设计模式总结:

模式的基本概念是增加一些抽象层，当人们对某事物进行抽象的时候，隔离特定的细节，使得变化的事物与不变的事物分离开。因此设计模式的目的就是封装变化。模式分为创建型、结构型和行为型。创建型用于怎样创建一个对象，通常包括隔离对象创建的细节，代码不依赖于对象是什么类型，因此在增加一种新的对象类型时不需要改变代码，包括单件模式。工厂模式和构建器模式。结构型影响对象之间的连接方式，确保系统的变化不需要改变对象间的连接。通常由工程项目限制条件来支配。包括代理模式和适配器模式。行为型在程序中处理具有特定操作类型的对象。包括命令模式、模板方法模式、状态模式、策略模式、职责链模式、观察者模式、多派遣模式和访问者模式。

一些保持代码简明的基本方法。信使将消息封装到一个对象中到处传递，而不是将消息的所有片段分开进行传递。收集参数的工作是从传递给它的函数中获取信息，当传递参数被传递给多个函数时使用。单件允许一个类有且仅有一个实例的方法，它的关键是防止客户程序员获得任何控制其对象生存期的权利，因此所有的构造函数为私有，并且防止编译器隐式生成任何构造函数，通过公有成员函数来提供对其对象的访问。命令模式可以消除代码间的耦合，它就是一个函数对象，通过将函数封装为对象就能够以参数的形式将其传递给其他函数或者对象，告诉它们在履行请求的过程中执行特定的操作，特点是允许向一个函数或者对象传递一个想要的动作。代理模式和状态模式都提供一个代理类。代理类派生自一个基类，由平行地派生自同一个基类的一个或多个类提供实际的实现。代理模式只有一个实现类，状态模式有多个实现。适配器模式接受一种类型并且提供一个对其他类型的接口，可以调整彼此的表达方式。模板方法模式的定义在基类中并且不能改动，它调用其他基类函数完成工作，但是客户程序员不必直接调用这些函数。策略模式可以用多个方法解决某个问题。职责链模式尝试采用一系列策略模式，在职责链中，一个函数调用自身，调用函数的一个实现，如此反复直至达到某个终止条件。工厂模式创建封装对象。构建器模式的目标是将对象的创建与它的表示法分开。观察者模式允许修改对象数量和更新发生的方式而不影响别的代码。