facade模式

场景

1、假设我们现在有两个子系统

2、子系统A，有多个内部模块，模块A、模块B和模块C

3、子系统B，现在有一个功能，需要同时依赖3个模块来实现一个功能

public interface ModuleA {

   void operate();

}

public class ModuleAImpl implements ModuleA {

   public void operate() {

       System.out.println("实现模块A的功能");

  }

}

public interface ModuleB {

   void operate();

}

public class ModuleBImpl implements ModuleB {

   public void operate() {

       System.out.println("实现模块B的功能");

  }

}

public interface ModuleC {

   void operate();

}

public class ModuleCImpl implements ModuleC {

   public void operate() {

       System.out.println("实现模块C的功能");

  }

}

【外观模式，就是将调用的组合，封装起来，统一对外】

public class SystemAFacade {

   public static void operate() {

       ModuleA moduleA = new ModuleAImpl();

       moduleA.operate();

       ModuleB moduleB = new ModuleBImpl();

       moduleB.operate();

       ModuleC moduleC = new ModuleCImpl();

       moduleC.operate();

  }

}

【调用处：直接调用外观模式下的对象】

public class SystemB {

   public static void main(String[] args) {

       SystemA.operate();

  }

}

外观模式其实很简单，就是说，假设我们有一个系统A吧，自己内部有多个代码组件，每个代码组件都可以实现一些功能。此时如果别的系统B要调用系统A实现某个功能，此时一种方法是，系统B调用系统A的多个代码组件的方法，最终实现一个功能。另外一个选择，就是**系统A里提供一个Facade类，系统B不需要care系统A内部有多少个代码组件，直接调用系统A的一个facade类的一个方法即可。**

就是说，大家可以想象一下，比如说**service就是一个经典的门面模式**

如果一个子系统里有多个DAO组件，然后如果别的子系统要实现一段业务逻辑，需要使用这个子系统的多个DAO组件，那就会导致别的子系统代码会很乱，需要care这个子系统很多DAO组件的实现。但是这个组件可以基于多个DAO组件封装出来一个service接口，然后别的系统就针对一个service接口开发即可。这个service不就是一个典型的门面吗？

这个模式在我们的电商系统里可以大量使用，就是用facade包装每个子系统内部的大量代码组件，对外暴露统一简单的接口

853769620