

Copilot

Java does not support multiple inheritance with classes directly to avoid complexity and ambiguity, such as the "diamond problem" where a class inherits from two classes that have a method with the same signature [1] [2]. However, Java achieves multiple inheritance through interfaces.

Multiple Inheritance with Interfaces

In Java, a class can implement multiple interfaces, allowing it to inherit the abstract methods from multiple sources. Here's a simple example:

```
interface Interface1 {  
    void method1();  
}  
  
interface Interface2 {  
    void method2();  
}  
  
class MyClass implements Interface1, Interface2 {  
    public void method1() {  
        System.out.println("Method1 from Interface1");  
    }  
  
    public void method2() {  
        System.out.println("Method2 from Interface2");  
    }  
  
    public static void main(String[] args) {  
        MyClass obj = new MyClass();  
        obj.method1();  
        obj.method2();  
    }  
}
```

In this example, MyClass implements both Interface1 and Interface2, thus inheriting methods from both interfaces [2] [3].

Default Methods in Interfaces

Java 8 introduced default methods in interfaces, which provide a way to include method implementations in interfaces. If a class implements multiple interfaces that have default

methods with the same signature, the class must override the conflicting method to resolve the ambiguity.

```
interface Interface1 {  
    default void method() {  
        System.out.println("Default Method from Interface1");  
    }  
}
```

```
interface Interface2 {  
    default void method() {  
        System.out.println("Default Method from Interface2");  
    }  
}
```

```
class MyClass implements Interface1, Interface2 {  
    @Override  
    public void method() {  
        Interface1.super.method();  
        Interface2.super.method();  
    }  
  
    public static void main(String[] args) {  
        MyClass obj = new MyClass();  
        obj.method();  
    }  
}
```

In this example, MyClass resolves the conflict by explicitly calling the default methods from both interfaces [1] [2].

If you have any specific questions or need further examples, feel free to ask!

Source: Conversation with Copilot, 9/30/2024

1. [Java and Multiple Inheritance - GeeksforGeeks](#)
2. [Multiple Inheritance in Java - Javatpoint](#)
3. [Does Java support Multiple inheritance? - BeginnersBook](#)
4. [Java Inheritance Tutorial: Explained with examples - Educative](#)
5. [Inheritance in Java - Javatpoint](#)