1. 单例模式：一个实例，节省内存
   1. 一个类只能有一个实例，并提供一个访问它的全局访问点
   2. 懒汉模式：需要创建的时候再创建
   3. 饿汉模式:提前创建好
2. 工厂模式：解耦、扩展（重用）
   1. 简单工厂：就是一个if判断，将产品（输出、返回值、结果、目的）抽象
   2. 方法工厂：将产品（根据工厂，让哪个类执行就返回哪个类的返回值）和工厂（执行类，让哪个类执行）都抽象
   3. 抽象工厂：就是在方法工厂的基础上，一个工厂可以加工多个产品（将方法工厂更加细化）
   4. 应用场景：
      1. 当创建一个对象，只需要一个简单的方法就可以完成的时候
      2. 当类之间需要增加依赖关系的时候，工厂可以降低耦合度
      3. 当系统需要较好的扩展性时，工厂可以是实现不同的产品由不同的工厂来创建。
3. 建造者模式：解耦、扩展（重用）
   1. 将一个复杂对象的构建（组成这个对象各个部分的过程）和他的表示（返回结果）分离，抽象他的构建，从而可以有多种不同的表示。
   2. 应用场景：
      1. 当创建一个对象 需要很多步骤的时候（这个对象有很多的属性）
      2. 将这个对象的创建和使用分离开来
4. 原型模式：是一种创建型模式，用于创建重复的对象，提供了一种创建对象的最佳方式之一。 （就是复制一个对象，以及对象其中的属性方法）
   1. 例如，一个对象需要在一个高代价的数据库操作之后被创建。我们可以缓存该对象，再下一个请求的时候，返回他的克隆，需要的时候，更新数据库就可以。
5. 观察者模式：订阅发布（微信的订阅号，当订阅号有消息更新的时候，那么就把更新的消息推送给订阅了这个订阅号的用户）
   1. 订阅号和订阅者
6. 中介者模式：用来降低多个对象和类之间的通信复杂性，这种模式提供了一个中介类，由该类处理不同类之间的通信。
   1. 例如：QQ群的群，打扑克的QQ斗地主平台，等都是中介，用户是用户。
7. 备忘录模式：对某个类的状态进行保存下来，需要的时候，可以在备忘录中进行恢复。
   1. 例如：电脑中的撤销，备份，电话备忘录