**Aim:-** To implement the concept of Multiple Inheritance.

**Objective :-** Implement multiple inheritance in a program to perform addition, multiplication and transpose operations on a matrix. Create an interface to hold prototypes of these methods and create a class input to read input. Inherit a new class from this interface and class. In main class create object of this child class and invoke required methods.

**Theory:-**

* In Multiple inheritance, one class can have more than one superclass and inherit features from all parent classes. Java does not support [multiple inheritance](https://www.geeksforgeeks.org/java-and-multiple-inheritance/) with classes. In java, we can achieve multiple inheritance only through [Interfaces](http://quiz.geeksforgeeks.org/interfaces-in-java/).
* An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.
* However, Java supports multiple interface inheritance where an interface extends more than one super interfaces.
* A class implements an interface, but one interface extends another interface.Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.
* The following is the syntax used to extend multiple interfaces in Java:

access\_specifier interface subinterfaceName extends superinterface1, superinterface2, …… {

// Body  
}

**Code:-**

interface DSEClass

{

public void comp();

}

class Student

{

void aids()

{

System.out.println(" aids branch total:10 student");

}

}

class StdentDetails extends Student implements DSEClass

{

public void comp()

{

System.out.println("comp branch total:25 student");

}

public static void main(String args[])

{

StdentDetails s= new StdentDetails();

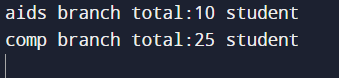
s.aids();

s.comp();

}

}

**Output:**

****

**Conclusion:-**

In conclusion, multiple inheritance is a powerful feature that offers flexibility and code reuse in object-oriented programming. However, it comes with certain complexities and challenges, particularly the diamond problem and potential for increased code maintenance. The choice to use multiple inheritance should be made carefully, considering the specific needs of the software project and the potential design trade-offs.