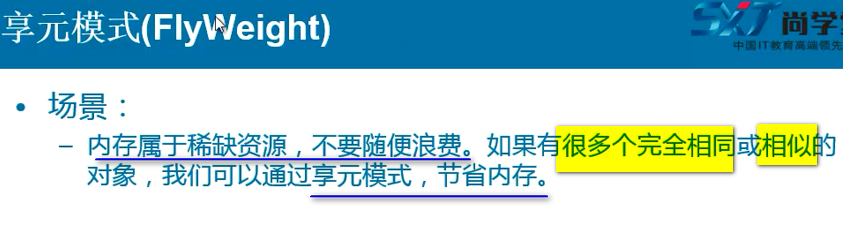
享元模式 Flyweight

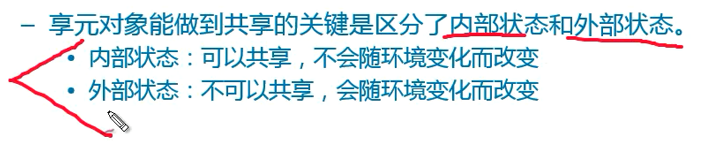
FlyWeight表示轻量级的意思

享元 --- 共享数据



时间换空间 节省了内存 一定程度提高了效率



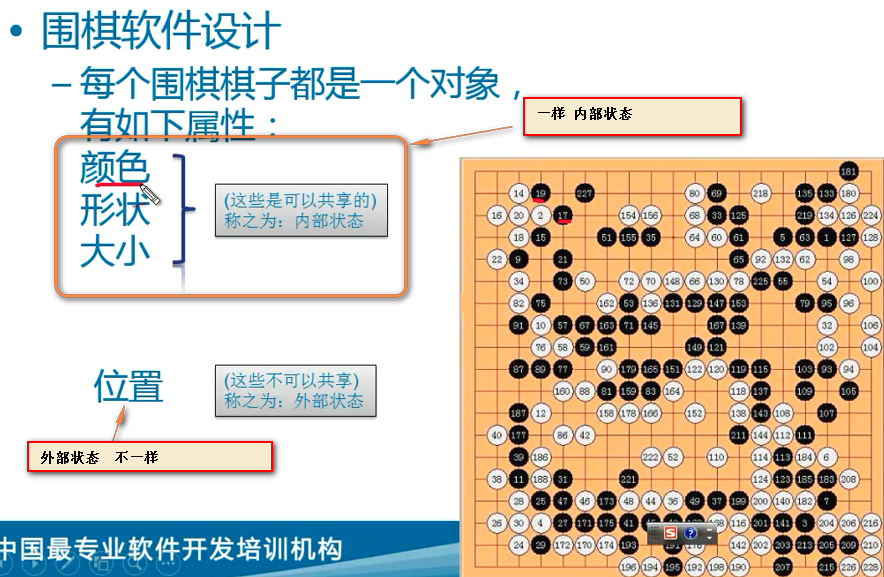


内部状态可以共享 不会随着环境变化而改变

所以 享元模式共享的是内部状态

假设设计一个围棋软件

每一个棋子都是一个对象 几百个

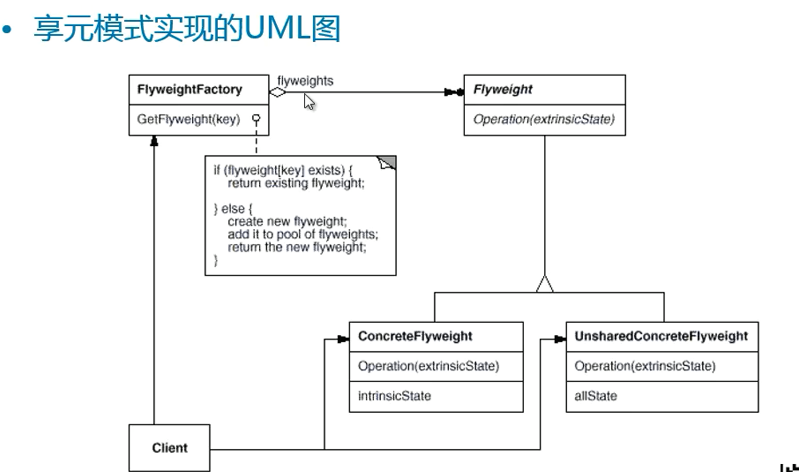


享元模式一定要区分出内部状态 和外部状态

----- 这样 大量的棋子 颜色 形状 大小的状态 就可以共享 但是 位置对象不可以共享

享元模式使用的时候 常常和工厂模式一起使用





FlyWeightFactory

享元池就是Map

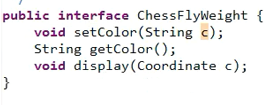
持有list

这个FlyWeight有ConcreteFlyweight和UnsharedConcreteFlyWeight

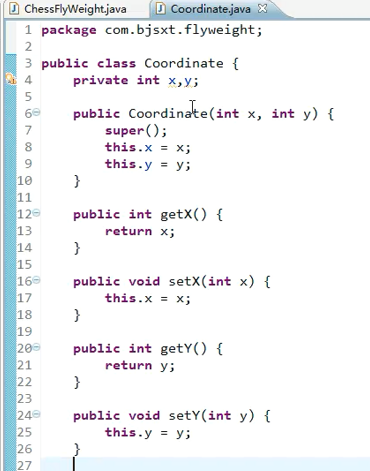
实现这个图中的围棋

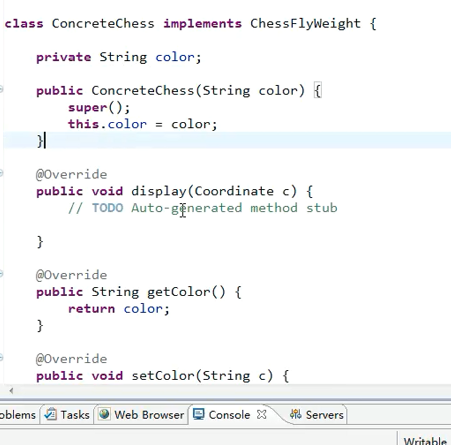
【享元 就是 共享状态相同的元数据 元数据就是属性】

先设计享元类

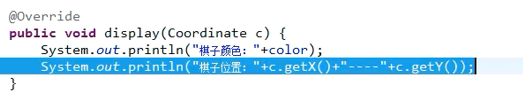


显示的位置不一样 --- 定义一个坐标类

 相当于外部状态 UnsharedConcreteFlyweight



现在是diplay



这是设计的享元类对象

 ---- 内部状态设计成成员变量 并进行存储 这里面的颜色color因为都一样 所以共享 所以设计成ConcreteFlyWeight类的属性 并在构造和方法中可以设置存储

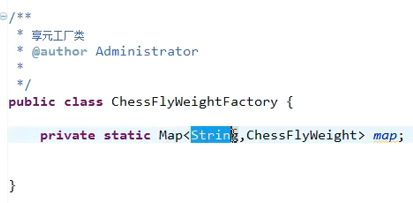


那么对于外部不一致的类 这里面的位置 就单独设计成一个新的类

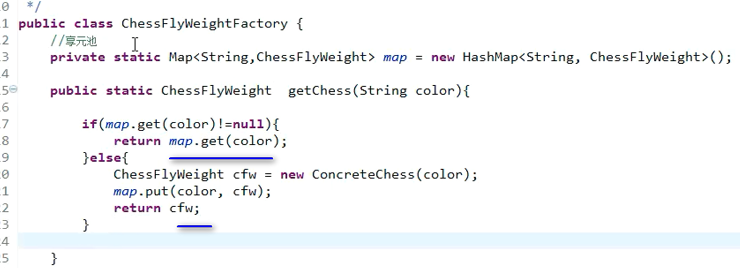
这里面是Coordinate类

现在这几享元工厂 ==== 没有享元工厂 就没有享元的优势

首先要设计一个Map

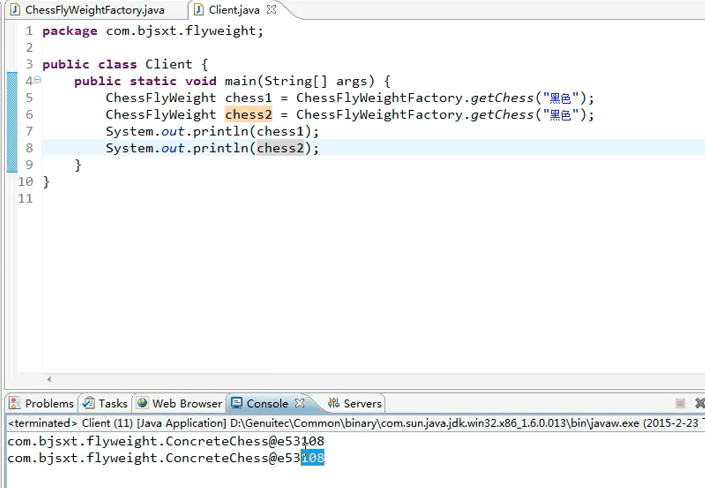


这个map就是享元池对象

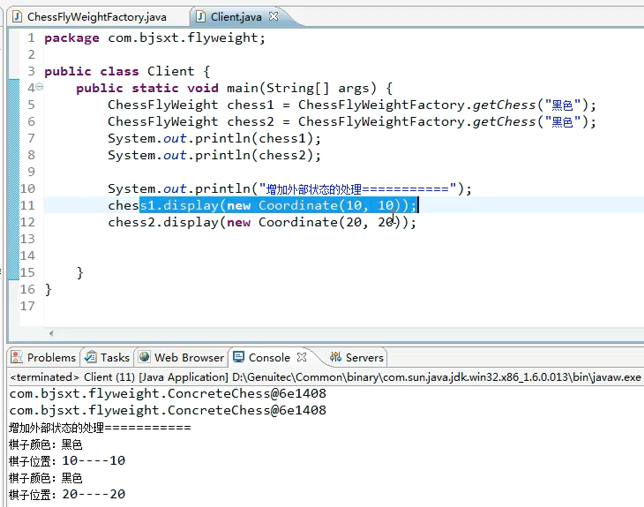


写完之后 测试一下

从工厂里面直接获取出我们的棋子



现在就是反复的 把棋子放到某个位置 增加外部状态的处理



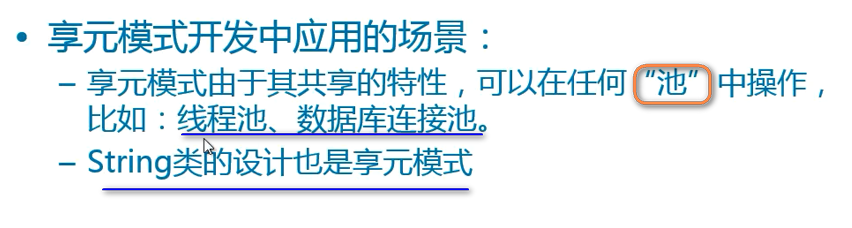
Xxx池都是 享元模式

这样Color实现了共享 不同而地方 外部传进来

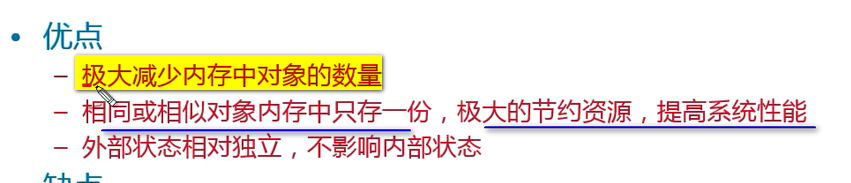
这样内部状态和外部状态分开

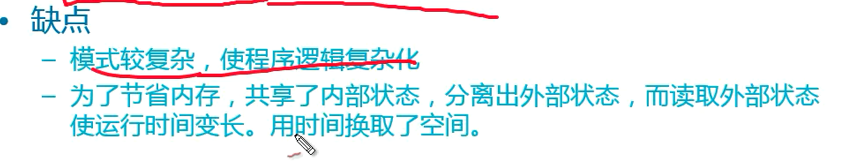
--- 要素 三点

---- 一定要有池化的工厂 内部状态使用成员变量存储你的状态 外部状态另起炉灶



String类 用的时候 有一个共享常量池中的对象 多个String可以共享池中的内容



 传递外部小对象 这样会增加运行时间

