空对象模式

一、空对象模式介绍

1、定义

空对象模式（Null Object Pattern）不属于GoF设计模式，但是它作为一种经常出现的模式足以被视为设计模式了。其具体定义为设计一个空对象取代NULL对象实例的检查。NULL对象不是检查控制，而是反映一个不做任何动作的关系。这样的NULL对象也可以在数据不可用的时候提供默认的行为，属于行为型设计模式。

2、空对象模式的应用场景

> 在Java编程中，经常会遇到空指针导致异常的错误信息，以及要经常判断是否空对象的情况，既使得程序不友好，也增加了程序的复杂性。利用空对象模式能很解决这些问题。

在Java编程的代码编写中。在向对象发送一个消息(也就是应用这个对象)之前，一般要检查对象是否存在，这样的检查的场景很容易出现多次。比如在电信业务受理系统中。可能会向一个业务受理对象索求它所相关的套餐对象.然后再验证这个套餐对象是否为null;

如果这个套餐对象存在，才能调用它的资费方法来设置这个受理业务的资费，在很多地方都是这样做的.造成很多重复的代码，空对象可以避免这样的情况。空对象的另一个应用场景是列表中的空对象和正常对象处理方式一样。减少了空对象并且简化了处理。使用空对象带来的好处是系统从来不会因为空对象而被破坏。由于空对象对所有外界请求的响应都和真实对象一样。所以系统行为总是正常的。空对象还有一个特点就是一定是常量，它们的任何属性都不会发生变化。因此可以用单例模式来实现它们。

空对象模式适用于以下应用场景。

（1）对象实例需要一个协作实例。空对象模式不会引入协作实例，它只是使用现有的协作实例。

（2）部分协作实例不需要做任何处理。

（3）从客户端中将对象实例不存在的代码逻辑抽象出来。

二、实现

1、空对象模式的UML类图

（1）抽象对象（AbstractCustomer）：定义所有子类公有的行为和属性。

（2）真实对象（RealCustomer）：继承AbstractCustomer类，并实现所有行为。

（3）空对象（NullCustomer）：继承AbstractCustomer类，对父类方法和属性不做实现和赋值。

2、空对象模式的写法

在空对象模式中，我们创建一个指定各种要执行的操作的**抽象类**和扩展该类的实体类，还创建一个未对该类做任何实现的空对象类，该空对象类将无缝地使用在需要检查空值的地方。

第一步、创建一个顾客抽象类

第二步、创建顾客实现类

第三步、创建顾客工厂，之创建三个对象

创建客户端测试

结果报错，因为工厂并没有Bob对象，所以返回了null，然后再用null调用getName方法，就会报空指针异常，只能通过if判断一下对象是否为空，在来下面操作，这样的话就很麻烦，如果对象太多，则每个对象都要判断是否

为空

下面我们创建一个空对象，让它也能和正常对象一样被使用

修改一下工厂

我们再来运行一下，结果并没有报空指针异常，空对象被当作正常对象一样处理了

3、空对象模式的优点

（1）它可以加强系统的稳固性，能有效地减少空指针报错对整个系统的影响，使系统更加稳定。

（2）它能够实现对空对象情况的定制化的控制，掌握处理空对象的主动权。

（3）它并不依靠Client来保证整个系统的稳定运行。

（4）它通过定义isNull()对使用条件语句==null的替换，显得更加易懂。

4、空对象模式的缺点

每一个要返回的真实的实体都要建立一个对应的空对象模型，那样会增加类的数量。

三、总结

空对象模式是一个很简单实用的模式，它可以使代码更清晰和简洁，提升代码的可读性，但是我们也不要滥用它。一般，一个对象在很多地方都被使用到，都需要对其进行空值判断，代码因此产生很多空值判断的逻辑，我们就可以根据实际场景考虑使用空对象模式了。