## 第25章 ——模版方法模式

## 25.1 其实手动挡车也不难开

熊小猫：兔小白，听说你自驾去草原玩了几天，感觉怎么样？

兔小白：景色非常美，就是开车有点累。我朋友的车是手动挡，但是我一直开自动挡，手动挡已经开的不熟练了。有一次起步时因为紧张，我忘记松手刹，开了几公里才发现。

熊小猫：不管手动挡还是自动挡，发动汽车起步的过程都差不多，只是细节上有些区别。你只要记住过程中大的步骤，就不会忘记松手刹了。

兔小白：流程我都知道，但是手动挡要操作离合器，手忙脚乱中我就忘了松手刹。

熊小猫：可以理解，毕竟是人就可能犯错嘛，人又不是机器。不过在软件开发中，是有办法解决此类问题的。

兔小白：是吗？但是软件也是人写的，不可能一点错误没有呀！

熊小猫：我的意思不是开发的软件没有问题。而是有一种设计模式可以保证这种模版化的操作不回漏掉某一个步骤。这种设计模式叫做模版方法模式。

你可以思考一下，咱们生活中的很多事情都是模版化的，大的流程都一样，只需要改变每一步执行的具体细节。例如学习，无论学什么内容，都要经历预习、听课、练习这三个步骤。你开发程序也一样的，大部分需求都需要开发Controller、Service、Dao类，还有单元测试类。一旦你开发完第一个需求，后面的需求都可以按照同样的套路开发。

兔小白：你让我想起做PPT，我一般都是找个模版改一改。你说的模版方法是不是类似？

熊小猫：还是有一点区别的。咱们做个练习，你就明白啦！

## 25.2

熊小猫：练习很简单，你回忆一下自动挡汽车的起步过程，写一段程序演示这个过程。

兔小白：自动挡汽车我熟悉，很快搞定！

10分钟后，兔小白写完了代码。

兔小白：我先定义了司机接口Driver，以便扩展。接口中只有一个startCar方法。

//代码

AtDriver是自动挡汽车司机类，实现Driver接口。按照自动挡汽车起步的步骤实现startCar方法。

//代码

客户端代码如下。

//代码

程序输出如下。

//代码

熊小猫：看来已经知道我要加需求了，接口都已经留好了。下面你再来实现手动档汽车驾驶。

5分钟后，兔小白完成了开发。

兔小白：我新增加了一个Driver接口的实现类MtCarDriver。

//代码

手动档和自动档汽车的起步过程差不多，所以我把AtCarDriver的startCar方法拷贝到MtCarDriver中，稍作修改就开发完成了。这就是你说的模版方法吧？

熊小猫：哈哈，你这就是复制粘贴，哪里是模版方法啊。你运行一下程序看看。

兔小白运行程序，结果如下。

//代码

熊小猫：咦？怎么没有挂档操作？汽车是怎么开起来的？

兔小白：哎呀......我拷贝完代码，可能在修改的时候被错误删除了。

熊小猫：因为startCar方法中的逻辑都在细节层面上，没有层次。其实汽车起步可以分为以下几个大的步骤。

1. 踩脚刹
2. 汽车点火
3. 挂档
4. 松手刹
5. 加油出发

无论是手动档还是自动档汽车，汽车起步都是这5个步骤。我们可以把流程这种不变的逻辑抽象到父类的startCar中，子类只要继承父类就一定会按照这5个步骤的顺序执行，但是每一个步骤的实现可以由子类自己定义。

兔小白：我按照你的思路想想该怎么改。

15分钟后，兔小白改好了程序。

兔小白