1.说一下设计模式？你都知道哪些?

答：设计模式总共有 23 种，总体来说可以分为三大类：创建型模式（ Creational Patterns ）、结构型模式（ Structural Patterns ）和行为型模式（ Behavioral Patterns ）。

2.什么是单例模式?

答：单例模式是一种常用的软件设计模式，在应用这个模式时，单例对象的类必须保证只有一个实例存在，整个系统只能使用一个对象实例。

> 优点：不会频繁地创建和销毁对象，浪费系统资源。
>
> 使用场景：IO 、数据库连接、Redis 连接等。

单例模式又分为懒汉式和饿汉式:

1. 饿汉式会直接创建出类的实例化;
2. 懒汉式则在需要的时候再创建类的实例化

饿汉式线程安全,懒汉式线程不安全,如果要保证线程安全则需要使用锁,但加锁的代价较大,因此大多时候会用到另一种方式,即双检查锁,但双检查锁有可能会造成无序写入的的问题,这是由于编译器优化所导致的,可以使用volatile关键字,不让编译器对其优化即可;

3.什么是简单工厂模式?

答：简单工厂模式又叫静态工厂方法模式，就是建立一个工厂类，对实现了同一接口的一些类进行实例的创建。比如，一台咖啡机就可以理解为一个工厂模式，你只需要按下想喝的咖啡品类的按钮（摩卡或拿铁），它就会给你生产一杯相应的咖啡，你不需要管它内部的具体实现，只要告诉它你的需求即可。

> 优点:
>
> 1. 工厂类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例，客户端可以免除直接创建产品对象的责任，而仅仅"消费"产品;
> 2. 简单工厂模式通过这种做法实现了对责任的分割，它提供了专门的工厂类用于创建对象;
> 3. 客户端无须知道所创建的具体产品类的类名，只需要知道具体产品类所对应的参数即可，对于一些复杂的类名，通过简单工厂模式可以减少使用者的记忆量;
> 4. 通过引入配置文件，可以在不修改任何客户端代码的情况下更换和增加新的具体产品类，在一定程度上提高了系统的灵活性。

缺点:

> 1. 不易拓展，一旦添加新的产品类型，就不得不修改工厂的创建逻辑;
> 2. 产品类型较多时，工厂的创建逻辑可能过于复杂，一旦出错可能造成所有产品的创建失败，不利于系统的维护。

4.什么是抽象工厂模式?

答：抽象工厂模式是在简单工厂的基础上将未来可能需要修改的代码抽象出来，通过继承的方式让子类去做决定。

5.什么是观察者模式?

观察者模式是定义对象间的一种一对多依赖关系，使得每当一个对象状态发生改变时，其相关依赖对象皆得到通知并被自动更新。观察者模式又叫做发布-订阅（Publish/Subscribe）模式、模型-视图（Model/View）模式、源-监听器（Source/Listener）模式或从属者（Dependents）模式。

> 优点：
>
> 1. 观察者模式可以实现表示层和数据逻辑层的分离，并定义了稳定的消息更新传递机制，抽象了更新接口，使得可以有各种各样不同的表示层作为具体观察者角色；
> 2. 观察者模式在观察目标和观察者之间建立一个抽象的耦合；
> 3. 观察者模式支持广播通信；
> 4. 观察者模式符合开闭原则（对拓展开放，对修改关闭）的要求。

缺点：

> 1. 如果一个观察目标对象有很多直接和间接的观察者的话，将所有的观察者都通知到会花费很多时间；
> 2. 如果在观察者和观察目标之间有循环依赖的话，观察目标会触发它们之间进行循环调用，可能导致系统崩溃；
> 3. 观察者模式没有相应的机制让观察者知道所观察的目标对象是怎么发生变化的，而仅仅只是知道观察目标发生了变化。

6.什么是装饰器模式?

答：装饰器模式是指动态地给一个对象增加一些额外的功能，同时又不改变其结构。

> 优点：装饰类和被装饰类可以独立发展，不会相互耦合，装饰模式是继承的一个替代模式，装饰模式可以动态扩展一个实现类的功能。
>
> 装饰器模式的关键：装饰器中使用了被装饰的对象。

7.什么是模板方法模式?

答：模板方法模式是指定义一个模板结构，将具体内容延迟到子类去实现。

> 优点：
>
> 1. 提高代码复用性：将相同部分的代码放在抽象的父类中，而将不同的代码放入不同的子类中；

2. 实现了反向控制：通过一个父类调用其子类的操作，通过对子类的具体实现扩展不同的行为，实现了反向控制并且符合开闭原则。

8.什么是代理模式?

代理模式是给某一个对象提供一个代理，并由代理对象控制对原对象的引用。

优点：

1. 代理模式能够协调调用者和被调用者，在一定程度上降低了系统的耦合度；
2. 可以灵活地隐藏被代理对象的部分功能和服务，也增加额外的功能和服务。

缺点：

1. 由于使用了代理模式，因此程序的性能没有直接调用性能高；
2. 使用代理模式提高了代码的复杂度。

9.什么是策略模式?

答：策略模式是指定义一系列算法，将每个算法都封装起来，并且使他们之间可以相互替换。

优点：遵循了开闭原则，扩展性良好。

缺点：随着策略的增加，对外暴露越来越多。

10.什么是适配器模式?

答：适配器模式是将一个类的接口变成客户端所期望的另一种接口，从而使原本因接口不匹配而无法一起工作的两个类能够在一起工作。

优点：

1. 可以让两个没有关联的类一起运行，起着中间转换的作用；
2. 灵活性好，不会破坏原有的系统。

缺点：

1. 过多地使用适配器，容易使代码结构混乱，如明明看到调用的是 A 接口，内部调用的却是 B 接口的实现。