澄清Java语言接口与继承的本质 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022__E6_BE_84_E6_B8_85Java_c97_138801.htm 大多数人认为，接口的意义在于顶替多重继承。众所周知Java没有c 那样多重继承的机制，但是却能够实作多个接口。其实这样做是很牵强的，接口和继承是完全不同的东西，接口没有能力代替多重继承，也没有这个义务。接口的作用，一言以蔽之，就是标志类的类别（type of class）。把不同类型的类归于不同的接口，可以更好的管理他们。OO的精髓，我以为，是对对象的抽象，最能体现这一点的就是接口。为什么我们讨论设计模式都只针对具备了抽象能力的语言（比如c 、java、c#等），就是因为设计模式所研究的，实际上就是如何合理的去抽象。（cowboy的名言是“ 抽象就是抽去像的部分 ”，看似调侃，实乃至理）。 设计模式中最基础的是工厂模式（Factory），在我最近的一个很简单的应用中，我想尽量的让我的程序能够在多个数据库间移植，当然，这涉及很多问题，单是如何兼容不同DBMS的SQL就让人头痛。我们不妨先把问题简单化，只考虑如何连接不同的数据库。 假设我有很多个类，分别是Mysql.java、SQLServer.java、Oracle.java、DB2.java，他们分别连接不同的数据库，统一返回一个Connection对象，并且都有一个close方法，用于关闭连接。只需要针对你的DBMS，选择不同的类，就可以用了，但是我的用户他会使用什么数据库？我不知道，我希望的是尽量少的修改代码，就能满足他的需要。我可以抽象如下接口： package org.bromon.test. public interface DB { java.sql.Connection openDB(String url,String

user,String password). void close(). } 这个接口只定义两个方法，没有任何有实际意义的代码，具体的代码由实作这个接口的类来给出，比如Mysql.java： Package org.bromon.test. import java.sql.*. public class Mysql implements DB { private String url= ”jdbc:mysql:localhost:3306/test ”. private String user= ”root ”. private String password= ” ”. private Connection conn. public Connection openDB(url,user,password) { //连接数据库的代码 } public void close() { //关闭数据库 } } 类似的当然还有Oracle.java等等，接口DB给这些类归了个类，在应用程序中我们这样定义对象： org.bromon.test.DB myDB. 使用myDB来操作数据库，就可以不用管实际上我所使用的是哪个类，这就是所谓的“ 开-闭 ”原则。但是问题在于接口是不能实例化的，myDB=new DB()，这样的代码是绝对错误的，我们只能myDB=new Mysql()或者myDB=new Oracle()。麻烦了，我还是需要指定具体实例化的是哪个类，用了接口跟没用一样。所以我们需要一个工厂： package org.bromon.test. public class DBFactory { public static DB Connection getConn() { Return(new Mysql()). } } 所以实例化的代码变成：myDB=DBFactory.getConn()； 这就是23种模式中最基础的普通工厂(Factory)，工厂类负责具体实例化哪个类，而其他的程序逻辑都是针对DB这个接口进行操作，这就是“ 针对接口编程 ”。责任都被推卸给工厂类了，当然你也可以继续定义工厂接口，继续把责任上抛，这就演变成抽象工厂(Abstract Factory)。 整个过程中接口不负责任何具体操作，其他的程序要连接数据库的话，只需要构造一个DB对象就OK，而不管工厂类如何变化。这就是接口的意义----抽象。 继承的概念

不用多说，很好理解。为什么要继承呢？因为你想重用代码？这绝对不是理由，继承的意义也在于抽象，而不是代码重用。如果对象A有一个run()方法，对象B也想有这个方法，所以有人就Class B extends A。这是不经大脑的做法。如果在B中实例化一个A，调用A的Run()方法，是不是可以达到同样的目的？如下： Class B { A a=new A(). a.run(). } 这就是利用类的聚合来重用代码，是委派模式的雏形，是GoF一贯倡导的做法。 那么继承的意义何在？其实这是历史原因造成的，最开始的OO语言只有继承，没有接口，所以只能以继承来实现抽象，请一定注意，继承的本意在于抽象，而非代码重用（虽然继承也有这个作用），这是很多Java烂书最严重的错误之一，它们所造成的阴影，我至今还没有完全摆脱，坏书害人啊，尤其是入门类的，流毒太大。什么时候应该使用继承？只在抽象类中使用，其他情况下尽量不使用。抽象类也是不能实例化的，它仅仅提供一个模版而已，这就很能说明问题。软件开发的万恶之源，一是重复代码而不是重用代码，二是烂用继承，尤以c 程序员为甚。Java中取缔多重继承，目的就是制止烂用继承，实是非常明智的做法，不过很多人都不理解。Java能够更好的体现设计，这是让我入迷的原因之一。