构造者模式

创建者模型： 学习了单例模式 工厂模式 （简单工厂 工厂方法 抽象工厂）

下面这个建造者模式 和原型模式都很重要

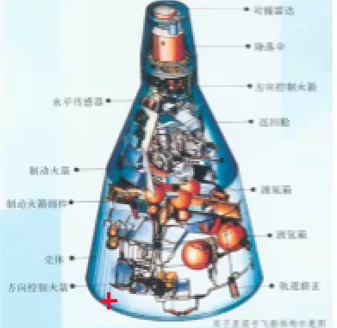


工厂模式 用来构建某一个产品或者某一个产品族

还有一中情况 要创建一个Iphone 有好多小组件 比如电池 主板 等等

弄好之后 还有一个装配的过程 --- 需要进行额外的装配过程

比如要构建神舟飞船



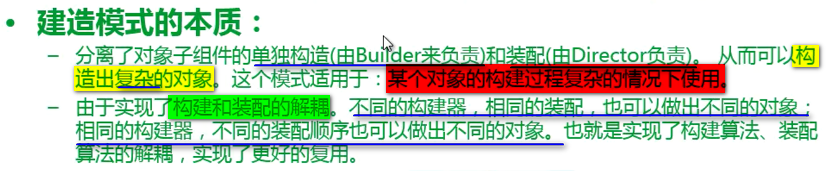
所以让各个工厂创造好之后 再把各个零件装配起来 需要一个装配的过程

实际开发的时候 建造者要和工厂模式搭配起来

每一个零件来自于工厂 然后通过构造者进行装配

这些模式是经常是**互相搭配**的

通常用来构建复杂的对象



可以用来构造非常复杂的对象

----- 不同的构建器 + 相同的装配 =🡺 不同的对象

低端汽车构建者 构建了低端的发动机和轮胎 底盘

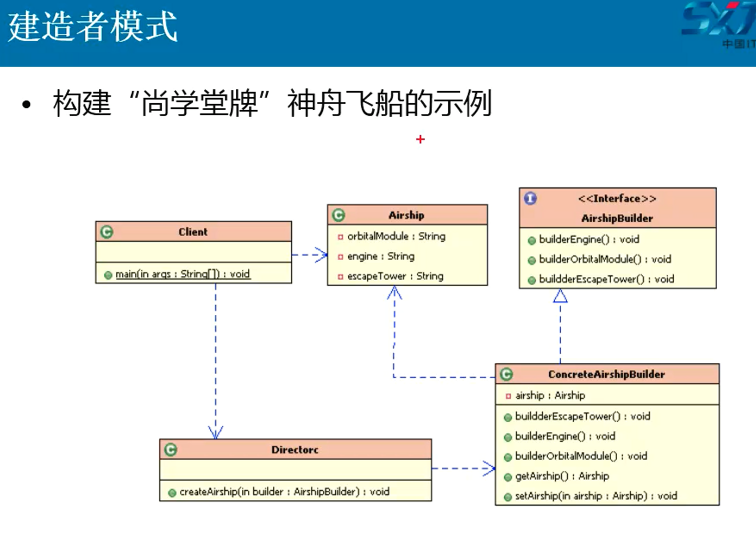
组装过程一样 构建出高端汽车 和低端汽车

相同的构造器 + 不同的装配顺序 =🡺 不同的对象

搭积木一样 不同的组装顺序搭建出不同的结果

构建和装配的解耦合 实现了更好的复用

构造者模式 分工出现了 单独的构造器 + 装配器 【实现了构建和装配的解耦合】



AirshipBuilder用来构建子组件

Drecterc用来进行装配 这个搞成接口实现类 就有多重装配

客户端调用的是Directerc 和 要什么 Airship

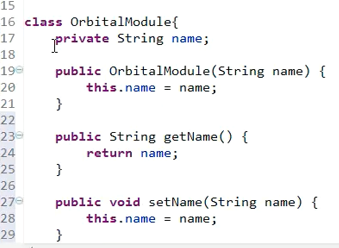
Builder+Director

先看宇宙飞船类 --- 轨道舱 + 发动机 + 逃逸塔

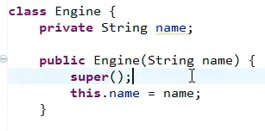
这些组件不能用字符串模拟 ---- 避免大家误解

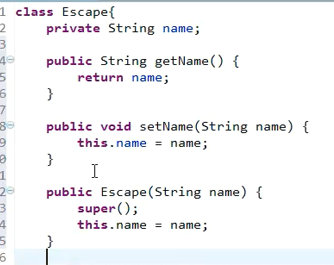
构建几个新的类

一个文件中建立好

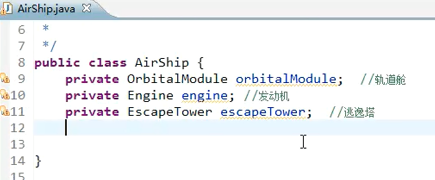
 --- 轨道舱

Engine类

 ---- 引擎



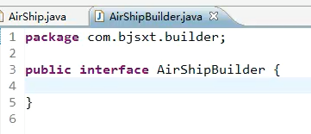
最后是宇宙飞船类



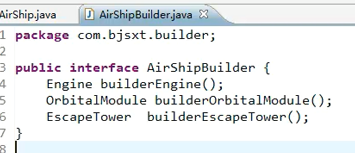
这样飞船有三个子组建 + setter getter

\*\*\*构建一个Build接口

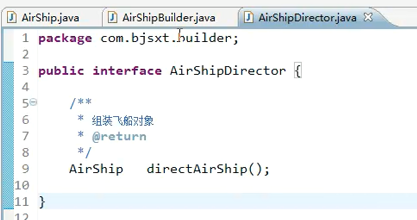
先提供一个build接口



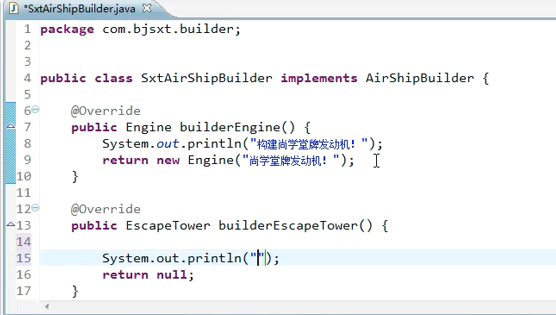
构建一个发动机 轨道舱 和 逃逸塔 生成自组建



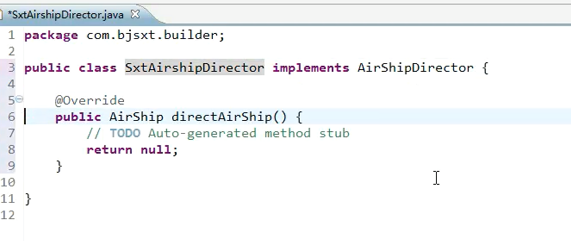
再创建一个接口 ---- Director 用来组装或者装配



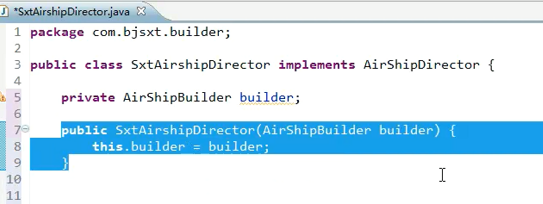
定义构建者的实现类 以Builder都是构建者模式 比如StringBuilder DomBuilder SAXBuilder等等 这些都用到了构建者模式



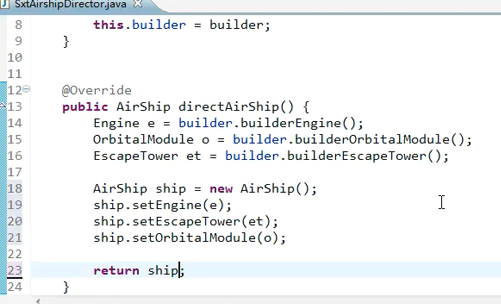
后面是类似的效果



希望返回一个AirShip的对象 注意 装配者一定调用构建者 ---- 构造出来

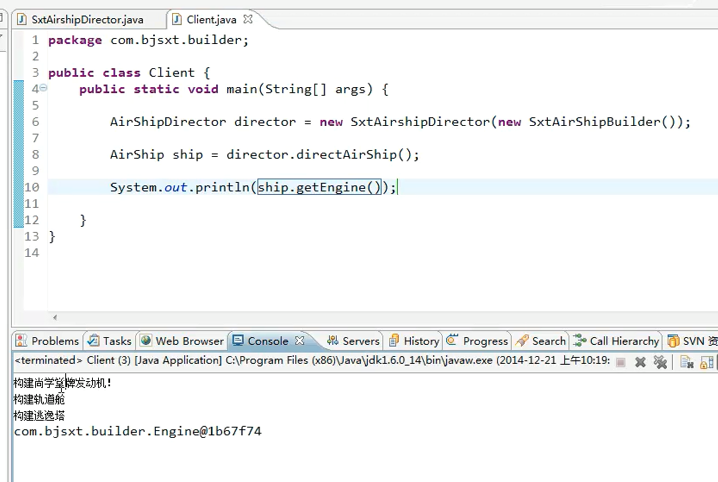


【构造的过程 就是把之前工厂模式中的Client的内容 拿到Director 也就是 之前是在Client端进行模拟 这个过程Client确定 现在是Director自己来确定】



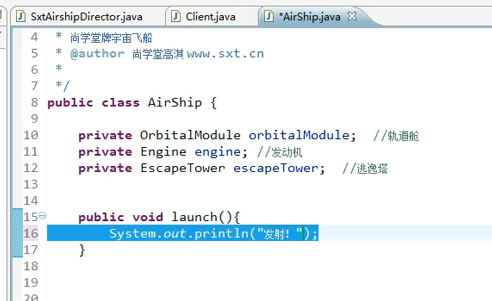
构建client

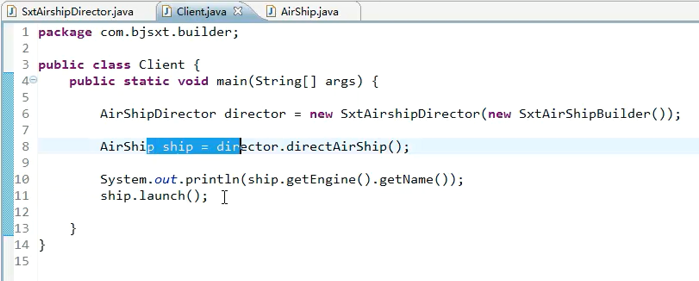
【构建什么 Director什么 就有什么的Builder Builder中就有子组建的creator】



【XxxDirector调用了XxxBuilder XxxBuilder实际上就是子组件的工厂 --- 简单工厂】

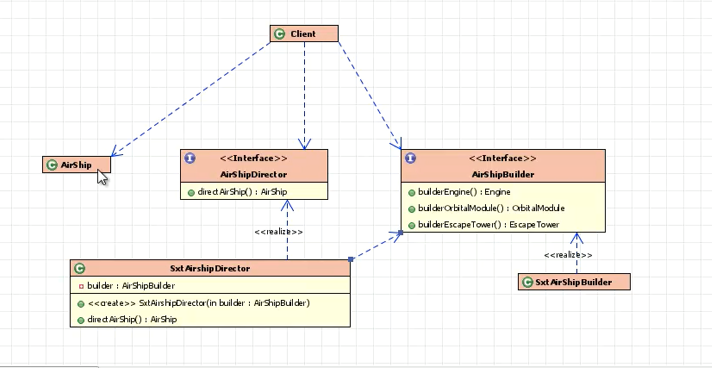
还可以对AirShip添加额外的方法

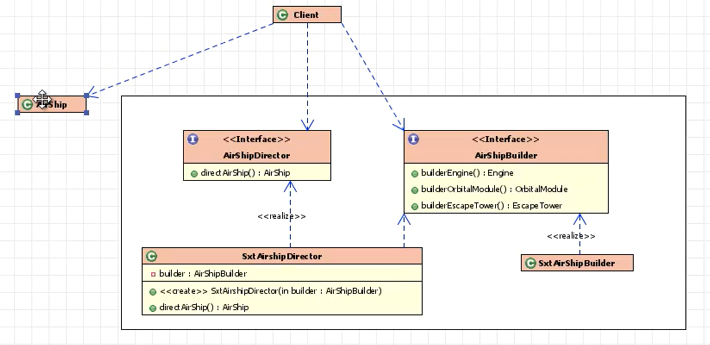




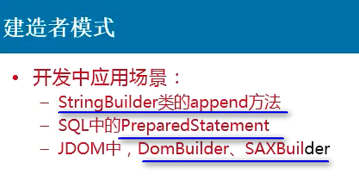
这就是建造者模式的简单类图

客户端只要知道AirShip AirShipDirector和 AirShipBuilder





建造一个对象 由先后关系 并且非常复杂



开发效果 构建这么复杂的对象 不多见 除非开发一个产品