c++编程思想第十章：

1. 创建型（Creational）：用于怎样创建一个对象。通常包括隔离对象创建的细节，这样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。、包含单件（Singleton）模式、工厂（Factory）模式和构建器（Builder）模式。
2. 2）结构型（Structural）：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。结构型模式常常由工程项目限制条件来支配。包含代（Proxy）模式和适配器（Adapter）模式。
3. 行为型（Behavioral）：在程序中处理具有特定操作类型的对象。这些对象封装要执行的操作过程，比如解释一种语言、实践一个请求、遍历一个序列（如像在一个迭代器内）或者实现一个算法。包含命令（Command）模式、模板方法（Template Method）模式、状态（State）模式、策略（Strategy）模式、职责链（Chain ofResponsibility）模式、观察者（Observer）模式、多派遣（Multiple Dispatching）模式和访问者（Visitor）模式。

单件（Singleton）也许是最简单的设计模式，它是允许一个类有且仅有一个实例的方法。创建一个单件模式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。为了做到这一点，声明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。

命令模式就是一个函数对象：一个作为对象的函数。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。可以说，命令模式是携带行为信息的信使。命令模式的主要特点是允许向一个函数或者对象传递一个想要的动作。在这里， 可以动态创建新的行为，某些事情通常只能通过编写新的代码来完成。

代理模式的一般 用途，描述如下：1）远程代理（Remote proxy）。为不同地址空间的对象提供代理。通过某些远程对象技术实现。

2）虚拟代理（Virtual proxy）。根据需要提供一种“惰性初始化”方式来创建高代价的对象。

3）保护代理（Protection proxy）。当不愿意客户程序员拥有被代理对象的全部访问权限时，使用保护代理。

4）巧妙引用（Smart reference）。当访问被代理的对象时，增加额外的活动

适配器模式：

适配器（Adapter）模式接受一种类型并且提供一个对其他类型的接口。当给定一个库或者具有某一接口的一段代码，同时还给定另外一个库或者与前面那段代码的基本思想相同的一段代码而只是表达方式不一致时，适配器模式将十分有用。通过调整彼此的表达方式以适配彼此，将会迅速产生解决方法

模板方法模式：应用程序结构框架中的一个 基本的概念是模板方法（Template Method）模式，它隐藏在覆盖的下方，通过调用基类的不同函数来驱动程序运行。模板方法模式的一个重要特征是它的定义在基类中（有时作为一个私有成员函数）并且不能改动—模板方法模式就是“坚持相同的代码”。它调用其他基类函数（就是那些被覆盖的函数）以便完成其工作，但是客户程序员不必直接调用这些函数。

策略模式：在程序运行时，可以插入变化的代码。策略模式也加入了“语境”，它可以是一个代理类，这个类控制着对特定策略对象的选择和使用—就像状态模式一样。

工厂模式：添加新类型将导致问题的出现。这个问题的解决方法就是强制用一个通用的工厂（factory）来创建对象，而不允许将创建对象的代码散布于整个系统。如果程序中所有需要创建对象的代码都转到这个工厂执行，那么在增加新对象时所要做的全部工作就是只需修改工厂。这种设计是工厂方法（Factory Method）模式的一种变体。由于每个面向对象应用程序都需要创建对象，并且由于人们可能通过添加新类型来扩展应用程序，工厂模式可能是所有设计模式中最有用的模式之一。抽象工厂（Abstract Factory）模式使用若干工厂方法（Factory Method）模式。每个工厂方法模式创建一个不同类型的对象。当创建一个工厂对象时，要决定将如何使用由那个工厂创建的所有对象。

构建器模式:这种模式的功能就是它将部件组合成为一个完整产品的算法与部件本身分开，这样就允许通过一个共同接口的不同实现来为不同的产品提供不同的算法。

观察者模式:观察者模式中有两个“变化的事件”：正在进行观察的对象的数量和更新发生的方式。这就是说，观察者模式允许修改这二者而不影响周围的其他代码。