## 老测试系统的不足点

1. 全局的control，control中的方法涉及面太广，控制了全局，即存在上帝类

**改进** 我们添加了control的接口（用于前台和后台之间的交互），即前台只能调用control的接口来与后台进行交互，而不能通过其他方式，同时根据page类与question类之间的关系，我们对我们的control做了一个映射，总的有个page control用于控制page的各种行为，但是当需要对某个question进行操作时，我们又有一个question control的类，对quesiton进行操作，当我们对某一内容需要进行操作，我们先调用page control，如果需要获取或者修改相关question内容时，我们由page control调用question control，question control操作后返回结果，由page control 接收，最后page control返回前台结果，即page control起到了中间交互点的作用，前台和后台的交互只能通过page control来完成。同时page control也不是上帝类，它不能完成所有的行为，而是要通过调用下级或者自身的函数来完成相关的操作。

即我们对上帝类进行了功能的拆分。

1. 没有相关factory，类的生成比较繁琐，同时容易出现问题。

**改进**

我们添加了AnswerFactory和Question简化了实体的生成，,达到封装效果,也就减少错误修改的机会。

1. 前台界面与Control层耦合严重

## 我们的设计

在这次开发中，为了提高效率我们对前台和后台的开发进行了分离，小组进行了讨论后统一了Control的接口，前台调用接口，后台实现接口完成前台给的任务，并且返还结果，这种定义好的接口极大的方便了开发，提高了各个部分的效率，同时统一的标准减少了争议以及前台和后台的疑惑，使我们的时间更多的应用于具体问题上，而不是前台需要的函数与后台所写的不一最后讨论这些不必要的时间浪费了。

## 心得

先进行讨论，每个人提出自己的建议，对整体系统进行剖析讨论，明确了职责后，前台和后台分开开发，定义好接口，明确了前台和后台交互的原则与具体实现，这样能够提升组员之间的效率，减少不必要的争议。

在开发过程中我们也发现后端接口的设计有很多缺陷，但是迫于时间所限，没有进行很大规模的重构。由于后端接口的设计与GUI开发时分开进行的，导致合并时遇到了很多问题，被迫修改了很多后端代码。未来开发希望能够尝试更多次迭代的方法来解决这种问题。

由于分工时没有正确估计工作量，导致最后又非常明显的工作量不均衡现象，出现了一个人提交了95%以上的代码更改的现象。