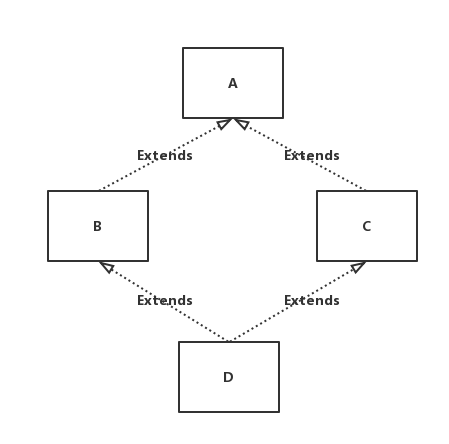
**JAVA中为什么不允许多重继承**

1. **方块问题**

多继承的菱形继承问题，也被称之为致命方块问题（diamond problem）。



假设类 A、B、C 内的方法都是 public 的，以方便讨论。

类 B 和类 C 继承自类 A，且都重写了类 A 中的同一个方法，而类 D 同时继承了类 B 和类 C，那么此时类 D 会继承 B、C 的方法，那对于 B、C 重写的 A 中的方法，类 D 会继承哪一个呢？

**多重继承会导致有歧义的情况存在。**

1. **设计时为了简单和安全性**

“JAVA omits many rarely used, poorly understood, confusing features of C++ that in our experience bring more grief than beneﬁt. This primarily consists of operator overloading (although it does have method overloading), multiple inheritance, and extensive automatic coercions.”——James Gosling

Java 去除了一些 C++ 中很少用、而且被经常误解而错用的功能，如操作符的重载(尽管 Java 仍旧保留方法的重载)，多重继承，以及广泛的自动强迫同型(extensive automatic coercions)。

C++被设计出来后，就会经常掉入多继承这个陷阱，虽然它也提出了相应的解决办法，但Java语言本着简单的原则舍弃了C++中的多继承，这样也会使程序更具安全性。

（C++语言是1983年由贝尔实验室的Bjarne Stroustrup在C语言的基础上推出的，Java语言是1995年由James Gosling和同事共同正式推出的）

1. **补充：为什么Java类可以集成多个接口?**

Java接口是行为性的，接口中的方法，是抽象的（从JDK1.8之后，接口中允许给出一些默认方法的实现，这里不考虑这个）。也就是说它只是定义某个行为的名称，而**具体的行为的实现是集成接口的类实现的**，因此就算两个接口中定义了两个名称完全相同的方法，当**某个类去集成这两个接口时，类中也只会有一个相应的方法**，这个方法的具体实现是这个类来进行编写的，所以并不会出现结构混乱的情况，即并不会出现二义性的问题。

1. **总结**
2. Java不支持多继承是由Java的定义决定的，Java最重要的定义，就是因为它是一种简单的面向对象解释型的语言。  
   　　2. Java不用多重继承是因为这种方法很少被使用，即使要使用也可以通过接口来实现多重继承问题。  
   　　3. Java的定义：  
   　　1） 因为Java: 一种简单的，面向对象的，分布式的，解释型的（译者注：Java既不是纯解释型也不是纯编译型的语言），健壮的，安全的，架构中立的，可移植的，高性能的，支持多线程的，动态语言。  
   　　2） 假设可以多重继承：  
   　　有两个类B和C继承自A；假设B和C都继承了A的方法并且进行了覆盖，编写了自己的实现；假设D通过多重继承继承了B和C，那么D应该继承B和C的重载方法，那么它应该继承的是B的还是C的？这就陷入了矛盾，所以Java不允许多重继承。