Java dilinde çoklu kalıtım engellenmiştir. Birden fazla interfaceden kalıtım alabilmemize rağmen birden fazla sınıftan kalıtım alamamızın sebebi temelde karmaşıklığı önlemektir. En basit şekilde anlatacak olursak bir C sınıfı düşünelim ve bu sınıf A ve B sınıflarını extends ediyor diyelim. A ve B sınıflarında display() metodu var ise java hangi sınıfın metodunun kullanılacağı konusundaki belirsizliği ortadan kaldırmış oluyor.

```
import java.io.*;
class A {
	void display() {
		System.out.println(“A”);
	}
}
class B {
	void display() {
		System.out.println(“B”);
	}
}
class C extends A, B {
	public static void main(String args[]) {
		C c = new C();
		c.display();
	}
}
```

C sınıfı tanımlanırken extends A'dan sonra { gerekli olduğu yönünde bir hata vererek java bize çoklu kalıtım yapamadığının uyarısını vermiş olur.

Bununla birlikte çoklu kalıtımın desteklendiği diller de vardır. Bunlar C++, Scala, Python, Perl gibi popüler dillerinde aralarından olduğu birçok dil mevcut. Burada kalıtımın uygunlandığı sınıf ve extends edilen sınıflardan başlayarak sırasıyla instance oluşur. Sonrasında ortak bir metod var ise burada işlem dilden dile farklılık gösterebilir. C++ ortak metodların hiçbirinin çalışmasına izin vermezken, Python extends edilme sırasına göre gezerek ilk bulduğu metodu çalıştırarak işleme devam eder. Burada java karışıklığı önleyerek güzel bir iş başarmış diyebiliriz. Java içerisinde interfaceler kullanarak birden fazla interface'i bir sınıf altında implemente edebiliriz.