开放封闭原则 孙悟空任弼马温一职

应用场景举例：

孙悟空从东海龙宫拿到定海神针如意金箍棒后回到花果山，和自己的部下过着自由自在的生活。那只好景不长，因为他在地狱删除了自己和花果山所有猴子的名单，同时又拿走了定海神针，不久便被阎王和龙王告上了天庭。玉帝正要下旨去捉拿妖猴问罪。忙被龙王劝止，龙王说孙悟空神通广大，阎王也深表赞同。玉帝有些迟疑，问众仙该如何是好，太白金星说不如封他一个天宫中的官职去做，这样明为封官，实际上在暗地里确进行压制。玉帝深表赞同。但是要封孙悟空一个什么官好呢？玉帝也一时想不出什么号的职位，于是就宣仙卿百官入朝，共同商讨此事。玉帝问道：“众爱卿，现在天庭什么地方可以空缺的职位啊？给那妖猴一个官去做。”众人都说现在天庭的各个职位人说爆满，无任何空缺职位。大家一时之间不知道该如何是好，此事太白金星灵机一动，向玉帝禀报道：“禀报玉帝，鉴于天庭个职位人员爆满，不如封他一个弼马温的职位。”玉帝问道：“弼马温？是何等职位啊？”太白金星说，弼马温就是用来管理天马的，反正现在天马无人管理，在天庭新设立一个弼马温的职位一方面有利于管理天马，另一方面可以不影响天庭现有的职位，最后还可以安抚妖猴。

之所以会出现上面职位难以安排的情况，这还要从天庭的官吏机制说起，是整个天庭的官吏制度导致了这种情况。在天庭中，法力越大的位置越高，相应的法力越低的位置就越低。而且法力高的由于可以得到各种相应的更多的仙桃啊、太上老君的金丹啊等就变的法力越来越大，也就导致了位置越来越高；另一方面，因为法力越来越大，所以寿命也就越来越长。这导致了什么结果你？导致了终身制！一个职位几乎由相应的仙人一直掌管，永远没有空缺的时候。在天庭是一个萝卜一个坑，官职一旦确定就很难更改！我们说托塔李天王，他就永

远是李天王，没人能够取代他的位置！那现在如何安排孙悟空呢？要做到既不影响天庭既有秩序和众仙的利益，又能够安抚孙悟空，那就只有新设立一个职位啦！

玉帝一听太白金星的想法，大悦。立刻派人把孙悟空请了上来。孙悟空早就听说天庭好玩，而且在天庭还有官职，喜出望外，欢欢喜喜的赴任去了。

定义：

开放封闭原则(Open-Closed Principle)：一个软件实体应当对扩展开放，则修改关闭。对扩展开放，意味着有新的需求或变化时，可以对现有代码进行扩展，以适应新的情况；对修改封闭，意味着类一旦设计完成，就可以独立完成其工作，而不要对类进行任何修改。

开放封闭原则是所有面向对象原则的核心。

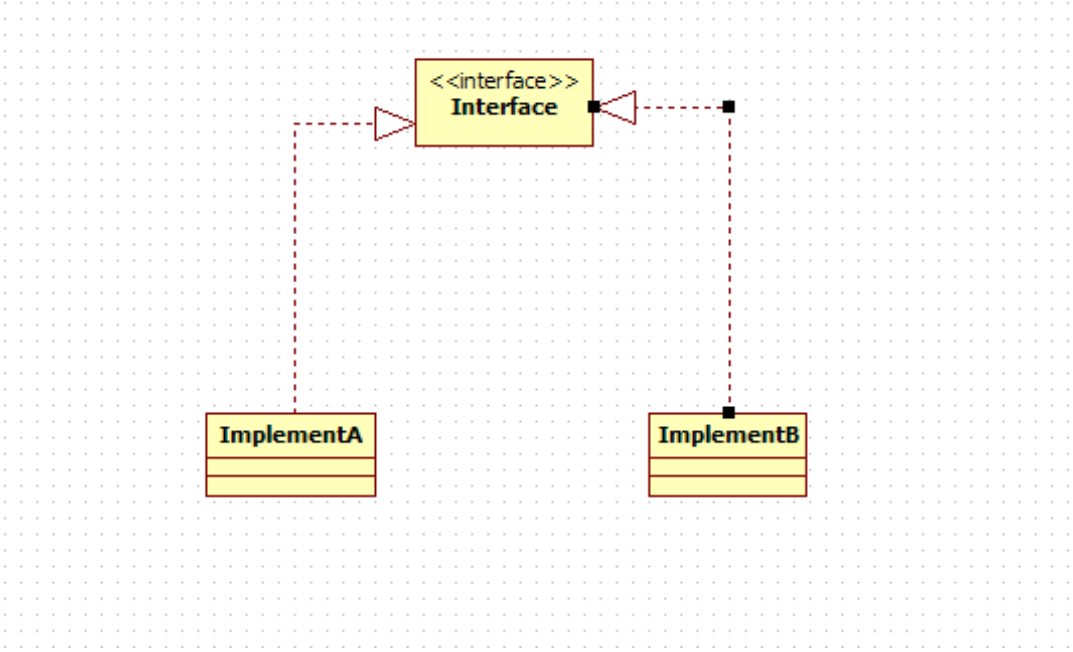
“需求总是在变化”。

“世界没有一个软件是不变的”。

如何做到开放封闭原则呢？答案是：封装变化，依赖接口和抽象类，而不要依赖具体实现类。要针对接口和抽象类编程，不要针对具体实现编程。因为接口和抽象类是稳定的，他们是一种对客户端的一种承诺，是相对不变的。如果以后需要扩展或者开发新的功能，只需要实现或者继承接口或者抽象类即可覆盖或者扩展新的功能，这样做同时也不回影响新的功能。这就很好的做到了对扩展开放、对修改关闭。

实际上讲，绝对封闭的系统式不存在的。无论我们怎么保持封闭，一个系统总会有要变化的地方，“世界上没有一个不边的软件”、“需求总是在改变”。我们要做的不是消灭变化，而是把变化隔离开来，并对其进行封装。我们无法控制变化，但是我们可以预则哪里会发生变法。把要变化的地方抽象起来，这样以后再面临变化的时候我们就可以尽量的扩展，而无须改变以后的代码。

如下图所示：



故事分析：

太白金星献计任孙悟空为弼马温一职就很好的体现了开放封闭原则。

为什么这么说呢？

这还要从天庭的官吏机制说起，在天庭中，法力越大的位置越高，相应的法力越低的位置就越低。而且法力高的由于可以得到各种相应的更多的仙桃啊、太上老君的金丹啊等就变的法力越来越大，也就导致了位置越来越高；另一方面，因为法力越来越大，所以寿命也就越来越长。这导致了什么结果你？导致了终身制！一个职位几乎由相应的仙人一直掌管，永远没有空缺的时候。在天庭是一个萝卜一个坑，官职一旦确定就很难更改！我们说托塔李天王，他就永远是李天王，没人能够取代他的位置！那现在如何安排孙悟空呢？要做到既不影响天庭既有秩序和众仙的利益，又能够安抚孙悟空，那就只有新设立一个职位啦！

总结一下：天庭既有秩序不变，扩展一个弼马温的位置给孙悟空。而且这种扩张不会影响到天庭的其它秩序。这不就是对修改关闭，对扩展开放吗？！

同时，可能不久又出现一个新的“孙悟空第二”，龙王可能又要告到天庭，“需求总是变法的”！这时候只需要按照针对孙悟空同样的思路就可以很好解决此类变化。

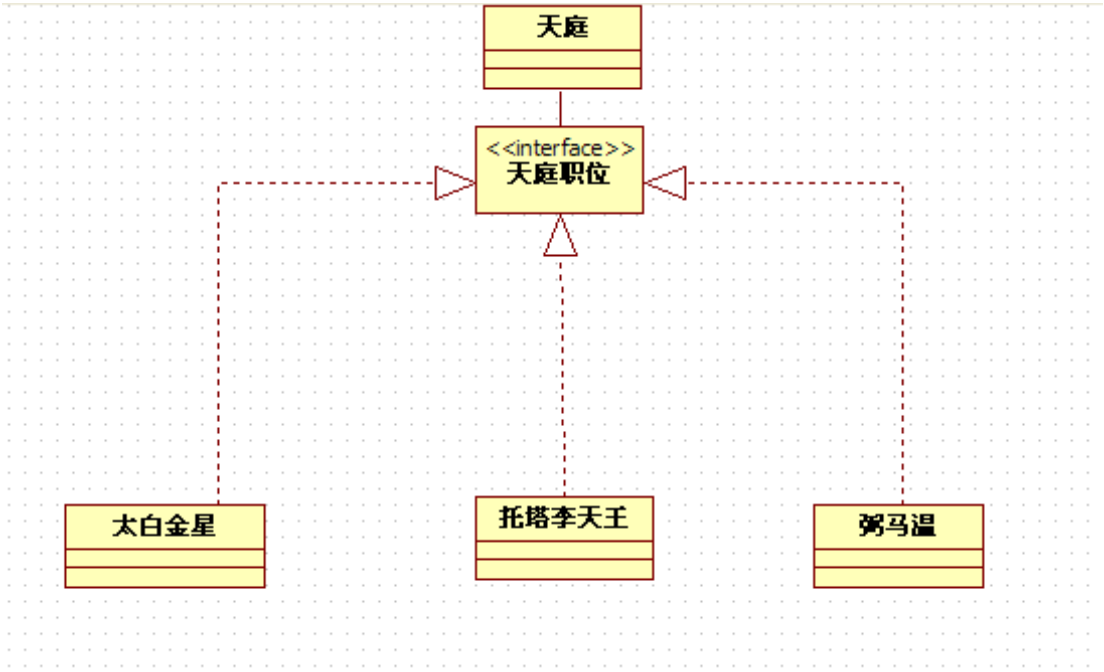
再次重温一下面的话：

“需求总是在变化。”

“世界上没有一个软件是不变的。”

“针对抽象编程，不要针对实现编程。”

如下图所示：



Java 代码实现：

新建一个职位的接口：

```
package com.dieremng.designPattern.OCP;
/*
 * 职位的接口
```

```
 */
public interface Position {
    /*
     * 定义职位的职责
     */
    public void duty();
}
```

太白金星、托塔李天王、弼马温分别实现上面的接口。代码依次如下：

```
package com.dieremng.designPattern.OCP.impl;

import com.dieremng.designPattern.OCP.Position;
/*
 * 太白金星对职位的实现
 */
public class Taibaijinxin implements Position {

    public void duty() {
        System.out.println("这里是太白金星");

    }

}
```

```
package com.dieremng.designPattern.OCP.impl;

import com.dieremng.designPattern.OCP.Position;
/*
 * 托塔李天王对职位的实现
 */
public class Tuotalitianwang implements Position{

    public void duty() {
        System.out.println("这里是托塔李天王");
    }

}
```

```
package com.dieremng.designPattern.OCP.impl;

import com.dieremng.designPattern.OCP.Position;
/*
 * 弼马温对职位的实现
 */
public class Bimawen implements Position {

    public void duty() {
        System.out.println("这里是弼马温");
    }

}
```

建立一个测试类，代码如下：

```
package com.dieremng.designPattern.OCP.client;

import com.dieremng.designPattern.OCP.Position;
import com.dieremng.designPattern.OCP.impl.Bimawen;
import com.dieremng.designPattern.OCP.impl.Taibaijinxin;
import com.dieremng.designPattern.OCP.impl.Tuotalitianwang;

public class PositionTest {

    /**
     * @param args
     */
    public static void main(String[] args) {
        Position taibaijinxin = new Taibaijinxin();
        Position tuotalitianwang = new Tuotalitianwang();
        Position bimawen = new Bimawen();

        taibaijinxin.duty();
        tuotalitianwang.duty();
        bimawen.duty();
    }

}
```

程序运行结果如下：

```
这里是太白金星
这里是托塔李天王
这里是弼马温
```

已有应用简介：

开放封闭原则是所有面向对象原则的核心。

软件的分析、设计、编码、维护等生命周期的各个阶段总是力求做到对修改关闭、对扩展开放。

著名的巴巴运动网生命周期的各个阶段就遵循了开放封闭原则。它把基本的 CRUD 操作做成了一个接口，同时采用了 JDK 5 引入的泛型技术，这样就可以保证以后做基本的添删改查操作时只需要实现该类即可。但由于引入了泛型技术，同时在后台提供了对接口的抽象实现，你甚至不用写一行代码，就可以自如的操作数据库。如果以后又需要扩展的地方，只需要扩展继承扩展自己的特有的操作即可，大大提高了生产效率。

温馨提示：

遵循开放封闭原则的关键是依赖于抽象，但是否依赖了抽象就一定遵循了开放封闭原则呢？这里面更核心的东西是对职责的分离和封装，只有分离出变化和不变的元素，把变化的地方抽象起来，这种抽象可以使用接口，也可以使用抽象类，针对分离好的抽象编程，而不是针对具体编程，这样才才是真正的遵循开放封闭原则。