

23CSE111
OBJECT ORIENTED PROGRAMMING
LAB REPORT



Department of Computer Science Engineering
Amrita School of Computing
Amrita Vishwa Vidyapeetham,
Amaravati Campus

Name: M.Nishanth

Roll No: AV.SC.U4CSE24217

Verified By :

INDEX

S.NO	EXPERIMENT	SIGNATURE
1.	Installing java in our device.	
2.	Program for entering student details.	
3.	Program to calculate the simple intrest	
4.	To find Fibonacci series upto some number.	
5.	To find temperature in Celsius.	
6.	To find factorial of a given number.	
7.	Program to convert from Celsius to fahrenheit	
8.	Program to calculate the area of a triangle using heron's formula.	
9.	To create java program with following instructions 1.Create a class with name car 2. Create four attributes named car_color ,Car_brand,fuel_type,mileage 3. Create three methods named start(), stop(). Service() 4. Create three objects named car1,car2 and car3	
10.	To create a class bankAccount with methods deposit() and withdrawl	
11.	Write a java program with class named "book". the class should contain various attributes such as title, author, year of publication. it should also contain a constructor with parameters which	

	initializes title, author, year of publication and create a method which displays the details of 2 books.	
12.	Write a java program with class named “myclass” with a static variable count of int type. initialize it to zero and a constant variable “pi” of type double initialized to “3.14” as attributes of that class. now define a constructor for “myclass”, that increments the count variable each time an object of “myclass” is created. finally, print the final values of ‘count’ and ‘pi’ variables and create 3 objects.	
13.	create a calculator using the operations add, subtarct, multiplication and division using multilevel inheritance and display the desired output.	
14.	<p>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 trucks and they need a program to store details about each vehicle, such as brand and speed.</p> <p>*Cars should have an additional property such as number_of_doors .</p> <p>*Bikes should have a property indicating whether they have gears or not.</p>	

	<p>*Truck should have a property of their capacity(in tons).</p> <p>*Every class should have a constructor.</p>	
15.	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.	
16	<p>Create a calculator class with overloaded methods to perform addition:</p> <p>1.add two integers</p> <p>2.add 2 doubles</p> <p>3.add 3 integers</p>	
17	Create a shape class with a method “calculateArea()” that is overloaded for different shapes(ex:square,rectangle,triangle).then create a subclass “Circle” that overrides the calcArea() method for a circle.	
18.	<p>A college is developing an automated admission system that verifies student’s eligibility for Undergraduate(UG) and Postgraduate(PG) programs. Each program has different eligibility criteria based on the student’s percentage in their previous qualification.</p> <p>CONDITION:</p>	

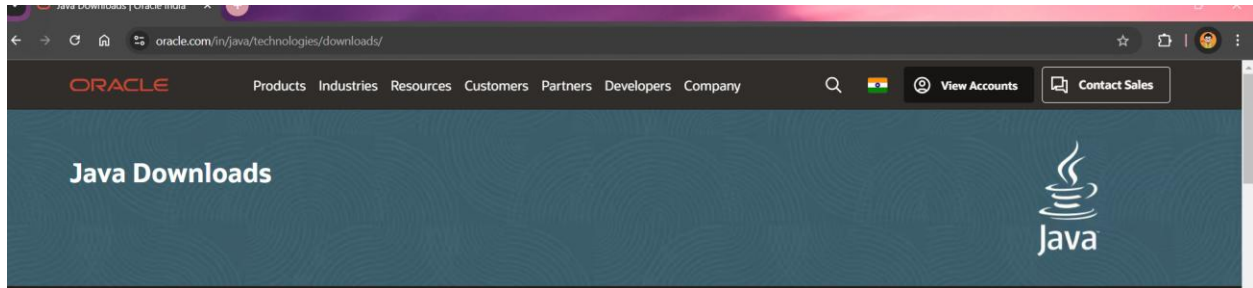
	1.UG admission require minimum of 60%,PG admission require minimum of 70%.	
19	WRITE A JAVA PROGRAM TO CREATE AN ABSTRACT CLASS “ANIMAL” WITH AN ABSTRACT METHOD CALLED “Sound()”. CREATE SUBCLASSES LION AND TIGER THAT EXTEND THE ANIMAL CLASS AND IMPLEMENT THE “Sound()” METHOD TO MAKE A SPECIFIC SOUND FOR EACH ANIMAL.	
20	WRITE A JAVA PROGRAM TO CREATE AN ABSTRACT CLASS “SHAPE3D” WITH ABSTRACT METHODS “calculateVolume()” AND “calculateSurfacArea()”. CREATE SUBCLASSES SPHERE AND CUBE THAT EXTEND THE SHAPE3D CLASS AND IMPLEMENT THE RESPECTIVE METHODS TO CALCULATE THE VOLUME AND SURAFEC AREA OF EACH SHAPE.	
21	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)” AND A CONCRETE METHOD TO DISPLAY THE PATTERN TITLE. IMPLEMENT 2 SUBCLASSES, 1.STAR PATTERN PRINTS A RIGHT ANGLED TRIANGLE OF STARS 2.NUMBER PATTERN PRINTS A RIGHT ANGLED TRAINGLE OF INCREASING NUMBERS.	

	IN THE MAIN METHOD, CREATE OBJECTS OF BOTH SUBCLASSES AND PRINT PATTERN FOR A GIVEN NUMBER OF ROWS.	
22	<p>WRITE A JAVA PROGRAM TO CREATE AN INTERFACE “Playable” WITH A METHOD “Play()” THAT TAKES NO ARGUMENTS AND RETURNS VOID.</p> <p>CREATE 3 CLASSES Football, Volleyball and Basketball THAT IMPLEMENTS THE Playable INTERFACE AND OVERRIDE THE play() METHOD TO PLAY RESPECTIVE SPORTS</p>	
23	<p>WRITE A JAVA PROGRAM TO CREATE AN INTERFACE “Shape” WITH THE “getPerimeter()” METHOD. CREATE 3 CLASSES “Rectangle”, “Circle” and “Triangle” THAT IMPLEMENTS THE SHAPE INTERFACE. IMPLEMENT THE “getPerimter()” METHOD FOR EACH OF 3 CLASSES.</p>	

OBJECT ORIENTED PROGRAMMING LAB REPORT

TASK 1: JAVA INSTALLATION ON WINDOWS

STEP 1: Go to the official website.



STEP 2: Install JDK 21.

JDK 23	JDK 21	GraalVM for JDK 23	GraalVM for JDK 21
Java SE Development Kit 21.0.6 downloads			
JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC) .			
JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the Java SE OTN License (OTN) and production use beyond the limited free grants of the OTN license will require a fee .			
Linux	macOS	Windows	
Product/file description		File size	Download
x64 Compressed Archive		185.92 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer		164.31 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer		163.06 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

STEP 2: Environmental Variable Set-Up.

STEP 3: Set the path to C:\Program Files\Java\JDK 21\Bin

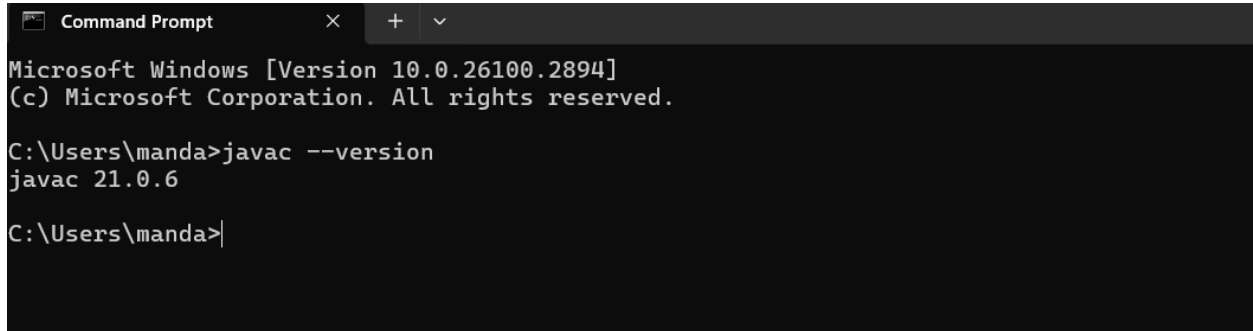
(The files will be stored in the bin)

STEP 4: Search for the environmental variables in the Environmental Variable Set-Up.

STEP 5: Select the path in system variables for multiple users.

STEP 7: Paste the above selected path in the “New” section.

STEP 8: Go to the Command Prompt Window and verify the installation by typing “—version” or “javac –version”.



```
Command Prompt
Microsoft Windows [Version 10.0.26100.2894]
(c) Microsoft Corporation. All rights reserved.

C:\Users\manda>javac --version
javac 21.0.6

C:\Users\manda>
```

TASK 2: First Java Program

AIM: Execute First Java Program with Student Details.

PROGRAM:

```
public class StudentDetails {
    public static void main(String[] args) {
        String name = "Nishanth";
        int age = 18;
        String course = "Computer Science and Engineering";
        String section = "CSE-C";

        System.out.println("Student Details:");
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Course: " + course);
        System.out.println("Section: " + section);
    }
}
```


OUTPUT:

```
C:\Users\manda\OneDrive\Desktop\WebDevProje
Student Details:
Name: Nishanth
Age: 18
Course: Computer Science and Engineering
Section: CSE-C
```

ERRORS:

1	Syntax error	Semicolon Added
2	Runtime Error	Copied correct path
3	Name Error	Rectified

WEEK-2

AIM: WRITE A JAVA PROGRAM TO CALCULATE THE SIMPLE INTEREST WITH INPUTS

PROGRAM:

```
public class SimpleInterest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the principal amount: ");
        double principal = scanner.nextDouble();

        System.out.print("Enter the rate of interest (in %): ");
```

```
double rate = scanner.nextDouble();

System.out.print("Enter the time period (in years): ");

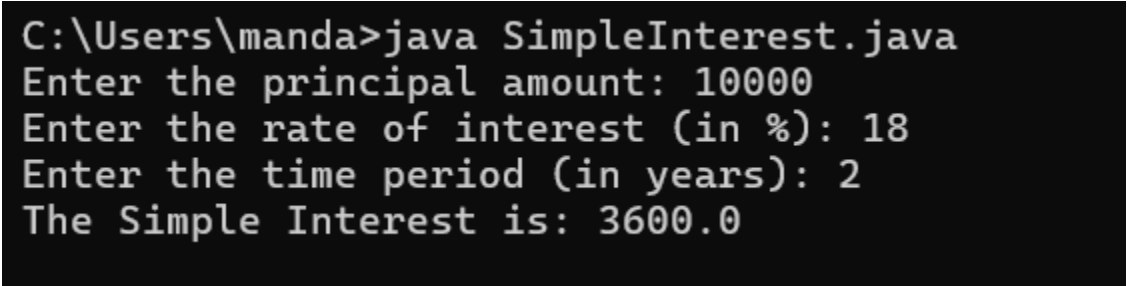
double time = scanner.nextDouble();

double simpleInterest = (principal * rate * time) / 100;

System.out.println("The Simple Interest is: " + simpleInterest);

scanner.close();
}
}
```

OUTPUT:



```
C:\Users\manda>java SimpleInterest.java
Enter the principal amount: 10000
Enter the rate of interest (in %): 18
Enter the time period (in years): 2
The Simple Interest is: 3600.0
```

NEGATIVE CASE:

```
C:\Users\manda>java SimpleInterestCalculator.java
Enter the principal amount: ten
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2573)
    at SimpleInterestCalculator.main(SimpleInterestCalculator.java:10)
```

1. When we give input in decimal form for principal value it will show error.

2. It can't compile the code and can't show the output

IMPORTANT POINTS:

1. We have used scanner method to give input to the code

2. Formula of simple interest is $(P \times T \times R) / 100$

3. We have assigned values for P, T, R

Errors:

S.No	Error Type	Reason for error	Rectification
1	Runtime error	Incorrect path	Copied correct path
2	Syntax error	{ missing	{ added

1. AIM: WRITE A JAVA PROGRAM TO FIND THE FACTORIAL OF A NUMBER

PROGRAM:

```
import java.util.Scanner;
```

```
public class FactorialCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```

        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        int result = factorial(number);
        System.out.println("Factorial of " + number + " is: " + result);
        scanner.close();
    }

    public static int factorial(int n) {
        if (n == 0) {
            return 1;
        } else {
            return n * factorial(n - 1);
        }
    }
}

```

OUTPUT:

```

C:\Users\manda>java Factorial.java
Enter a number: 5
Factorial of 5 is: 120

```

NEGATIVE CASE:

```

C:\Users\manda>java Factorial.java
Enter a number: 12.5
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
    at Factorial.main(Factorial.java:7)

```

1. When we give input in decimal form for principal value it will show error.
2. It can't compile the code and can't show the output.

ERROR:

S.No	Error type	Reason for error	rectification
1.	Logical error	Incorrect input	Correcting input
2.	Runtime error	Incorrect path	Using correct path

IMPORTANT POINTS:

1. Importing Scanner:

- The code imports the Scanner class from the java.util package to read user input.

2. Main Method:

- The main method is the entry point of the program. It creates a Scanner object to read user input and prompts the user to enter a number.

3. Factorial Calculation: The factorial method is defined as a static method that takes an integer n and returns the factorial of n.

- method uses recursion to calculate the factorial. If n is 0, it returns 1 (base case). Otherwise, it returns $n * \text{factorial}(n - 1)$.

4. Closing Scanner:

- The scanner.close() method is called to close the Scanner object and release the resources associated with it.

5. Recursion:

The use of recursion in the factorial method is a key point. It repeatedly calls itself with decremented values of n until it reaches the base case

3 . AIM:WRITE A JAVA PROGRAM TO CONVERT CELCIUS TO FAHRENHEIT

PROGRAM :

```
import java.util.Scanner;

public class CelsiusToFahrenheit {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter temperature in Celsius: ");
        double celsius = scanner.nextDouble();
        double fahrenheit = celsiusToFahrenheit(celsius);
        System.out.println("Temperature in Fahrenheit: " + fahrenheit);
        scanner.close();
    }

    public static double celsiusToFahrenheit(double celsius) {
        return (celsius * 9/5) + 32;
    }
}
```

OUTPUT:

```
C:\Users\manda>java New.java
Enter temperature in Celsius: 88
Temperature in Fahrenheit: 190.4
```

NEGATIVE CASE:

```
C:\Users\manda>java New.java
Enter temperature in Celsius: twenty
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2573)
    at CelsiusToFahrenheit.main(New.java:7)
```

IMPORTANT POINTS TO NOTE:

1. Importing Scanner Class:

- The Scanner class is imported from java.util.Scanner to take user input.

2. Class Declaration:

- The class name CelsiusToFahrenheit follows Java naming conventions (PascalCase).

3. Main Method:

- The public static void main(String[] args) method is the entry point of the program

S.No	Error type	Reason for error	rectification
1.	Logical error	Incorrect input	Correcting input
2.	Runtime error	Incorrect path	Using correct path

3.AIM: JAVA PROGRAM TO CONVERT FAHRENHEIT TO CELCIUS

PROGRAM:

```
import java.util.Scanner;
```

```
public class FahrenheitToCelsius {
```

```

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter temperature in Fahrenheit: ");
    double fahrenheit = scanner.nextDouble();
    double celsius = fahrenheitToCelsius(fahrenheit);
    System.out.println("Temperature in Celsius: " + celsius);
    scanner.close();
}

public static double fahrenheitToCelsius(double fahrenheit) {
    return (fahrenheit - 32) * 5/9;
}
}

```

OUTPUT:

```

C:\Users\manda\OneDrive\Desktop\java funda
Enter temperature in Fahrenheit: 545
Temperature in Celsius: 285.0

```

NEGATIVE CASE:

```

C:\Users\manda\OneDrive\Desktop\java fundamentals>15>
The syntax of the command is incorrect.

```

IMPORTANT POINTS:

- `import java.util.Scanner;` is used to include the Scanner class to take user input.

1.Class Declaration:

- The class is named FahrenheitToCelsius which follows Java naming conventions.

2. Main Method:

- The program execution starts from the main() method:

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

3.Scanner Class:

- The Scanner object is created to take input from the user:

```
Scanner scanner = new Scanner(System.in);
```

ERRORS:

S.No	Error type	Reason for error	Rectification
1.	Logical error	Due to incorrect input	Corrected by giving correct input
2.	Runtime error	Incorrect path	Using correct path

5 . AIM: WRITE A JAVA PROGRAM TO FIND THE AREA OF A TRIANGLE

PROGRAM:

```
import java.util.Scanner;
```

```
public class TriangleArea {
```

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
    System.out.print("Enter the base of the triangle: ");  
    double base = scanner.nextDouble();  
    System.out.print("Enter the height of the triangle: ");  
    double height = scanner.nextDouble();  
    double area = calculateArea(base, height);  
    System.out.println("The area of the triangle is: " + area);  
    scanner.close();  
}
```

```
public static double calculateArea(double base, double height) {  
    return (base * height) / 2;  
}  
}
```

OUTPUT:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>  
Enter the base of the triangle: 12  
Enter the height of the triangle: 15  
The area of the triangle is: 90.0
```

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>java AreaofTriangle.java
Enter the base of the triangle: 88"
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2573)
    at TriangleArea.main(AreaofTriangle.java:7)
```

IMPORTANT POINTS:

User Input:

- The user is prompted to enter:
 - Base of the triangle
 - Height of the triangle

Method Call:

- The method calculateArea(base, height) is called to calculate the area of the triangle.

Return Value:

- The method returns the area value to the main() method.

S.No	Error type	Reason for error	rectification
2.	Runtime error	Incorrect path	Using correct path
3.	Syntax error	No semicolon	Using semicolon

ERRORS:

6. AIM: WRITE A JAVA PROGRAM TO FIND THE AREA OF A RECTANGLE

PROGRAM:

```
import java.util.Scanner;

public class RectangleArea {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the length of the rectangle: ");

        double length = scanner.nextDouble();

        System.out.print("Enter the width of the rectangle: ");

        double width = scanner.nextDouble();

        double area = calculateArea(length, width);

        System.out.println("The area of the rectangle is: " + area);

        scanner.close();

    }

    public static double calculateArea(double length, double width) {

        return length * width;

    }

}
```

OUTPUT:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals
Enter the length of the rectangle: 7
Enter the width of the rectangle: 3
The area of the rectangle is: 21.0
```

NEGATIVE CASE:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>java AreaofRect.java
Enter the length of the rectangle: ten
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextDouble(Scanner.java:2573)
    at RectangleArea.main(AreaofRect.java:8)
```

IMPORTANT POINTS:

1. Purpose: The program calculates the area of a rectangle based on the user-provided length and width.

2. User Input:

- It prompts the user to input the rectangle's length and width.
- The input is read using the Scanner class and stored as double values to allow for decimal measurements.

3.Modularity:

- The calculation logic is placed in a separate method, calculateArea(), which takes length and width

ERRORS:

1	Syntax error	Semicolon added
---	--------------	-----------------

2.	Name error	rectified
----	------------	-----------

7.AIM: WRITE A JAVA PROGRAM FOR FIBONACCI SEQUENCE

PROGRAM:

```
import java.util.Scanner;
```

```
public class Fibonacci {
```

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

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

```
        int num;
```

```
        int f3;
```

```
        int f1 = 0;
```

```
        int f2 = 1;
```

```
        int i = 2;
```

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

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

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

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

```
        while(i < num) {
```

```

        f3 = f1 + f2;

        f1 = f2;

        f2 = f3;

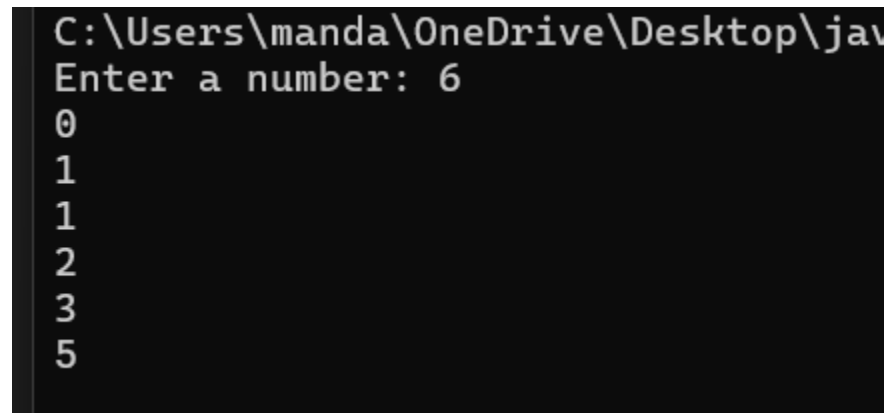
        System.out.println(f3);

        i = i + 1;
    }

    sc.close();
}
}

```

OUTPUT:

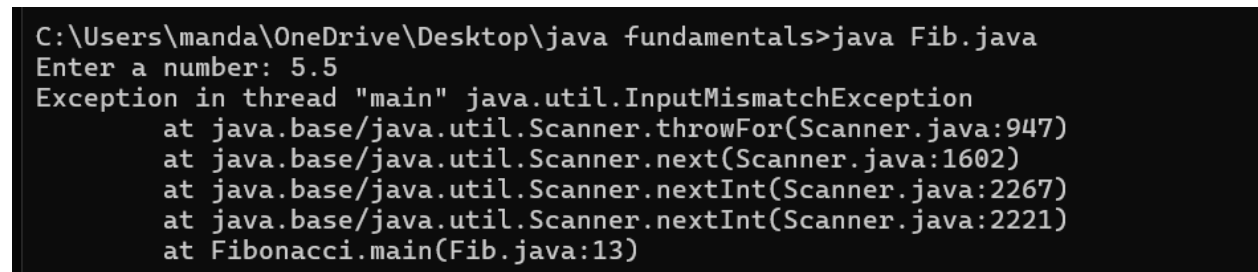


```

C:\Users\manda\OneDrive\Desktop\java fundamentals>java Fib.java
Enter a number: 6
0
1
1
2
3
5

```

NEGATIVE CASE:



```

C:\Users\manda\OneDrive\Desktop\java fundamentals>java Fib.java
Enter a number: 5.5
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:947)
    at java.base/java.util.Scanner.next(Scanner.java:1602)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2267)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2221)
    at Fibonacci.main(Fib.java:13)

```

1.I have rearranged the starting two elements from 0 and 1 to 1 and 0.

2. Therefore, the order of Fibonacci series is changed.

ERRORS:

S.No	Error type	Reason for error	Rectification
1	Name error	Incorrect usage of function	Correcting by using correct formula
2	Syntax error	No semicolon	Semicolon added
3	Runtime error	Incorrect path	Copied correct path

IMPORTANT POINTS:

1. Importing Scanner:

- The code imports the Scanner class from the java.util package to read user input.

2. Class Declaration:

- The code declares a class named fibo.

3. Main Method:

- The main method is the entry point of the program. It creates a Scanner object to read user input and prompts the user to enter a number.

4. Variable Initialization:

- The code initializes variables:
 - num: The number of Fibonacci terms to be generated.
 - f3: Used to store the next Fibonacci term.

- f1 and f2: The first two terms of the Fibonacci sequence, initialized to 0 and 1, respectively.
- i: Counter variable initialized to 2 since the first two terms are already known.

WEEK-3

Aim:

To create java program with following instructions

- 1.Create a class with name car**
- 2. Create four attributes named car colour ,Car brand, fuel type, mileage**
- 3. Create three methods named start(), stop(). Service()**
- 4. Create three objects named car1, car2 and car3**

PROGRAM:

```
import java.util.*;

class car
{

    public String Car_color;

    public String Car_brand;

    public String fuel_type;

    public int mileage;

    public void start()
```

```
{  
  
    System.out.println("Car Started:");  
    System.out.println("Car color is :"+Car_color);  
    System.out.println("Car Brand is:"+Car_brand);  
    System.out.println("Car fuel type is:"+fuel_type);  
    System.out.println("Car mileage is:"+mileage);  
}
```

```
public void service()
```

```
{  
    System.out.println("Car Started:");  
    System.out.println("Car color is :"+Car_color);  
    System.out.println("Car Brand is:"+Car_brand);  
    System.out.println("Car fuel type is:"+fuel_type);  
    System.out.println("Car mileage is:"+mileage);  
}
```

```
public void stop()
```

```
{  
    System.out.println("Car Started:");  
    System.out.println("Car color is :"+Car_color);  
    System.out.println("Car Brand is:"+Car_brand);
```

```
        System.out.println("Car fuel type is:"+fuel_type);
        System.out.println("Car mileage is:"+mileage);
    }

    public static void main(String args[])
    { System.out.println("\nNishanth\n\n");

        car car1 = new car();
        car1.Car_color = "Black";
        car1.Car_brand = "Mercedes";
        car1.fuel_type = "Diesel";
        car1.mileage = 100;
        car1.start();

        car car2 = new car();
        car2.Car_color = "White";
        car2.Car_brand = "BMW";
        car2.fuel_type = "Petrol";
        car2.mileage = 200;
        car2.stop();

        car car3 = new car();
        car3.Car_color = "Red";
        car3.Car_brand = "Skoda";
        car3.fuel_type = "Petrol";
        car3.mileage = 300;
        car3.service();
```

```
}  
}
```

OUTPUT:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>ja  
Nishanth  
  
Car Started:  
Car color is :Black  
Car Brand is:Mercedes  
Car fuel type is:Diesel  
Car mileage is:100  
Car Started:  
Car color is :White  
Car Brand is:BMW  
Car fuel type is:Petrol  
Car mileage is:200  
Car Started:  
Car color is :Red  
Car Brand is:Skoda  
Car fuel type is:Petrol  
Car mileage is:300
```

CLASS DIAGRAM:



- Car_color: String - Car_brand: String - fuel_type: String - mileage: int
+ start(): void + service(): void + stop(): void

IMPORTANT POINTS:

1. The car class has four attributes: Car_color, Car_brand, fuel_type, and mileage.
2. It also has three methods: start(), service(), and stop().
3. The start(), service(), and stop() methods all print the same details about the car.
4. Each method prints the car's color, brand, fuel type, and mileage to the console
5. The main method creates three instances of the car class: car1, car2, and car3.
6. Each car object is assigned specific values for Car_color, Car_brand, fuel_type, and mileage.

AIM:

WRITE A JAVA CLASS FOR A BANK ACCOUNT WITH DEPOSIT() AND WITHDRAW() AS METHODS.

PROGRAM:

```
class BankAccount {  
    private double balance;  
  
    public BankAccount(double initialBalance) {  
        if (initialBalance > 0) {  
            this.balance = initialBalance;  
        } else {  
            this.balance = 0;  
        }  
    }  
  
    public void deposit(double amount) {  
        if (amount > 0) {  
            balance += amount;  
            System.out.println("Deposited $" + amount);  
        } else {  
            System.out.println("Deposited amount must be positive");  
        }  
    }  
  
    public double getBalance() {
```

```

        return balance;
    }
}

public class Main1 {
    public static void main(String args[]) {
        BankAccount account = new BankAccount(50000);
        account.deposit(25000);
        System.out.println("Current Balance is:" + account.getBalance());
    }
}

```

OUTPUT:

```

C:\Users\manda\OneDrive\Desktop\java fundamentals>java BankAccount
Enter Account holder name: Hari
Enter Account number: 24210
Enter current amount: 100000.0
Deposited amount is :
Deposited amount is: 10000
Total current amount is: 110000.0
withdraw amount is: 500
Current amount is: 109500.0

```

CLASS DIAGRAM:

BankAccount
-balance: double
+BankAccount(double initialBalance

<pre>+deposit(doubleamount): void + getBalance(): double</pre>
--

IMPORTANT POINTS:

The Bank Account class has a private attribute balance to store the account balance.

- The class has a constructor, BankAccount(double initialBalance), which initializes the balance. If the initial balance is not positive, it sets the balance to 0.
- The deposit(double amount) method adds a positive amount to the balance and prints a message. If the deposit amount is not positive, it prints an error message.
- The getBalance() method returns the current balance of the account.
- The Main1 class contains the main method, which serves as the entry point of the program.
- In the main method, an instance of Bank Account is created with an initial balance of 1000.

WEEK-4

1.AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “Book”. THE CLASS SHOULD CONTAIN VARIOUS ATTRIBUTES SUCH AS TITLE, AUTHOR, YEAR OF PUBLICATION. IT SHOULD ALSO CONTAIN A CONSTRUCTOR WITH PARAMETERS WHICH INITIALIZES TITLE, AUTHOR, YEAR OF PUBLICATION AND CREATE A METHOD WHICH DISPLAYS THE DETAILS OF 2 BOOKS.

PROGRAM:


```
public class Book {  
    public String title;  
    public String author;  
    public int year;  
  
    Book(String title, String author, int year) {  
        this.title = title;  
        this.author = author;  
        this.year = year;  
    }  
  
    public void displayDetails() {  
        System.out.println("Title: " +title);  
        System.out.println("Author: " +author);  
        System.out.println("Year of Publication" +year);  
    }  
  
    public static void main(String[] args) {  
        Book b1 = new Book("Math", "Ramanujan", 1950);  
        Book b2 = new Book("Physics", "CV Raman", 1960);  
  
        b1.displayDetails();  
        b2.displayDetails();  
    }  
}
```

```
}  
  
}
```

OUTPUT:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>java Book.java  
Title: Math  
Author: Ramanujan  
Year of Publication1950  
Title: Physics  
Author: CV Raman  
Year of Publication1960
```

NEGATIVE CASE:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>javac Book.java  
Book.java:23: error: ';' expected  
        b2.displayDetails()  
                        ^  
1 error
```

ERROR:

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

CLASS DIAGRAM:

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

IMPORTANT POINTS:

1. Constructor:

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

2.Method:

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

3. Object Creation:

- Two objects `b1` and `b2` are created using the constructor.

2.AIM: WRITE A JAVA PROGRAM WITH CLASS NAMED “MyClass” WITH A STATIC VARIABLE COUNT OF INT TYPE. INITIALIZE IT TO ZERO AND A CONSTANT VARIABLE “Pi” OF TYPE DOUBLE INITIALIZED TO “3.14” AS ATTRIBUTES OF THAT CLASS. NOW DEFINE A CONSTRUCTOR FOR “MyClass”, THAT INCREMENTS THE COUNT VARIABLE EACH TIME AN OBJECT OF “MyClass” IS CREATED. FINALLY, PRINT THE FINAL VALUES OF ‘COUNT’ AND ‘PI’ VARIABLES AND CREATE 3 OBJECTS.

PROGRAM:

```
public class MyClass {  
    static int count = 0;
```

```
static final double pi = 3.14;
```

```
MyClass() {  
    count++;  
}
```

```
public static void main(String[] args) {  
    MyClass obj1 = new MyClass();  
    MyClass obj2 = new MyClass();  
    MyClass obj3 = new MyClass();  
  
    System.out.println("Count: " +count);  
    System.out.println("Pi: " +pi);  
}  
  
}
```

OUTPUT:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>java MyClass.java  
Count: 3  
Pi: 3.14
```

NEGATIVE CASE:

```
C:\Users\manda\OneDrive\Desktop\java fundamentals>java MyClass.java
error: no class declared in source file
```

ERROR:

S.No	Error Type	Reason for error	Rectification
1.	No class	No class name declared	Created class named 'MyClass'
2.	Syntax error	Not added keyword	Added keyword named 'new'

CLASS DIAGRAM:

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

IMPORTANT POINTS:

1.Static Keyword

- Static members belong to the **class, not to individual objects**.
- Only one copy of the static variable is maintained for all objects.

2.Static Variable

- **static int count:**
 - Shared among all objects of the class.
 - It is initialized only once and not for every object.
 - It increments every time the constructor is called.

3.Final Variable

- **static final double pi:**

- The **final** keyword makes the variable constant.
- Its value **cannot be changed** once assigned.
- It must be initialized at the time of declaration.

WEEK-5

AIM: Create a calculator using the operations add, subtract, multiplication and division using multilevel inheritance and display the desired output.

PROGRAM:

```
import java.util.Scanner;
```

```
class Addition {
```

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

```
        return a + b;
```

```
    }
```

```
}
```

```
class Subtraction extends Addition {
```

```
    public int subtract(int a, int b) {
```

```
        return a - b;
```

```
    }
```

```
}
```

```
class MultiplicationDivision extends Subtraction {  
    public int multiply(int a, int b) {  
        return a * b;  
    }  
  
    public double divide(int a, int b) {  
        if (b == 0) {  
            System.out.println("Division by zero is not allowed.");  
            return 0;  
        }  
        return (double) a / b;  
    }  
}
```

```
public class Calculator {  
    public static void main(String[] args) {  
        int num1 = 25;  
        int num2 = 2;  
  
        MultiplicationDivision calculator = new MultiplicationDivision();  
  
        System.out.println("Number 1: " + num1);  
        System.out.println("Number 2: " + num2);  
    }  
}
```

```

        System.out.println("Result (Addition): " + calculator.add(num1, num2));

        System.out.println("Result (Subtraction): " + calculator.subtract(num1,
num2));

        System.out.println("Result (Multiplication): " + calculator.multiply(num1,
num2));

        System.out.println("Result (Division): " + calculator.divide(num1, num2));

    }
}

```

OUTPUT:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-X
roject\bin' 'Calculator'
Number 1: 25
Number 2: 2
Result (Addition): 27
Result (Subtraction): 23
Result (Multiplication): 50
Result (Division): 12.5
PS C:\Users\manda>

```

NEGATIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeData
roject\bin' 'Calculator'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error, insert "]" to complete ClassBody

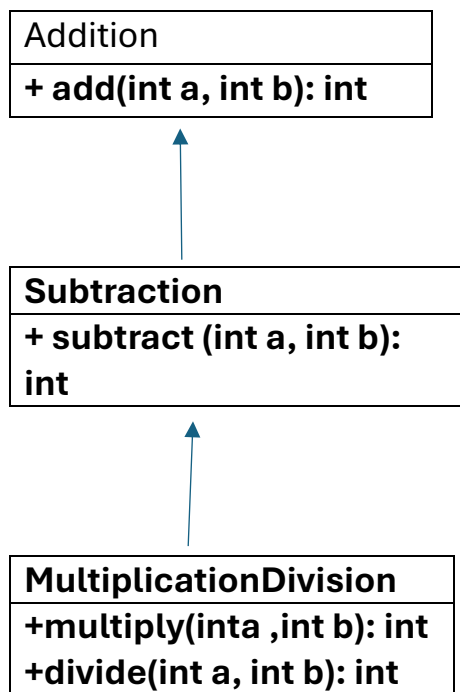
    at Calculator.main(Calculator.java:41)
PS C:\Users\manda>

```

ERROR TABLE:

S.No	Error	Rectificaton
1.	Compilation error	Removed the 'public' from main class

CLASS DIAGRAM:



IMPORTANT POINTS:

1. Inheritance Hierarchy

- The code demonstrates **multi-level inheritance**:
 - Addition → Subtraction → MultiplicationDivision
- Each child class **extends** the previous one, inheriting its methods while adding new functionality.

2. Method Overloading vs. Overriding

- There is **no method overriding** here (no method is redefined in child classes).
- Instead, each subclass **adds new methods**:

- Addition → add()
- Subtraction → subtract()
- MultiplicationDivision → multiply(), divide()

AIM:

Create a base class BankAccount with methods deposit and withdraw. create a 2 subclasses SavingsAcc and CheckingAcc and override the withdraw method in each subclass to impose different withdraw limits and fees and create a constructor and withdraw method in base class.

PROGRAM:

```
public class BankAcc {  
    protected double balance;  
  
    public BankAcc(double initialBalance) {  
        this.balance = initialBalance;  
    }  
  
    public void deposit(double amount) {  
        if (amount > 0) {  
            balance += amount;  
            System.out.println("Deposited: Rupees" + amount);  
        } else {
```

```
        System.out.println("Deposit amount must be positive.");
    }
}
```

```
public void withdraw(double amount) {
    if (amount > 0 && amount <= balance) {
        balance -= amount;
        System.out.println("Withdrawn: rupees" + amount);
    } else {
        System.out.println("Insufficient balance or invalid amount.");
    }
}
```

```
public void displayBalance() {
    System.out.println("Current Balance: Rupees" + balance);
}
}
```

```
class SavingsAccount extends BankAcc {
    private static final double WITHDRAW_LIMIT = 1000.0;

    public SavingsAccount(double initialBalance) {
        super(initialBalance);
    }
}
```

```
}
```

```
@Override
```

```
public void withdraw(double amount) {
```

```
    if (amount > WITHDRAW_LIMIT) {
```

```
        System.out.println("Withdrawal amount exceeds the limit of Rupees" +  
WITHDRAW_LIMIT);
```

```
    } else if (amount > 0 && amount <= balance) {
```

```
        balance -= amount;
```

```
        System.out.println("Withdrawn from Savings Account: Rupees" +  
amount);
```

```
    } else {
```

```
        System.out.println("Insufficient balance or invalid amount.");
```

```
    }
```

```
}
```

```
}
```

```
class CheckingAccount extends BankAcc {
```

```
    private static final double TRANSACTION_FEE = 2.0;
```

```
    public CheckingAccount(double initialBalance) {
```

```
        super(initialBalance);
```

```
    }
```

@Override

```
public void withdraw(double amount) {  
    double totalAmount = amount + TRANSACTION_FEE;  
    if (amount > 0 && totalAmount <= balance) {  
        balance -= totalAmount;  
        System.out.println("Withdrawn from Checking Account: Rupees" +  
amount + " (Fee: Rupees" + TRANSACTION_FEE + ")");  
    } else {  
        System.out.println("Insufficient balance or invalid amount.");  
    }  
}  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        SavingsAccount savings = new SavingsAccount(2000.0);  
        savings.displayBalance();  
        savings.deposit(500.0);  
        savings.withdraw(1200.0); // Exceeds limit  
        savings.withdraw(800.0); // Valid withdrawal  
        savings.displayBalance();  
  
        System.out.println();  
    }  
}
```

```

        CheckingAccount checking = new CheckingAccount(1500.0);
        checking.displayBalance();
        checking.deposit(300.0);
        checking.withdraw(100.0); // Includes transaction fee
        checking.withdraw(2000.0); // Insufficient balance
        checking.displayBalance();
    }
}

```

OUTPUT:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe'
n'
Current Balance: Rupees2000.0
Deposited: Rupees500.0
Withdrawal amount exceeds the limit of Rupees1000.0
Withdrawn from Savings Account: Rupees800.0
Current Balance: Rupees1700.0

Current Balance: Rupees1500.0
Deposited: Rupees300.0
Withdrawn from Checking Account: Rupees100.0 (Fee: Rupees2.0)
Insufficient balance or invalid amount.
Current Balance: Rupees1698.0
PS C:\Users\manda>

```

NEGATIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCod
n'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:

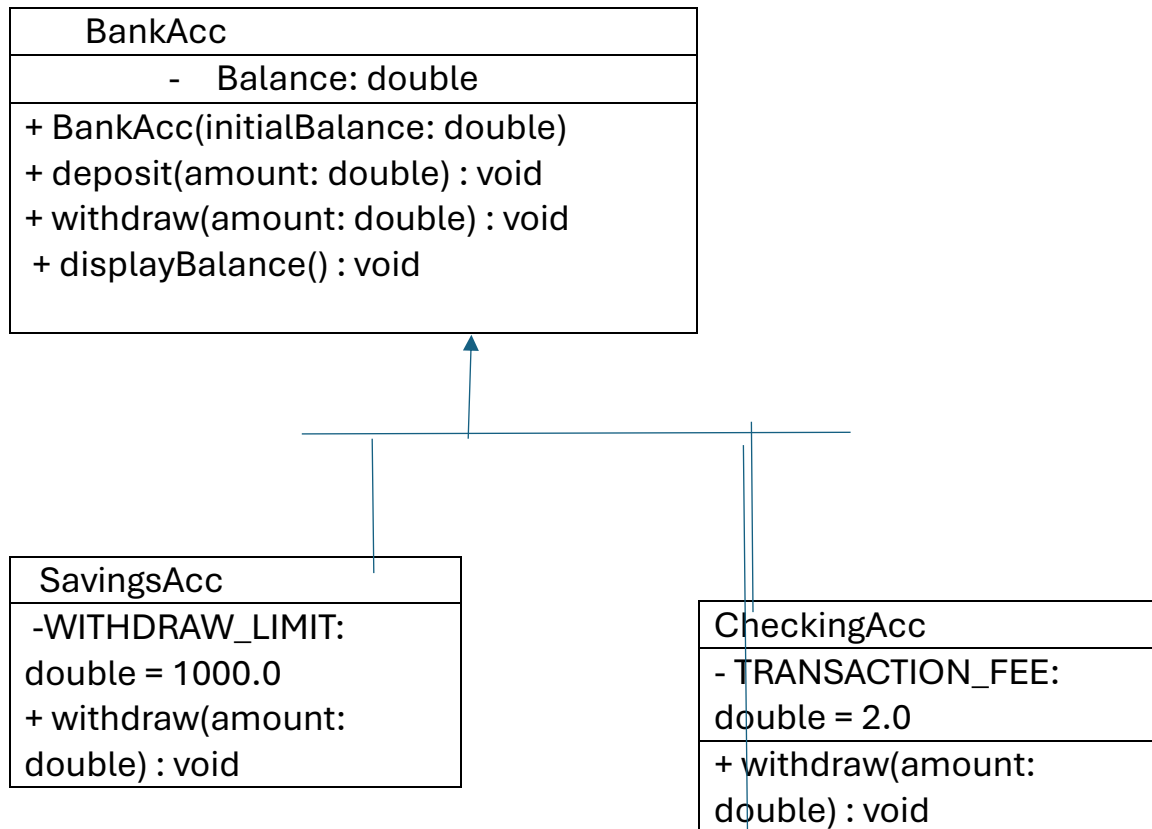
    at Main.main(BankAcc.java:71)

```

ERROR TABLE:

S.No	Error	Rectification
1.	Compilation error	Removed the 'public' from main class
2.	Syntax Error	Added parenthesis

CLASS DIAGRAM:



IMPORTANT POINTS:

1. BankAcc is the parent class with:
 - A protected balance attribute
 - Public methods for deposit, withdraw, and displayBalance
 - A constructor
2. SavingsAccount extends BankAcc and:

- Has a constant WITHDRAW_LIMIT
 - Overrides the withdraw method to enforce the limit
3. CheckingAccount extends BankAcc and:
- Has a constant TRANSACTION_FEE
 - Overrides the withdraw method to include the fee

AIM:

A vehicle rental company wants to develop a system that maintains

Information about different types of vehicles available for rent

The Company rents out cars, bikes and truck and they need a program to

Store details about each vehicle, such as brand and speed

Cars should have an additional property: number of doors

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

The system should also include a function to display details about each vehicle

And indicate when a vehicle is starting

PROGRAM:

```
class vehicle{  
    String brand;  
    int speed;  
  
    public vehicle(String brand,int speed){  
        this.brand=brand;
```



```

        this.speed=speed;
    }

    public static void main(String[] args) {

        car obj1=new car("ford",34,4);
        bike obj2=new bike("hero",100,true);
        truck obj3=new truck("tata",60,40);

    }
}

class car extends vehicle{

    int noofdoors;

    public car(String brand, int speed,int noofdoors) {
        super(brand, speed);
        this.noofdoors=noofdoors;
        System.out.println("Brand of car is:"+brand);
        System.out.println("Speed of car is:"+speed);
        System.out.println("no of doors of car:"+noofdoors);
    }
}

```

```
}
```

```
class bike extends vehicle{
```

```
    boolean gears;
```

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

```
        super(brand, speed);
```

```
        this.gears=gears;
```

```
        System.out.println("Brand of bike is:"+brand);
```

```
        System.out.println("Speed of bike is:"+speed);
```

```
        System.out.println("Gears of bike:"+gears);
```

```
    }
```

```
}
```

```
class truck extends vehicle{
```

```
    int weight;
```

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

```
        super(brand,speed);
```

```
        this.weight=weight;
```

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

```
        System.out.println("Speed of Truck is:"+speed);
```

```
        System.out.println("Weight of load is"+weight);
```

```
    }
```

}

OUTPUT:

```
PS C:\Users\manda> java -cp 'C:\Program Files\Java\jdk-9.0.4\bin\java.exe' .\Vehicle.class
Brand of car is:ford
Speed of car is:34
no of doors of car:4
Brand of bike is:hero
Speed of bike is:100
Gears of bike:true
Brand name is:tata
Speed of Truck is:60
Weight of load is40
PS C:\Users\manda>
```

NEGATIVE CASE:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:

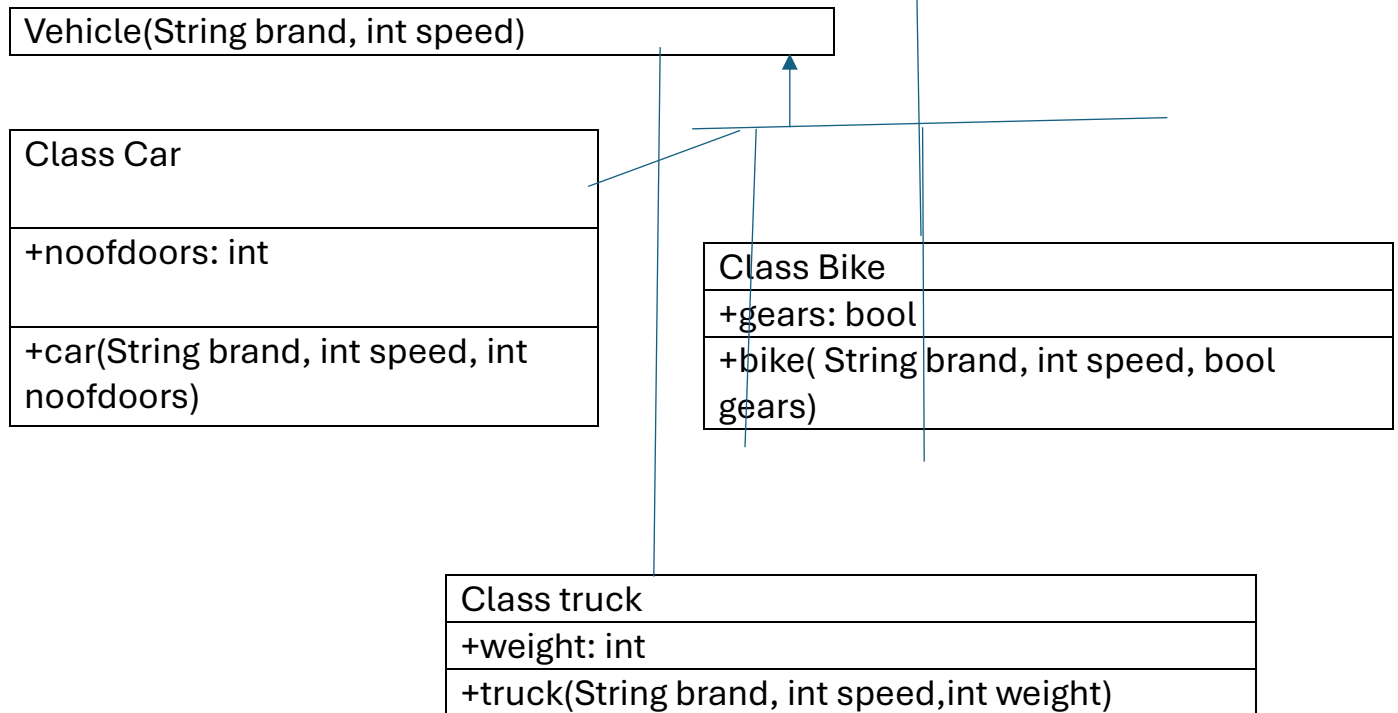
    at vehicle.main(Vehicle.java:10)
PS C:\Users\manda>
```

ERROR TABLE:

S.No	Error	Rectification
1.	Compilation error	Removed the 'public' from main class
2.	Syntax Error	Added parenthesis

CLASS DIAGRAM:

Class Vehicle
+ Brand: string
+speed: int



IMPORTANT POINTS:

1. Inheritance Hierarchy:

- The code demonstrates a simple inheritance hierarchy where car, bike, and truck classes all inherit from the base vehicle class.

2. Base Class (vehicle):

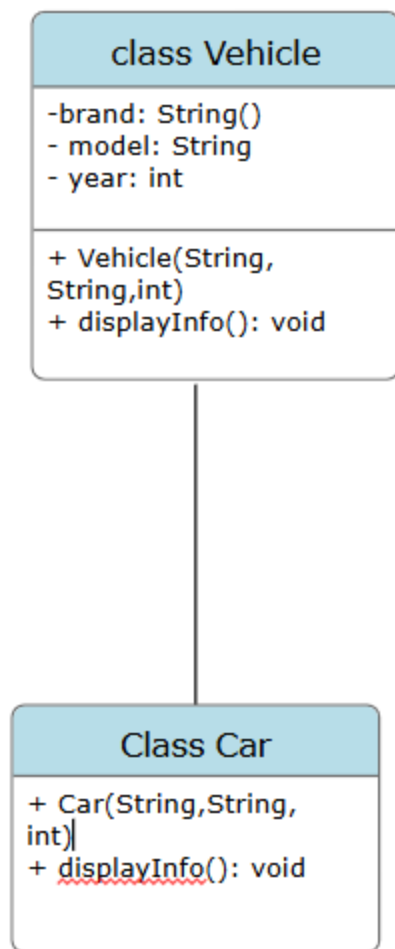
- Contains common properties for all vehicles: brand (String) and speed (int)
- Has a constructor that initializes these properties
- The main method is placed in the base class (unconventional but valid)

WEEK-6

AIM:

WRITE A JAVA PROGRAM TO CREATE A “VEHICLE” CLASS WITH A METHOD “displayInfo()”. OVERRIDE THIS METHOD IN THE SUBCLASS TO PROVIDE SPECIFIC INFO ABOUT THE CAR.

CLASS DIAGRAM:



PROGRAM:

```
public class Vehicle {  
    public String brand;  
    public String model;  
    public int year;
```

```
public Vehicle(String brand, String model, int year) {  
    this.brand = brand;  
    this.model = model;  
    this.year = year;  
}
```

```
public void displayInfo() {  
    System.out.println("Vehicle Information:");  
    System.out.println("Brand: " + brand);  
    System.out.println("Model: " + model);  
    System.out.println("Year of Manufacturing: " + year);  
}
```

```
public static void main(String[] args) {  
    System.out.println("Nishanth,24217,cse-c");  
    Vehicle vehicle = new Vehicle("Toyota", "Camry", 1990);  
    Car car = new Car("Mitsubishi", "Pajero", 1995);  
  
    vehicle.displayInfo();  
    car.displayInfo();  
}
```

```
class Car extends Vehicle {  
    public Car(String brand, String model, int year) {  
        super(brand, model, year);  
    }  
  
    public void displayInfo() {  
        System.out.println("Car Details:");  
        System.out.println("Brand: " + brand);  
        System.out.println("Model: " + model);  
        System.out.println("Year of Manufacturing: " + year);  
    }  
}
```

OUTPUT:

POSITIVE CASE:



```
ta\Local\Temp\vscodesws_75646\jdt_ws\jdt.ls-java-project\bin' 'Vehicle  
Nishanth,24217,cse-c  
Vehicle Information:  
Brand: Toyota  
Model: Camry  
Year of Manufacturing: 1990  
Car Details:  
Brand: Mitsubishi  
Model: Pajero  
Year of Manufacturing: 1995  
PS C:\Users\manda>
```


NEGATIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\manda\AppData\Local\Temp\vscodesws_e8a29\jdt_ws\jdt.ls-java-proje  
icle'  
Exception in thread "main" java.lang.Error: Unresolved compilation problem:  
    No enclosing instance of type Vehicle is accessible. Must qualify the allocation with an enclosing instance of type Vehicle (e.g. x.new A() where x is an instance of Vehicle).  
  
    at Vehicle.main(Vehicle.java:41)  
PS C:\Users\manda>
```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Compilation error	Added the public static to the subclass
2.	Syntax error	Added the parenthesis

IMPORTANT POINTS:

1. Encapsulation:

- The fields brand, model, and year are public, which is not ideal for encapsulation. Typically, fields should be private, and accessors (getters) and mutators (setters) should be used to access and modify them.

Method Overriding:

The Car class overrides the displayInfo() method from the Vehicle class.

Static Inner Class:

By making Car a static inner class, it can be instantiated directly using new Car without needing an instance of Vehicle.

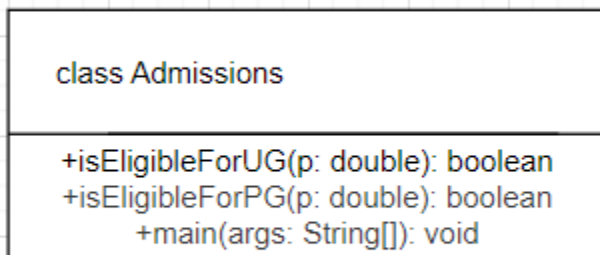
AIM:

A college is developing an automated admissions system that verifies student eligibility for ug and pg programs. Each program has different eligibility criteria based on the students percentages in their previous qualifications.

1. UG requires minimum 60%

2. PG requires minimum 70%

CLASS DIAGRAM:



PROGRAM:

```
public class Admissions {

    public boolean isEligibleForUG(double percentage) {
        return percentage >= 60.0;
    }

    public boolean isEligibleForPG(double percentage) {
```

```
        return percentage >= 70.0;
    }

    public static void main(String[] args) {
        Admissions admissions = new Admissions();

        double studentPercentageUG = 65.05;
        double studentPercentagePG = 72.45;

        if (admissions.isEligibleForUG(studentPercentageUG)) {
            System.out.println("Nishanth,24217,cse-c");
            System.out.println("Student is eligible for UG program.");
        } else {
            System.out.println("Student is not eligible for UG program.");
        }

        if (admissions.isEligibleForPG(studentPercentagePG)) {
            System.out.println("Student is eligible for PG program.");
        } else {
            System.out.println("Student is not eligible for PG program.");
        }
    }
}
```

OUTPUT:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' -Xms1G -Xmx1G -jar %cd%\ta\Local\Temp\vscodesws_75646\jdt_ws\jdt.ls-java-project\bin\jdt.ls-java-project.jar %cd%\Nishanth,24217,cse-c
Student is eligible for UG program.
Student is eligible for PG program.
PS C:\Users\manda>
```

NEGATIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsws_e8a29\jdt_ws\jdt.ls-java-project\bin' 'Admissions'
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
    The method println(String) is undefined for the type PrintStream
    Syntax error, insert "}" to complete ClassBody

    at Admissions.main(Admissions.java:25)
PS C:\Users\manda>
```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Syntax error	Added the bracket “}”
2.	Undefined PrintStream	“System.out.println”

IMPORTANT POINTS:

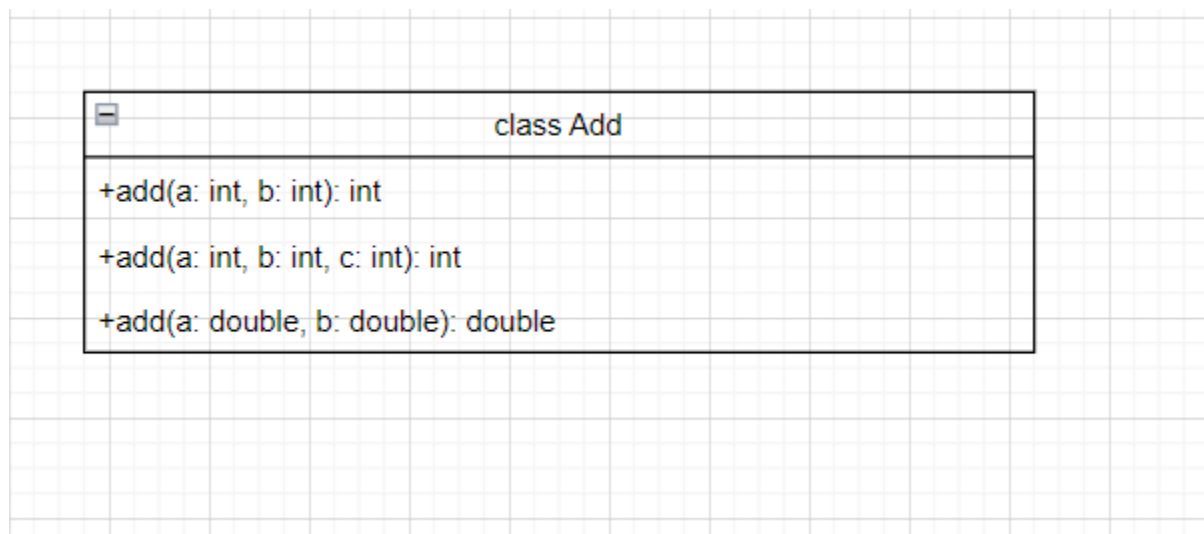
1. Class Definition: The Admissions class checks student eligibility for UG and PG programs.
2. UG Eligibility Method: isEligibleForUG returns true if the percentage is 60% or higher.
3. PG Eligibility Method: isEligibleForPG returns true if the percentage is 70% or higher.

AIM:

CREATE A CALCULATOR CLASS WITH OVERLOADED METHODS TO PERFORM ADDITION.

1. ADD TWO INTEGERS
2. ADD 2 DOUBLES
3. ADD 3 INTEGERS

CLASS DIAGRAM:



PROGRAM:

```
public class Add {  
    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 static void main(String[] args) {  
        Add calculator = new Add();  
  
        int sumIntegers = calculator.add(5, 10);  
        System.out.print("Nishanth,24217,cse-c\n");  
        System.out.println("Sum of two integers: " + sumIntegers);  
  
        double sumDoubles = calculator.add(5.5, 10.5);  
        System.out.println("Sum of two doubles: " + sumDoubles);  
  
        int sumThreeIntegers = calculator.add(5, 10, 15);  
        System.out.println("Sum of three integers: " + sumThreeIntegers);  
    }  
}
```

```
}  
  
}
```

OUTPUT:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java  
project\bin' 'Add'  
Nishanth,24217,cse-c  
Sum of two integers: 15  
Sum of two doubles: 16.0  
Sum of three integers: 30  
PS C:\Users\manda>
```

NEGATIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCode  
codesws_e8a29\jdt_ws\jdt.ls-java-project\bin' 'Add'  
Exception in thread "main" java.lang.Error: Unresolved compilation problems:  
    The method add(int, int) is undefined for the type Calculator  
    The method add(double, double) is undefined for the type Calculator  
    The method add(int, int, int) is undefined for the type Calculator  
  
    at Add.main(Add.java:15)  
PS C:\Users\manda>
```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Undefined methods	Replaced Calculator with add in the main method
2.	Syntax error	Added the semicolon
3.	Compilation error	Rectified the main method

IMPORTANT POINTS:

Method Overloading (Core Concept)

- Three add() methods with:
 - Different **number of parameters**: 2 and 3
 - Different **parameter types**: int and double

Addition Methods:

add(int a, int b): Adds two integers and returns the result.

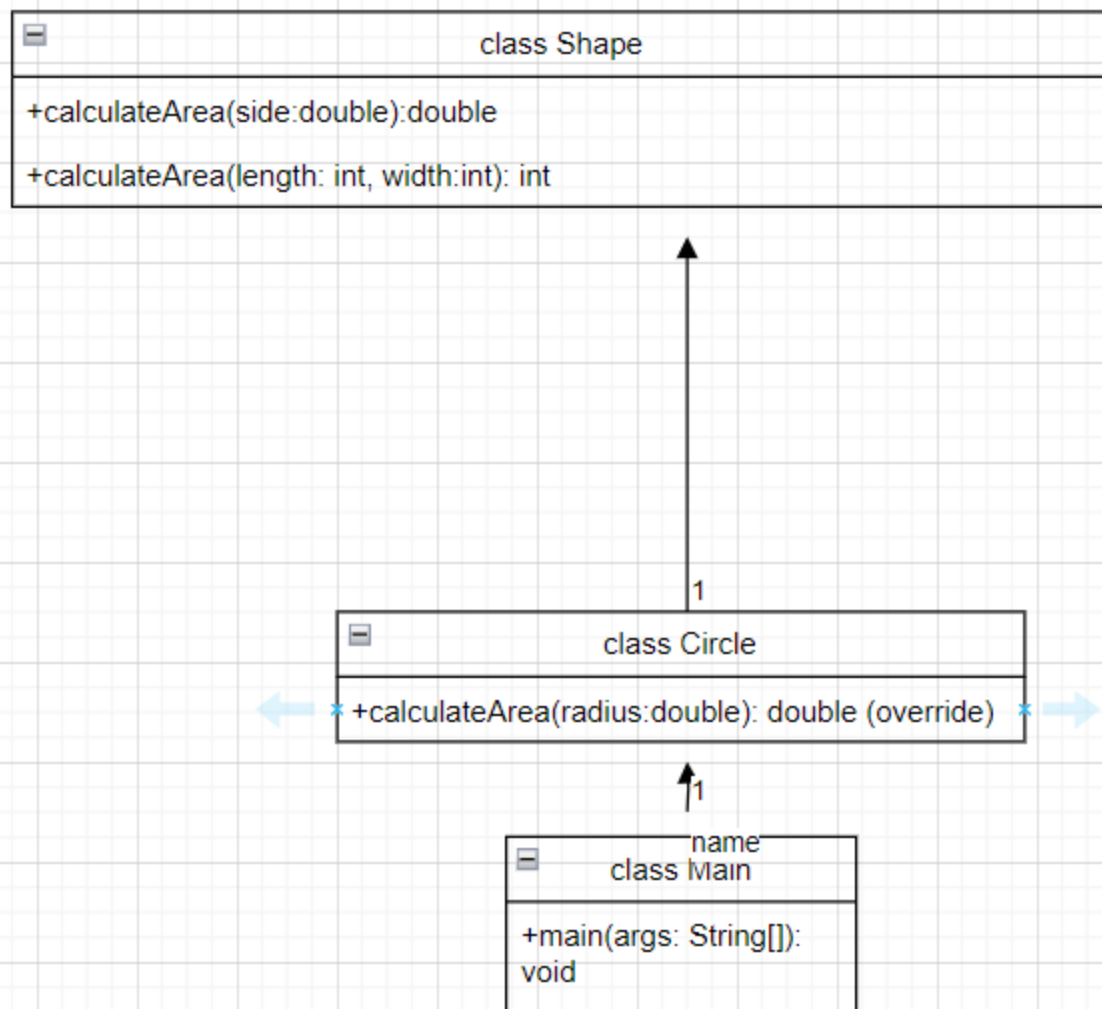
add(double a, double b): Adds two doubles and returns the result.

add(int a, int b, int c): Adds three integers and returns the result.

AIM:

CREATE A SHAPE CLASS WITH A METHOD “calculateArea()” THAT IS OVERLOADED FOR DIFFERENT SHAPES (E.G., SQUARE, CIRCLE, TRAINGLE,ETC). THEN CREATE A SUBCLASS “Circle” THAT OVERRIDES THE “calculateArea()” METHOD FOR A CIRCLE:

CLASS DIAGRAM:



PROGRAM:

```
public class Shape {  
    public double calculateArea(double side) {  
        return side * side;  
    }  
    public int calculateArea(int length, int width) {  
        return length * width;  
    }  
}
```

```
}  
}
```

```
class Circle extends Shape {  
    @Override  
    public double calculateArea(double radius) {  
        return 3.14 * radius * radius;  
    }  
}
```

```
class Main {  
    public static void main(String args[]) {  
        Circle obj1 = new Circle();  
        Shape obj2 = new Shape();  
  
        System.out.println("The area of a square is: " + obj2.calculateArea(2.0));  
        System.out.println("The area of a circle is: " + obj1.calculateArea(3.0));  
        System.out.println("The area of a rectangle is: " + obj2.calculateArea(3, 4));  
    }  
}
```

OUTPUT:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetails' '-Djava.class.path=Local\Temp\vscodesws_e8a29\jdt_ws\jdt.ls-java-project\bin' 'Main'
The area of a square is: 4.0
The area of a circle is: 28.259999999999998
The area of a rectangle is: 12
PS C:\Users\manda>

```

NEGATIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetails' '-Djava.class.path=Local\Temp\vscodesws_e8a29\jdt_ws\jdt.ls-java-project\bin' 'Main'
The area of a square is: 4.0
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error on token "*", Expression expected after this token

    at Circle.calculateArea(Shape.java:13)
    at Main.main(Shape.java:23)
PS C:\Users\manda>

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetails' '-Djava.class.path=Local\Temp\vscodesws_e8a29\jdt_ws\jdt.ls-java-project\bin' 'Main'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error, insert "}" to complete ClassBody

    at Main.main(Shape.java:25)
PS C:\Users\manda>

```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Incorrect formula	Corrected formula
2.	Syntax error	Inserted “}”

IMPORTANT POINTS:

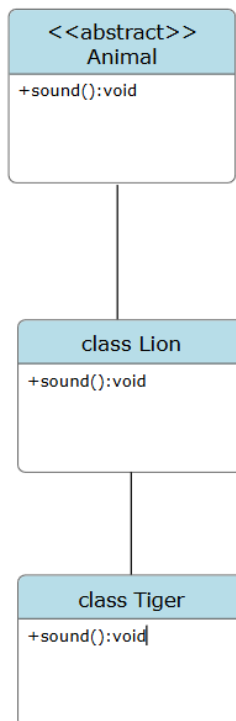
1. Class Shape demonstrates method overloading with calculateArea() for square and rectangle.
2. Class Circle extends Shape and overrides the calculateArea(double) method.
3. The overridden method in Circle calculates the area **of a** circle using πr^2

WEEK-7

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:



PROGRAM:

```
abstract class Animal {  
    abstract void sound();  
}
```

```
class Lion extends Animal {  
    @Override  
    void sound() {  
        System.out.println("A Lion roars!");  
    }  
}
```

```
class Tiger extends Animal {  
    @Override  
    void sound() {  
        System.out.println("A Tiger growls!");  
    }  
}
```

```
public class Sounds {  
    public static void main(String[] args) {  
        System.out.println("Nishanth,24217,cse-c");  
        Lion lion = new Lion();  
        lion.sound();  
    }  
}
```

```

    Tiger tiger = new Tiger();

    tiger.sound();

}

}

```

OUTPUT:

POSITIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\
codesws_75646\jdt_ws\jdt.ls-java-project\bin' 'Sound'
Nishanth,24217,cse-c
A Lion roars!
A Tiger growls!
PS C:\Users\manda>

```

NEGATIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCode
ta\Local\Temp\vscodesws_28956\jdt_ws\jdt.ls-java-project\bin' 'Main'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:

    at Main.main(Sounds.java:21)
PS C:\Users\manda> 

```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Compilation error	Named the main class with another name
2.	Syntax error	Corrected spelling

IMPORTANT POINTS:

1. Abstract Class:

- Animal is an abstract class that cannot be instantiated directly.
- It contains an abstract method sound() which must be implemented by all subclasses.

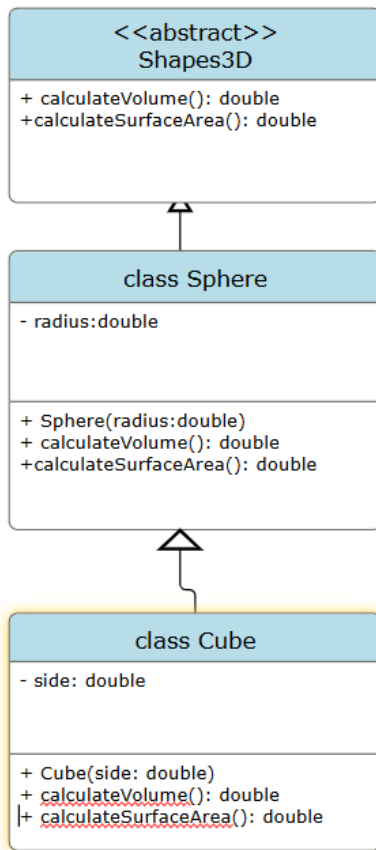
2. Method Overriding:

- Both Lion and Tiger override the sound() method of Animal to provide their own implementation

AIM:

WRITE A JAVA PROGRAM TO CREATE AN ABSTRACT CLASS “SHAPE3D” WITH ABSTRACT METHODS “calculateVolume()” AND “calculateSurfacArea()”. CREATE SUBCLASSES SPHERE AND CUBE THAT EXTEND THE SHAPE3D CLASS AND IMPLEMENT THE RESPECTIVE METHODS TO CALCULATE THE VOLUME AND SURAFEC AREA OF EACH SHAPE.

CLASS DIAGRAM:



PROGRAM:

```
abstract class Shapes3D {  
    public abstract double calculateVolume();  
    public abstract double calculateSurfaceArea();  
}
```

```
class Sphere extends Shapes3D {  
    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 * Math.PI * Math.pow(radius, 2);
    }
}

class Cube extends Shapes3D {
    private double side;

    public Cube(double side) {
        this.side = side;
    }

    public double calculateVolume() {
        return Math.pow(side, 3);
    }
}
```

```
public double calculateSurfaceArea() {  
    return 6 * Math.pow(side, 2);  
}  
}
```

```
public class Shapes3DImplementation {  
    public static void main(String[] args) {  
        Sphere sphere = new Sphere(5);  
        System.out.print("Nishanth,24217,cse-c\n");  
        System.out.println("Sphere Volume: " + sphere.calculateVolume());  
        System.out.println("Sphere Surface Area: " +  
            sphere.calculateSurfaceArea());  
  
        Cube cube = new Cube(3);  
        System.out.println("Cube Volume: " + cube.calculateVolume());  
        System.out.println("Cube Surface Area: " + cube.calculateSurfaceArea());  
    }  
}
```

OUTPUT:

POSITIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe'
project\bin' 'Shapes3DImplementation'
Nishanth,24217,cse-c
Sphere Volume: 523.5987755982989
Sphere Surface Area: 314.1592653589793
Cube Volume: 27.0
Cube Surface Area: 54.0
PS C:\Users\manda>
```

NEGATIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java
project\bin' 'Shapes'
Error: Unable to initialize main class Shapes
Caused by: java.lang.NoClassDefFoundError: [LSttring;
PS C:\Users\manda> █
```

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCode
project\bin' 'Shapes'
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
    Syntax error, insert "}" to complete ClassBody

    at Shapes.main(Shapes.java:47)
PS C:\Users\manda> █
```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Unable to initialize main class Shapes	Spelling error in "Sttring"
2.	Syntax error	Inserted "}"

IMPORTANT POINTS:

1. Demonstrates abstraction with an abstract class Shapes3D, and method overriding in Sphere and Cube to provide specific implementations for volume and surface area.
2. Shows constructor usage, encapsulation with private fields, and **polymorphic design potential** for 3D shapes.

AIM:

WRITE A JAVA PROGRAM USING AN ABSTRACT CLASS TO DEFINE A METHOD FOR PATTERN PRINTING.

CRAETE AN ABSTRACT CLASS NAMED “PatternPrinter” WITH AN ABSTRACT METHOD “Printpattern(int)” AND A CONCRETE METHOD TO DISPLAY THE PATTERN TITLE.

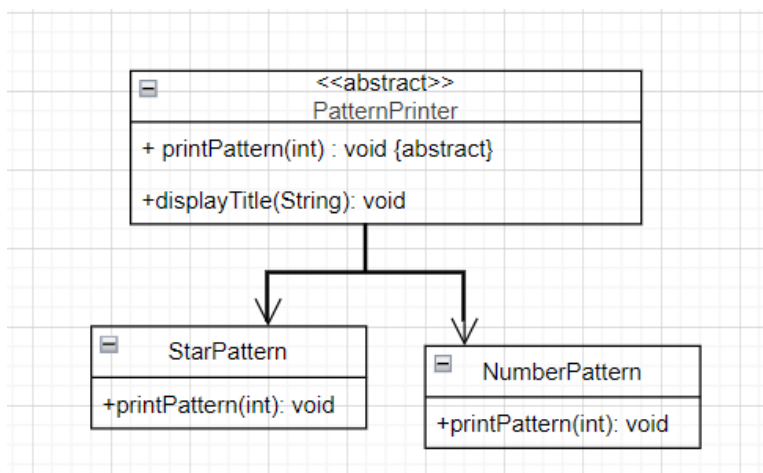
IMPLEMENT 2 SUBCLASSES,

1.STAR PATTERN PRINTS A RIGHT ANGLED TRIANGLE OF STARS

2.NUMBER PATTERN PRINTS A RIGHT ANGLED TRAINGLE OF INCREASING NUMBERS.

IN THE MAIN METHOD, CREATE OBJECTS OF BOTH SUBCLASSES AND PRINT PATTERN FOR A GIVEN NUMBER OF ROWS.

CLASS DIAGRAM:



PROGRAM:

```
abstract class PatternPrinter {  
    public abstract void printPattern(int rows);  
}
```

```
public void displayTitle(String title) {  
    System.out.println(title);  
}  
}
```

```
class StarPattern extends PatternPrinter {  
    @Override  
    public void printPattern(int rows) {  
        for (int i = 1; i <= rows; i++) {  
            for (int j = 1; j <= i; j++) {  
                System.out.print("* ");  
            }  
            System.out.println();  
        }  
    }  
}
```

```
class NumberPattern extends PatternPrinter {  
    public void printPattern(int rows) {  
        int number = 1;  
        for (int i = 1; i <= rows; i++) {  
            for (int j = 1; j <= i; j++) {  
                System.out.print(number + " ");  
            }  
        }  
    }  
}
```

```
        number++;  
    }  
    System.out.println();  
}  
}  
}
```

```
public class PatternPrint {  
    public static void main(String[] args) {  
        PatternPrinter starPattern = new StarPattern();  
        starPattern.displayTitle("Star Pattern:");  
        starPattern.printPattern(5);  
  
        PatternPrinter numberPattern = new NumberPattern();  
        numberPattern.displayTitle("Number Pattern:");  
        numberPattern.printPattern(5);  
    }  
}
```

OUTPUT:

POSITIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInInt
ternPrint'
Star Pattern:
*
* *
* * *
* * * *
* * * * *

Number Pattern:
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
PS C:\Users\manda>
```

NEGATIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' '-XX:+ShowCodeDetailsInInt
ternPrint'
Error: Main method not found in class PatternPrinter, please define the main method as:
    public static void main(String[] args)
or a JavaFX application class must extend javafx.application.Application
PS C:\Users\manda>
```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Main method not found in class PatternPrinter	Named the class with main function as PatternPrint

IMPORTANT POINTS:

Class StarPattern:

- This class extends PatternPrinter and provides an implementation for the printPattern method.

Class NumberPattern:

- This class also extends PatternPrinter and implements the printPattern method.

Abstract Class PatternPrinter:

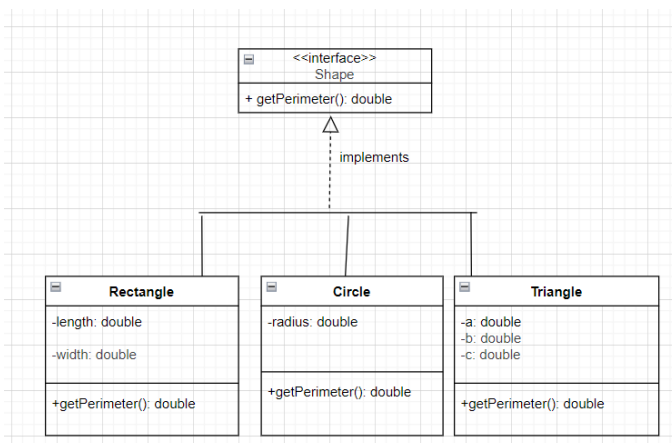
- This is an abstract class that defines a blueprint for pattern printing.
- It contains an abstract method `printPattern(int rows)` which must be implemented by any subclass.

WEEK-8

AIM:

WRITE A JAVA PROGRAM TO CREATE AN INTERFACE “Shape” WITH THE “getPerimeter()” METHOD. CREATE 3 CLASSES “Rectangle”, “Circle” and “Triangle” THAT IMPLEMENTS THE SHAPE INTERFACE. IMPLEMENT THE “getPerimter()” METHOD FOR EACH OF 3 CLASSES.

CLASS DIAGRAM:



PROGRAM:

```
interface Shape {  
    double getPerimeter();  
}
```

```
class Rectangle implements Shape {  
    private double length;  
    private double width;
```

```
public Rectangle(double length, double width) {  
    this.length = length;  
    this.width = width;  
}  
  
public double getPerimeter() {  
    return 2 * (length + width);  
}  
}
```

```
class Circle implements Shape {  
    private double radius;  
  
    public Circle(double radius) {  
        this.radius = radius;  
    }  
  
    public double getPerimeter() {  
        return 2 * Math.PI * radius;  
    }  
}
```

```
class Triangle implements Shape {  
    private double a;  
    private double b;  
    private double c;  
  
    public Triangle(double a, double b, double c) {  
        this.a = a;  
        this.b = b;  
        this.c = c;  
    }  
  
    public double getPerimeter() {  
        return a + b + c;  
    }  
}
```

```
public class ShapeTEST {  
    public static void main(String[] args) {  
        System.out.println("Nishanth,av.sc.u4cse24217,cse-c");  
        Shape rectangle = new Rectangle(2, 3);  
        System.out.println("Perimeter of rectangle: " + rectangle.getPerimeter());  
  
        Shape circle = new Circle(2);
```

```

        System.out.println("Perimeter of circle: " + circle.getPerimeter());

        Shape triangle = new Triangle(3, 4, 5);

        System.out.println("Perimeter of triangle: " + triangle.getPerimeter());
    }
}

```

OUTPUT:

POSITIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-
roject\bin' 'ShapeTEST'
Nishanth,av.sc.u4cse24217,cse-c
Perimeter of rectangle: 10.0
Perimeter of circle: 12.566370614359172
Perimeter of triangle: 12.0
PS C:\Users\manda>

```

NEGATIVE CASE:

```

PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe' -XX:+ShowCodeDetails
roject\bin' 'ShapeTEST'
Exception in thread "main" java.lang.Error: Unresolved compilation problems:
    Shape cannot be resolved to a type
    Shape cannot be resolved to a type
    Shape cannot be resolved to a type

    at ShapeTEST.main(ShapeTEST.java:53)
PS C:\Users\manda>

```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
------	-------	---------------

1.	Exception in thread "main" java.lang.Error	Corrected the syntax error from "interfaces" to interface
-----------	--	--

IMPORTANT POINTS:

1. Interface Shape:

- Defines a contract with a single method `getPerimeter()`.
- Any class implementing this interface must provide an implementation for the `getPerimeter()` method.

Class ShapeTEST:

- Contains the main method, serving as the entry point for the program.
- Prints a header with a name and identifier.
- Creates instances of Rectangle, Circle, and Triangle using the Shape interface reference.
- Demonstrates polymorphism by using the Shape interface to reference different shape objects (Rectangle, Circle, Triangle).
- Allows for calling the `getPerimeter()` method on these objects through the interface reference.

AIM:

WRITE A 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 RESPECTIVE SPORTS.

CLASS DIAGRAM:

PROGRAM:

```
interface Playable {  
    void play();  
}
```

```
class Football implements Playable {  
    public void play() {  
        System.out.println("I play Football.");  
    }  
}
```

```
class Volleyball implements Playable {  
    public void play() {  
        System.out.println("I play Volleyball.");  
    }  
}
```

```
class Basketball implements Playable {  
    public void play() {  
        System.out.println("I play Basketball.");  
    }  
}
```

```
}  
}  
  
class PlayableMain {  
    public static void main(String[] args) {  
        System.out.println("Nishanth, av.sc.u4cse24217, cse-c");  
        Playable p1 = new Football();  
        Playable p2 = new Volleyball();  
        Playable p3 = new Basketball();  
        p1.play();  
        p2.play();  
        p3.play();  
    }  
}
```

OUTPUT:

POSITIVE CASE:

```
PS C:\Users\manda> & 'C:\Program Files\Java\jdk-21\bin\java.exe'  
yableMain'  
Nishanth, av.sc.u4cse24217, cse-c  
I play Football.  
I play Volleyball.  
I play Basketball.  
PS C:\Users\manda>
```

NEGATIVE CASE:


```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:  
  Syntax error, insert "]" to complete ClassBody  
  
    at PlayableMain.main(PlayableMain.java:32)  
PS C:\Users\manda>
```

ERROR TABLE:

S.NO	ERROR	RECTIFICATION
1.	Syntax error	Inserted “”

IMPORTANT POINTS:

- 1.The Playable interface type is used as a reference (Playable p1, p2, p3) to objects of different implementing classes.
- 2.This demonstrates runtime polymorphism — the method that gets called is determined by the actual object type.
3. New instances of Football, Volleyball, and Basketball are created and assigned to interface-type references.