设计模式

1. 设计模式分类

创建型模式：用于描述“如何创建对象”，主要特点是将对象的创建和使用分离。

结构型模式：如何将类和对象按照某种布局构成更大的结构。

行为型模式：描述类和对象之间协作共同完成单个对象无法单独完成的任务，以及如何分配职责。

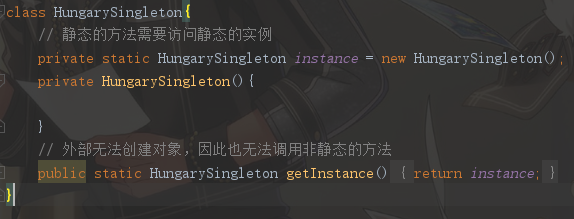
1. 创建型模式

①单例模式

涉及到一个单一的类，该类负责创建自己的对象，同时确保只有单个对象被创建。该类提供了一种访问其唯一对象的方式，不需要实例化该类。

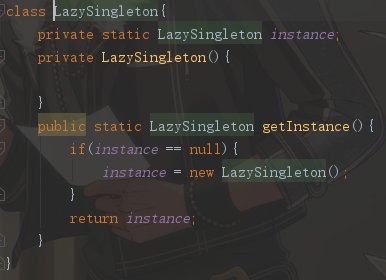
1. 饿汉式

在类初始化时就实例化。

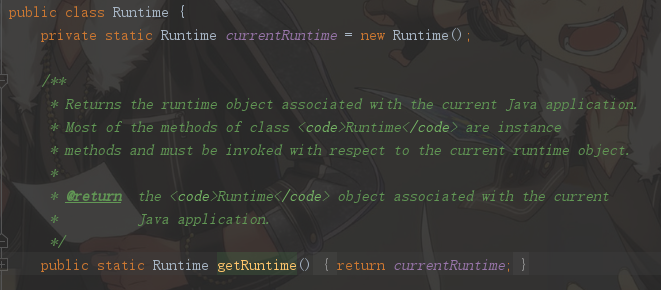


1. 懒汉式

在首次使用该对象时才进行实例化。



1. 具体实现：JDK Runtime



②工厂模式

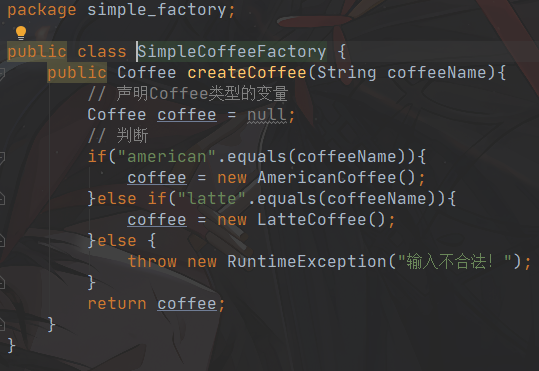
如果创建对象时直接new一个对象，就会对该对象耦合严重，如果要更换对象，所有new对象的地方都需要修改一遍。

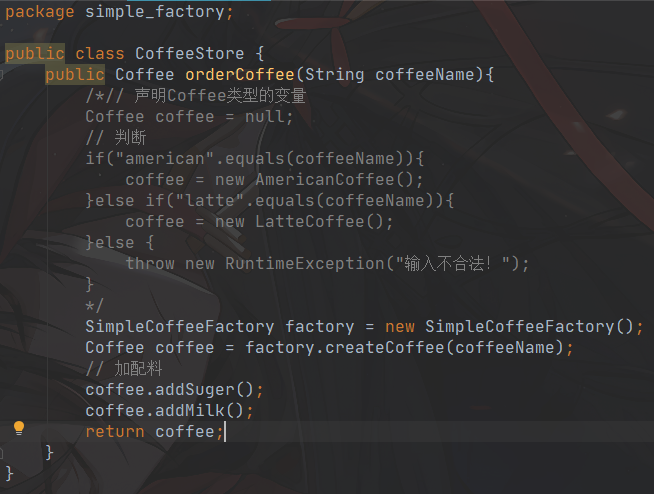
如果用工厂来生产对象，当需要更换对象时，只需要在工厂里进行修改。

最大的优点：解耦。

1. 简单工厂

组成：抽象产品（定义产品的规范），具体产品（抽象产品的实现或者子类），具体工厂（获取产品）





现在CoffeeStore不生产具体的咖啡，直接使用简单工厂生产的产品，解除了依赖。

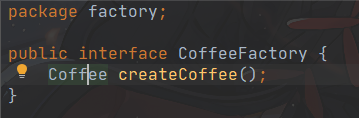
把对象的创建和业务逻辑层分离，避免了修改客户代码（Coffee Store）。

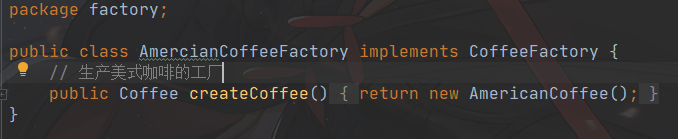
缺点：仍然不满足开放-封闭原则，当需要添加一种新的产品时，需要修改简单工厂类。

1. 工厂模式

定义一个用于创建对象的接口，让子类决定实例化哪个产品对象。

组成：抽象产品，具体产品，**抽象工厂**（创建对象的接口），具体工厂（生产具体产品）。





满足开放-封闭原则：当需要生产新的产品时，只需要创建一个新的具体工厂，实现抽象工厂接口即可，不需要修改工厂类。

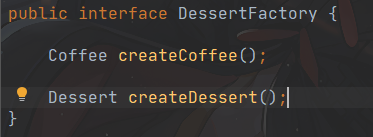
缺点：每增加一个新的产品都必须增加一个新的具体工厂和一个新的具体产品，增加了系统的复杂度。

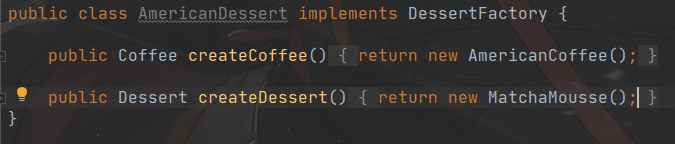
1. 抽象工厂模式

抽象工厂模式考虑多等级产品的生产（甜品店不止卖不同种类的咖啡，还卖各种甜品）。

产品族：同一个具体工厂生产的，位于不同等级（不同种类）的一组产品。

组成：抽象产品，具体产品，抽象工厂（包含多个创建产品的方法，可以生产同族但不同等级的产品），具体工厂（实现抽象工厂的多个方法）。







优点：只使用同一个产品族里的对象。

缺点：当产品族中需要新增一个产品时，需要修改所有的工厂类。

适用场景：需要创建的对象是一系列相互关联或相互依赖的产品族，每次只使用其中的某一族的产品。

③原型模式

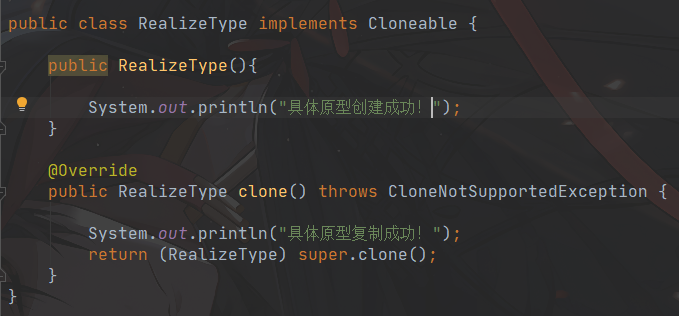
用一个已经创建的实例作为原型，通过复制该原型对象来创建一个和原型对象相同的新对象。

组成：抽象原型类（规定了clone方法），具体原型类（实现抽象原型类），访问类（使用具体原型类的clone方法复制该原型对象）。

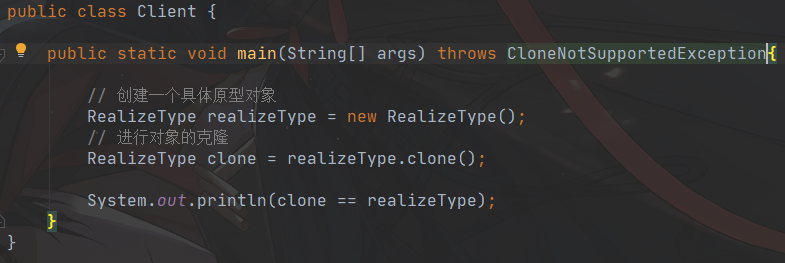
克隆方法：

（1）浅克隆：非基本类型属性（如引用），仍然指向原有属性所指向对象的内存地址；

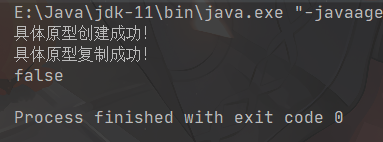
（2）深克隆：属性中引用的其它对象也会被克隆。



注意：以Cloneable作为抽象原型类。



输出结果：



第一个具体原型对象是通过构造函数创建的，第二个具体原型对象是通过克隆创建的，两个对象不是同一个对象。

适用场景：

1. 对象的创建非常复杂；
2. 性能和安全要求比较高。

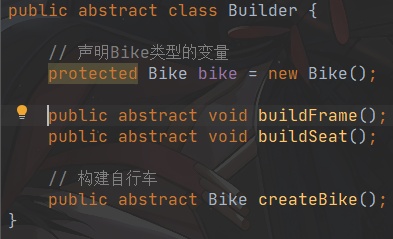
④建造者模式

将一个复杂对象的构建和表示分离，使得同样的构建过程可以创建不同的表示。

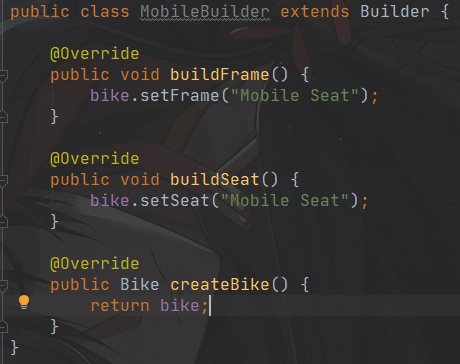
用户只需要指定复杂对象的类型，无须知道其内部的具体构造细节。

组成：产品类（复杂对象），抽象建造者类（规定实现复杂对象哪些部分的创建），具体建造者类（完成复杂对象各个部分的具体创建），指挥者类（调用具体的建造者来创建复杂对象的各个部分，装配）。

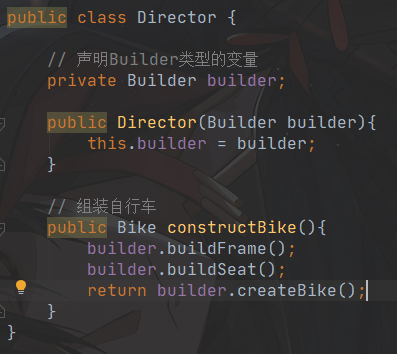
抽象建造者类：



具体建造者类：



指挥者类：



优点：

（1）封装较好。将主要的业务逻辑（装配的过程）封装在指挥者类中，稳定性较好。

（2）客户端无须知道产品内部组成的细节。

（3）可以更加精细地控制产品的创建过程。将复杂对象的创建步骤分解到不同的方法中。

（4）建造者模式很容易进行扩展。如果有新的需求，只需要添加一个新的具体建造者类即可。

缺点：

要求创建的产品具有较多的共同点。

适用场景：

1. 创建的对象较为复杂，由多个部件构成；
2. 创建复杂对象的算法独立于对象的组成部分以及装配方式。

⑤创建型模式对比

1. 工厂/抽象工厂模式：注重整体对象的创建方式
2. 建造者模型：部件构建/装配的过程。