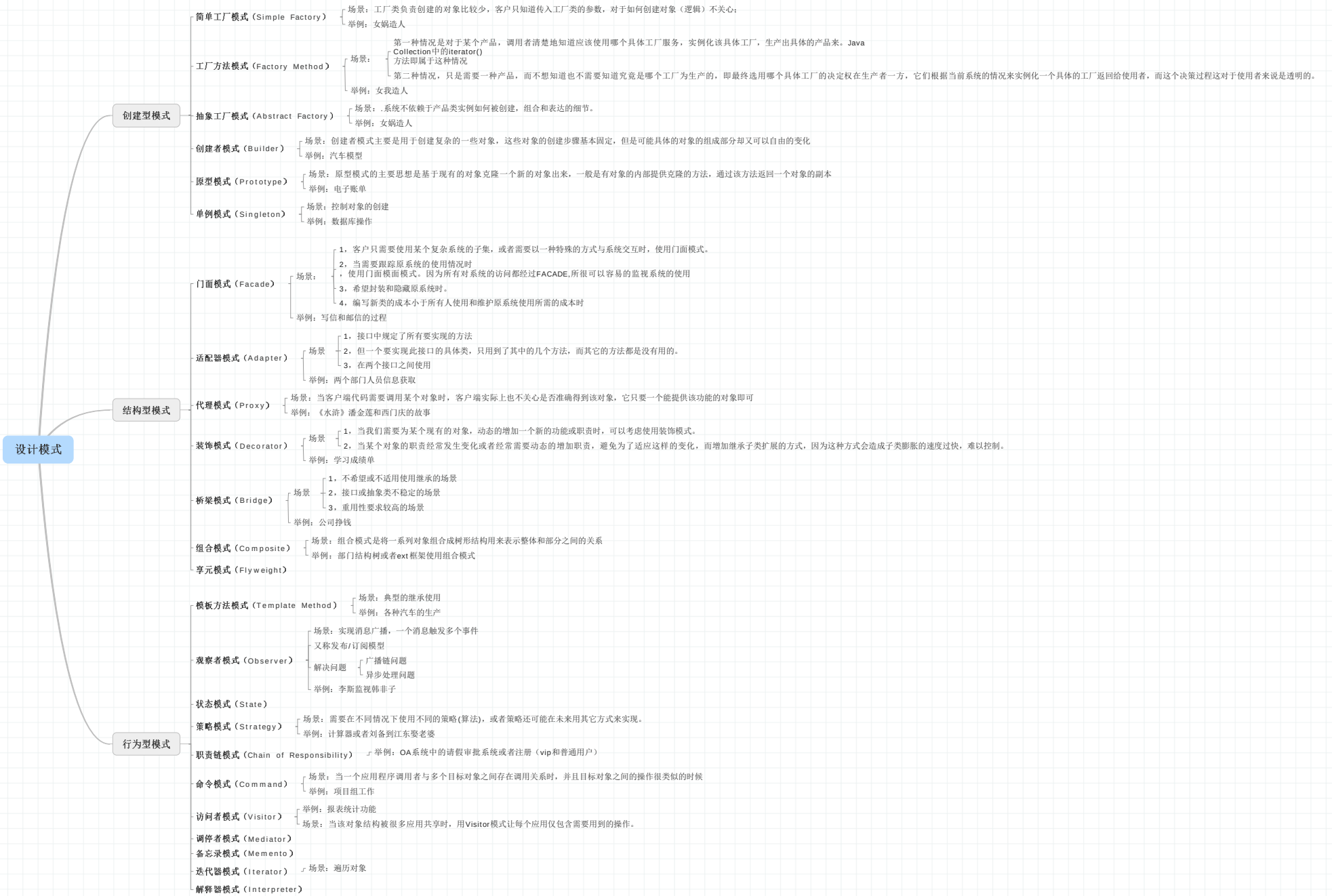
设计模式
创建型模式
简单工厂模式（Simple Factory）
场景：工厂类负责创建的对象比较少，客户只知道传入工厂类的参数，对于如何创建对象（逻辑）不关心；
举例：女娲造人
工厂方法模式（Factory Method）
场景：
第一种情况是对于某个产品，调用者清楚地知道应该使用哪个具体工厂服务，实例化该具体工厂，生产出具体的产品来。Java Collection中的iterator()
方法即属于这种情况
第二种情况，只是需要一种产品，而不想知道也不需要知道究竟是哪个工厂为生产的，即最终选用哪个具体工厂的决定权在生产者一方，它们根据当前系统的情况来实例化一个具体的工厂返回给使用者，而这个决策过程这对于使用者来说是透明的。
举例：女我造人
抽象工厂模式（Abstract Factory）
场景：.系统不依赖于产品类实例如何被创建，组合和表达的细节。
举例：女娲造人
创建者模式（Builder）
场景：创建者模式主要是用于创建复杂的一些对象，这些对象的创建步骤基本固定，但是可能具体的对象的组成部分却又可以自由的变化
举例：汽车模型
原型模式（Prototype）
场景：原型模式的主要思想是基于现有的对象克隆一个新的对象出来，一般是有对象的内部提供克隆的方法，通过该方法返回一个对象的副本
举例：电子账单
单例模式（Singleton）
场景：控制对象的创建
举例：数据库操作
结构型模式
门面模式（Facade）
场景：
1，客户只需要使用某个复杂系统的子集，或者需要以一种特殊的方式与系统交互时，使用门面模式。
2，当需要跟踪原系统的使用情况时
，使用门面模面模式。因为所有对系统的访问都经过FACADE,所很可以容易的监视系统的使用
3，希望封装和隐藏原系统时。
4，编写新类的成本小于所有人使用和维护原系统使用所需的成本时
举例：写信和邮信的过程
适配器模式（Adapter）
场景
1，接口中规定了所有要实现的方法
2，但一个要实现此接口的具体类，只用到了其中的几个方法，而其它的方法都是没有用的。
3，在两个接口之间使用
举例：两个部门人员信息获取
代理模式（Proxy）
场景：当客户端代码需要调用某个对象时，客户端实际上也不关心是否准确得到该对象，它只要一个能提供该功能的对象即可
举例：《水浒》潘金莲和西门庆的故事
装饰模式（Decorator）
场景
1，当我们需要为某个现有的对象，动态的增加一个新的功能或职责时，可以考虑使用装饰模式。
2，当某个对象的职责经常发生变化或者经常需要动态的增加职责，避免为了适应这样的变化，而增加继承子类扩展的方式，因为这种方式会造成子类膨胀的速度过快，难以控制。
举例：学习成绩单
桥梁模式（Bridge）
场景
1，不希望或不适用使用继承的场景
2，接口或抽象类不稳定的场景
3，重用性要求较高的场景
举例：公司挣钱
组合模式（Composite）
场景：组合模式是将一系列对象组合成树形结构用来表示整体和部分之间的关系
举例：部门结构树或者ext框架使用组合模式
享元模式（Flyweight）
行为型模式
模板方法模式（Template Method）
场景：典型的继承使用
举例：各种汽车的生产
观察者模式（Observer）
场景：实现消息广播，一个消息触发多个事件
又称发布/订阅模型
解决问题
广播链问题
异步处理问题
举例：李斯监视韩非子
状态模式（State）
策略模式（Strategy）
场景：需要在不同情况下使用不同的策略(算法)，或者策略还可能在未来用其它方式来实现。
举例：计算器或者刘备到江东娶老婆
职责链模式（Chain of Responsibility）
举例：OA系统中的请假审批系统或者注册（vip和普通用户）
命令模式（Command）
场景：当一个应用程序调用者与多个目标对象之间存在调用关系时，并且目标对象之间的操作很类似的时候
举例：项目组工作
访问者模式（Visitor）
举例：报表统计功能
场景：当该对象结构被很多应用共享时，用Visitor模式让每个应用仅包含需要用到的操作。
调停者模式（Mediator）
备忘录模式（Memento）
迭代器模式（Iterator）
场景：遍历对象
解释器模式（Interpreter）