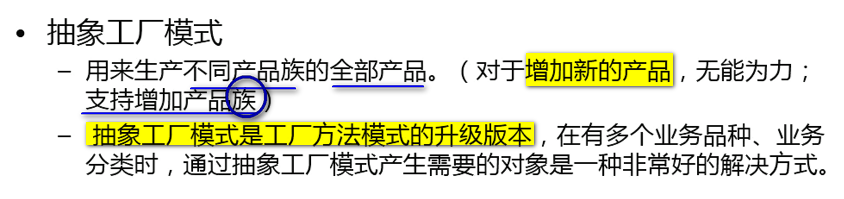
抽象工厂模式

简单工厂 和 工厂方法都是用来创建Car接口下面的子类

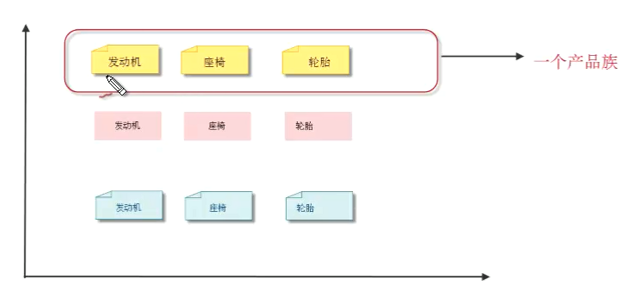
现在要创建多个接口 每一个接口下面有多个实现类

这时候 简单工厂 工厂方法都无能为力

创建接口和多个产品的时候怎么办？引入产品簇



----- 抽象工厂模式 对新增加的产品是无能为力的



三种接口

粉色的是高端的实现类 ---- 高端的产品族

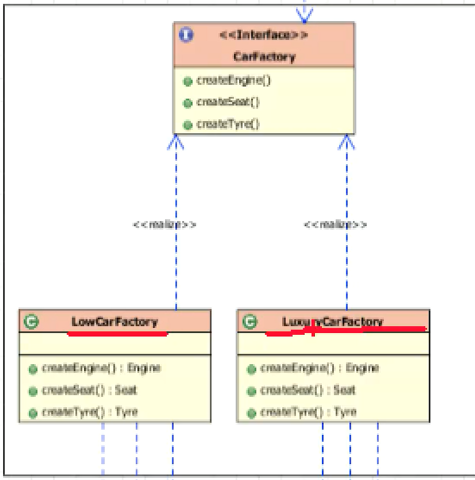
蓝色的是低端的实现类 --- 低端的产品族

抽象工厂模式对增加产品族是非常方便的 但是 对增加某一个产品是无能为力的

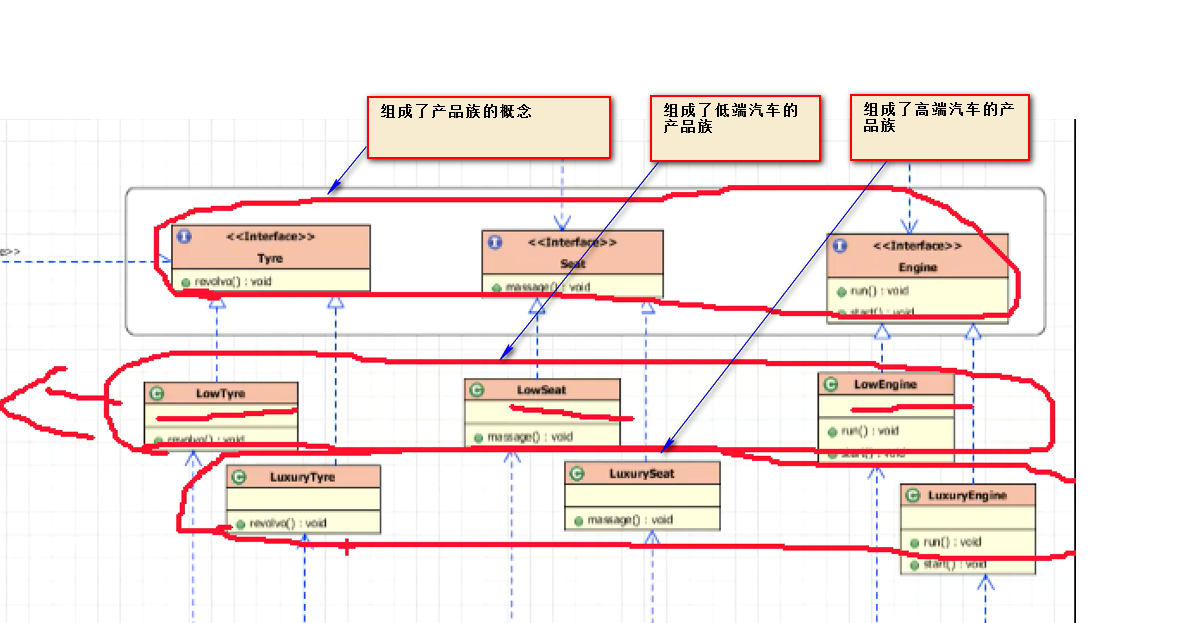
工厂方法是简单工厂的升级版 ---- 单个产品

抽象工厂实际上是另一种业务情况 ----- 产品族

我们编码以这张图为例

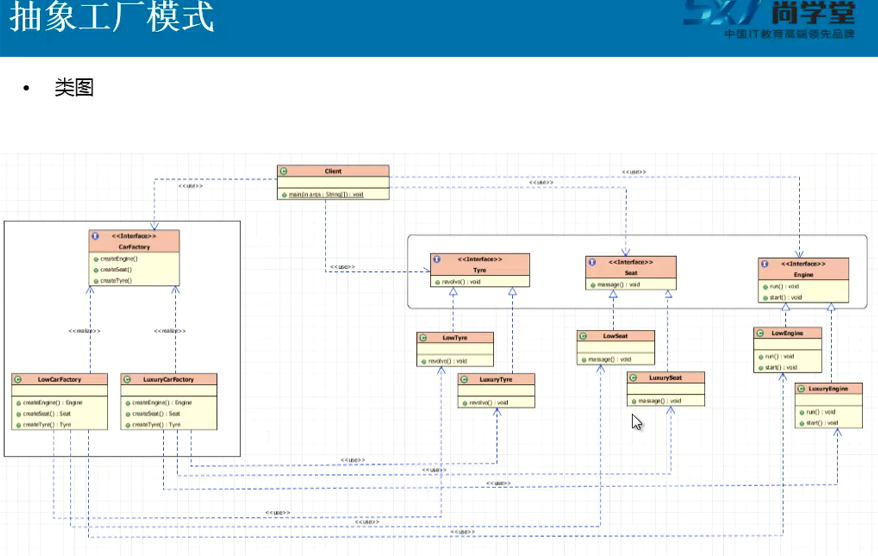


这里面有两个实现类 高端汽车工厂和低端汽车工厂

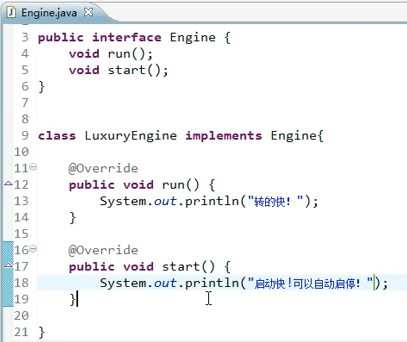


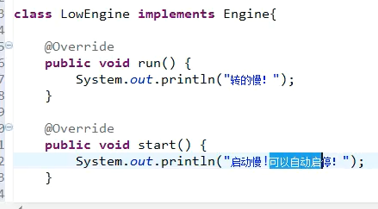
还有可能是 高端轮胎 +低端座椅 +低端发动机构成终端中端汽车 这样CarFactory就多了一个中端汽车

这样 我的Client就跟接口和CarFactory打交道就可以了



创建产品族的接口



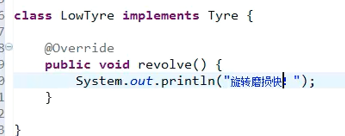


针对座椅增加接口



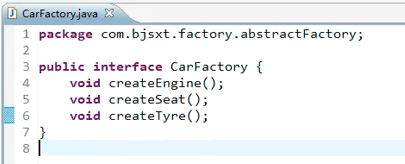
还有一个是轮胎





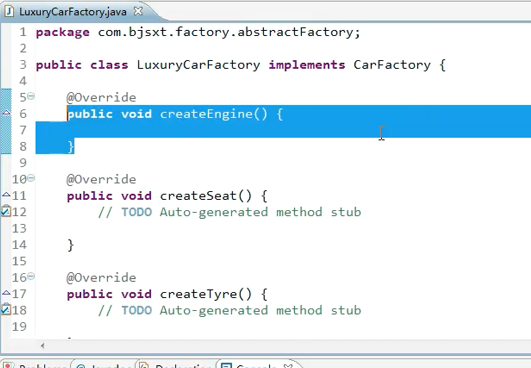
这样产品就加好了

构建CarFactory

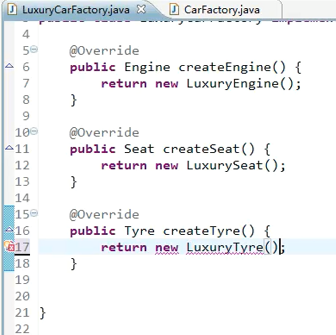


创建具体的工厂

【产品族是可以任意组合的】



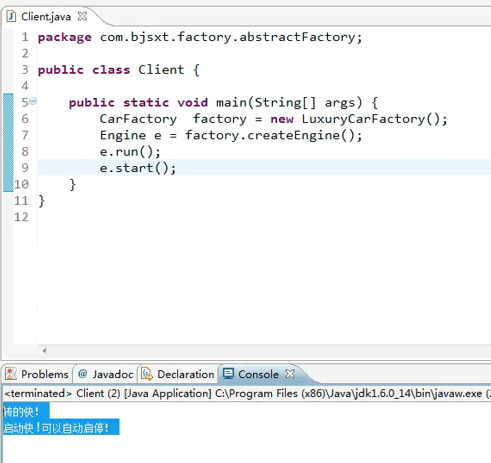
创建我们的



这就是高端汽车的产品

创建一个低端汽车工厂

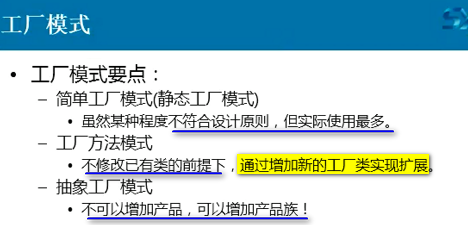
建立一个Client



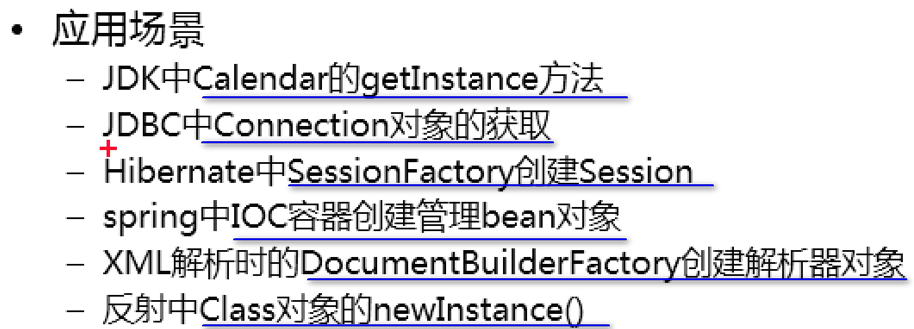
创建出来高端引擎之后 就能直接使用了

【实际上 缺一个模板方法 来调度这个高端的车 没有的话 就直接在Client中自己调用】

总结工厂模式



工厂模式的应用场景



工厂模式出现 有时候和单例模式组成

Servlet就是 工厂模式和单例模式组成的

工厂模式会套用其他模式

实际上 见名知意是最基本的 开源中遇到XxxxFactory 一定用的就是工厂模式