Template(模板模式)

By:Zhi Li

1.什么是模板模式(What?)

模板模式的定义：定义一个操作中的算法骨架，而将算法的一些步骤延迟到子类中，使得子类可以不改变该算法结构的情况下重定义该算法的某些特定步骤的模式。它是一种行为型模式

2.为什么要使用模板模式(Why?)

在有些特定的情境下，处理的方式是有不同种类的，这些处理的方式虽然种类不同，但是他们的实现步骤大同小异。只要将这些大同小异的步骤全部抽象出来，代码便可以实现很高的复用性。
我个人将模板模式理解为是一种“提取公因式”的设计模式，其实这种提取公因式的思想在我们日常生活中也随处可见。例如做菜就是一件这样一件事

一般来说做一道鱼香肉丝包含了四个步骤：

买菜、洗菜、切菜、炒菜

但是我们不可能每天都吃鱼香肉丝，兴许哪天我们喜欢吃清蒸蛋羹

那么做一道清蒸蛋羹同样也包含了四个步骤：

买菜、洗菜、切菜、蒸菜

以此类推，做任何一道烹饪方式不同的菜（炒、煎、煮等等）都需要经历四个步骤，而且你会很惊讶的发现似乎做任何一道菜的前三个步骤他们似乎都是一模一样的，即买菜、洗菜、切菜（当然我在这里举的清蒸蛋羹的例子可能不太恰当，鸡蛋可能不需要切。但是这个思想却是通用的）。因此，这就是模板模式的意义所在：它将这些共同的步骤像公因式一样全部提取了出来，以后我们每增加一个新的菜式的时候，只需要沿用我们提取的公因式走过前面买菜、洗菜、切菜的三个步骤，唯独留下最后一个步骤根据菜式自行决定就行了。通过这种方法，代码的复用率可以明显的增加。

3.怎么使用模板模式(How?)

一个模式的核心之一就是该模式所涉及的角色。模板模式的角色只有一个，那就是模板(Template)。

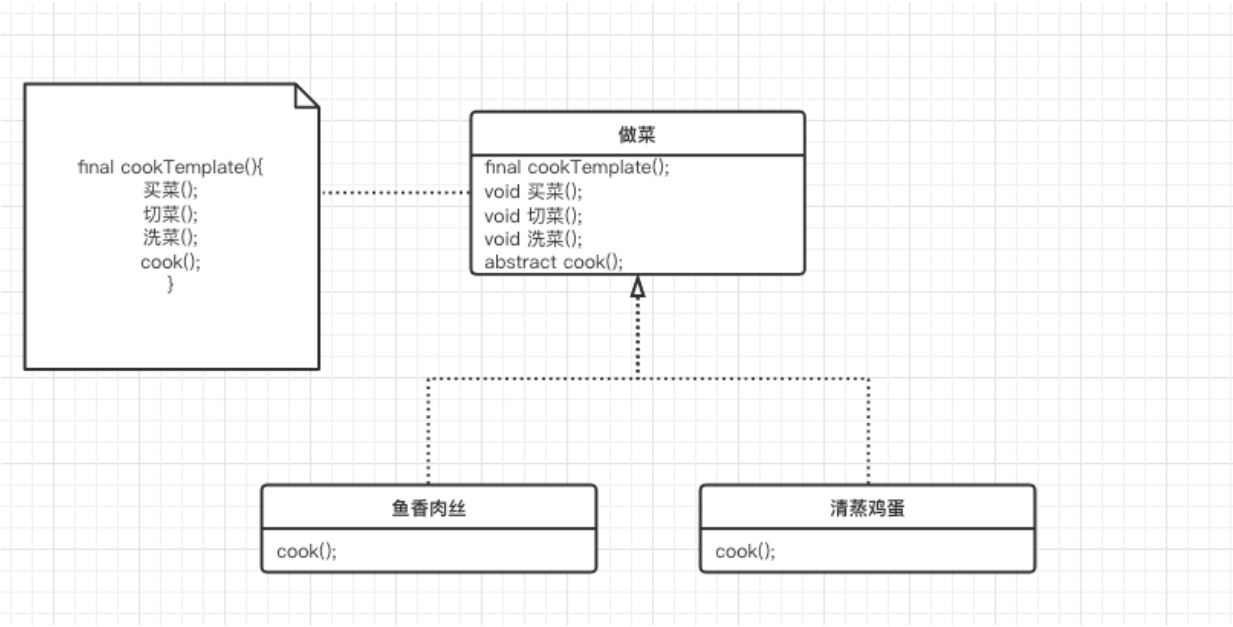
这里插入一个题外话，在设计模式的语境之下，你应当需要清楚的一点就是每当我们提到’角色’这个词的时候，它一定对应的是一个Java中的抽象类或者是接口，而非一个实例。这是因为设计模式是一种高度凝练以及抽象的开发问题的解决方案，就好比算法一样。算法是处于函数这个层级的抽象，它的主体是一个函数，你只需要决定输入、函数体以及输出就行了，然而设计模式则是在函数层级之上的架构级别的抽象，你可以将它理解为是多个算法，多个函数组合在一起的层级的抽

象。所以在未来的设计文档之中我不会再强调实例或者是对象的概念，每当提到'角色'的时候，请默认将其当作是一个接口或者抽象类。
模板角色仅仅只由函数组成，这些函数在分类上分为两种：模板函数，以及基本函数

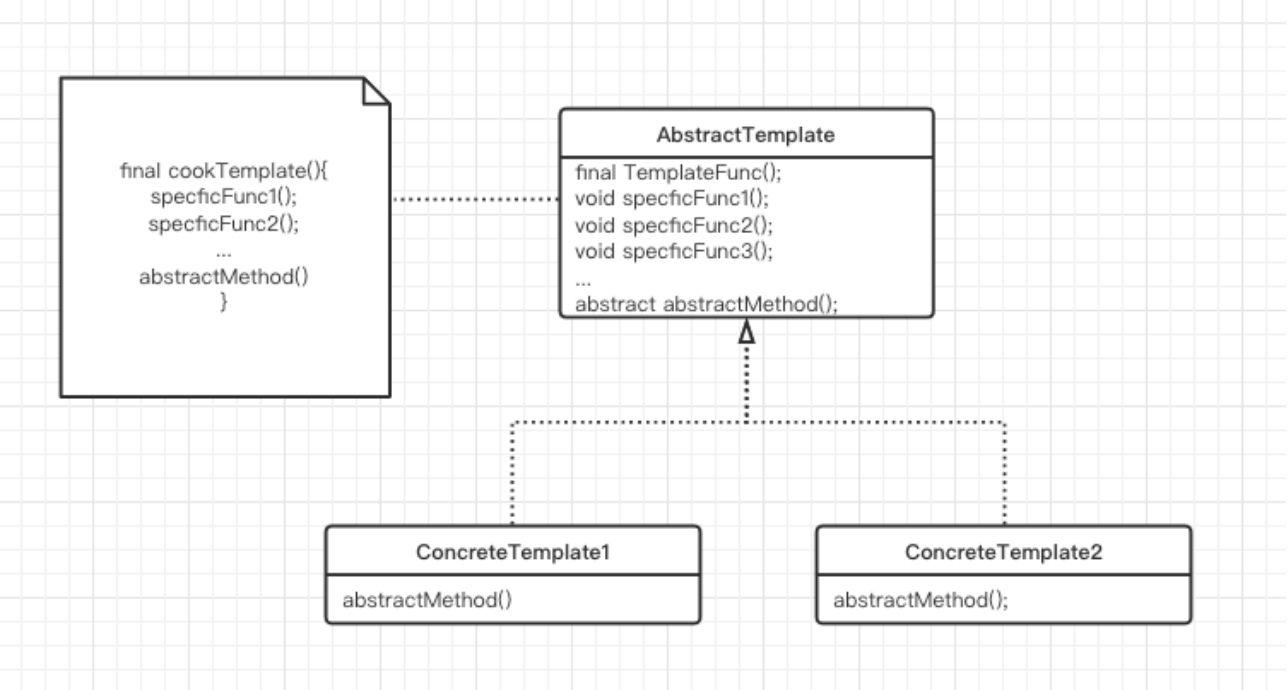
模板函数是一个指定解决方案的执行顺序的函数，它的存在指定了这一类问题的执行顺序。还是按照上面做菜作为例子，做菜的步骤是买菜、洗菜、切菜、蒸菜。那么在这个模板函数之中，我们需要做的就是依次调用这四个函数。(被final修饰)

基本函数是可以自行根据需求设计的函数，由于我们提取的公因式只有买菜、洗菜、切菜，那么最后一个步骤蒸菜在这个语境中就是基本函数，我们可以自定义它为炒，也可以自定义它为煎等等。基本函数按照严格定义又应该分为抽象基本函数、具体基本函数等等。但是我觉得没有必要认知到这个程度，毕竟这不是应试。所以我们只需要简单的理解为：只要这个函数在模板类中是一个能被重写的函数，那么它就是基本函数（不被final修饰）。

综上所述，我们首先推理出来做菜这个情景按照模板模式实现以后的类图是这样的。



由此情景图推理可得，一个模板模式的类图应该是这个样子的 (注意方法的调用顺序以及这些方法是抽象方法还是实际方法全部都是自定义的，不需要完全按照类图上的示意来)



4. 在这个项目之中如何使用

登陆功能天生就是一个适合适用模板方法的功能

登陆的基本步骤包含下列几步：

1.接受学生的学号作为输入

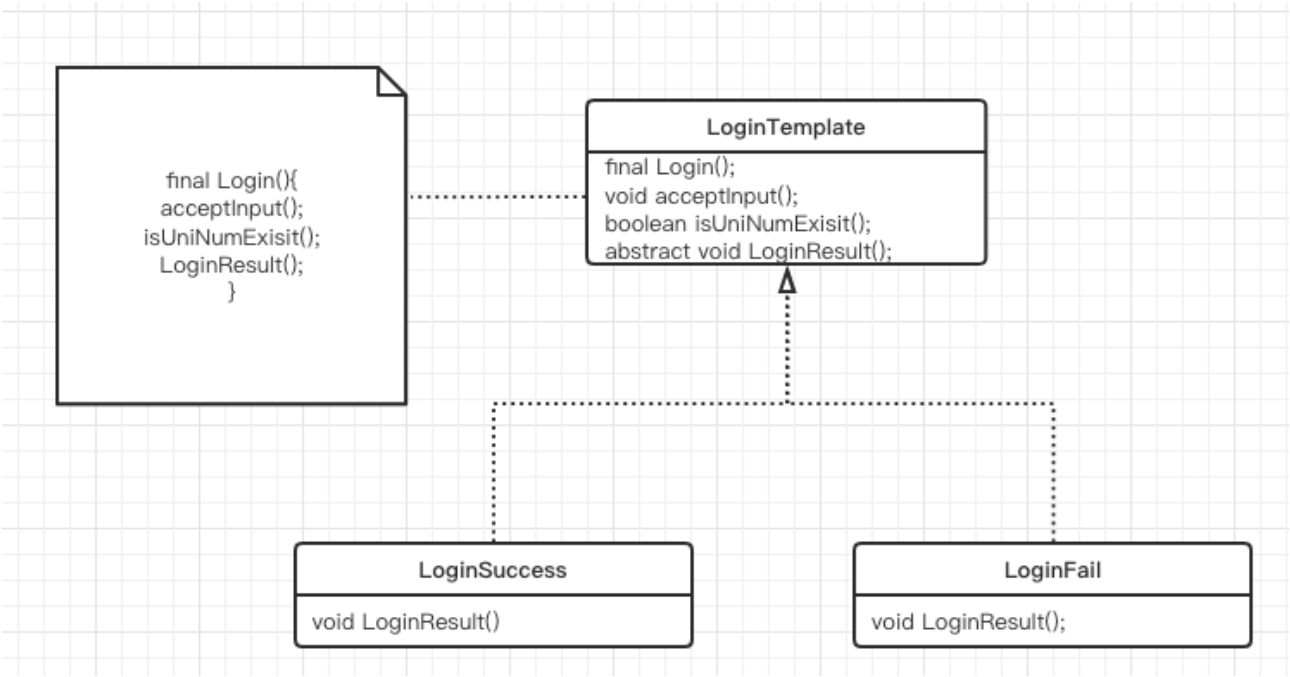
2.判断该学号是否存在于数据库中

3.若存在则返回该学生的状态等信息，弹出欢迎界面，若不存在则提示登录失败然后返回

前两个步骤就是我们所谓的“公因式”，而第三个步骤需要根据学生状态来自定义。因此这个功能和上述所提到的做菜的例子匹配度极高，是一个特别适合使用模板模式实现的功能。

整个类图如下：抽象模板为LoginTemplate, 实际模板为LoginSuccess与LoginFailed,分别对应登录成功和登录失败的两种情景。
在抽象模板中，模板函数为Login, 基本函数为accpetInput ,isUniNumExist, LoginResult.其中模板函数Login所做的事情就是按照上文的三个步骤顺序调用acceptInput、isUniNumExist以及LoginResult。

在实例模板LoginSuccess与LoginFail中，唯一需要重写的基本方法为LoginResult。在LoginSuccess中,这个LoginResult所实现的就是登录成功后的逻辑，包括呈现学生当前的注册状态以及展示欢迎语等等，而LoginFail对应的就是登录失败后的逻辑，包括返回失败提示等等。

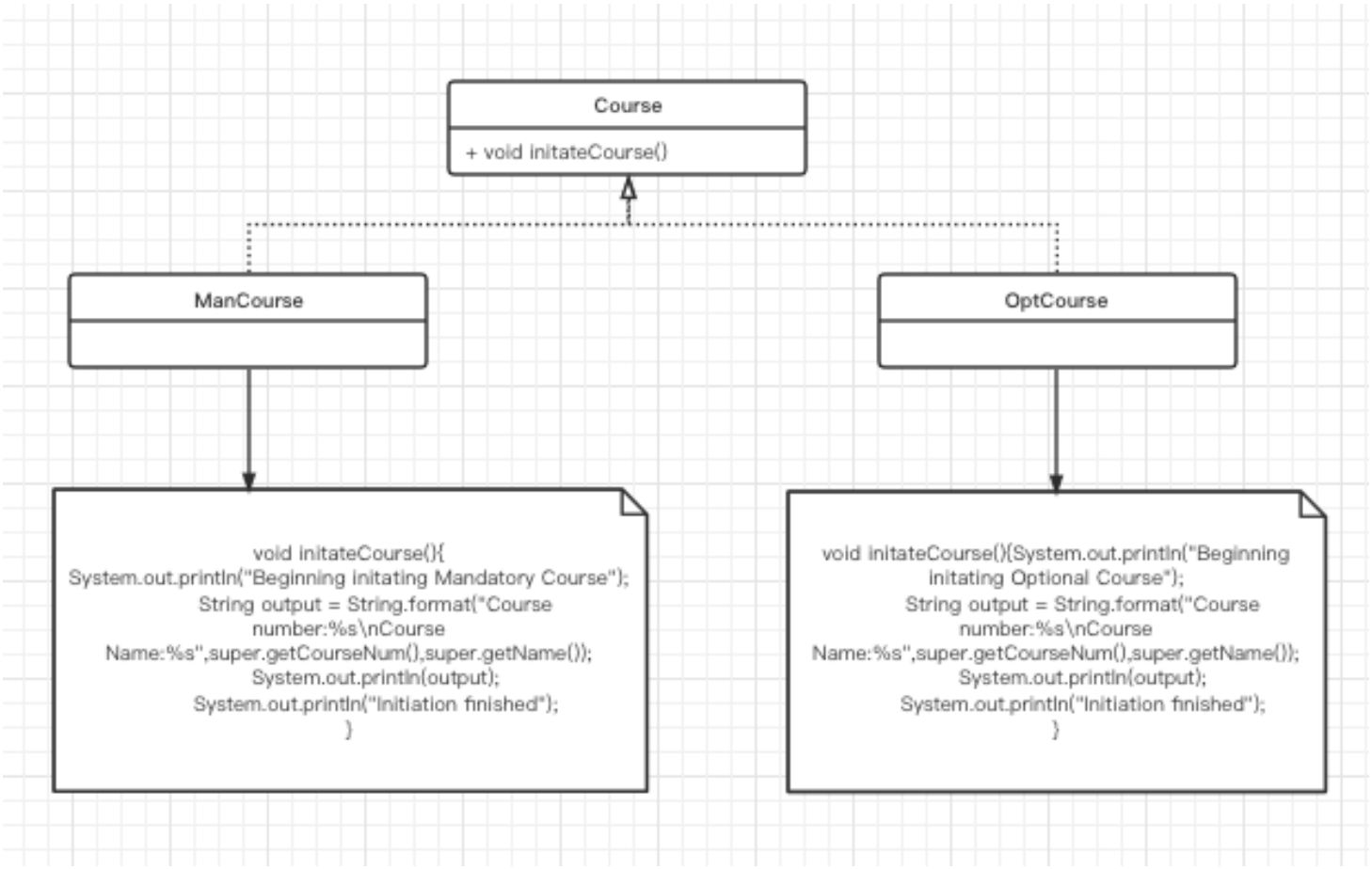


注意：由于接受输入的逻辑还没有确定通过哪种方式实现（命令行或前端），因此acceptInput方法默认返回的输入值是一个模拟的值。同时学生的存储方式也没确定，因此无法查询学生是否已经存在。具体实现请结合代码注释阅读

新增: (after commit f436e)

新的模板模式应用在课程的初始化的过程中,用于区分该次课程的初始化是选修课还是必修课:

其类图如下所示:


Course
+ void initateCourse()
ManCourse
OptCourse
void initateCourse(){
System.out.println("Beginning initating Mandatory Course");
String output = String.format("Course number:%s\nCourse Name:%s",super.getCourseNum(),super.getName());
System.out.println(output);
System.out.println("Initiation finished");
}
void initateCourse(){System.out.println("Beginning initating Optional Course");
String output = String.format("Course number:%s\nCourse Name:%s",super.getCourseNum(),super.getName());
System.out.println(output);
System.out.println("Initiation finished");
}