设计模式：

设计模式的分类：

总体来说设计模式分为三大类：

创建型模式：

工厂方法模式

抽象工厂模式

单例模式

建造者模式

原型模式

结构型模式：

适配器模式

装饰器模式

代理模式

外观模式

桥接模式

组合模式

享元模式

行为模式：

策略模式

模板方法模式

观察者模式

迭代子模式

责任链模式

命令模式

备忘录模式

状态模式

访问者模式

中介者模式

解释器模式

设计模式的六大原则

总原则：开闭原则

开闭原则就是说**对修改关闭对扩展开放**。在程序需要扩展的时候，不能去修改原有的代码，而是扩展原有的代码，实现一个热插拔的效果。所以一句话概括就是：为了使程序的扩展性好，易于维护和升级。想要达到这样的效果，我们需要使用接口和抽象类等，后面的具体设计中我们会提到这点。

1. 单一职责原则

不要存在多于一个导致类变更的原因，也就是说每个类应该实现单一的职责，如果不然就应该把类拆分。

1. 里氏替换原则

里氏替换原则是面向对象设计的基础原则之一。里氏代换原则中说，任何基类可以出现的地方，子类一定可以出现。LSP是继承复用的基石，只有当衍生类可以替换掉基类，软件单位的功能不受影响时，基类才能真正被复用，而衍生类也能够在基类的基础上增加新的行为。里氏替换原则是对于开闭原则的补充。实现开闭原则的关键步骤就是抽象化，而基类与子类的继承关系就是抽象化的具体实现，所以里氏替换原则是对实现抽象化的具体步骤的规范。

里氏替换原则中，子类对父类的方法尽量不要重写和重载，因为父类代表了定义好的结构，通过这个规范的接口与外界交互，子类不应该随意破坏它。

1. 依赖倒转原则

这个是开闭原则的基础，具体内容：面向接口编程，依赖于抽象而不依赖于具体。写代码时用到具体类时，不与具体类交互，而与具体类的上层接口交互。

1. 接口隔离原则

这个原则的意思是：每个接口中不存在子类用不到却必须实现的方法，如果不然，就要拆分接口。使用多个隔离的接口，比使用单个接口要好。

1. 迪米特原则（最少知道原则）

就是说：一个类对自己依赖的类知道的越少越好。也就是说无论被依赖的类多么的复杂，都应该将逻辑封装在方法的内部，通过public方法提供给外部，这样被依赖的类变化时，才能最小的影响该类。

1. 合成复用原则

原则是尽量首先使用合成/聚合的方式，而不是使用继承。

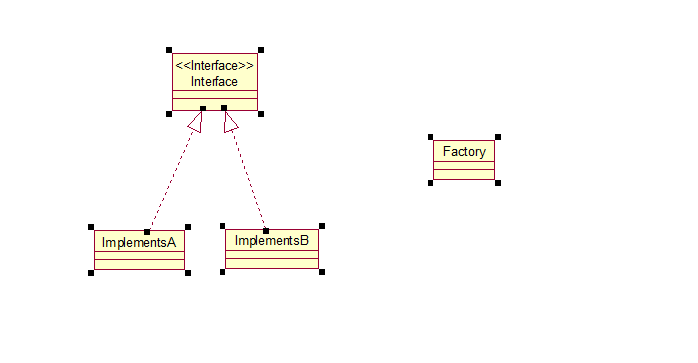
JAVA的23中设计模式

创建模式：

简单工厂模式：

* 普通

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。**

****

工厂类就是用来创建ImplementsA和ImplementsB的实例的。