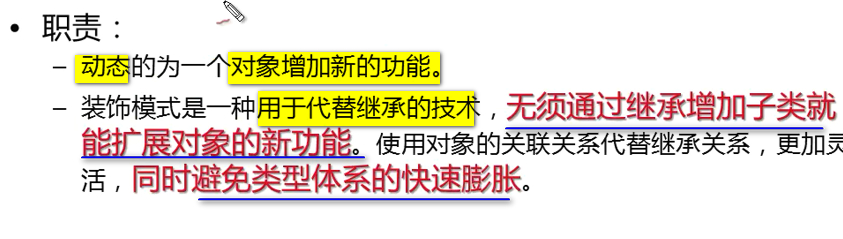
装饰器模式



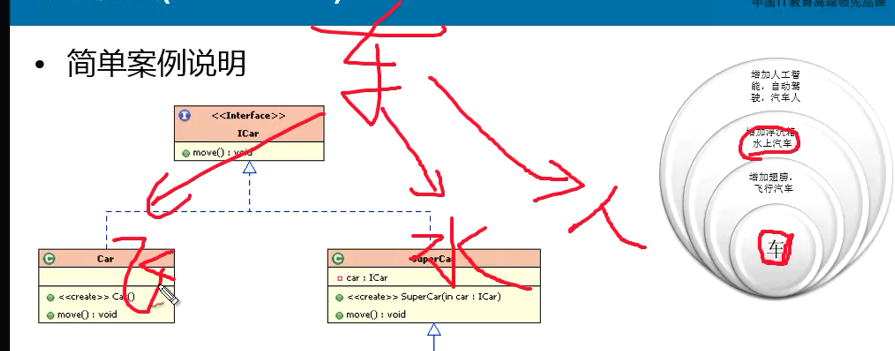
【桥接模式也有避免子类膨胀的作用 但是 前提是有两个以上不同的维度 ---- 现在是单个维度 所以 】

替代的是继承的技术

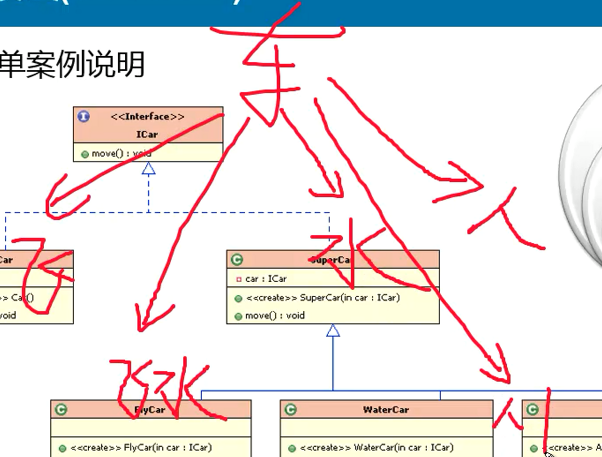
给对象新家功能 可以通过子类添加 但是 使用装饰器模式 可以避免子类；类型的扩张

现在有一个例子

车下面有一个飞行的汽车 水上汽车 还有自动驾驶的人工智能的汽车



如果我要一个汽车 又能飞 又能水上漂



这样这种组合功能 就会增加出来非常多的子类

这样排列组合 增加很多的子类 每增加一个新功能 就要增加很多子类

如果还要潜水 这样就会造成子类的迅速膨胀

【如果把这些功能 进行分成维度 就可以使用桥接模式】

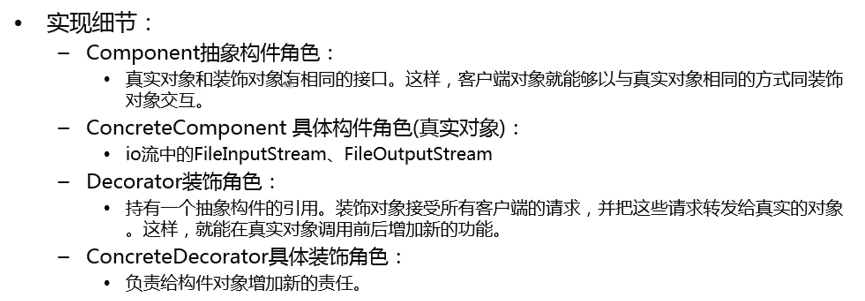
使用装饰模式 就可以避免就可以



我们就定义蓝色的几个子类 飞行的车 水上漂的车 人工智能的车 和 潜水的车

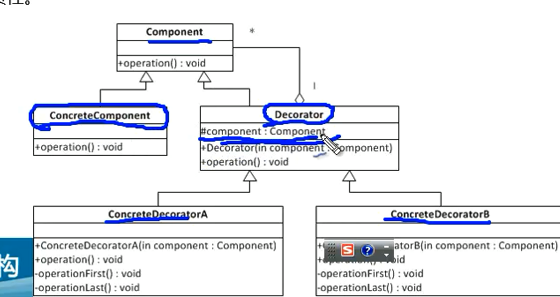
这样就避免了类型的膨胀 通过对象组合关系 揪出来一个组合的功能

这就是装饰模式的好处



【前面说过 转换流失适配器模式】

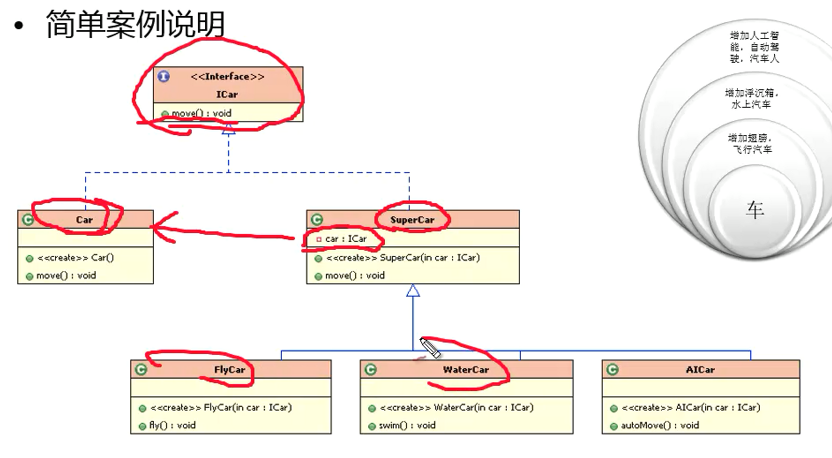
利用对象的组合关系 组合一下这个案例



装饰器持有真实对象的引用

我们这个ICar就是抽象组件 某一个move方法

有真实的对象对应的类 Car和SuperCar



这样 FlyCar WaterCar 和AICar

装饰器模式 就把新增的功能作为一个类结构 作为存储

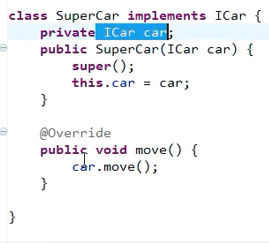
这样 新的功能就在SuperCar下面扩张 不会影响类的层次结构 不会暴增

首先是抽象的接口



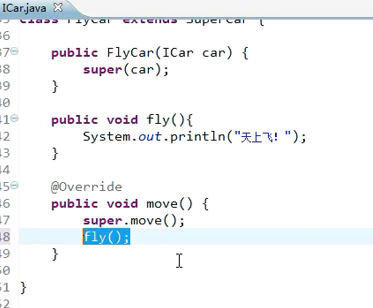


首先作为装饰器 要有真实对象的引用

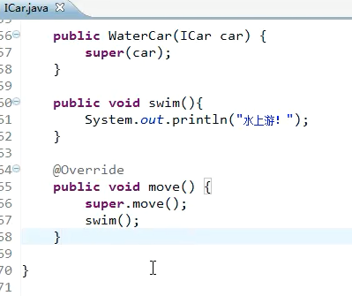


装饰器的角色

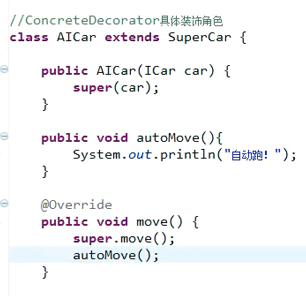
下面就是增加具体的装饰器

 FlyCar

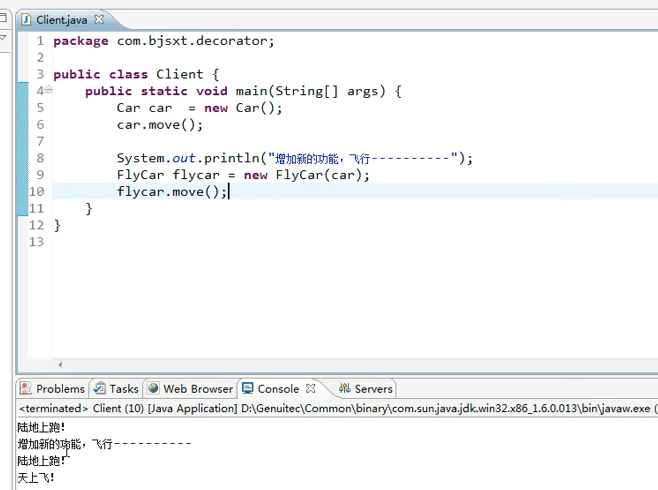
下面是具体装饰对象



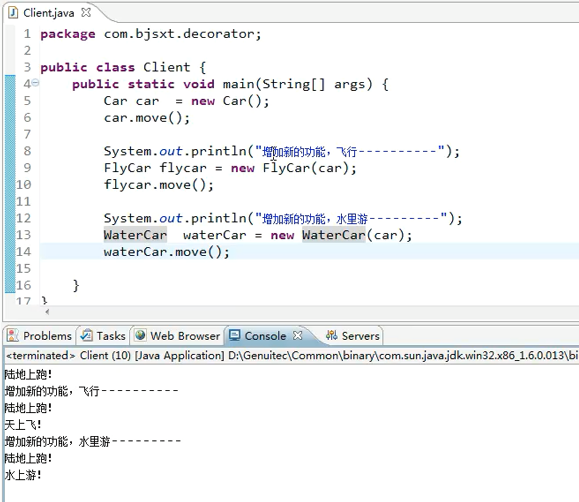
还有一个AICar



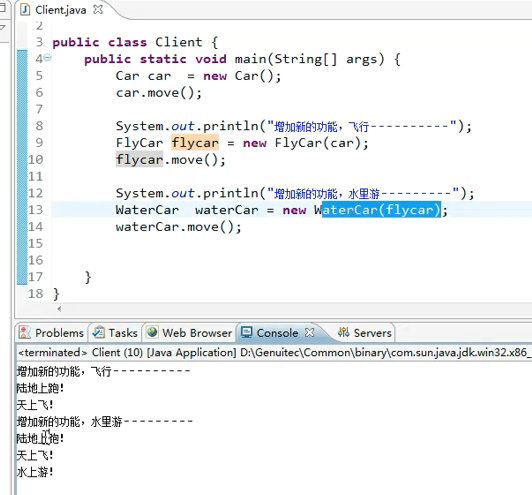
现在做一个测试



继续动态增加 水里游

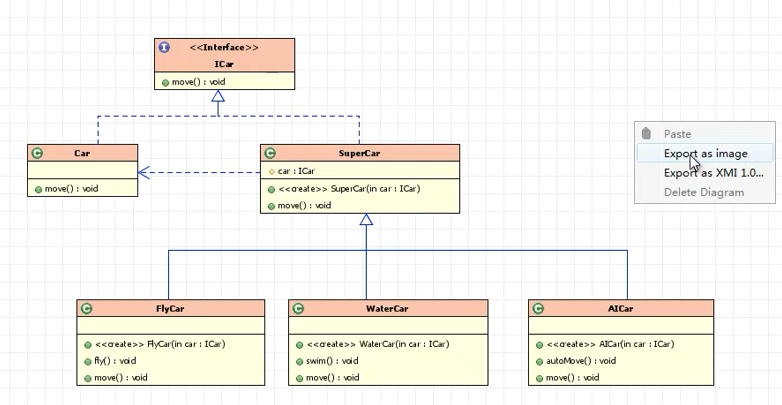


问题是 我想传入flyCar



这是一个复合的类型

画一下类图



有一个前面的IO流就是包装流



非常典型的装饰器模式



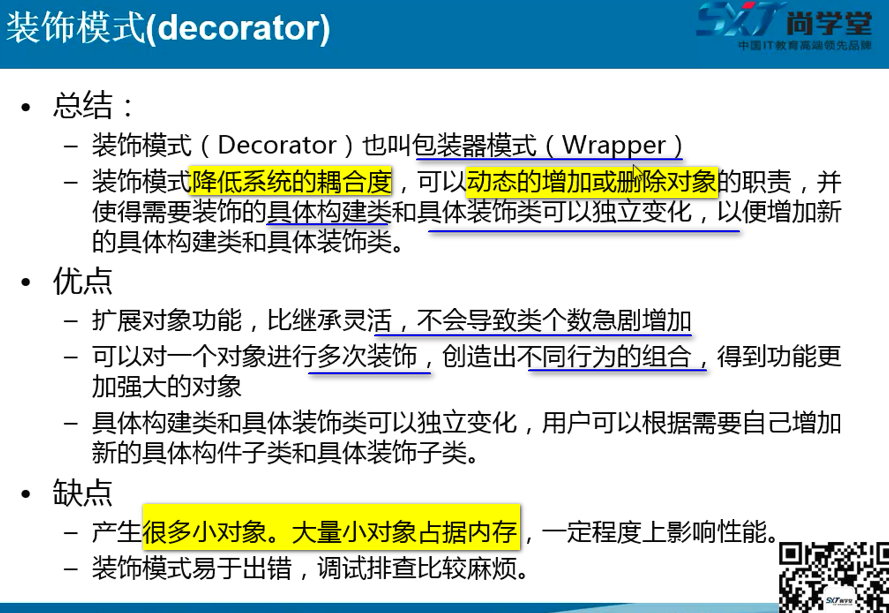
IO中的输入流 输出流都是给予InputStrean OutputStream Reader Writer都是基础

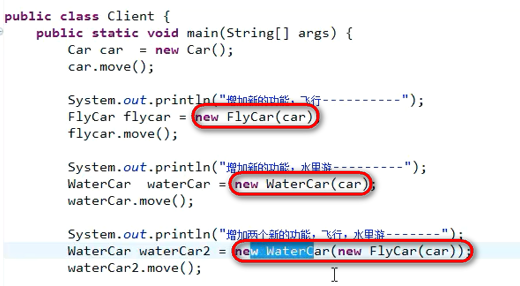


Swing包 也用到了装饰器模式

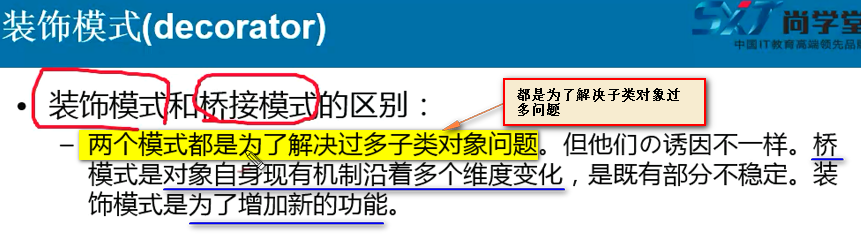
Servlet API中提供了request和response增加了对应的Wrapper

装饰模式 也叫包装模式 XxxWrapper 基本都是装饰器模式





装饰模式 和 桥接模式都可以替代继承 怎么区分？



桥接模式是用来解决多层继承的问题 ----- 多层继承会造成子类膨胀 ------ 变化维度就比较多 本来就不稳定

装饰模式就是本来很稳定 就是为了增加新的功能的 --- 增加的功能是新的组合的时候 就可以使用装饰模式

两者的关注度不一样