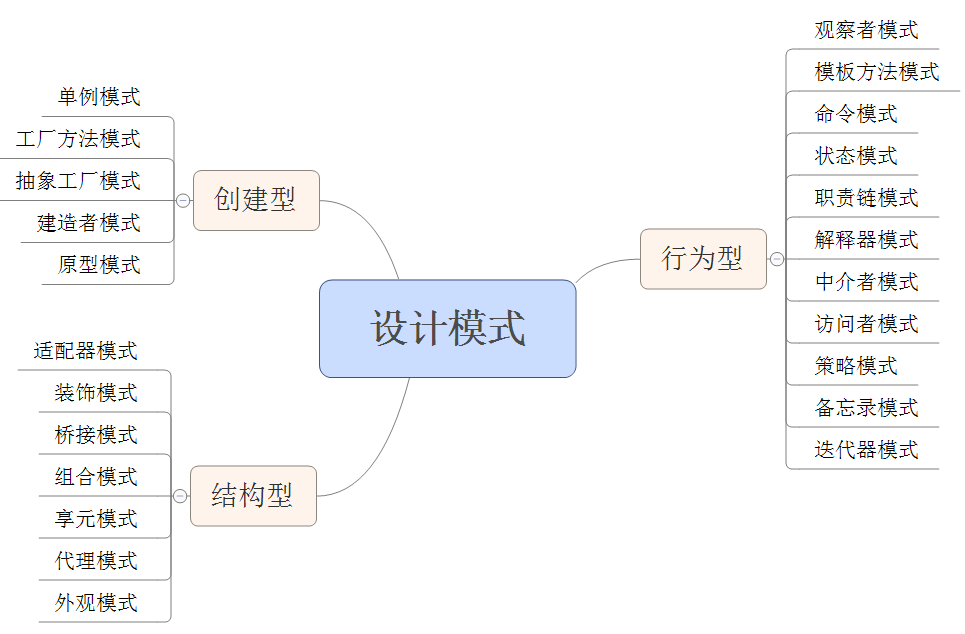
这段时间，学习状态比较一般，空闲时基本都在打游戏，和研究如何打好游戏，之后会有一篇文章介绍LOL英雄介绍（隐藏发布），玩笑了。为了和“学习”之间的友谊小船不翻，决定对以往学习过的GOF设计模式做一个简单的回顾，这儿俺只是“知识的搬运工”哈。程杰老师的《大话设计模式》一书非常的棒，虽然出版有小10年了，但知识点清晰易懂，尤其是最后的虚拟的设计模式选美大赛，非常精辟的对设计模式行进了剖析比较，便于实战中设计模式的选用和落地。



GOF模式总结：



1. 工厂方法：定义一个用于创建对象的接口，让子类决定实例化哪一个类，工厂方法使一个类的实例化延迟到子类。其是一切创建型模式的基础，其他均是在不同的场景下，对其进行的不同演化。



1. 抽象工厂：提供一个创建一系列或相关依赖对象的接口，而不需指定它们具体的类。看起来结构很复杂，但其结构是相似的，很好理解。



1. 建造者：将一个复杂对象的构件和它的表示分离，使得同样的构件过程可以创建不同的表示。



1. 原型：用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。在实际中，实现IClone接口的类就是原型模式的实现。



1. 单例：最常用的设计模式之一，服务类一般会考虑选用单例模式，保证一个类仅有一个实例，并提供一个访问它的全局访问点。



1. 适配器：将一个类的接口转换成客户希望的另一个接口，使得原来不兼容的接口可以一起工作，非常简单，常见的就是修改参数列表即可。



1. 桥接：将抽象部分和实现分离，使得它们可以独立地变化，通过使用聚合来代替集成来解耦。针对两个易变对象，为了使得它们的变化产生尽可能少的影响，通过他们的抽象进行交互，可以查考博主holly的博文<http://www.cnblogs.com/houleixx/archive/2008/02/23/1078877.html>。



1. 组合：将对象组合成树形结构以表示“部分-整体”的层次结构，组合模式使得用户对单个对象和组合对象的使用具有一致性，最常见的就是DOM树。



1. 装饰：动态地给一个对象添加一些额外的职责，就增加功能来说，装饰模式相比生成子类更加灵活。



1. 外观：为子系统中的一组接口提供一个一致的界面，外观模式定义了一个高层接口，这个接口使得这一子系统更加容易使用。



1. 享元：为运用共享技术有效地支持大量细粒度的对象。



1. 代理：为其他对象提供一种代理以控制对这个对象的访问。这儿补充一下，代理、外观和适配器模式的区别，代理对象代表一个单一对象而外观对象代表一个子系统，代理的客户对象无法直接访问目标对象，由代理提供对目标对象的访问控制，而外观通常是对子系统各元件的简化。而适配器不需要虚构一个代表，只需要提供特定方法即可。



1. 观察者(Observer)：定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有历来于它的对象都得到通知并被自动更新，.NET程序中的事件机制既是最常见示例。此外，常见的MVC架构模式，其实就是观察者、组合和策略模式的组合。



1. 模板方法：定义一个操作的算法骨架，而将一些步骤延迟到子类中，模板方法使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤。这个在开发中也经常用到，在有一些面向切面的应用场景时，如果不使用第三方动态代理库，就可以通过简单模板方法达到相同目的，比如共用异常处理和日志记录代码。



1. 命令：将一个请求封装为一个对象，从而使你可以用不同的请求对客户进行参数化；可以对请求排队或记录请求日志，以及支持可撤销的操作。



1. 状态：允许一个对象在其内部状态改变时改变它的行为，让对象看起来修改它的类，可以使常见的If-Else分支结构的可读性和可维护性变强。



1. 职责链：使多个对象都有机会处理请求，从而避免请求的发送者和接收者之间的耦合关系，将这些对象连成一条链，并沿着这条链传递请求，直到有一个对象处理为止，类似现实中的审批流程，层层提交，知道有权限处理的人为止，常见的MVC过滤器就是该模式的具体实现。



1. 解释器：给定一个语言，定义它文法的一种表示，并定义一个解释器。其使用场景是，如果一个特定类型的问题发生的频率足够高，那么可能就值得将该问题的各个实例表述为一个简单语言中的句子，这样就可以构建一个解释器，其通过解释这些句子来解决问题。比如当一个语言需要解释执行，而且你可将该语言中的句子表示为一个抽象语法树时，可以使用解释器模式。



优点：容易通过继承来改变和扩展文法，也比较容易实现文法，因为定义抽象语法树中各个节点的类的实现大体相似。

缺点；由于需要为每一条规则至少定义一个类，因此当规则多复杂时，难以维护和管理，推荐使用语法分析程序或编译器生成器来处理。

1. 中介者：用一个中介对象来封装一系列的对象交互，中介者使各对象不许显式的互相引用，从而达到解耦的目的，符合最小知识原则。



1. 访问者模式：表示一个作用于某对象结构中的各元素的操作，它使你可以在不改变各元素的前提下定义作用于这些元素的新操作，其是一个双分配的结构，比较复杂，需要多练习。



优点：它将数据结构和作用在其上的操作解耦开来，使得操作集合可以相对自由的演化。当ObjectStructure稳定时，添加新的操作变得非常简单，增加一个新的访问者即可。

缺点：使增加新的数据结构变得复杂，同时增加了系统的复杂程度。

1. 策略：定义一系列的算法，把他们封装起来，并且使得他们可以相互替换，独立于使用它的客户而变化。



1. 备忘录：在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态，这样以后就可以将该对象恢复到原先保存的状态。



1. 迭代器：提供一种方法顺序访问一个聚合对象中的各个元素，而不需要暴露其内部表示，.NET迭代器就是其实现示例。



Tip：

补充事件和委托的辨析，委托是对函数的封装，可以当做给方法的特征指定的一个名称，而事件是委托的一种特殊形式，当发生有意义的事情时，事件对象处理通知过程。此外，委托是一种引用方法的类型，一旦为委托分配了方法，委托将与海方法具有完全相同的行为，而事件是在发生其他类或对象关注的事情时，类或对象可通过事件通知他们。

参考资料

1. 程杰. 大话设计模式[M]. 北京:清华大学出版社, 2007.
2. [美], Erich, Gamma, [美], Richard, Helm, [美], Ralph, Johnson. 设计模式：可复用面向对象软件的基础[M]. 北京:机械工业出版社, 2007.