

一．类之间的关系

1.继承(泛化 is a extends)

2.实现关系（like a implements）

3.依赖关系（Uses A 方法参数，方法体里面用）

4.关联：组合和聚合，就是类属性。

5.聚合：Has a 类属性，没有也可以存在，如老师没有部门属性，也可以存在。

6.组合：关系较强，has a，没有则不能单独存在，如房子没有房间是不行的。

二．软件设计5大原则



**1.六大原则-单一职责原则**

单一职责 ：只做一件事，做好一件事

原则思想：一个方法只负责一件事情。描述：单一职责原则很简单，一个方法 一个类只负责一个职责，各个职责的程序改动，不影响其它程序。 这是常识，几乎所有程序员都会遵循这个原则。优点：降低类和类的耦合，提高可读性，增加可维护性和可拓展性，降低可变性的风险。

**2.六大原则-里氏替换原则** 子类可以完全替其父类或者接口

原则思想：使用的基类可以在任何地方使用继承的子类，完美的替换基类。描述：子类可以扩展父类的功能，但不能改变父类原有的功能。子类可以实现父类的抽象方法，但不能覆盖父类的非抽象方法，子类中可以增加自己特有的方法。优点：增加程序的健壮性，即使增加了子类，原有的子类还可以继续运行，互不影响。**3.六大原则-依赖倒置原则** 抽象不应该依赖细节，细节需要依赖抽象。高级模块不应该依赖低级模块，两者都应该依赖抽象。

原则思想：高层次的模块不应该依赖于低层次的模块，他们都应该依赖于抽象，抽象不应该依赖于具体实现，具体实现应该依赖于抽象。描述：类A直接依赖类B，假如要将类A改为依赖类C，则必须通过修改类A的代码来达成。这种场景下，类A一般是高层模块，负责复杂的业务逻辑；类B和类C是低层模块，负责基本的原子操作；假如修改类A，会给程序带来不必要的风险。优点：可以减少需求变化带来的工作量，做并行开发更加友好。

**4.六大原则-接口隔离原则** 接口要隔离，客户端不应该依赖它不需要的接口。

原则思想：类和类之间应该建立在最小接口的上。描述：类A通过接口依赖B，类C通过接口依赖D，如果接口类A和类B不是最小的接口，则依赖的类B和类D必须要实现他们不需要的方法。优点：提高程序的灵活度，提高内聚，减少对外交互，使得最小的接口做最多的事情。

**5.六大原则-迪米特法则**

原则思想：一个对象应当对其他对象有尽可能少地了解，简称类间解耦描述：一个类尽量减少自己对其他对象的依赖，原则是低耦合，高内聚，只有使各个模块之间的耦合尽量的低，才能提高代码的复用率。优点：低耦合，高内聚。

**6.六大原则-开放封闭原则**

增加扩展，不要修改。

原则思想：尽量通过扩展软件实体来解决需求变化，而不是通过修改已有的代码来完成变化描述：一个软件产品在生命周期内，都会发生变化，既然变化是一个既定的事实，我们就应该在设计的时候尽量适应这些变化，以提高项目的稳定性和灵活性。优点：单一原则告诉我们，每个类都有自己负责的职责，里氏替换原则不能破坏继承关系的体系。

三．具体设计模式

**形容词，有什么功能，用接口**

**名词，具体的东西，抽象一点的，用抽象类**

1. 单例模式
2. 饿汉
3. 内部类
4. 双重检查锁
5. 枚举

应用场景举例子:spring bean factory

1. 策略模式

a.comparator

应用场景举例子:Java各个arrays collections stream的sort比较器。

1. 工厂模式
2. 静态工厂 静态方法(不太算)，getInstance();
3. 简单工厂 一个工厂，生产所有的
4. 工厂方法，分几个工厂，生产各自产品；感觉就是策略模式，只不过是生产了对象。（产品维度扩展，族方向不能扩展）
5. 抽象工厂 生产任意产品，产品族，一个工厂生多个产品，(工厂方法就是能生产一个产品的抽象工厂。)（产品族方向扩展，产品方向不好扩展）

抽象工厂虽然能生产产品族，但是每个产品是固定的，不好扩展，所以想在两个方向扩展。就是Spring的IOC（或者DI）了，每个对象注入新的对象，可以理解为产品，对象即为工厂。可根据唯一id来任意组合产品。扩展性任意。

应用场景举例子:spring bean factory

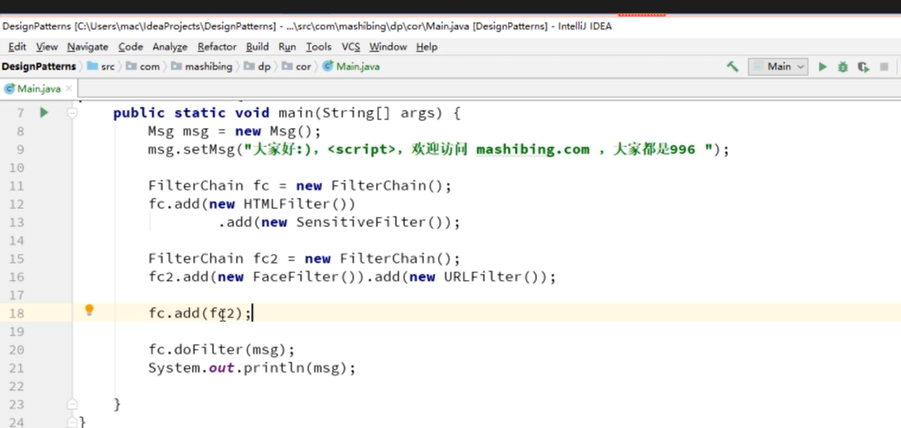
1. 门面模式：对外，一条龙服务，统一对外接口。（OpenApi，统一对外接口）。解耦。
2. Mediator调停者模式：系统间，只和一个服务打交道。例如时间服务（ESC，消息中间件）。

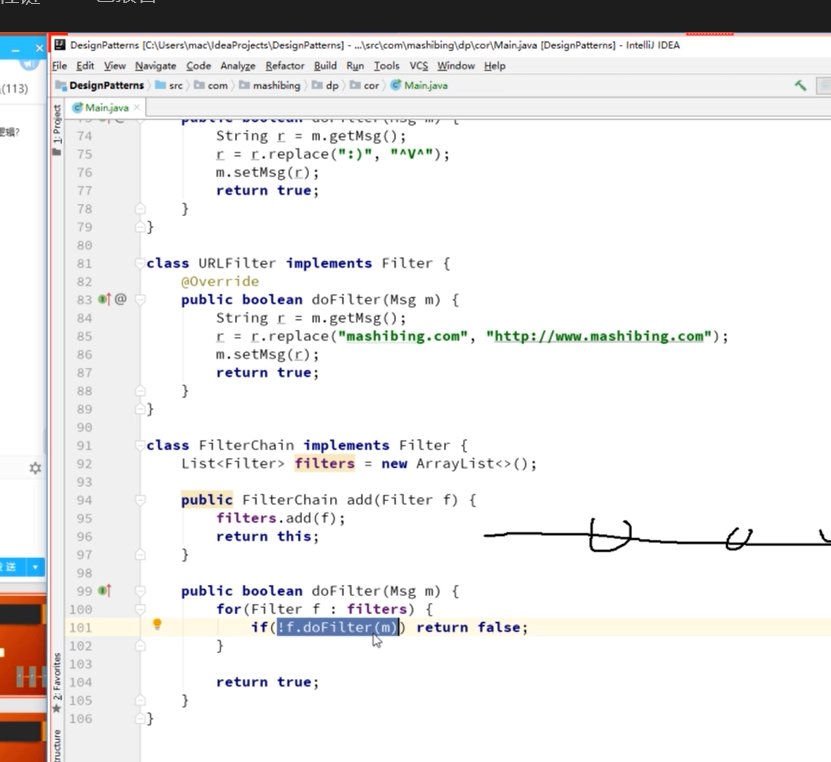
4和5可以是一个服务。

1. 装饰器模式(和静态代理特别像)

加点东西，收费计算中的改进。Java IO中的写法。

1. 责任链模式（倒数第三难，第一是响应式，第二是代理）





Servelet的过滤器，spring拦截器，struts的过滤器

1. 观察者模式（监听模式）(hook callback 钩子函数，回调函数)

事件源（被观察者）生产 事件 观察者。

发生一件事（主动或者被动），去处理。

https://www.kancloud.cn/cyyspring/more/1383971

模板方法+钩子函数

public abstract class HummerModel {

protected abstract void start(); //发动

protected abstract void stop(); //停止

protected abstract void alarm(); //鸣笛

protected abstract void engineBoom(); //轰鸣

final public void run() { //车总归要跑

this.start();

this.engineBoom();

if(this.isAlarm()) {//想让它叫就叫，不想就不叫 控制的，类比事件，由子类自己控制要不要出发这个事件。

this.alarm();//钩子

}

this.stop();

}

protected boolean isAlarm() { //我们加了一个判断方法，默认返回true

return true;

}

}

JS中的例子



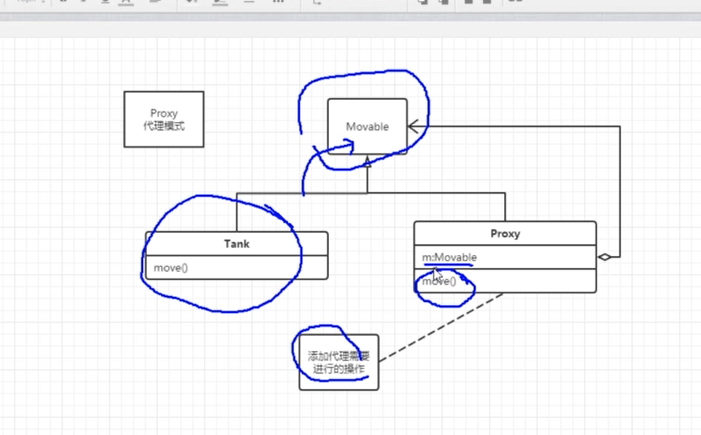
1. 组合模式，树状（递归遍历，非递归遍历-）栈式遍历），Node。
2. 享元模式：**共享**元数据，Int的-128-127。 池子（不是资源，是元数据，静态的。）（池化是动态的，链接，线程等。）。

JVM的常量池子。

结合组合才比较负责。

1. 代理模式（静态，动态，aop,sdk动态代理 || cglib,）

a.静态代理：（怎么感觉是装饰器呢，像IO那种感觉。）



b.动态代理 jdkproxy

c．动态代理之cglib

d.aop

1. D
2. D
3. Dddddd
4. D
5. D
6. D
7. D
8. D
9. D
10. D
11. D
12. D
13. D