Java and Multiple Inheritance

Multiple Inheritance is a feature of object oriented concept, where a class can inherit properties of more than one parent class. The problem occurs when there exist methods with same signature in both the super classes and subclass. On calling the method, the compiler cannot determine which class method to be called and even on calling which class method gets the priority.

Why Java doesn't support Multiple Inheritance?

Consider the below Java code. It shows error.

```
// First Parent class
class Parent1
{
    void fun()
    {
        System.out.println("Parent1");
    }
}

// Second Parent Class
class Parent2
{
    void fun()
    {
        System.out.println("Parent2");
    }
}

// Error : Test is inheriting from multiple
// classes
class Test extends Parent1, Parent2
{
   public static void main(String args[])
   {
       Test t = new Test();
       t.fun();
   }
}
```

Output :

```
Compiler Error
```

From the code, we see that, on calling the method fun() using Test object will cause complications such as whether to call Parent1's fun() or Parent2's fun() method.

1. The Diamond Problem:

```
          GrandParent
           /     \
          /       \
      Parent1      Parent2
          \       /
           \     /
             Test
```

```
// A Grand parent class in diamond
class GrandParent
{
    void fun()
    {
        System.out.println("Grandparent");
    }
}

// First Parent class
class Parent1 extends GrandParent
{
    void fun()
    {
        System.out.println("Parent1");
    }
}

// Second Parent Class
class Parent2 extends GrandParent
{
    void fun()
    {
        System.out.println("Parent2");
    }
}


// Error : Test is inheriting from multiple
// classes
class Test extends Parent1, Parent2
{
   public static void main(String args[])
   {
       Test t = new Test();
       t.fun();
   }
}
```

Run on IDE

From the code, we see that: On calling the method fun() using Test object will cause complications such as whether to call Parent1's fun() or Beta's fun() method.

Therefore, in order to avoid such complications Java does not support multiple inheritance of classes.

2. Simplicity – Multiple inheritance is not supported by Java using classes , handling the complexity that causes due to multiple inheritance is very complex. It creates problem during various operations like casting, constructor chaining etc and the above all reason is that there are very few scenarios on which we actually need multiple inheritance, so better to omit it for keeping the things simple and straightforward.

How are above problems handled for Default Methods and Interfaces ?

Java 8 supports default methods where interfaces can provide default implementation of methods. And a class can implement two or more interfaces. In case both the implemented interfaces contain default methods with same method signature, the implementing class should explicitly specify which default method is to be used or it should override the default method.

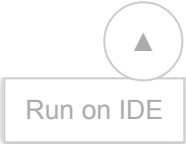
```
// A simple Java program to demonstrate multiple
// inheritance through default methods.
interface PI1
{
    // default method
    default void show()
    {
        System.out.println("Default PI1");
    }
}

interface PI2
{
    // Default method
    default void show()
    {
        System.out.println("Default PI2");
    }
}

// Implementation class code
class TestClass implements PI1, PI2
{
    // Overriding default show method
    public void show()
    {
        // use super keyword to call the show
        // method of PI1 interface
        PI1.super.show();

        // use super keyword to call the show
        // method of PI2 interface
        PI2.super.show();
    }

    public static void main(String args[])
    {
        TestClass d = new TestClass();
        d.show();
    }
}
```



Output:

```
Default PI1
Default PI2
```

If we remove implementation of default method from “TestClass”, we get compiler error. See this for a sample run.

If there is a diamond through interfaces, then there is no issue if none of the middle interfaces provide implementation of root interface. If they provide implementation, then implementation can be accessed as above using super keyword.

```
// A simple Java program to demonstrate how diamond
// problem is handled in case of default methods

interface GPI
{
    // default method
    default void show()
    {
        System.out.println("Default GPI");
    }
}

interface PI1 extends GPI { }

interface PI2 extends GPI { }

// Implementation class code
class TestClass implements PI1, PI2
{
    public static void main(String args[])
    {
        TestClass d = new TestClass();
        d.show();
    }
}
```

Run on IDE

Output:

```
Default GPI
```

Also See – http://qa.geeksforgeeks.org/510/why-java-doesnt-support-multiple-inheritance

This article is contributed by **Vishal S**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.