设计模式完全手册

讲师：丁宋涛 夏曹俊

行为型模式

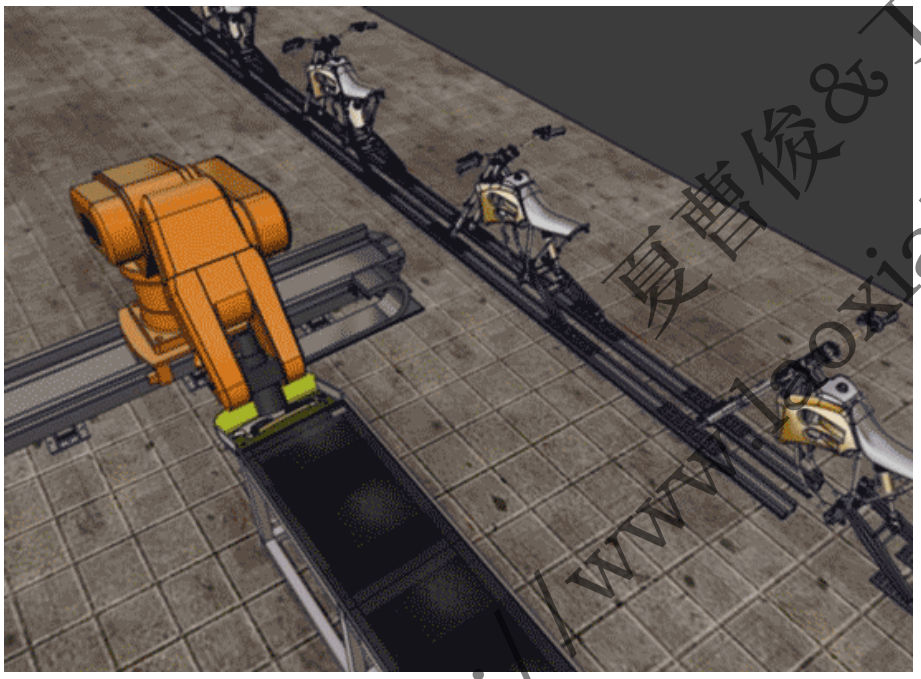
- 对不同对象之间划分责任和算法的抽象。这一类型中，我们将进一步厘清面向对象设计的概念——“动作，行为也可以抽象为一类对象”

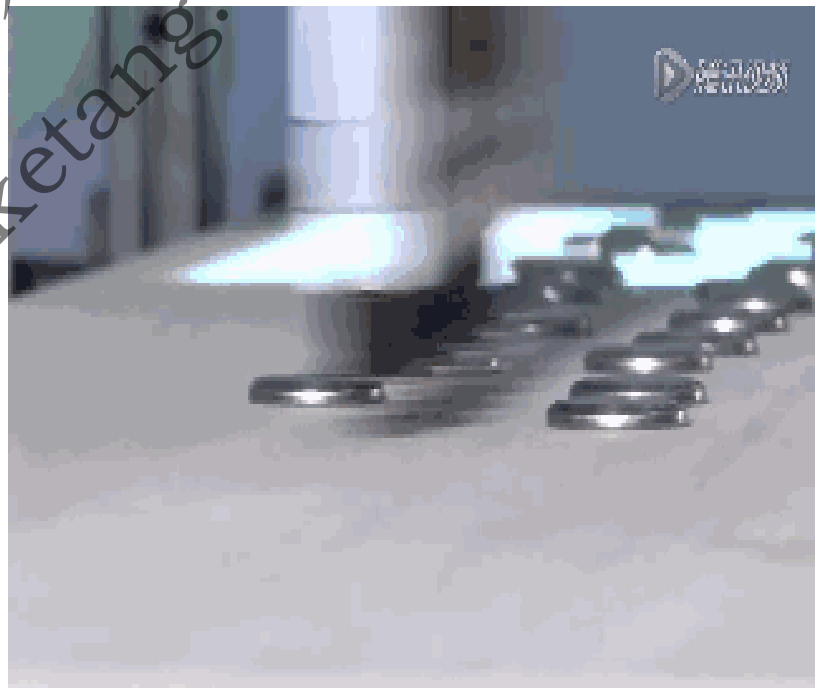
行为型模式之模板模式

- 学习模板模式
- 一：模板模式的介绍-定义、结构、参考实现、场景问题
- 二：模板模式的典型疑问与优缺点评价
- 三：模板模式的应用案例与思考

场景

- 定义一个操作中的算法的骨架，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。





模板方法与抽象类

- 那就是通常在“既要约束子类的行为，又要为子类提供公共功能”的时候使用抽象类。
- 按照这个原则来思考模板方法模式的实现，模板方法模式需要固定定义算法的骨架，这个骨架应该只有一份，算是一个公共的行为，但是里面具体的步骤的实现又可能是各不相同的，恰好符合选择抽象类的原则。
- 变与不变
- 程序设计的一个很重要的思考点就是“变与不变”，也就是分析程序中哪些功能是可变的，哪些功能是不变的，把不变的部分抽象出来，进行公共的实现，把变化的部分分离出去，用接口来封装隔离，或用抽象类来约束子类行为。
- 模板方法模式很好的体现了这一点。模板类实现的就是不变的方法和算法的骨架，而需要变化的地方，都通过抽象方法，把具体实现延迟到子类，还通过父类的定义来约束了子类的行为，从而使系统能有更好的复用性和扩展性。

模板方法与好莱坞法则

- 什么是好莱坞法则呢？简单点说，就是“不要找我们，我们会联系你”。
- 模板方法模式很好的体现了这一点，做为父类的模板会在需要的时候，调用子类相应的方法，也就是由父类来找子类，而不是让子类来找父类。
- 这是一种反向的控制结构，按照通常的思路，是子类找父类才对，也就是应该是子类来调用父类的方法，因为父类根本就不知道子类，而子类是知道父类的，但是在模板方法模式里面，是父类来找子类，所以是一种反向的控制结构。

模板方法在实际开发中的应用案例

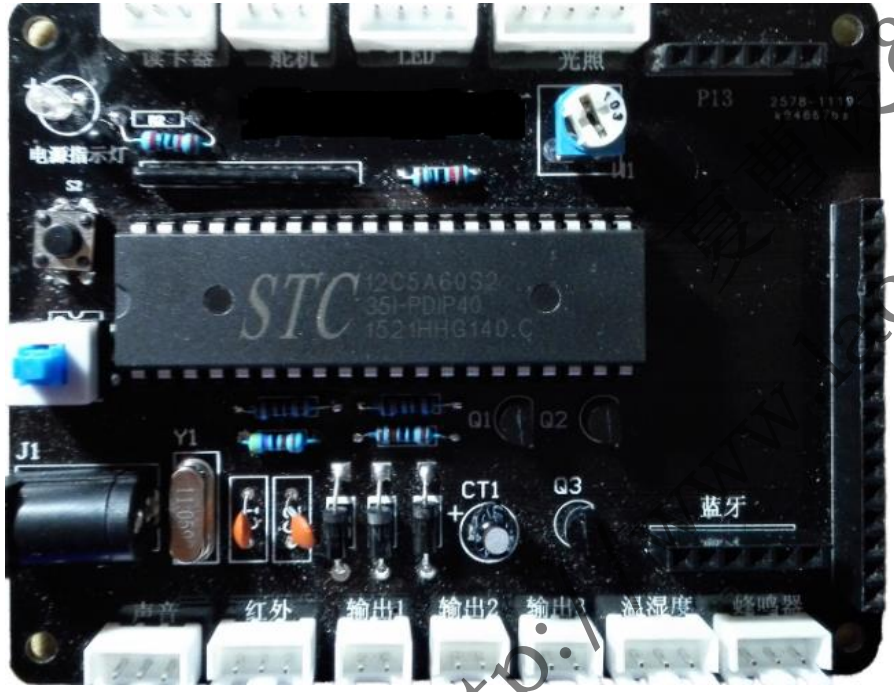
- 钩子函数的设置
- 单元测设工具gtest

观察者模式

- 学习观察者模式
- 一：观察者模式的介绍-定义、结构、参考实现、场景问题
- 二：观察者模式的典型疑问与优缺点评价：饿汉，懒汉与多线程安全
- 三：观察者的应用案例与思考：缓存

场景

- 考虑这样一个实际应用：当一个对象的状态发生改变的时候，如何让依赖于它的所有对象得到通知，并进行相应的处理呢？



观察者模式

- 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，所有依赖于它的对象都得到通知并被自动更新。
- 回想一下：邮局订阅报纸的过程：
- 出版者+订阅者就是观察者模式

观察者模式的本质-触发联动

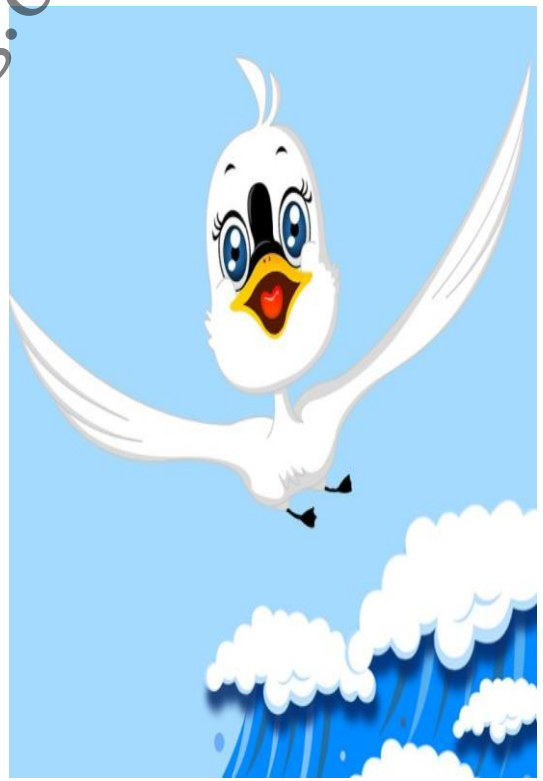
- 1：当一个抽象模型有两个方面，其中一个方面的操作依赖于另一个方面的状态变化，那么就可以选用观察者模式。
- 2：当一个对象必须通知其它的对象，但是你又希望这个对象和其它被它通知的对象是松散耦合的，也就是说这个对象其实不想知道具体被通知的对象，这种情况可以选用观察者模式，这个对象就相当于是目标对象，而被它通知的对象就是观察者对象了

策略模式

- 学习策略模式
- 一：策略模式的介绍-定义、结构、参考实现、场景问题
- 二：策略模式的典型疑问与优缺点评价
- 三：策略模式的应用案例与思考

场景（is a还是 has a）

- 考虑这样一个场景：有这样一个动画场景…,会飞的鸭子?



如何设计

- 从封装的角度来说，我们希望抽象出数据的共同属性，但是抽象出一个超类Duck就可以了吗？

策略模式定义

- **1：策略模式的功能**
- **策略模式的功能是把具体的算法实现，从具体的业务处理里面独立出来，实现成为单独的算法类，从而形成一系列的算法，并让这些算法可以相互替换。策略模式的重心不是如何来实现算法，而是如何组织、调用这些算法，从而让程序结构更灵活、具有更好的维护性和扩展性**

策略模式的UML示例与代码

策略模式的本质-里氏替换原则

- 里氏替换原则，子类可以完全代替父类的行为。
- 策略模式是一个扁平结构，一系列的实现算法其实是兄弟关系，都是实现同一个接口或者继承的同一个父类。这样只要使用策略的客户保持面向抽象类型编程，就能够使用不同的策略的具体实现对象来配置它，从而实现一系列算法可以相互替换。

策略模式的工程应用-if/else的代码坏味道

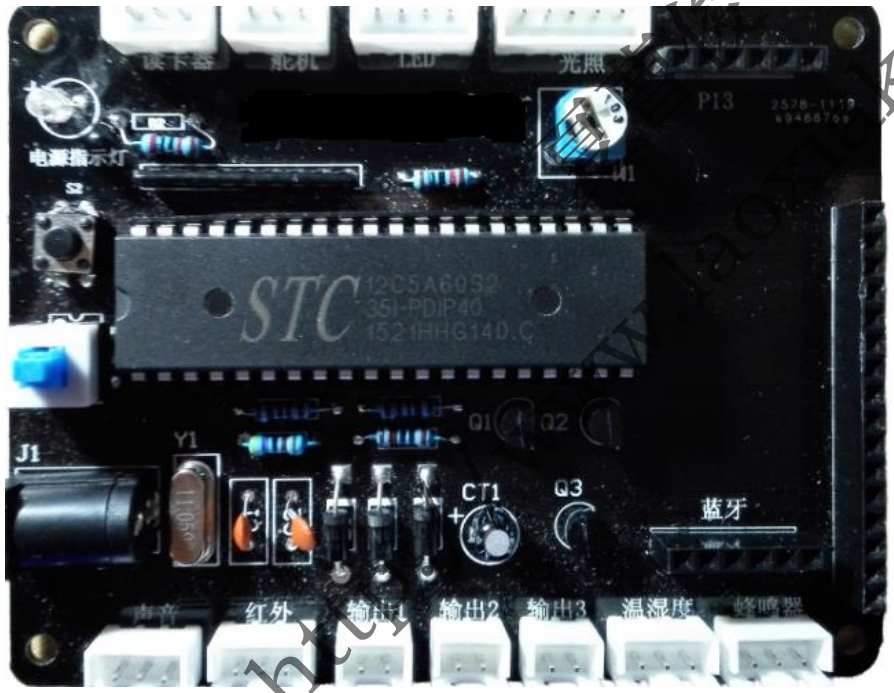
- 支付交易系统中的账户转账——不能用账户作为对象
- 同时将，转账作为账户的动作，而应该将转账抽象成策略对象，组合进入，账户。
- 数据采集系统的信令采集

命令模式

- 学习命令模式
- 一：命令模式的介绍-定义、结构、参考实现、场景问题
- 二：命令模式的典型疑问与优缺点评价
- 三：命令模式的应用案例与思考

场景

- 考虑这样一个实际应用：在智能控制的板卡上有多个插槽slot，每一个插槽可以关联一个设备，现在希望能够有效的记录下这些设备的申请运行。



如何设计

- 从封装的角度来说，设备本身就具有了开关的功能。我们需要记录他们的状态，而这一系列的状态恰恰是我们需要处理的抽象。

命令模式

- **1：模式的功能**
- **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，以及支持可撤销的操作。**

责任链模式（Chain of responsbility）

- 学习责任链模式
- 一： 责任链模式的介绍-定义、结构、参考实现、场景问题
- 二：责任链模式的典型疑问与优缺点评价
- 三：责任链模式的应用案例与思考

场景

- 考虑这样一个实际应用：企业工作流，财务报销流程：
- 1不同级别的领导，审批的费用金额不等
- 2 申请报销由申请人进行发起，提交领导审查，如果同意则可以领取，如果不同意就通知不被批准
- 3假定我们有如下的一个报销流程，项目经理可以审批500元，部门经理可以批准1000元，CEO不限数额。

如何设计

- 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，并沿着这条链传递该请求，直到有一个对象处理它为止 。

责任链模式

- 1：模式的功能
- 如何构建链
- 这是个大问题，实现的方式也是五花八门，归结起来大致有以下一些方式。
- 首先是按照实现的地方来说：
- （1）：可以实现在客户端，在提交请求前组合链，也就是在使用的时候动态组合
- 链，称为外部链；
- （2）：可以在Handler里面实现链的组合，算是内部链的一种；
- （3）：可以在各个职责对象里面，由各个职责对象自行决定后续的处理对象，这种
- 实现方式要求每个职责对象除了进行业务处理外，还必须了解整个业务流程。

责任链模式的应用-功能链

- 考虑这样一个功能，在实际应用开发中，在进行业务处理之前，通常需要进行权限检查、通用数据校验、数据逻辑校验等处理，然后才开始真正的业务逻辑实现。可以把这些功能分散到一个功能链中，这样做的目的是使程序结构更加灵活，而且复用性会更好，比如通用的权限检查就只需要做一份，然后就可以在多个功能链中使用了
- 过滤器与事件冒泡

备忘录模式（Memento）

- 学习Memento模式
- 一： Memento模式的介绍-定义、结构、参考实现、场景问题
- 二： Memento模式的典型疑问与优缺点评价
- 三： Memento模式的应用案例与思考

场景

- 考虑这样一个实际应用：如何在不破坏类的封装性的前提下，来保存和恢复对象的状态。消息管理器的应用。

如何设计

- 我们抽象出Message类以后，设计一个retreatMsg的方法?

备忘录模式-广义备忘录模式

- 1：ODBC与数据库
- 2：Http的Session机制
- 3：浏览器的cookie。

调停者模式（Mediator）

- 学习调停者模式
- 一：调停者模式的介绍-定义、结构、参考实现、场景问题
- 二：调停者模式的典型疑问与优缺点评价
- 三：调停者模式的应用案例与思考

场景

- 考虑这样一个实际应用：电脑的组成与微软 Win95划时代的“即插即用”

调停者模式

- 1：模式的功能
- 调停者的功能非常简单，就是封装对象之间的交互。如果一个对象的操作
- 会引起其它相关对象的变化，或者是某个操作需要引起其它对象的后续或连带操
- 作，而这个对象又不希望自己来处理这些关系，那么就可以找调停者，把所有的
- 麻烦扔给它，只在需要的时候通知调停者，其它的就让调停者去处理就可以了。
- 反过来，其它的对象在操作的时候，可能会引起这个对象的变化，也可以
- 这么做。最后对象之间就完全分离了，谁都不直接跟其它对象交互，那么相互的
- 关系，全部被集中到调停者对象里面了，所有的对象就只是跟调停者对象进行通
- 信，相互之间不再有联系。
- 把所有对象之间的交互都封装在调停者当中，无形中还得到另外一个好
- 处，就是能够集中的控制这些对象的交互关系，这样有什么变化的时候，修改起
- 来就很方便。

调停者模式的应用-多对多关系的解耦合

状态模式（state）

- 学习状态模式
- 一：状态模式的介绍-定义、结构、参考实现、场景问题
- 二：状态模式的典型疑问与优缺点评价
- 三：状态模式的应用案例与思考

场景

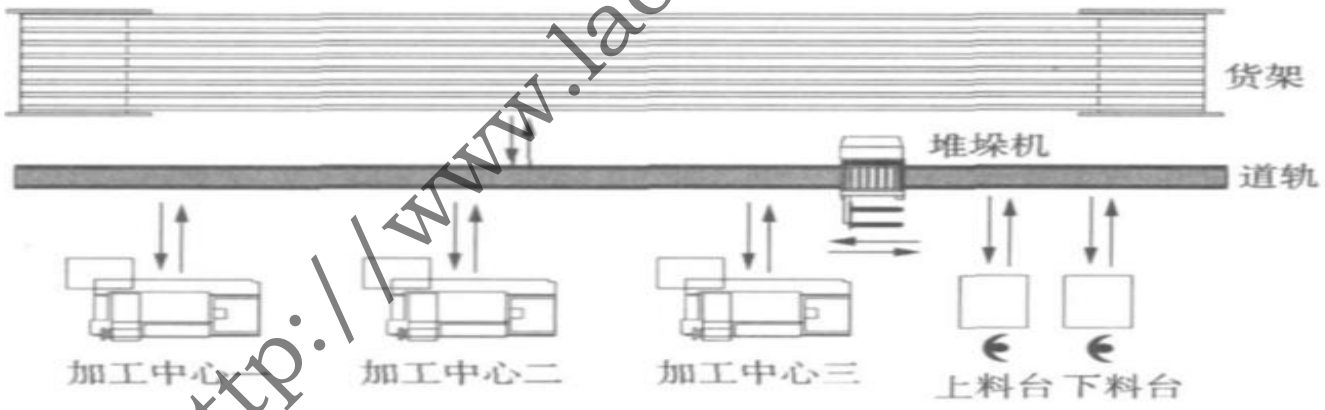
- 考虑这样一个实际应用：考虑一个在线下载的应用，要实现控制在1s内同一个用户只能单线程下载一个文件，如果一个用户反复下载，而且同时下载的并发数超过3，则判定为恶意下载，要取消该用户的资格。如果一个用户的下载个数超过5，将进入黑名单，禁止再登录本系统。”

状态模式

- **1：模式的功能**
- **状态模式的功能就是分离状态的行为，通过维护状态的变化，来调用不同的状态对应的不同的功能。**

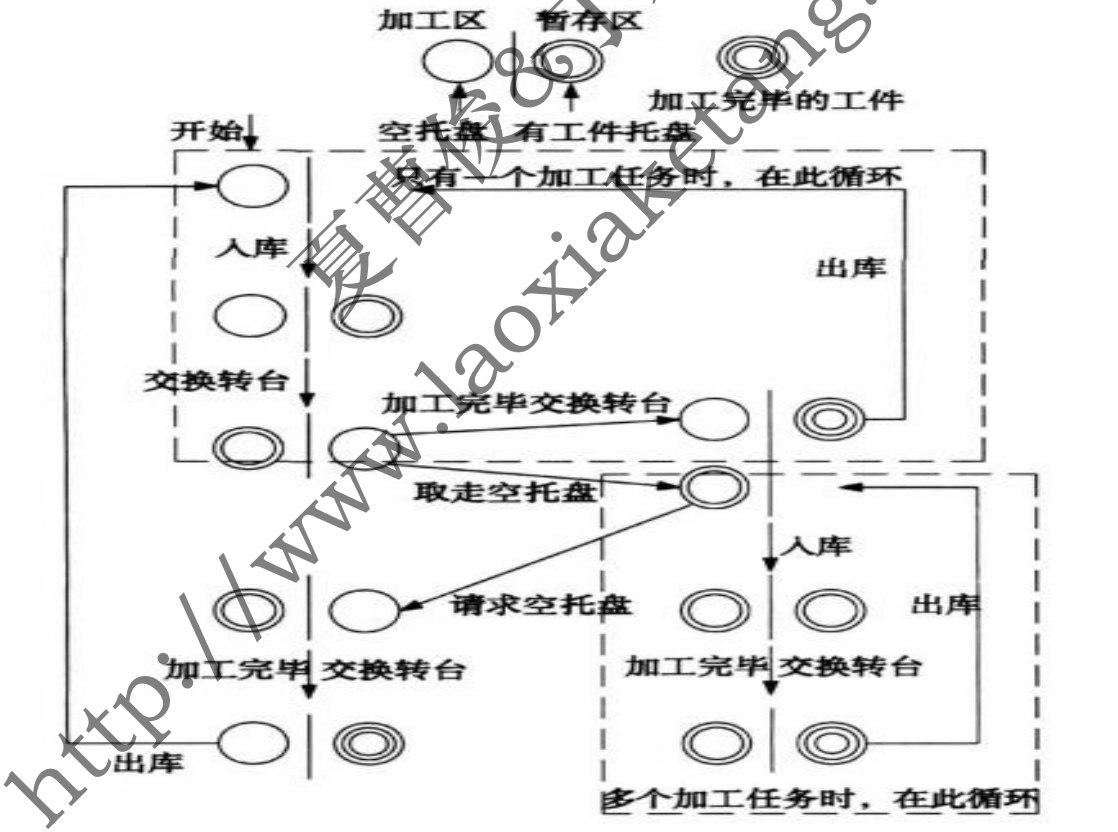
状态模式的应用-复杂if-else结构的解耦合

- 柔性生产中的if-else解耦加工中心共有 5个状态 :空闲 、急停 、故障 、关闭与运行 。其中运行状态只处理工件的正常加工及加工结束时的机床状态调整工作 ， 相应的物流动作发生在机床空闲状态 ,需要注意的是 ,这里的空闲状态只是定义的一个状态 ,实际机床可能在做一些动作



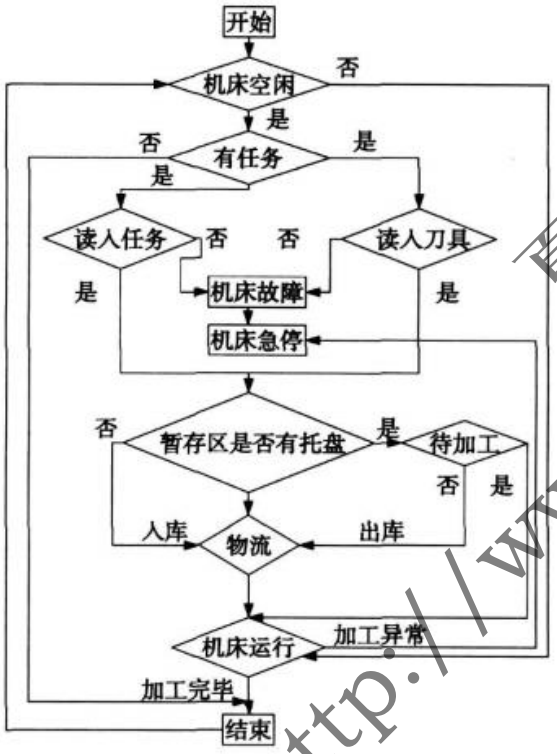
状态模式重构if-else结构

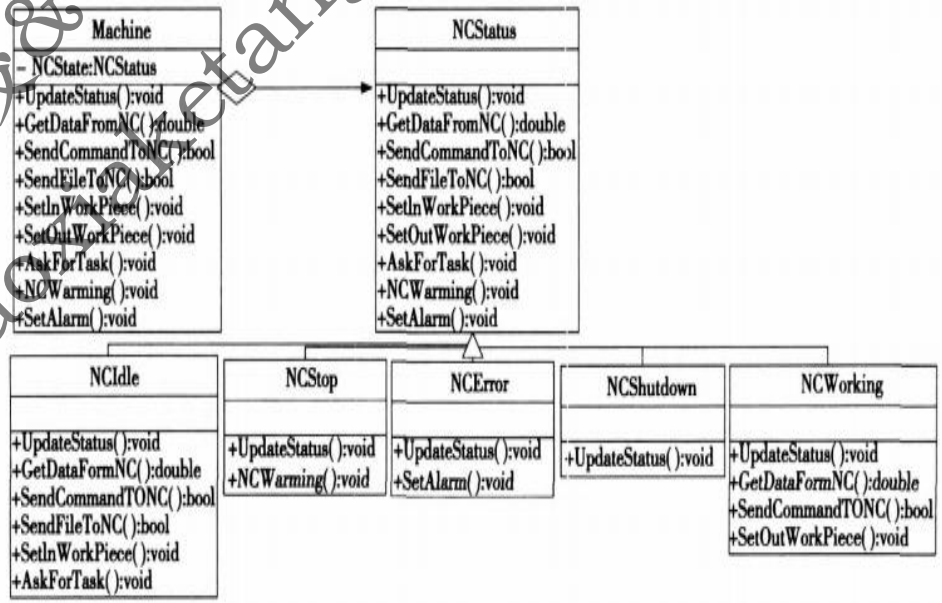
- **机床的行为取决于它的状态,在机床的运行时刻,物流的行为也是根据机床的状态做出改变,取走暂存区的空托盘或加工完毕的工件可以发生在机床加工区加工结束时或者是加工中,两者的改变带来了状态的变化,状态的变化依赖大量的分支判断来实现,在需求变化时~~~~**



状态模式的应用-复杂if-else结构的解耦合

- 加工流程





解释器模式（Interpretor）

- 学习解释器模式
- 一：解释器模式的介绍-定义、结构、参考实现、场景问题
- 二：解释器模式的典型疑问与优缺点评价
- 三：解释器模式的应用案例与思考

场景

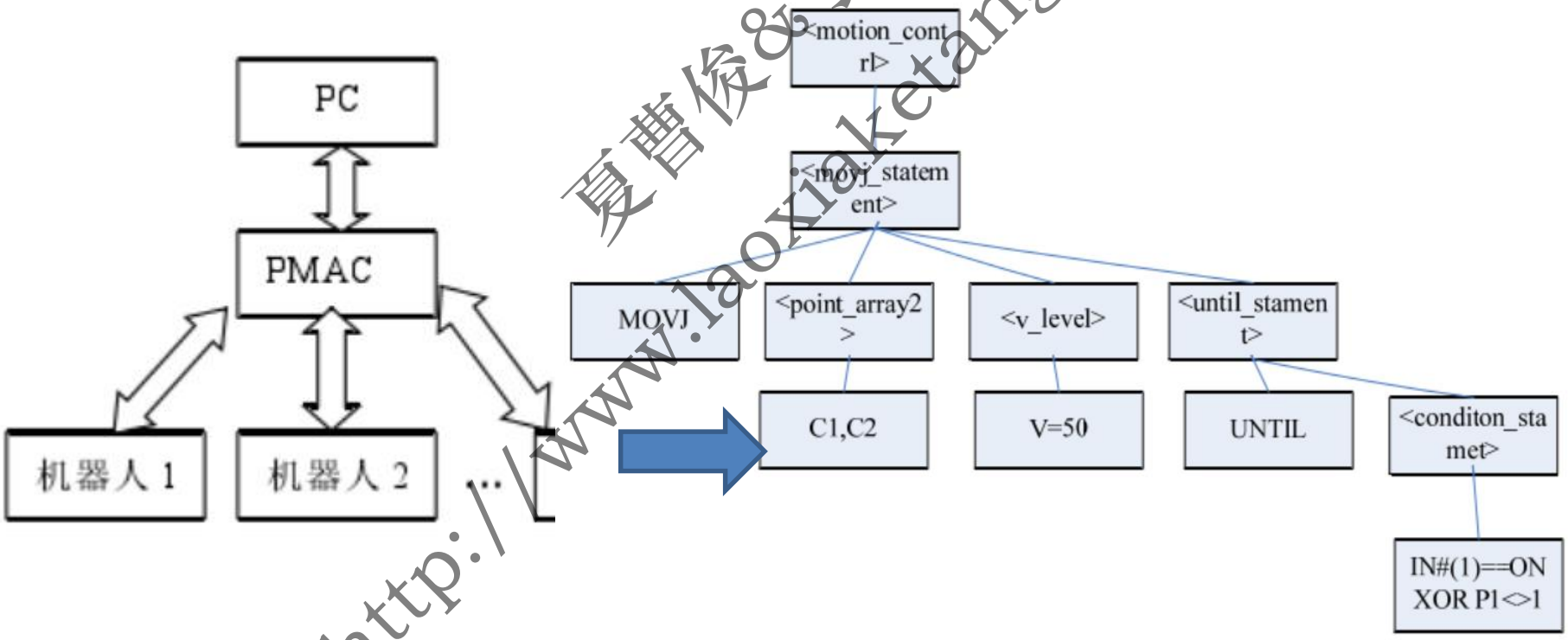
- 考虑这样一个应用：制作一个个位数加减法的计算表达式解析

解释器模式

- **1：模式的功能**
- **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，这个解释器使用该表示来解释语言中的句子。**

解释器的应用案例-开放式工业机器人运动语言解释器

控制系统可以对不同的机器人进行伺服控制。可以用PMAC单独的控制一个机器人,也可以用PMAC同时对多个不同的机器人进行伺服控制，使其协同工作。这对于工业生产具有非常重大的意义: 对不同的工作,更换不同的机器人来完成工作；在工业自动化生产流水线上，可以让多台不同的机器人协同工作，完成不同的工作



访问者模式（Visitor）

- 学习访问者模式
- 一：访问者模式的介绍-定义、结构、参考实现、场景问题
- 二：访问者模式的典型疑问与优缺点评价
- 三：访问者模式的应用案例与思考

场景

- 考虑这样一个应用：假设我们有一系列的手机需要进行管理、测试，编写应用…

Visitor模式

- **1：模式的功能**
- **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类的前提下定义作用于这些元素的新操作。**

Visitor的应用案例-工单派送系统

由于业务受理和工单施工两大业务流程在时间上是允许异步的,并且业务受理和工单施工在业务上应该是没有太多逻辑关联的。那么如何在不影响业务受理的情况下将业务受理与工单施工进行有效的衔接,这是系统设计需要考虑的首要问题。因为,对于业务受理来说,受理产生的受理单本身不知道对应的施工工单是什么样的格式,也不知道施工的流程,也不知道要拆分成如何样的施工工单同样对于施工受理方,并不关心业务受理的情况,只关心发送过来的工单格式。

所以为了要达到受理单和工单在不受“污染”的情况下各自工作,就要求系统的派单系统来处理这种转换。

此外,业务受理产生的受理单种类繁多,每种单对应的施工单的格式和消息都很不一样,并且对应的流转方式也不尽相同如一些审核的工单可以流转到系统的数据库表中等待审批、而一些购买信息则需要做远程接口实时发送到商平台。所以,工单派送子系统应该有为每种不同类别的受理单处理转换的能力。

随着业务的扩展越来越多的受理业务和管理部门的加入,施工过程也是个相对变化较大的模块,这样就要求施工过程能灵活的扩展,尽量达到在施工业务扩展的情况下不影响受理业务。从而使业务的扩展影响的范围缩小在最小范围内。

Visitor的应用案例-模块管理的双重分发