# Design Patterns

|  |  |  |  |
| --- | --- | --- | --- |
| 模式名称 | 描述 | 是否在 [*《设计模式》*](http://zh.wikipedia.org/zh-cn/%E8%AE%BE%E8%AE%A1%E8%8C%83%E4%BE%8B) | 是否在 [*《代码大全》*](http://zh.wikipedia.org/w/index.php?title=%E4%BB%A3%E7%A0%81%E5%A4%A7%E5%85%A8&action=edit&redlink=1)[[1]](http://zh.wikipedia.org/zh-cn/%E8%BD%AF%E4%BB%B6%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F#_note-McConnell2004-0) |
| [**创建型模式**](http://zh.wikipedia.org/w/index.php?title=%E5%88%9B%E5%BB%BA%E5%9E%8B%E6%A8%A1%E5%BC%8F&action=edit&redlink=1) |  |  |  |
| [抽象工厂模式](http://zh.wikipedia.org/zh-cn/%E6%8A%BD%E8%B1%A1%E5%B7%A5%E5%8E%82%E6%A8%A1%E5%BC%8F)(Abstract factory) | 为一个产品族提供了统一的创建接口。当需要这个产品族的某一系列的时候，可以从抽象工厂中选出相应的系列创建一个具体的工厂类。 | 是 | 是 |
| [工厂方法模式](http://zh.wikipedia.org/zh-cn/%E5%B7%A5%E5%8E%82%E6%96%B9%E6%B3%95%E6%A8%A1%E5%BC%8F)  (Factory Method) | 定义一个接口用于创建对象，但是让子类决定初始化哪个类。工厂方法**把一个类的初始化下放到子类**。 | 是 | 是 |
| [生成器模式](http://zh.wikipedia.org/zh-cn/%E7%94%9F%E6%88%90%E5%99%A8%E6%A8%A1%E5%BC%8F)  (Builder) | 将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示 | 是 | 否 |
| [懒惰初始化模式](http://zh.wikipedia.org/w/index.php?title=%E6%87%92%E6%83%B0%E5%88%9D%E5%A7%8B%E5%8C%96%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Lazy Initialization) | 推迟对象的创建、数据的计算等需要耗费较多资源的操作，只有在第一次访问的时候才执行。 | 否 | 否 |
| [对象池模式](http://zh.wikipedia.org/w/index.php?title=%E5%AF%B9%E8%B1%A1%E6%B1%A0%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Object Pool) | 通过回收利用对象避免获取和释放资源所需的昂贵成本。 An object pool is a set of initialised [objects](http://en.wikipedia.org/wiki/Object_%28computer_science%29) that are kept ready to use, rather than allocated and destroyed on demand. ThreadPool. | 否 | 否 |
| [原型模式](http://zh.wikipedia.org/zh-cn/%E5%8E%9F%E5%9E%8B%E6%A8%A1%E5%BC%8F)  (Prototype) | 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。原型模式多用于创建复杂的或者耗时的实例, 或者创建值相等，只是命名不一样的同类数据. | 是 | 否 |
| [单例模式](http://zh.wikipedia.org/zh-cn/%E5%8D%95%E4%BE%8B%E6%A8%A1%E5%BC%8F)  (Singleton) | 确保一个类只有一个实例，并提供对该实例的全局访问。 | 是 | 是 |
| [**结构型模式**](http://zh.wikipedia.org/w/index.php?title=%E7%BB%93%E6%9E%84%E5%9E%8B%E6%A8%A1%E5%BC%8F&action=edit&redlink=1) |  |  |  |
| [适配器模式](http://zh.wikipedia.org/zh-cn/%E9%80%82%E9%85%8D%E5%99%A8%E6%A8%A1%E5%BC%8F)  (Adaptor) | 将某个类的接口转换成客户端期望的另一个接口表示。适配器模式可以消除由于接口不匹配所造成的类兼容性问题。(类适配器通过多继承,对象适配器通过包含一个被适配对象) | 是 | 是 |
| [桥接模式](http://zh.wikipedia.org/zh-cn/%E6%A9%8B%E6%8E%A5%E6%A8%A1%E5%BC%8F)  (Bridge) | 将一个抽象与实现解耦，以便两者可以独立的变化。把事物对象和其具体行为、具体特征分离开来，使它们可以各自独立的变化。(IDrawingAPI, DrawingAPIImpl1,  DrawingAPIImpl2) Runtime Injection? | 是 | 是 |
| [组合模式](http://zh.wikipedia.org/w/index.php?title=%E7%BB%84%E5%90%88%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Composite) | 把多个对象组成树状结构来表示局部与整体，这样用户可以一样的对待单个对象和对象的组合。The intent of composite is to "compose" objects into tree structures to represent part-whole hierarchies. | 是 | 是 |
| [修饰模式](http://zh.wikipedia.org/zh-cn/%E4%BF%AE%E9%A5%B0%E6%A8%A1%E5%BC%8F)  Decorator | 向某个对象动态地添加更多的功能。修饰模式是除类继承外另一种扩展功能的方法。(在内部包含一个需要扩展的对象; 通常和内部对象拥有相同的接口) | 是 | 是 |
| [外观模式](http://zh.wikipedia.org/zh-cn/%E5%A4%96%E8%A7%80%E6%A8%A1%E5%BC%8F)  (Facade) | 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。 | 是 | 是 |
| [享元模式](http://zh.wikipedia.org/w/index.php?title=%E4%BA%AB%E5%85%83%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (FlyWeight) | 通过共享以便有效的支持大量小颗粒对象。小颗粒对象不能对使用它的环境(外部状态)做作出假设, 必要时可以通过参数的方式把环境传递给小颗粒对象. | 是 | 否 |
| [代理模式](http://zh.wikipedia.org/w/index.php?title=%E4%BB%A3%E7%90%86%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Proxy) | 为其他对象提供一个代理以控制对这个对象的访问, 比如需要时才创建/访问实际数据(比如硬盘上的图像或者远程的资源), 只访问需要用到的数据. 远程代理,保护代理, 虚代理(按需访问), 智能指引(如引用计数). | 是 | 否 |
| [**行为模式**](http://zh.wikipedia.org/w/index.php?title=%E8%A1%8C%E4%B8%BA%E6%A8%A1%E5%BC%8F&action=edit&redlink=1) |  |  |  |
| [责任链模式](http://zh.wikipedia.org/zh-cn/%E8%B4%A3%E4%BB%BB%E9%93%BE%E6%A8%A1%E5%BC%8F)  (Chain Of Responsibility) | 为解除请求的发送者和接收者之间耦合，而使多个对象都有机会处理这个请求。将这些对象连成一条链，并沿着这条链传递该请求，直到有一个对象处理它。类似于delegate. | 是 | 否 |
| [命令模式](http://zh.wikipedia.org/zh-cn/%E5%91%BD%E4%BB%A4%E6%A8%A1%E5%BC%8F)  (Command) | 将一个请求封装为一个对象，从而使你可用不同的请求对客户进行参数化；对请求排队或记录请求日志，以及支持可取消的操作。  Command需要保存内部状态。  它尝试以物件来代表实际行动。命令物件可以把行动(action) 及其参数封装起来，于是这些行动可以被：   * 重复多次 Execute * 取消（如果该物件有[实作](http://zh.wikipedia.org/w/index.php?title=%E5%AF%A6%E4%BD%9C&action=edit&redlink=1)的话） Undo * 取消后又再重做 Redo   WPF Command; Marcro Command.Command 常使用回调（调用某个对象的方法）。 | 是 | 否 |
| [解释器模式](http://zh.wikipedia.org/w/index.php?title=%E8%A7%A3%E9%87%8A%E5%99%A8%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Intepreter) | 给定一个语言, 定义它的文法的一种表示，并定义一个解释器, 该解释器使用该表示来解释语言中的句子。解释器为文法中每一条规则至少定义一个类(e.g. XXXXExpression which derives from Abstract expression and has a method of Interpret). XXXXExpression can contain other expressions. | 是 | 否 |
| [迭代器模式](http://zh.wikipedia.org/w/index.php?title=%E8%BF%AD%E4%BB%A3%E5%99%A8%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Iterator) | 提供一种方法顺序访问一个聚合对象中各个元素, 而又不需暴露该对象的内部表示。 | 是 | 是 |
| [Mediator](http://zh.wikipedia.org/w/index.php?title=Mediator_pattern&action=edit&redlink=1) | Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently. | 是 | 否 |
| [备忘录模式](http://zh.wikipedia.org/w/index.php?title=%E5%A4%87%E5%BF%98%E5%BD%95%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)  (Momento) | Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later. | 是 | 否 |
| [观察者模式](http://zh.wikipedia.org/zh-cn/%E8%A7%82%E5%AF%9F%E8%80%85%E6%A8%A1%E5%BC%8F)  (Observer) | 在对象间定义一个一对多的联系性，由此当一个对象改变了状态，所有其他相关的对象会被通知并且自动刷新。Subject-Observer.  注意**意外更新**，特别是在WinForm编程中。 | 是 | 是 |
| [状态模式](http://zh.wikipedia.org/w/index.php?title=%E7%8A%B6%E6%80%81%E6%A8%A1%E5%BC%8F&action=edit&redlink=1) | Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.  外部context类和用户交互，并且保有state类，将用户请求发送给state。所有state类根据请求作出反应，必要时设置context的state到另一个state. 比如TCPEstablishedState::Close方法就会设置TcpContext的state到TCPCloseState. | 是 | 否 |
| [策略模式](http://zh.wikipedia.org/zh-cn/%E7%AD%96%E7%95%A5%E6%A8%A1%E5%BC%8F) | 定义一个算法的系列，将其各个分装，并且使他们有交互性。策略模式使得算法在用户使用的时候能独立的改变。  使用运行时绑定，可以动态替换算法。  在一定程度上可以替代继承，更加灵活。 | 是 | 是 |
| [Specification](http://zh.wikipedia.org/w/index.php?title=Specification_pattern&action=edit&redlink=1) | Recombinable Business logic in a boolean fashion | 否 | 否 |
| [模板方法模式](http://zh.wikipedia.org/zh-cn/%E6%A8%A1%E6%9D%BF%E6%96%B9%E6%B3%95%E6%A8%A1%E5%BC%8F) | Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure. 在继承中非常常见。 | 是 | 是 |
| [访问者模式](http://zh.wikipedia.org/zh-cn/%E8%AE%BF%E9%97%AE%E8%80%85%E6%A8%A1%E5%BC%8F) | Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the  elements on which it operates.  在被访问类中，有accept方法调用VisitXXX方法；增加新的被访问者就得增加visitor的方法。  Optimize the implementation of a visitor that is allocated, used only once, and then deleted by **Single-Serving Visistor**  使用Enter, Leave方法来实现**Hierarchical Visitor**, 保持树形结构。 | 是 | 否 |
| [**并发模式**](http://zh.wikipedia.org/w/index.php?title=%E5%B9%B6%E5%8F%91%E6%A8%A1%E5%BC%8F&action=edit&redlink=1) |  |  |  |
| [Active Object](http://zh.wikipedia.org/w/index.php?title=Active_Object&action=edit&redlink=1) | Decouples method execution from method invocation that reside in their own [thread](http://en.wikipedia.org/wiki/Thread_%28computing%29) of control.  Active objects run in their own thread(often single thread and in sequence to avoid conflicts) instead of the thread of the caller. There will be a queue to hold the requests and the requests get handled one by one. After handled, the caller thread gets notified by using signals or callbacks. | 否 | 否 |
| [Double checked locking](http://zh.wikipedia.org/w/index.php?title=Double_checked_locking_pattern&action=edit&redlink=1) | Reduce the overhead of acquiring a [lock](http://en.wikipedia.org/wiki/Lock_%28computer_science%29) by first testing the locking criterion (the "lock hint") in an unsafe manner; only if that succeeds does the actual lock proceed. | 否 | 否 |
| [Guarded Suspension](http://zh.wikipedia.org/w/index.php?title=Guarded_suspension&action=edit&redlink=1) | Managing operations that require both a [lock](http://en.wikipedia.org/wiki/Lock_%28computer_science%29) to be acquired and a [precondition](http://en.wikipedia.org/wiki/Precondition) to be satisfied before the operation can be executed.  wait in the guarded method and the guard notify the wait in another method when the condition is met. | 否 | 否 |
| [Leaders/followers](http://zh.wikipedia.org/w/index.php?title=Leaders_followers_pattern&action=edit&redlink=1) | 领导者/追随者模式用于管理许多飞机场出租车候车台。在该用例中，出租车扮演“线程”角色，排在第一辆的出  租车成为领导者，剩下的出租车成为追随者。同样，到达出租车候车台的乘客构成了必须被多路分解给出租车的事件，一般以先进  先出排序。一般来说，如果任何出租车可以为任何顾客服务，该场景就主要相当于非绑定句柄/线程关联 | 否 | 否 |
| [Monitor object](http://zh.wikipedia.org/w/index.php?title=Monitor_(synchronization)&action=edit&redlink=1) | a **monitor** is an [object](http://en.wikipedia.org/wiki/Object_%28computer_science%29) intended to be used safely by more than one [thread](http://en.wikipedia.org/wiki/Thread_%28computer_science%29). Its methods are executed with [mutual exclusion](http://en.wikipedia.org/wiki/Mutual_exclusion). That is, at each point in time, at most one [thread](http://en.wikipedia.org/wiki/Thread_%28computer_science%29) may be executing any of its [methods](http://en.wikipedia.org/wiki/Method_%28computer_science%29). Like an Account class in bank system.  C#: Monitor class; Locker on an whole object. | 否 | 否 |
| [Read write lock](http://zh.wikipedia.org/w/index.php?title=Read_write_lock_pattern&action=edit&redlink=1) | allows [concurrent](http://en.wikipedia.org/wiki/Concurrency_control) read access to an [object](http://en.wikipedia.org/wiki/Object_%28computer_science%29) but requires exclusive access for write operations.  System.Threading.ReaderWriterLockSlim. The VDC Cache Service can use this lock. EnterReadLocker, TryEnterWriterLocker.. | 否 | 否 |
| [线程池模式](http://zh.wikipedia.org/w/index.php?title=%E7%BA%BF%E7%A8%8B%E6%B1%A0%E6%A8%A1%E5%BC%8F&action=edit&redlink=1) | a number of [threads](http://en.wikipedia.org/wiki/Thread_%28computer_science%29) are created to perform a number of tasks(delegates), which are usually organized in a [queue](http://en.wikipedia.org/wiki/Queue_%28data_structure%29). Typically, there are many more tasks than threads. As soon as a thread completes its task, it will request the next task from the queue until all tasks have been completed. | 否 | 否 |
| [Thread-specific storage](http://zh.wikipedia.org/w/index.php?title=Thread-Specific_Storage&action=edit&redlink=1) | Uses [static](http://en.wikipedia.org/wiki/Static) or global [memory](http://en.wikipedia.org/wiki/Computer_storage) local to a [thread](http://en.wikipedia.org/wiki/Thread_%28computer_science%29).  class FooBar  {  [ThreadStatic] static int foo;  }  It is also used in OpenMP. | 否 | 否 |
| [Reactor](http://zh.wikipedia.org/w/index.php?title=Reactor_pattern&action=edit&redlink=1) | Handle service requests delivered [concurrently](http://en.wikipedia.org/wiki/Concurrency_%28computer_science%29) to a service handler(demultiplexer) by one or more inputs(e.g. IO). The service handler(demultiplexer) then demultiplexes the incoming requests and dispatches them synchronously to the associated request handlers(concrete handlers). | 否 | 否 |

## [**生成器模式**](http://zh.wikipedia.org/zh-cn/%E7%94%9F%E6%88%90%E5%99%A8%E6%A8%A1%E5%BC%8F)**(Builder)**

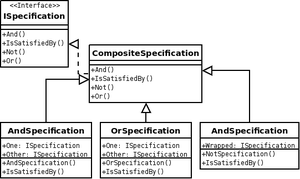
将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。

客户创建Director对象(负责构建)，并用它所想要的Builder对象进行配置。

* 一旦产品部件被生成，导向器就会通知生成器。
* 生成器处理导向器的请求，并将部件添加到该产品中(负责生成表示)。
* 客户从生成器中检索产品。

|  |
| --- |
| [**Specification**](http://zh.wikipedia.org/w/index.php?title=Specification_pattern&action=edit&redlink=1) |

business logic can be recombined by chaining the business logic together using boolean logic.



从CompositeSpecification继承，拥有And, Or, Not等逻辑操错。缺省实现And, Or, NotSpecification.

具体的business logic采用实现检查business的logic来实现. 用**IsSatisfied**来进行条件判断。

OverDueSpecification OverDue = new OverDueSpecification();

NoticeSentSpecification NoticeSent = new NoticeSentSpecification();

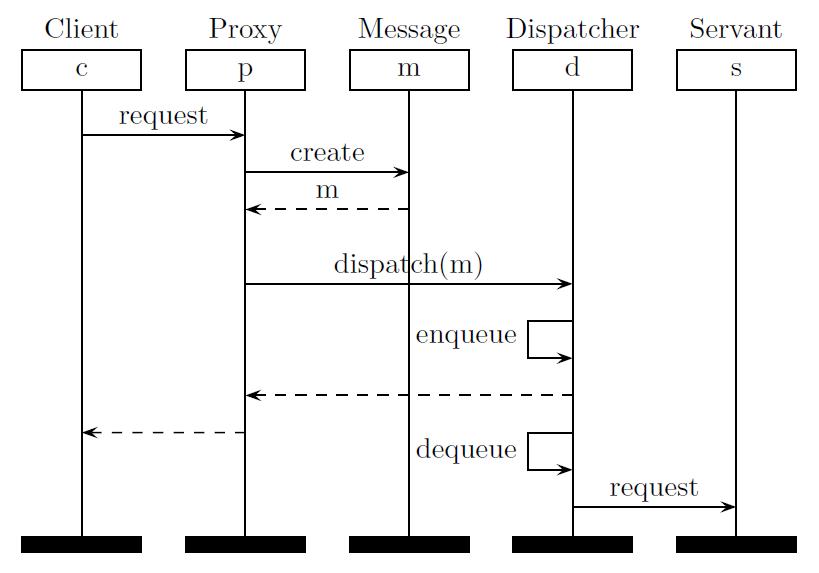
ISpecification SendToCollection = OverDue.And(NoticeSent).And(InCollection.Not());

其本质是将用And, Or, NotSpecification来连接需要的business logic.

## [**Active Object**](http://zh.wikipedia.org/w/index.php?title=Active_Object&action=edit&redlink=1)

Decouples method execution from method invocation that reside in their own [thread](http://en.wikipedia.org/wiki/Thread_%28computing%29) of control.

Active objects run in their own thread(often single thread and in sequence to avoid conflicts) instead of the thread of the caller. There will be a queue to hold the requests and the requests get handled one by one. After handled, the caller thread gets notified by using signals or callbacks.



## [**Double checked locking**](http://zh.wikipedia.org/w/index.php?title=Double_checked_locking_pattern&action=edit&redlink=1)

1. Check that the variable is initialized (without obtaining the lock). If it is initialized, return it immediately.
2. Obtain the lock.
3. Double-check whether the variable has already been initialized: if another thread acquired the lock first, it may have already done the initialization. If so, return the initialized variable.
4. Otherwise, initialize and return the variable.

if (xxx == null) //first check

{

    lock(this) //lock

  {

      if (xxx== null) //second check.

            xxx = new xxxClass();

  }

 return xxx;

}