设计模式分为三大类：

**创建型模式**，共五种：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**，共七种：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。

**行为型模式**，共十一种：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。

其实还有两类：并发型模式和线程池模式。

设计模式的六大原则：

**总原则－开闭原则**

对扩展开放，对修改封闭。在程序需要进行拓展的时候，不能去修改原有的代码，而是要扩展原有代码，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，否则就应该把类拆分。

**2、里氏替换原则（Liskov Substitution Principle）**

任何基类可以出现的地方，子类一定可以出现。里氏替换原则是继承复用的基石，只有当衍生类可以替换基类，软件单位的功能不受到影响时，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 氏代换原则是对“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，所以里氏代换原则是对实现抽 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 便破坏它。

**3、依赖倒转原则（Dependence Inversion Principle）**

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。

**4、接口隔离原则（Interface Segregation Principle）**

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，就要将接口拆分。使用多个隔离的接口，比使用单个接口（多个接口方法集合到一个的接口）要好。

**5、迪米特法则（最少知道原则）（Demeter Principle）**

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

最 少知道原则的另一个表达方式是：只与直接的朋友通信。类之间只要有耦合关系，就叫朋友关系。耦合分为依赖、关联、聚合、组合等。我们称出现为成员变量、方 法参数、方法返回值中的类为直接朋友。局部变量、临时变量则不是直接的朋友。我们要求陌生的类不要作为局部变量出现在类中。

**6、合成复用原则（Composite Reuse Principle）**

尽量首先使用合成/聚合的方式，而不是使用继承。

之前已经陆续整理了9种设计模式，链接如下，接下来一段时间陆续把剩余的过一遍，整理出来，理解设计模式还是很重要的。

创建型模式：[工厂方法模式](http://blog.csdn.net/jason0539/article/details/23020989)、[抽象工厂模式](http://blog.csdn.net/jason0539/article/details/44976775)、[单例模式](http://blog.csdn.net/jason0539/article/details/23297037)、[建造者模式](http://blog.csdn.net/jason0539/article/details/44992733)、[原型模式](http://blog.csdn.net/jason0539/article/details/23158081)

结构型模式：[适配器模式](http://blog.csdn.net/jason0539/article/details/22468457)、[装饰者模式](http://blog.csdn.net/jason0539/article/details/22713711)、[代理模式](http://blog.csdn.net/jason0539/article/details/22974405)、[外观模式](http://blog.csdn.net/jason0539/article/details/22775311)、[桥接模式](http://blog.csdn.net/jason0539/article/details/22568865)、[组合模式](http://blog.csdn.net/jason0539/article/details/22642281)、[享元模式](http://blog.csdn.net/jason0539/article/details/22908915)

行为型模式：[策略模式](http://blog.csdn.net/jason0539/article/details/45007553)、[模板方法模式](http://blog.csdn.net/jason0539/article/details/45037535)、[观察者模式](http://blog.csdn.net/jason0539/article/details/45055233)、[迭代子模式](http://blog.csdn.net/jason0539/article/details/45070441)、[责任链模式](http://blog.csdn.net/jason0539/article/details/45091639)、[命令模式](http://blog.csdn.net/jason0539/article/details/45110355)、[备忘录模式](http://blog.csdn.net/jason0539/article/details/45126489)、[状态模式](http://blog.csdn.net/jason0539/article/details/45021055)、[访问者模式](http://blog.csdn.net/jason0539/article/details/45146271)、[中介者模式](http://blog.csdn.net/jason0539/article/details/45216585)、解释器模式

还有两类：并发型模式和线程池模式