23CSE111

OOPS

(Object Oriented Programming System)

LAB MANUAL

A logo with pink letters

Description automatically generated

Department of CSE

Amrita School of Engineering

Amrita Vishwa Vidyapeetham, Amaravati Campus

**NAME: P. Pranav**

**ROLLNO:AV.SC.U4CSE24244**

**SECTION: CSE-C**

**WEEK-1:**

**Aim:** How to install jdk and first program on

printing student details*.*

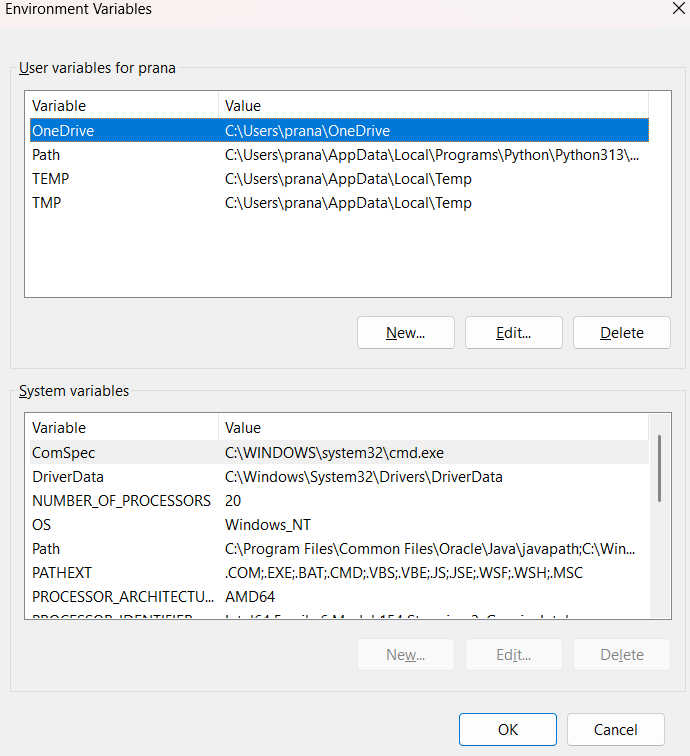
**Step-1:** Download JDK-21 from oracle website

**

**Step-2:**Install the JDK-21 with accepting terms and

conditions according to the respective windows.

**Step-3**:Setting up environmental variables.



\*Windows c -> C-drive -> program files ->Java -

>JDK-21->select bin

\*Select and open environmental variable in search

bar-> either select system variables or user

variables-> select path-> click edit->New-> paste

the bin-> finish the setup(apply the changes).

~for verifying the installed version

Open cmd-> type java --version

~command propt

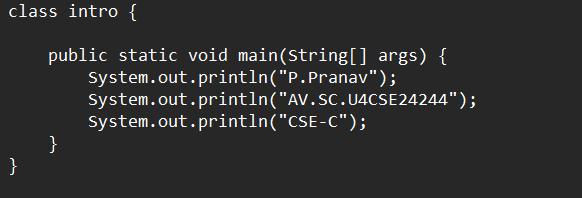
Javac filename.java ->compiling.

Java filename.java ->display

**PROGRAM-1:**

**Aim:** Write a program in java for displaying

student details.



**Output:**

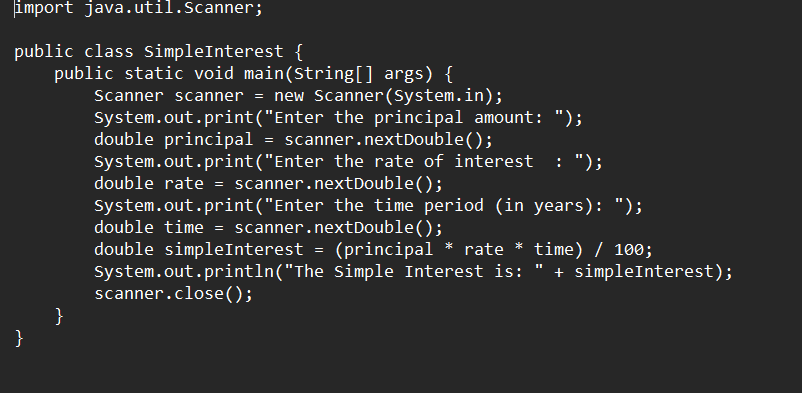


|  |  |  |
| --- | --- | --- |
| **Code \** | **Error** | **Rectification** |
| System.out.println  ("Class: CSE-C") | Semi colon(;) is  missing at the  end. | Add a semi colon(;) at the  end.  System.out.println("Class:  CSE-C"); |

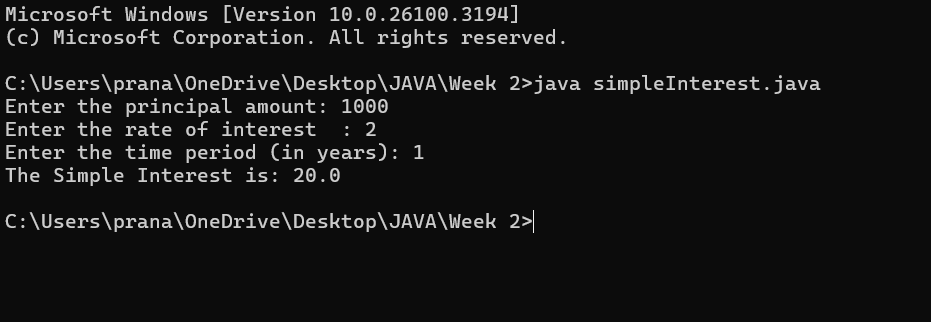
**WEEK-2:**

**PROGRAM-1:**

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

**

**Output:**

******

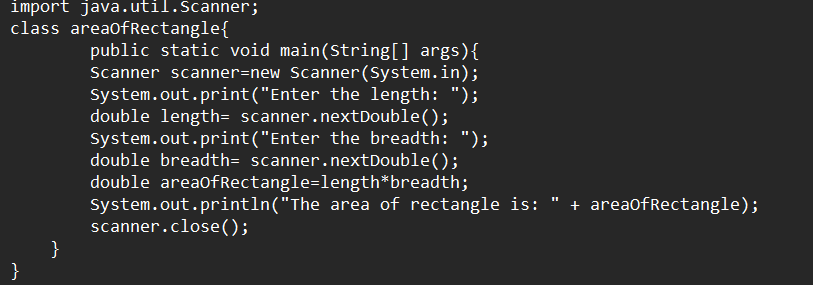
**ERROR TABLE*:***

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not using the closing semi colon. | 1.Should not give space between next and Double.  2.We must put semi colon after each line when required. |

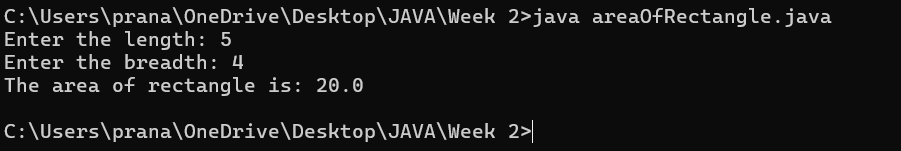
**PROGRAM-2:**

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

rectangle.

**

**Output:**

******

IMPORTANT POINTS:

1. Area of a rectangle is area = l\*b, where

L = length of a side of the rectangle,

B= breadth of a side of the rectangle.

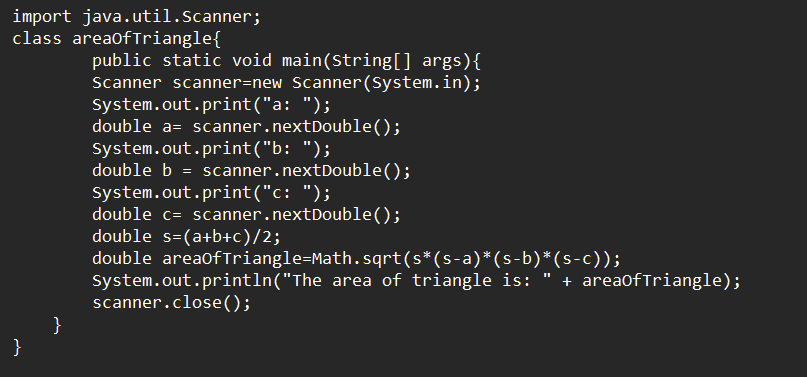
1. Here, we must be sure that all the expressions/conditions inside for the for loop must be given correctly.

**ERROR TABLE:**

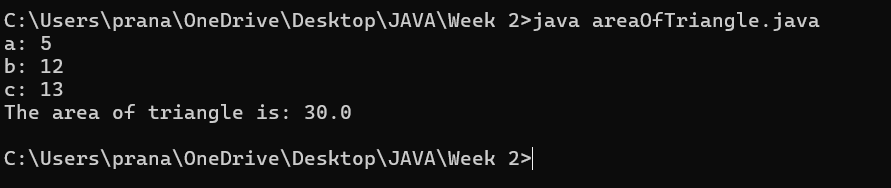
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly.  2.Declaring the data type. | 1.We should give iterative statements correctly.  2.We should give the data type. |

**PROGRAM-3:**

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



**Output:**

******

IMPORTANT POINTS:

1. Here, we’re finding the area of a triangle using heron’s formula.
2. Heron’s formula for finding a triangle is:

S = (a +b +c)/2

Where S is the semi-perimeter of the triangle.

Now the area formula is:

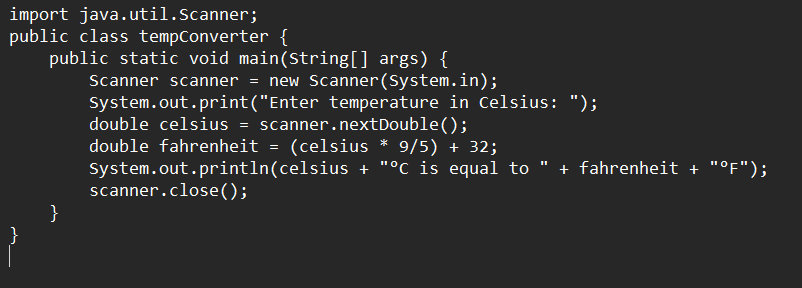
Area = sqrt(s\*(s-a)\*(s-b)\*(s-c)).

ERROR TABLE:

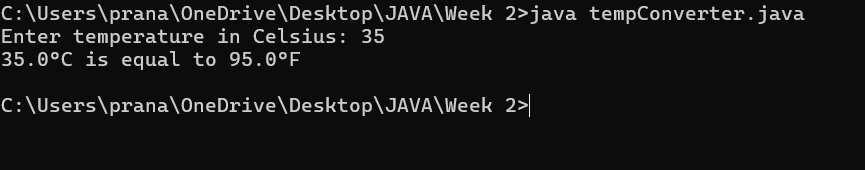
|  |  |
| --- | --- |
| **Code Error** | Code rectification |
| 1.While printing the variable not giving + sign.  2.Declaring the data type. | 1.We should give correct indentation.  2.We should give the data type. |

**PROGRAM-4(a):**

**Aim:** Write a program in java for converting temperature from Celsius to Fahrenheit.

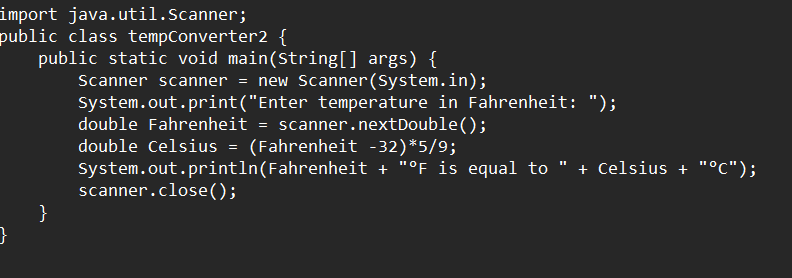


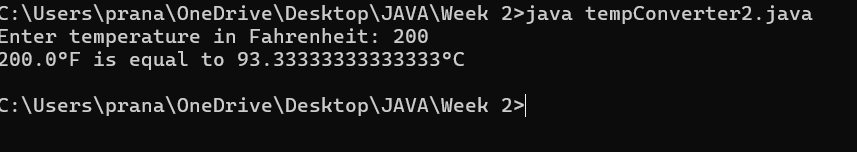
OUTPUT:



**PROGRAM-4(b):**

**Aim:** Write a program in java for converting temperature from Fahrenheit to Celsius.

****Output:**

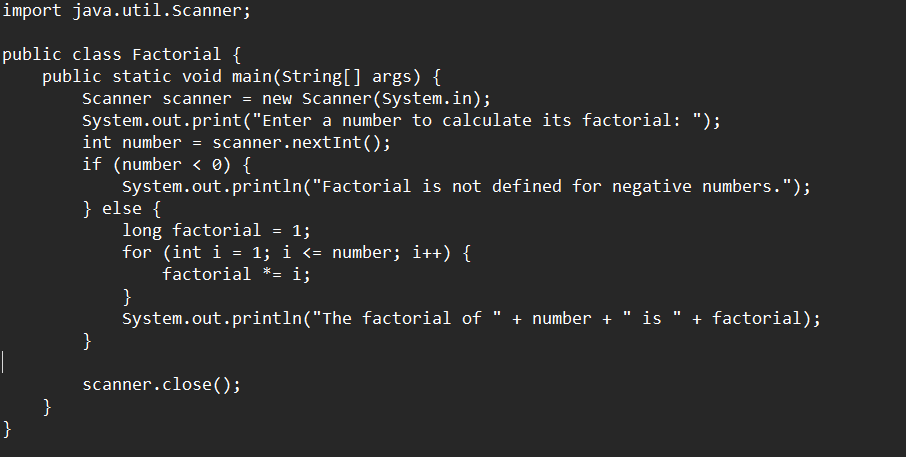
******

**ERROR TABLE:**

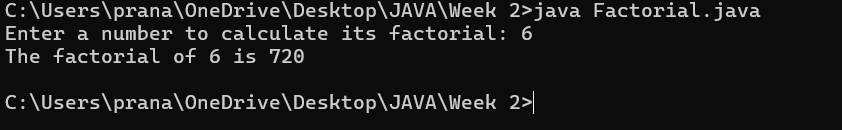
|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While printing the variable not giving + sign.  2.Not closing the scanner. | 1.We should give correct indentation.  2.Closing the scanner is must. |

**PROGRAM-5:**

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

******

OUTPUT:

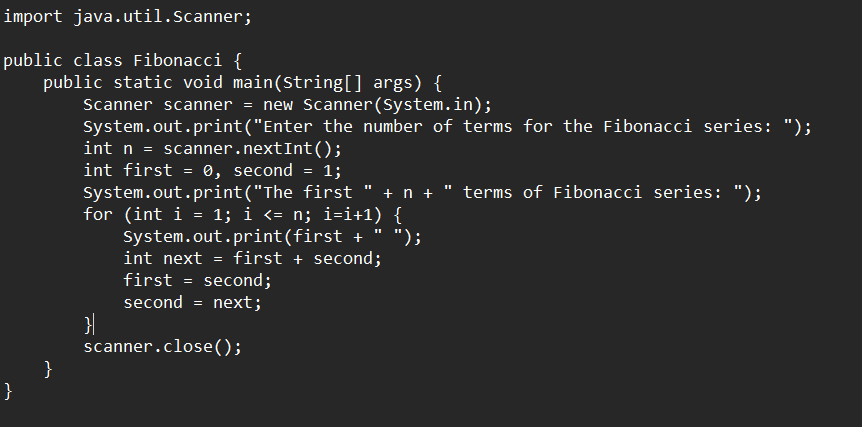


ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.While using for iteration, not giving the conditions correctly. | 1.We should give iterative statements correctly. |

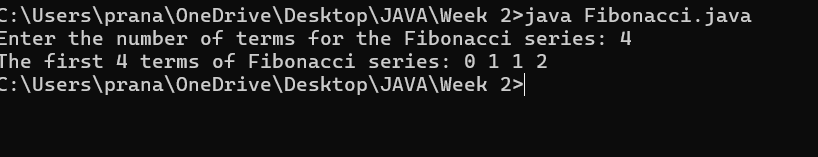
**PROGRAM-6:**

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



IMPORTANT POINTS:

1. In the Fibonacci sequence, the sum value is given to the second variable, and the value of the second variable is given to the first variable.
2. This process is repeated a certain number of times until the conditions are met.

OUTPUT: 

ERROR TABLE:

|  |  |
| --- | --- |
| **Code Error** | **Code rectification** |
| 1.Giving space between next and Double.  2.Not putting flower brackets in the end. | 1.Should not give space between next and Double.  2.We must put giving flower brackets in the end. |

**WEEK -3:**

**PROGRAM-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 these methods named start (), stop (), service ()

4.Create the objects named car, car1, car2

**CODE:**

public class Car {

private String car\_color;

private String car\_brand;

private String fuel\_type;

private String mileage;

public void start() {

System.out.println("car is started");

}

public void stop() {

System.out.println("car is stopped");

}

public void service() {

System.out.println("car is for service");

}

public static void main(String args[]) {

Car car = new Car();

car.car\_color = "white";

car.car\_brand = "audi";

car.fuel\_type = "petrol";

car.mileage = "20";

car.start();

System.out.println("car\_color: " + car.car\_color + " car\_brand: " + car.car\_brand + " fuel\_type: " + car.fuel\_type + " mileage: " + car.mileage);

Car car1 = new Car();

. car1.car\_color = "white";

car1.car\_brand = "audi";

car1.fuel\_type = "petrol";

car1.mileage = "20";

car1.stop();

System.out.println("car\_color: " + car1.car\_color + " car\_brand: " + car1.car\_brand + " fuel\_type: " + car1.fuel\_type + " mileage: " + car1.mileage);

Car car2 = new Car();

car2.car\_color = "white";

car2.car\_brand = "audi";

car2.fuel\_type = "petrol";

car2.mileage = "20";

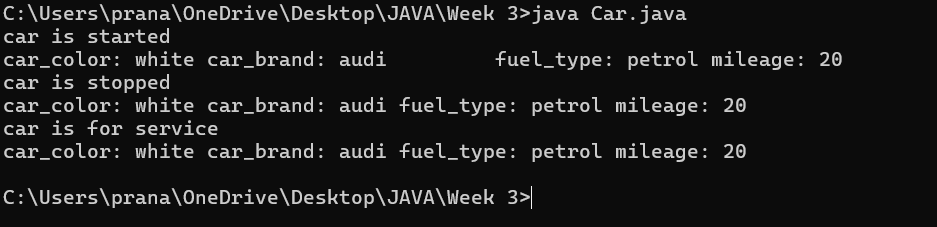
car2.service();

System.out.println("car\_color: " + car2.car\_color + " car\_brand: " + car2.car\_brand + " fuel\_type: " + car2.fuel\_type + " mileage: " + car2.mileage);

}

}

**OUTPUT:**

****

**Error table:**

|  |  |  |
| --- | --- | --- |
| S.no | error | Rectification |
| 1 | Missing ‘;‘ | ‘;‘ added |
| 2 | Mispelled Variable call | Rectified with  Correct variable name |
| 3 | Uppercase and lowercase | rectified |

IMPORTANT POINTS:

1. Before calling the function we should write the method properly.
2. Here, the “public void start( )” indicates that we are writing a method to call the function.
3. When we call a certain method, the process inside it will be printed as an output of the code.
4. Here the details inside the function are called objects, we can give any objects

**Class diagram:**

|  |
| --- |
| **car**  **----------------------**-  -car\_color:string  -car\_brand:string  -fuel\_type:string  -milage:double  ----------------------  +start():void  +stop():void  +service():void |

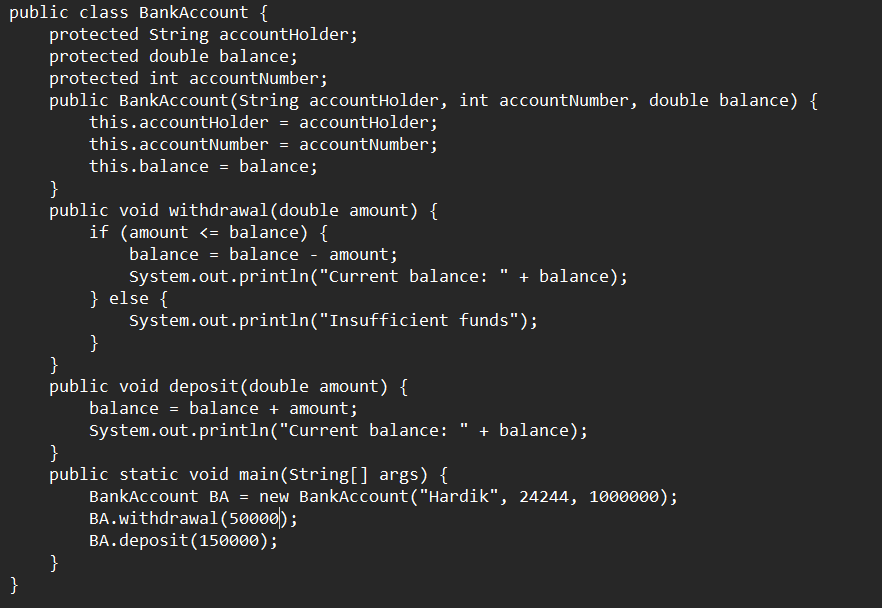
**PROGRAM-2:**

**Aim:** To create a class BankAccount with methods deposit() and withdraw() . create two subclasses savingsaccount and checkingaccount override the withdraw () method in each subclass to impose different withdrawal limits and fees

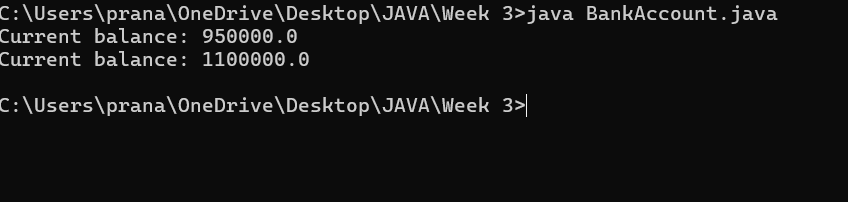
**Class diagram:**

|  |
| --- |
| **BankAccount**  ----------------------------------------------------------  -balance: double  ----------------------------------------------------------  +BankAccount(intialBalance: double)  +deposit(amount: double):void  +withdraw(amount: double):void |

Code:

****

**OUTPUT:**

****

IMPORTANT POINTS:

1. The condition inside the if statement must be correct.
2. It explains that if the withdrawal money is less than the money in the bank account, then we can withdraw the amount.

**Error table:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Error name | Error name | Rectification |
| 1 | Name Error | Undefined name | Correct variable  Name replaced |
| 2 | Syntax Error | Missing Parenthesis | Parenthesis Added |
| 3 | Logical Error | Incorrect Condition | Condition Rectified |

**WEEK-4:**

**PROGRAM-1:**

**Aim:** Write a java program with class named book .The class should contain various attributes such as Title ,Author and Year of Publication .It should also contain a constructor with parameter which initializes Title ,Author and Year of publication .Create a method which displays the details of the book .Display the details of two books.

**CODE:**

class book{

public String title;

public String author;

public String year\_of\_publication;

public void book(){

this.title=title;

this.author=author;

this.year\_of\_publication=year\_of\_publication;

}

public static void main(String[] args){

book book1=new book();

book book2=new book();

book1.book();

book1.title="Sherlock Holmes ";

book1.author="Arthur Conan Doyle";

book1.year\_of\_publication="1887";

book2.book();

book2.title="Harry Potter";

book2.author="J.K. Rowling";

book2.year\_of\_publication="1997";

System.out.println("Book-1");

System.out.println("Title :" +book1.title);

System.out.println("Author :" +book1.author);

System.out.println("Year of publication :" +book1.year\_of\_publication);

System.out.println("Book-2");

System.out.println("Title :" +book2.title);

System.out.println("Author :" +book2.author);

System.out.println("Year of publication :" +book2.year\_of\_publication);

}

}

**OUTPUT:**

****

**Class Diagram:**

|  |
| --- |
| **Book** |
| * Title: String * Author: String * Year of publication: int |
| + Book(title: String,  Author: String;  Year of publication: int  + displayDetails( ): void |

IMPORTANT POINTS:

1. While defining two classes for a code, we must be sure that we save both the classes in separate files.
2. While defining a method we should also define a function to call that method.

**Error Table:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Error** | **Rectification** |
| **1.** | **Missing “;” after calling method.** | **Added “;”** |

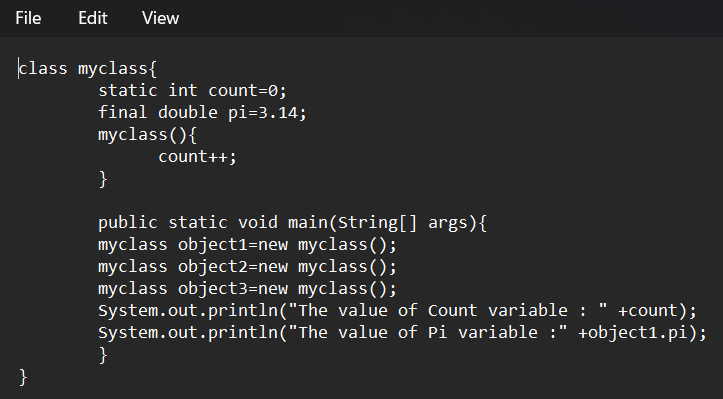
**PROGRAM-2:**

**Aim:** Create a java program with class named myclass with a static variable count of int type initialized to 0 and a constant variable “Pi” of type double initialized to 3.14 has attributes of that class .Now define a constructor for “myclass” that incerements the count variable each time an object of myclass is created finally print final values count and Pi variables .Create three objects

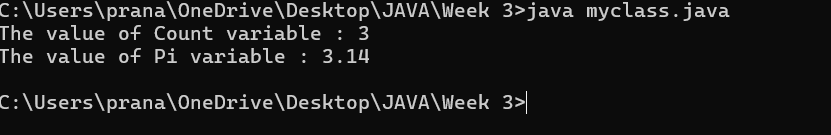
**Class Diagram:**

|  |
| --- |
| Myclass |
| - Count: int  - Pi: double |
| + myclass( )  + main(args: String[]): void |

Code:



**OUTPUT:**

****

IMPORTANT POINTS:

1. We must declare the initial value of the variable before declaring the final one.
2. Here the main objective is to increase the count according to the number of objects we make, i.e the count increases when the no.of objects are increasing.

Error Table:

|  |  |  |
| --- | --- | --- |
| S.No | Error | Rectification |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| 1. | Not typing “}“ at the end of the code. | Added “}”. |

**WEEK-5:**

**PROGRAM-1**

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

**Class Diagram:**

|  |
| --- |
| **Basic Operations** |
| + add (a,b)  +subtract (a,b) |

|  |
| --- |
| **Multiplication** |
| +Multiply (a,b) |

|  |
| --- |
| **Division** |
| + Divide (a,b) |

|  |  |
| --- | --- |
| **Subtraction** | |
| + subtraction(a,b) | |
|  | |  | |

|  |
| --- |
| **Calculator** |
| +calculate (op,a,b) |

**Code:** class Calculator {

}

class add extends Calculator {

    public int addition(int a, int b) {

        return a + b;

    }

}

class sub extends add {

    public int subtraction(int a, int b) {

        return a - b;

    }

}

class mul extends sub {

    public int multiplication(int a, int b) {

        return a \* b;

    }

}

class div extends mul {

    public int division(int a, int b) {

        return a / b;

    }

}

public class Multilevel {

    public static void main(String[] args) {

        div obj = new div();

        System.out.println("Addition: " + obj.addition(10, 2));

        System.out.println("Subtraction: " + obj.subtraction(10, 2));

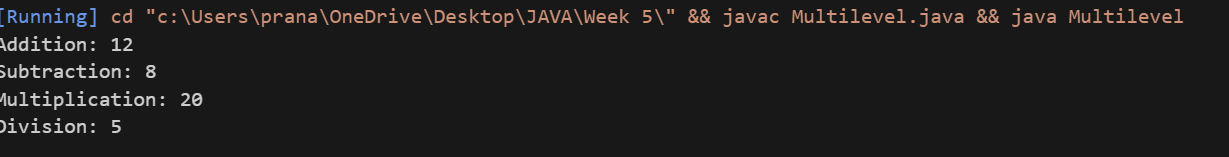
        System.out.println("Multiplication: " + obj.multiplication(10, 2));

        System.out.println("Division: " + obj.division(10, 2));

    }

}

Output:

****

**ERROR TABLE :**

|  |  |  |
| --- | --- | --- |
| **s.no** | **Error** | **Error rectification** |
| **1** | **Syntax error on token ")”** | **‘{’ should be mentioned** |
| **2** | **calculator cannot be resolved to a type** | **‘c’ should be capital.** |

**IMPORTATNT POINTS:**

**We use multilevel inheritance in this**

**Each subclass inherits methods from its parent class, gaining access to addition, subtraction,**

**Multiplication and division.**

**PROGRAM-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 they need a program to store details about each vehicle such as brand & speed.**

**Class Diagram:**

|  |
| --- |
| Vehicle |
| * Brand: String * speed: int |
| + Vechile(String,b int)  + Start()  + DisplayDetails() |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Car | | * numberofdoors: int * seatingCapacity: int | | + car(String,int,int,int)  + displaydetails() | | |  | | --- | | Bike | | -hasGears: boolean | | + Bike(String,int,Boolean)  +displayetails() | |

|  |
| --- |
| Truck |
| -capacity: double |
| + truck(String,int,double)  +showtruckdetails()  +displaydetails() |

Code:

class Vehicle{

    void company(){

        System.err.println("Vehicle Company");

    }

    void start(){

    System.out.println("Vehicle is starting");

    }

}

class cars extends Vehicle{

    void car(){

        System.out.println("It has 4 doors");

    }

}

class bikes extends Vehicle{

    void bike(){

        System.out.println("It has gears");

    }

}

class trucks extends Vehicle{

    void truck(){

        System.out.println("It has gears");

    }

}

public class Hierarchial{

    public static void main(String[] args) {

        cars c = new cars();

        c.car();

        c.start();

        c.company();

        bikes b = new bikes();

        b.bike();

        b.start();

        b.company();

        trucks t = new trucks();

        t.truck();

        t.start();

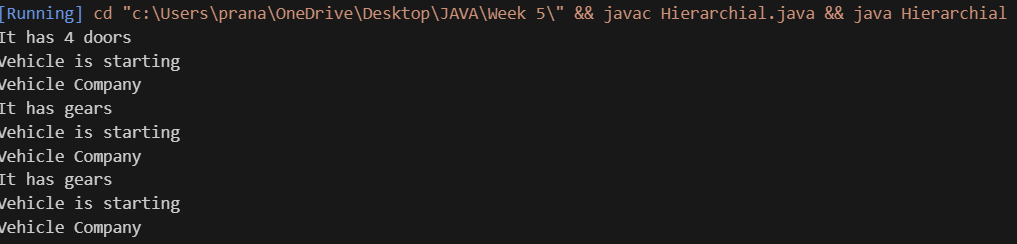
        t.company();

    }

}

ERROR TABLE :

|  |  |  |
| --- | --- | --- |
| s.no | Error | Error rectification |
| 1 | Syntax error on token ")” | ‘{’ should be mentioned |
| 2 | Heirarchial cannot be resolved to a type | ‘h’ should be capital |

Output: 

IMPORTATNT POINTS:

The code demonstrates inheritance using a parent class Vehicle and three child classes: Car, Bike, and

Truck.

Each subclass has its own constructor that extends the parent class constructor

WEEK-6

Program 1

Aim: Write a JAVA program to create a vehicle class with a method displayInfo.

Override this method in the Car subclass to provide specific information about a car.

Class Diagram:

|  |
| --- |
| Vehicle |
| +displayinfo():void |

|  |
| --- |
| Car\_model:String Brand:String Fuel\_type:String |
| + car(String car\_model,String Brand,String Fuel\_type)  +displayinfo(): void |

Code:

class vehicle{

}

class car extends vehicle{

    void displayinfo(){

    System.out.println("This is a Ferrari 488 GTB.");

    System.out.println("Engine: 3.9-liter twin-turbocharged V8");

    System.out.println("Horsepower: 661 hp (at 8,000 rpm)");

    System.out.println("Starting price: Around $250,000 USD");

    }

}

public class ovveride{

    public static void main(String[] args) {

        car c1=new car();

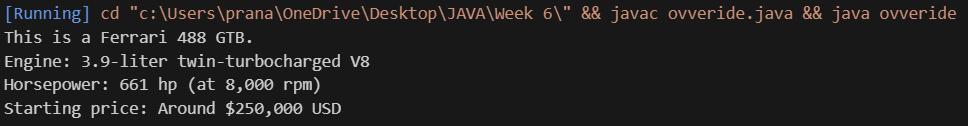
        c1.displayinfo();

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| 1. | Semi colon (;) | Give the semi colon (;) in each line where it is required |
| 2. | Syntax Error | Giving Capital ‘S’ in printing statements (System.out.println) |

Output: 

**IMPORTANT POINTS:**

In order to do this, we have to use inheritance concept. Here we used the multi-inheritance concept.

Program 2

Aim: A college is developing an automated admission system that verifies students eligibility for undergraduate(UG) and post graduate(PG) programs Each program has different eligibility criteria based on theh student’s percentage in the previous qualification.

UG admission require a minimum of 60%.

PG admission require a minimum of 70%.

**Class Diagram:**

|  |
| --- |
| **adm** |
| elg():void |

|  |  |
| --- | --- |
| ug | pg |
| +elg():void | +elg():void |

Code:

class student{

    String name;

    double percentage;

public student(String name,double percentage){

    this.name=name;

    this.percentage=percentage;

    }

}

class UG extends student{

    public UG(String name,double percentage){

        super(name,percentage);

    }

    void eligibility(){

        if(percentage>=60){

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

        }

        else{

            System.err.println(name+" is not eligible for UG addmission.");

        }

    }

}

class PG extends student{

    public PG(String name,double percentage){

        super(name,percentage);

    }

    void eligibility(){

        if(percentage>=70){

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

        }

        else{

            System.err.println(name+" is not eligible for PG addmission.");

        }

    }

}

public class Admission{

    public static void main(String[] args) {

        UG s1=new UG("Dhanush",90);

        PG s2=new PG("Karthik",90);

        s1.eligibility();

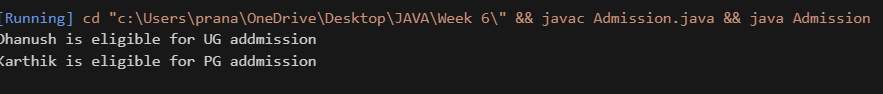
        s2.eligibility();

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| **Sl.No** | **Error name** | **Error Rectification** |
| **1.** | **Semi colon (;)** | **Give the semi colon (;) in each line where it is required** |
| **2.** | **Syntax Error** | **Giving Capital ‘S’ in printing statements (System.out.println)** |

Output: 

**IMPORTANT POINTS:**

Super keyword is used take the method,variable,constructor from the super class.

Program 3

Aim: Create a calculator class with overloaded methods to perform addition.

1. Add 2 integers.
2. Add 2 doubles.
3. Add 3 integers.

Class Diagram:

|  |
| --- |
| calculator |
| +add(int a,int b):int  +add(double a,double b):double  +add(int a,int b,int c):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;

    }

}

public class Overload {

    public static void main(String[] args) {

        Calculator a = new Calculator();

        System.out.println("Sum is " + a.add(5, 6));

        System.out.println("Sum is " + a.add(5.6, 6));

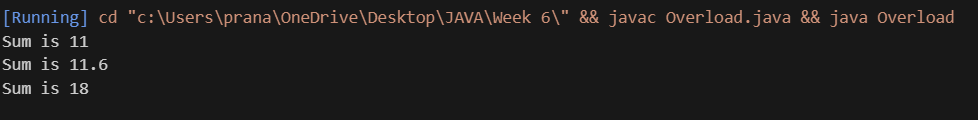
        System.out.println("Sum is " + a.add(5,6,7));

    }

}

Error table:

|  |  |  |
| --- | --- | --- |
| S.No | Expected Error | Reason |
| 1 | Setting the parameters inside the constructor | We cannot pass the values inside constructor without setting them first |
| 2 | } | Ending the class and main method is required |

Output: 

Program 4

AIM: Create a shape class with a method calculateArea() that is overloaded for different shapes. Then create a subclass circle that overrides the calculator method for a circle.

**CLASS DIAGRAM:**

|  |
| --- |
| shape |
| +calarea(float side):float  +calarea(float l,float b):float  +calarea(float c):float |

|  |
| --- |
| Circle |
| +calarea(double r):double |

Code:

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 3.14\*radius\*radius;

    }

}

public class overrides {

    public static void main(String[] args) {

        circle c=new circle();

        shape s=new shape();

        shape r=new shape();

        System.out.println("Area of circle = "+c.calculatearea(5));

        System.out.println("Area of square = "+s.calculatearea(5));

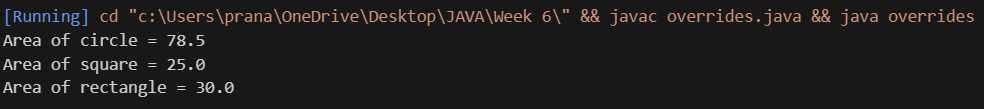
        System.out.println("Area of rectangle = "+s.calculatearea(5,6));

    }

}

**ERROR TABLE:**

|  |  |  |
| --- | --- | --- |
| Sl.No | Error name | Error Rectification |
| 1. | Semi colon (;) | Give the semi colon (;) in each line where it is  required |
| 2. | Syntax Error | Giving Capital ‘S’ in printing statements (System.out.println) |

Output: 

**IMPORTANT POINTS:**

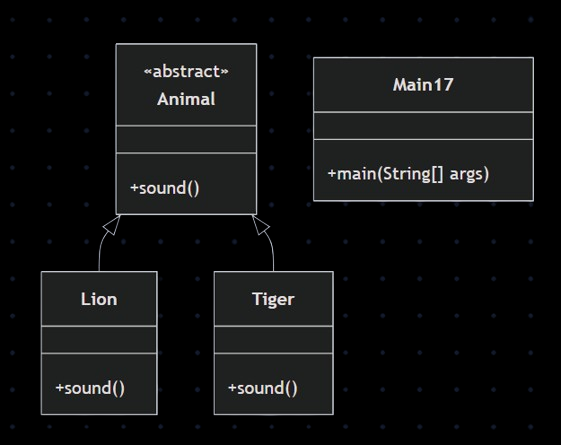
In this program we use both method overloading and overriding to calculate area of different shapes.

WEEK-7

Program 1

AIM: write a java program to create an abstract class animal with an abstract method called sound(). Create subclasses lion and tiger that extend the animal class and implement the sound() method to make a specific sound for each animal.

Class Diagram:



Code:

abstract class Animal {

    abstract void sound();

}

class Lion extends Animal {

    void sound() {

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

    }

}

class Tiger extends Animal {

    void sound() {

        System.err.println("Tiger Roars");

    }

}

public class Main17 {

    public static void main(String[] args) {

        System.err.println("P.Pranav,24244,CSE-C");

        Lion l = new Lion();

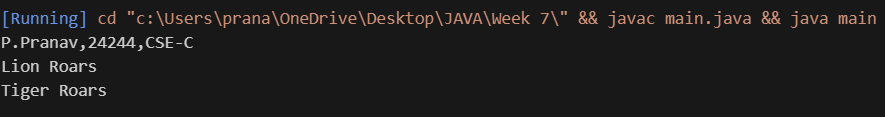
        Tiger t = new Tiger();

        t.sound();

    l.sound();

    }

}

Output:  IMPORTANT POINTS:

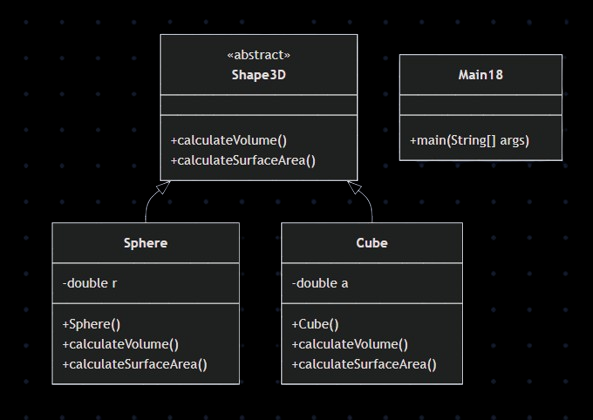
Both Lion and Tiger override the sound () method from the Animal abstract class to provide their own specific implementation.

The class Animal is declared as abstract meaning it cannot be instantiated directly. It serves as a blueprint for other classes.

Program 2

AIM: write a java program to create an abstract class Shape3D with abstract methods calcluateVolume() 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:



Code:

abstract class Shape3D{

    abstract double calculateVolume();

    abstract double calculateSurfaceArea();

}

class Sphere extends Shape3D{

    private double radius;

    public Sphere(double radius){

        this.radius=radius;

    }

    public double calculateVolume(){

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

    }

    public double calculateSurfaceArea(){

        return 4.0\*Math.PI\*Math.pow(radius,2);

    }

}

class Cube extends Shape3D{

    private double side;

    public Cube(double side){

        this.side=side;

    }

    public double calculateVolume(){

        return Math.pow(side,3);

    }

    public double calculateSurfaceArea(){

        return 6.0\*Math.pow(side,2);

    }

}

public class Main18{

    public static void main(String[] args){

        System.err.println("P.Pranav,24244,CSE-C");

        Sphere s=new Sphere(5);

        System.err.println("Surface area of Sphere= "+s.calculateSurfaceArea());

        System.err.println("Volume of Sphere= "+s.calculateVolume());

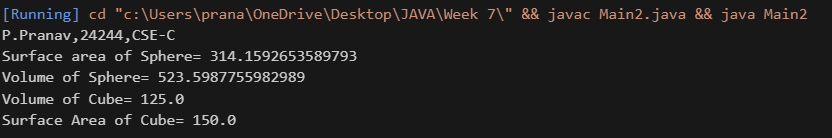
        Cube c=new Cube(5);

        System.err.println("Volume of Cube= "+c.calculateVolume());

        System.err.println("Surface Area of Cube= "+c.calculateSurfaceArea());

    }

}

Output: 

IMPORTANT POINTS:

Each shape class:

Calculates and prints volume and surface area using its respective formulas.

Demonstrates method overriding with specific implementations in each subclass.

**Program 3**

AIM: write a java program using an abstract class to define a method for pattern printing

create an abstract class named PatternPrinter with an abstract method printPattern(int n) and a concrete method to display the

pattern title.

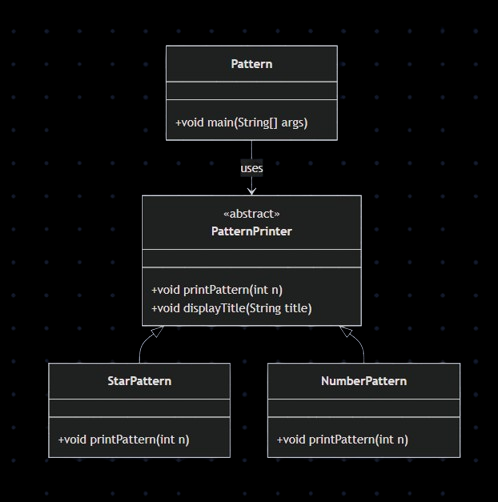
implement the two subclasses:

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

2. NumberPattern-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:



Code:

import java.util.Scanner;

abstract class PatternPrinter {

abstract void printPattern(int n);

void displayTitle(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) {

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

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

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

}

System.out.println();

}

}

}

public class Pattern {

public static void main(String[] args) {

System.err.println("P.Pranav,24244,CSE-C");

Scanner input = new Scanner(System.in);

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

int rows = input.nextInt();

PatternPrinter star = new StarPattern();

star.displayTitle("Star Pattern:");

star.printPattern(rows)

PatternPrinter number = new NumberPattern();

number.displayTitle("Number Pattern:");

number.printPattern(rows);

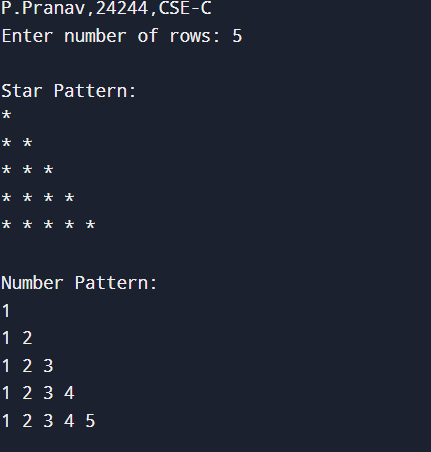
input.close();

}}

ERROR TABLE:

|  |  |  |
| --- | --- | --- |
| s.no | ERRORS | Error rectification |
| 1. | Output lacks spacing between patterns for clarity. | Add a  System.out.println(); between the two patterns. |

Output:



IMPORTANT POINTS:

Method Overriding  
 print Pattern () is overridden in both Star Pattern and Number Pattern.

Polymorphism  
 Parent reference (Pattern Printer) is used to call subclass methods.