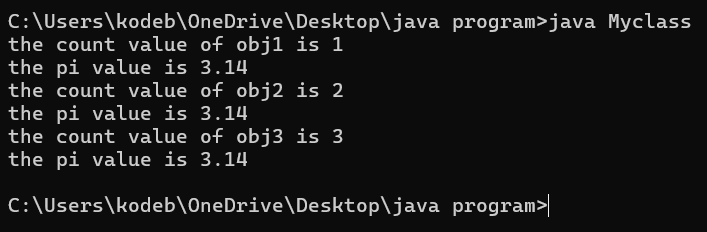
Class Myclass

- count : int

-pi : double

+Count() :int

**OUTPUT:**

****

**ERRORS:**

|  |  |
| --- | --- |
| Errors Found | Errors Rectification |
| error: ';' expected  Myclass obj3=new Myclass() | Rectified by keeping ; at the end of Myclass obj3=new Myclass() |

**IMPORTANT POINTS:**

1. The static variable is a variable shared among all instance of the class.
2. Syntax of static variable - **static datatype varname = value;**
3. Final keyword is used to fix the value to the variable.
4. Syntax to final a variable – **final datatype varname = value;**

**WEEK-5**

**PROGRAM-1:**

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

**Code:**

class calculator {

protected double a, b;

public calculator(double a, double b) {

this.a = a;

this.b = b;

}

}

class Addition extends calculator {

public Addition(double a, double b) {

super(a, b);

}

public double add() {

return a + b;

}

}

class Subtraction extends Addition {

public Subtraction(double a, double b) {

super(a, b);

}

public double subtract() {

return a - b;

}

}

class Multiplication extends Subtraction {

public Multiplication(double a, double b) {

super(a, b);

}

public double multiply() {

return a \* b;

}

}

class Division extends Multiplication {

public Division(double a, double b) {

super(a, b);

}

public double divide() {

if (b != 0) {

return a / b;

} else {

System.out.println("Error");

return Double.NaN;

}

}

}

class Final extends Division {

public Final(double a, double b) {

super(a, b);

}

public void displayResults() {

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

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

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

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

}

}

import java.util.Scanner;

public class allcalculator {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

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

        double a = input.nextDouble();

        System.out.println("Enter b number: ");

        double b = input.nextDouble();

        Final calc = new Final( a,  b);

        calc.displayResults();

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

        input.close();

    }}

**CLASS DIAGRAM:**

Calculator

-a: double

-b: double

+Calculator(a,b)

Addition

+add(): double

Subtraction

+subtract(): double

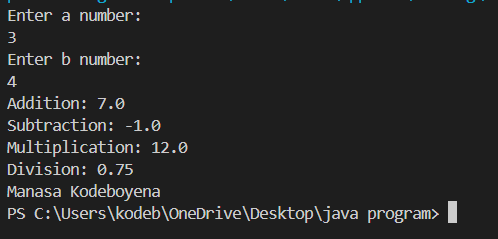
Multiplication

+multiply(): double

Division

+divide(): double

**Output:**

****

**ERROR:**

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1. not providing the return method correctly. 2. Not mentioning super to obtain the super class constructor. | 1. After declaring methods, we must provide the return method correctly. 2. To obtain the super class we need to mention super. |

**Porgram:2**

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

1. cars should have an additional property: no.of doors
2. Bikes should have a property indicating whether they have gears or not.
3. The system should also include a function to display details about each vehicle and indicate when a vehicle is starting.
4. Every class should have a constructor

Question:

1. Which oops concept is used in the above program
2. If the company decides to add a new type of vehicle, Truck, how would you modify the program?
3. Truck should include an additional property capacity (in tons)
4. Create a showTruckdetails() method to display the truck’s capacity.
5. Write a constructor for Truck that initializes all properties
6. Implement the truck class and update the main method to create a Truck object and also create an object for car and bike sub classes Finally, display the details.

**CODE:**

public class vehicle {

public String brand;

public int speed;

public vehicle(String brand, int speed) {

this.brand = brand;

this.speed = speed;

}

public void start() {

System.out.println(brand + " is starting");

}

public void showDetails() {

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

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

}

}

class Car extends vehicle {

private int noOfDoors;

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

super(brand, speed);

this.noOfDoors = noOfDoors;

}

public void showDetails() {

super.showDetails();

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

}

}

class Bike extends vehicle {

private boolean hasGears;

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

super(brand, speed);

this.hasGears = hasGears;

}

public void showDetails() {

super.showDetails();

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

}

}

class Truck extends vehicle {

private int capacity;

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

super(brand, speed);

this.capacity = capacity;

}

public void showTruck() {

super.showDetails();

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

}

}

public class rent {

    public static void main(String[] args) {

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

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

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

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

        car.start();

        car.showDetails();

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

        bike.start();

        bike.showDetails();

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

        truck.start();

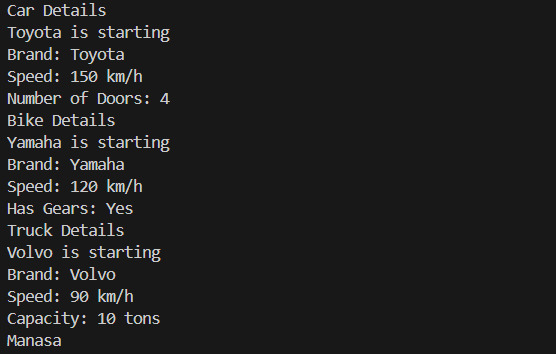
        truck.showTruck();

        System.out.println("Manasa”);

    }

}

**Output:**

****

**CLASS DIAGRAM:**

Vehcile

* Brand : str
* Speed : int

+ init(brand,speed)

+ start\_vehicle()

+display\_details()

Car

-no.of doors: int

+init (brand,speed displaydetails(), no.of doors)

+displaydetails()

+showtruckdetails()

Truck

-capacity: float

Bike

-has gears: bool

+init (brand, speed, has gears)

+displaydetails()

**ERRORS:**

|  |  |
| --- | --- |
| Code Error | Code rectification |
| 1. Declaring two superclasses inside the same file. 2. Not declaring the variable using ‘this’ keyword inside the constructor. | 1. Make two separate files to save the two super classes. 2. Declare the variable using this keyword to run the program. |

**WEEK – 6**

**PROGRAM :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.

**Class Diagram :**

**Vehicle**

- brand : string   
- type : String

+ displayinfo: void

**Car info**

- brand : string   
- type : String

+ displayinfo: void

**Code**

class Vehicle\_info{

private String brand;

private String type;

public Vehicle\_info(String brand,String type){

this.brand = brand;

this.type = type;

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

System.out.println("Type of the vehicle: "+type);

}

public void displayinfo(){

System.out.println("Vehicle is a car");

}

}

class Car\_info extends Vehicle\_info{

public Car\_info(String brand, String type) {

super(brand, type);

}

public void displayinfo(){

System.out.println("Vehicle is a car");

}

}

class Information{

public static void main(String[] args) {

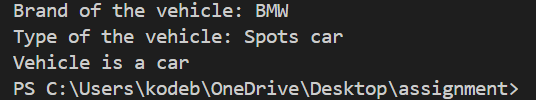
Car\_info car = new Car\_info("BMW","Spots car");

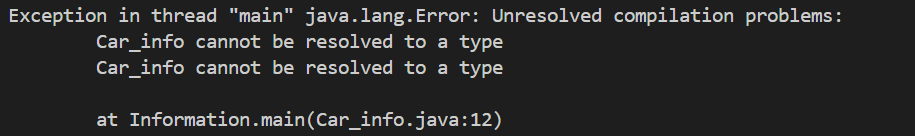
car.displayinfo();

}

}

**OUTPUT:**

****

**Negative case :**

**Error**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| I wrote display info type as string | I rectified it as void type |

**PROGRAM : 2**

A college is developing an automated admission system that verifies student eligibility for undergraduate (UG) and postgraduate (PG) programs. Each program has different eligibility criteria based on the student’s percentage in their previous qualifications. Qualifications: • UG admissions require a minimum of 60%. • PG admissions require a minimum of 70%

**CLASS DIAGRAM :**

**Student**

#name : string   
# percentage : int

+ Student : (string name ,int percentage)

**postgraduate**

# name : string   
# percentage : int

+ postgraduate : (string name ,int percentage)

+ eligibility : (int percentage)

**undergraduate**

# name : string   
# percentage : int

+ undergraduate : (string name ,int percentage)

+ eligibility : (int percentage)

**`**

**CODE :**

import java.util.Scanner;

public class Admission { public static void main(String[]args)

{

Scanner input = new Scanner(System.in);

System.out.println("K. Manasa\nAV.SC.U4CSE24206\nCSE-C ");

System.out.println("Enter student name: ");

String name = input.nextLine();

System.out.println("Enter qualification percentage: ");

double percentage = input.nextDouble();

input.nextLine();

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

String program = input.next().toUpperCase();

if (program.equals("UG"))

{

if (percentage>=60)

{

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

}

Else

{

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

}

}

else if (program.equals("PG"))

{

if (percentage>= 70)

{

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

}

else {

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

}

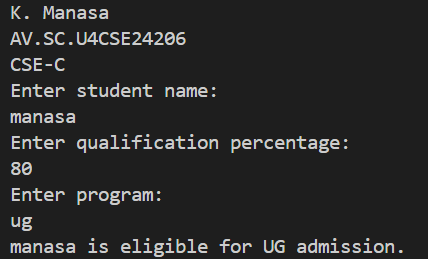
}

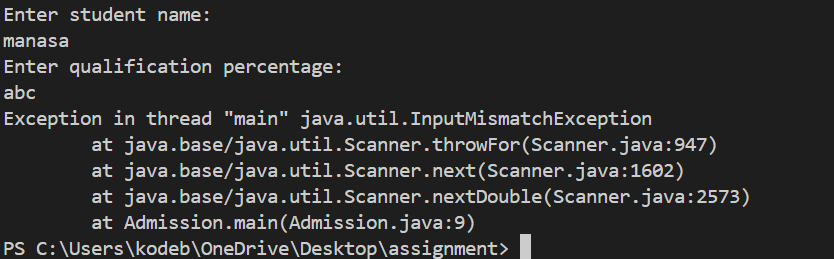
else { System.out.println("Invalid course entered."); } input.close(); } }

**Error :**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| I wrote display info type as string | I rectified it as void type |

**OUTPUT :**

****

**Negative case :**

**PROGRAM : 3**

Create a calculator class with overloaded methods to perform addition.

• Add two integers. • Add two doubles. • Add three integers.

**CLASS DIAGRAM :**

Calculator

+ add : (int,int)   
+ add : (double,double)  
+ add : (int,int,int)

**CODE :**

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 Final{

    public static void main(String[] args) {

        calculator c = new calculator();

        System.out.println("Addition of two doubles:"+c.add(9.0, 9.0));

        System.out.println("Addition of two integers :"+c.add(4,3));

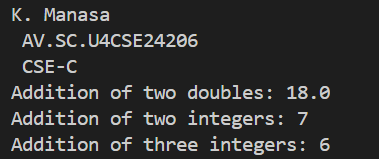
        System.out.println("Addition of three intergers:"+c.add(2,2 ,2 ));

    }

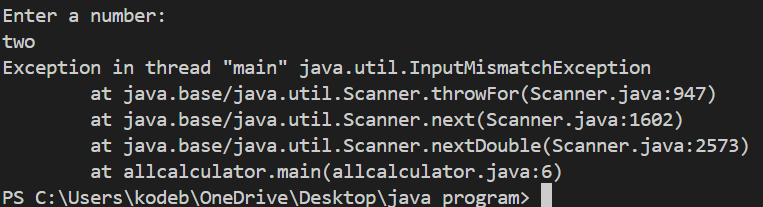
}

**OUTPUT :**

**Positive case :**

****

**Negative case :**

****

**Error :**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| I wrote display info type as string | I rectified it as void type |

**IMPORTANT POINTS :**

All methods share the same name (add) but different signatures

Return type varies based on input parameters (int or double)

The compiler determines which method to call based on argument types and count

**Program : 4**

**Class diagram :**

**Shapes**

+ side : double   
+ length : double   
+breadth : double

+ calculateArea : double

**Circle**

+ radius : double

+ calculateArea : double

**Code :**

public class Shape {

public double calculateArea(double side){

        return side\*side;

    }

    public double calculateArea(double length,double breadth){

        return length\*breadth;

    }

}

class Circle extends Shape{

    public double calculateArea(double radius){

        return Math.PI \* radius \* radius;

    }

}

class Shape\_Area{

    public static void main(String[] args) {

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

        System.out.println("AV.SC.U4CSE24206");

        Circle c = new Circle();

        System.out.println("the Area of circle is :"+c.calculateArea(2.5));

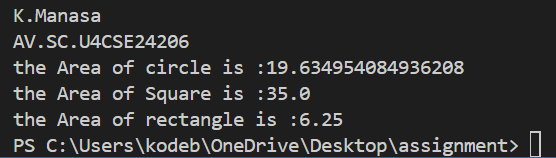
        Shape s = new Shape();

        System.out.println("the Area of Square is :"+s.calculateArea(5,7));

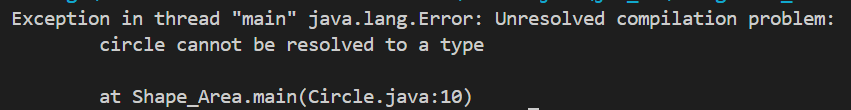
        System.out.println("the Area of rectangle is :"+s.calculateArea(2.5));

    }

}

**Output :**

**Negative case**

****

**Error :**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| I wrote display info type as string | I rectified it as void type |

**Important points :**

The last output line is misleading - it calculates a square's area but is labeled as "rectangle"

The inheritance hierarchy shows a Circle IS-A Shape relationship

All area calculations use double precision for accuracy

**WEEK – 7**

**PROGRAM – 1:**

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

**CLASS DIAGRAM :**

**Animal**

Sound () : void

**Tiger**

Sound () : void

**Lion**

Sound () : void

**CODE:**

 abstract public class animal {

    abstract public void sound();

}

class lion extends animal{

    @Override

    public void sound(){

        System.out.println("Roar");

    }

}

class tiger extends animal{

    @Override

    public void sound(){

        System.out.println("grrr");

    }

}

class A16{

    public static void main(String[] args){

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

        System.out.println("AV.SC.U4CSE24206");

        animal a =new lion();

        a. sound();

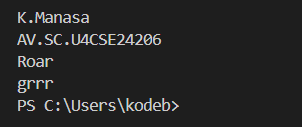
        animal b = new tiger();

        b.sound();

    }

}

**OUTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| **ERROR** | **RECTIFICATION** |
| Error: Main method not found in class animal, please define the main method as: | We should place @Override at the starting of the abstract class |

**IMPORTANT POINTS:**

Abstraction (through abstract class/method)

Inheritance (lion/tiger extending animal)

Polymorphism (same method call producing different results)

Method overriding (subclasses providing specific implementations)

**PROGRAM -2 :**

Write a java program to create an abstract class Shape3D with abstract methods calculateVolume() and calculateSurfaceArea(). Create subclasses Sphere and Cube that extends Shape3D class and implement the respective methods to calculate the Volume and SurfaceArea of each shape.

**CLASS DIAGRAM :**

**Shapes3D**

CalculateVolume () : double

calculateSurfaceArea() : double

**Cube**

Side : double

Cube () : side

calculateVolume (): double

calculateSurfaceArea () : double

**Sphere**

Radius () : double

Sphere () : radius

calculateVolume() : double

calculateSurfaceArea () : double

**CODE:**

abstract class Shape3D {

public abstract double calculateVolume();

public abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D{

    private double radius;

    public Sphere(double radius){

        this.radius=radius;

    }

    @Override

    public double calculateVolume(){

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

    }

    @Override

    public double calculateSurfaceArea(){

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

    }

    }

    class Cube extends Shape3D{

        private double sideLength;

        public Cube(double sideLength){

            this.sideLength = sideLength;

        }

            @Override

            public double calculateVolume(){

                return Math.pow(sideLength,3);

            }

            @Override

            public double calculateSurfaceArea(){

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

            }

            public class Shape3Dtest{

                public static void main(String[] args){

                    Sphere Sphere = new Sphere(5);

                    Cube Cube = new Cube(4);

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

                    System.out.println("AV.SC.U4CSE24206");

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

                    System.out.println("The volume of the Sphere is "+Sphere.calculateVolume());

                    System.out.println("The surface Area of the sphere is "+Sphere.calculateSurfaceArea());

                    System.out.println("The volume od the Cube is"+Cube.calculateVolume());

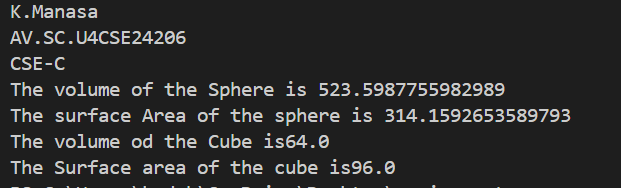
                    System.out.println("The Surface area of the cube is"+Cube.calculateSurfaceArea());

                }

            }

    }

**OUTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| I types it as “math.pi” | I rectified it as “Math.PI”. |

**IMPORTANT POINTS:**

**Abstraction:** Shape3D defines what operations should exist without implementation

**Encapsulation:** Private fields with public methods for calculations

**Polymorphism:** Different implementations of the same methods in different shapes

**Inheritance:** Subclasses extend the abstract Shape3D class

**PROGRAM – 3**

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

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

• Implement two subclasses :

1. Star Pattern - prints a right angled triangle of stars(\*)

2. Number Pattern – prints a right angled triangle of increasing numbers. In the main() method, Create objects of both subclasses and print the patterns for a given no.of row

**CLASS DIAGRAM :**

**PatternPrinter**

n : int ()

title : String

PrintPattern () : int

Patterntitle () : String

**StarPattern**

n : int

PrintPattern (): int

**NumberPattern**

n : int

PrintPattern : int

**CODE:**

abstract class PatternPrinter {

    public int n;

    private String title;

abstract void PrintPattern(int n);

public void Patterntitle(String title) {

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

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

    }

}

class StarPattern extends PatternPrinter {

    public void PrintPattern(int n) {

        this.n = n;

        for (int i = 0; i <= n; i++) {

            for (int j = 0; j <= i; j++) {

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

            }

            System.out.println(); // Move to the next line after inner loop

        }

    }

}

class NumberPattern extends PatternPrinter {

    public void PrintPattern(int n) {

        this.n = n;

        for (int i = 0; i <= n; i++) {

            for (int j = 0; j <= i; j++) {

                System.out.print(j + 1 + " ");

            }

            System.out.println(); // Move to the next line after inner loop

        }

    }

}

class Pattern {

    public static void main(String[] args) {

        StarPattern s = new StarPattern();

        s.Patterntitle("StarPattern");

        s.PrintPattern(5);

        NumberPattern num = new NumberPattern();

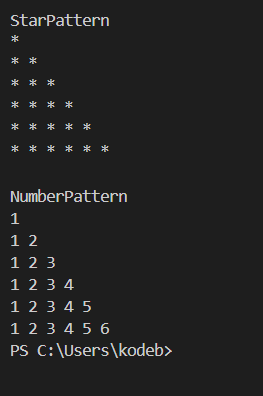
        num.Patterntitle("NumberPattern");

        num.PrintPattern(5);

    }

}

**OTPUT:**

****

**ERROR:**

|  |  |
| --- | --- |
| **ERROR** | **RECTIFICATION** |
| Exception in thread "main" java.lang.Error: Unresolved compilation problem:  Type mismatch: cannot convert from int to boolean | for (int j = 0; j <= i; j++) |

**IMPORTANT POINTS:**

Abstraction → PatternPrinter defines a structure without full implementation.

Inheritance → Subclasses reuse Patterntitle() from the parent class.

Polymorphism → Same method (PrintPattern) behaves differently in each subclass.

Encapsulation → title is private, accessed via a public method.

**WEEK -8**

**PROGRAM -1**

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

**CLASS DIAGRAM :**

**<shapePerimeter>**

getPerimeter () : void

**Triangle**

Side1 : double

Side2:double

Side3 : double

Triangle (double side1,

double side2, double side3)

getPerimeter ():void

**Rectangle**

length : double

breadth :double

Rectangle (double length , double breadth)

getPerimeter () : void

**Circle**

radius : double

Circle (double radius)

getPerimeter () : void

**CODE :**

interface shapePerimeter {

    void getPerimeter();

}

class circle implements shapePerimeter{

    double radius;

    final double pi =3.14;

    public circle( double radius){

        this.radius = radius;

    }

    @Override

    public void getPerimeter(){

        double p = 2\*pi\*radius;

        System.out.println("the perimeter of a circle is "+p);

    }

}

class rectangle implements shapePerimeter{

    double length;

    double breadth;

    public rectangle( double length ,double breadth){

        this.length = length;

        this.breadth = breadth;

    }

    @Override

    public void getPerimeter(){

        double p = 2\*(length +breadth);

        System.out.println("the perimeter of a rectangle is "+p);

    }

}

class triangle implements shapePerimeter{

    double side1;

    double side2;

    double side3;

    public triangle( double side1 ,double side2 , double side3){

        this.side1 = side1;

        this.side2 = side2;

        this.side3 = side3;

    }

    @Override

    public void getPerimeter(){

        double p = side1 +side2 +side3;

        System.out.println("the perimeter of a square is "+p);

    }

}

class A20{

    public static void main (String[] args){

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

        System.out.println("AV.SC.U4CSE24206");

        shapePerimeter c = new circle(5.0);

        c.getPerimeter();

        shapePerimeter r = new rectangle(5.0,10.0);

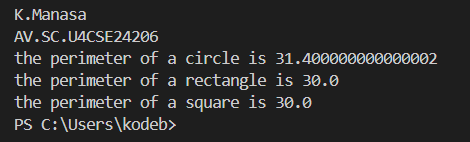
        r.getPerimeter();

        shapePerimeter t = new triangle(5.0,10.0,15.0);

        t.getPerimeter();

}}

**OUTPUT :**

****

**ERRORS :**

|  |  |
| --- | --- |
| Error Found | Error Rectification |
| I types it as “math.pi” | I rectified it as “Math.PI”. |

**IMPORTANT POINTS :**

**Interface Implementation** → Ensures all shapes define getPerimeter().

**Encapsulation** → Private fields with constructor initialization.

**Polymorphism** → Same method call (getPerimeter()) behaves differently based on the object type.

**Code Reusability** → Different shapes follow a common structure but compute perimeters differently.

**PROGRAM -2**

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

**CLASS DIAGRAM :**

**<Playable>**

Play () :void

**VolleyBall**

play () : void

**FootBall**

play () : void

**CODE :**

interface Playable{

    public void play();

}

class FootBall implements Playable {

    public void play(){

        System.out.println("The player is playing Football.");

    }

}

class VolleyBall implements  Playable {

    public void play(){

        System.out.println("The player is playing Volleyball.");

    }

}

class BasketBall implements  Playable {

    public void play(){

        System.out.println("The player is playing Basketball.");

    }

}

class A25{

    public static void main(String[] args){

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

        System.out.println("AV.SC.U4CSE24206");

        FootBall f1 = new FootBall();

        f1.play();

        VolleyBall v1 = new VolleyBall();

        v1.play();

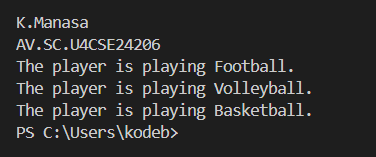
        BasketBall b1 = new BasketBall();

        b1.play();

    }

}

**OUTPUT :**

****

**ERRORS :**

|  |  |
| --- | --- |
| **ERROR** | **RECTIFICATION** |
| Exception in thread "main" java.lang.Error: Unresolved compilation problem:  override cannot be resolved to a type | O should be capital at @override.that means @Override is correct |

**IMPORTANT POINTS :**

**Interface Usage**: The Playable interface enforces a contract (the play() method) that implementing classes must follow.

**Loose Coupling**: The Sports class interacts with objects through the Playable interface rather than concrete classes, promoting flexibility.

**Method Overriding**: Each sport class overrides play() to provide its own behavior.