系统设计之后，我们需要对我们的类模型进行细化，具体过程如下

1. **实现用例**

根据用例的场景，我们为类添加一些操作，使我们的用例能够通过类运行起来

1. **设计算法**

一般来说，20%的操作消耗80%的执行时间，对于可能存在性能问题的操作，我们可以给出一个初步的算法伪代码（例如，每次查找key为“abc”的对象，我们需要都遍历10000个对象，每个对象是有序排序的，这时候我们可以使用折中算法）

在一些复杂的算法中，我们需要为类添加一些行为，让算法更容易理解（如下面的伪代码，我们在word中添加一个“包含”方法）

如下是一个查找字符的算法伪代码

for line in doc

for word in line

If word.包含(“需要查找的字符”)

Return word

End if

End for

End for

1. **向下递归设计操作**

如下一个站点登录用例：

用户访问登录页面

系统返回登录页面

用户输入用户名密码

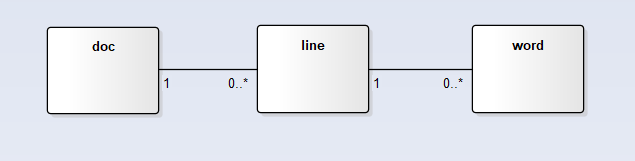
系统验证用户名密码正确

系统跳转至后台主页

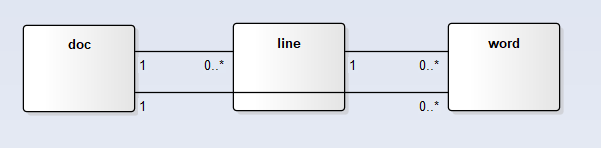
在这段用例中，我们可以在AccountController中添加“返回登录页面”方法，“验证用户名密码”方法，“验证用户名密码”方法会调用我们的“用户仓库”类的“获取用户”方法获取我们的“用户”类，然后调用我们“用户”类的“验证密码”方法验证我们密码的正确性

上面的操作中，除了AccountController的“返回登录页面”方法和“验证用户名密码”方法可以从用例中发现外，其余的方法需要我们自己去推导设计

1. **为高效访问而增加冗余**



有时候我们为了高效访问会增加一些冗余关联，如下



1. **应用缓存**

对于一些消耗性能的操作，我们可以缓存其结果

1. **行为具体化**

即将我们的操作提升为类

如果我们的行为有状态，需要在运行时存储或修改或传递，则我们应该将行为提升为类

1. **调整继承**

某些类的行为具有相似性，这时候我们应该从不同的抽象层次去看待，可能我们就会发现他们具有共同的父类