大家好啊.
今天我们将遇到我们的第一个套路–策略模式.

[产品乱入]: 小许啊, 我们这里有个需求文档, 看看吧.
XX 市管理局:
贵公司好, 我市希望请贵公司为我市设计一个管理系统, 统计市民的生活情况.
现在先请贵公司设计一个类, 模拟一个市民. 此市民可以吃饭, 睡觉, 工作.
注: 不同市民的生活方式可以不一样, 我们会提供不同市民的动作.
注 2: 具体行为的变化可能较快, 希望贵公司谅解.
注 3: 我们会尽可能快的发来下一篇需求文档.

谢谢

哎呀, 今天晚上我吃臭鸡蛋和瓜皮吧 (编的真烂)

算了, 先分析需求.

需求不就是让我们写一个市民类吗, 里面有 eat,sleep,work3 个方法, 老简单了.
小许很快写出了第一版代码 basic.cpp

这时, 小许突然收到了一封邮件.

PM: 客户刚刚说不同市民的行为可能不一样.

小许: 切, 我把函数改称支持多态的不就好了 (PS.C++ 函数默认带 final)
很快, 小许就写好了第二版的代码 (Change_Virtual.cpp), 交给了 BILL 审批
BILL: 嘿嘿嘿, 你和张大胖犯了同一个错误. 去问他吧.

小许: 请问张大胖在吗?

张大胖: 在.
小许:BILL 说我这版代码不合格, 为什么?
张大胖满脸自豪 (BILL 优秀小学教师实锤): 你知道设计模式吗?
小许: 布吉岛
张大胖: 就是一群前人总结的经验而已
小许: 那我去背不就是啦?
张大胖:NO,NO,NO, 你要理解其精华.
小许: 那其精华是什么呢???
张大胖: 你最讨厌什么?
小许:RubbishPM 改需求!
张大胖: 那你就要让改需求时更方便.
小许: 什么意思?
张大胖: 就是要让容易改变的地方做的容易改变, 容易复用.
小许: 我这里只要一个类就可以了呀?
张大胖: 我问你, 如果有 100 个吃饭, 睡觉方法互不相同的包子师傅, 你要用几个类?
小许:100 个.
张大胖: 产品要改一下包子的制作方法.
小许:emmmm,100 个类要改死我啊
张大胖: 你看,BILL 就把你的代码退回来了吧.
小许: 哦! 谢谢张大胖.
张大胖: 不谢 (BILL 还送我个小黄鸭, 不错)

小许想了想, 决定增加一个 SKILL 类 (源代码我其实直接写成了 Function-Pointer)
然后增加了 SKILL 类成员 Eat,Sleep,Work. 在 eat,sleep,work 里调用.(Pattern_used.cpp)

BILL 看了一眼: 不错, 不过你还是把 SKILL 改成类吧, 毕竟这个 SKILL 里可能会发生一些神奇的事情. 改完就能 Commit 了
很快, 小许写出了最终版代码 (Pattern_Result.cpp) 提交了.
BILL: 小许啊, 做的不错, 你知道这是什么模式吗?
小许: 什么模式?
BILL: 这叫做–策略模式, 那个 SKILL 类就是” 策略”

小许: 哦, 涨芝士了.
这时, 小许的邮箱又响了..... 未完待续.