重构：引导课

改善既有代码的设计

配合：软件质量保障

- 这些代码有哪些问题?
- 面向对象设计的特点
- 重构的原则
- 重构的基本方法
- 如何阅读《重构》一书?

这些代码有什么坏味道?

- public Map<String, Object> userqry(PageResult<ScuUser> pageResult,
- String name, String phone,String jobNember, Integer sex, String startDate, String endDate)throws Exception;

- Map<String, Object> suggestQry(PageResult<RafflesSuggestBean> pageResult,
- String name, String phone, Integer startAge, Integer endAge,
- Integer sex, String suggestDateStart, String suggestDateEnd)throws Exception;

过大的类

重复的代码

过长的函数、方法

解决办法

问题：过长参数

解决方案：

方法参数用对象，而非一长串参数值

面向对象设计

- 抽象
- 封装
- 多态
- 继承

“重构：在不改变软件之可察行为的前提下，调整软件内部结构，提高可理解性，降低其修改成本”

–Martin Fowler

为何重构?

- 为了改进设计
- 为了提高可理解性
- 为了写出稳定健壮的代码
- 为了提高编程效率

何时重构?

- 添加功能时
- 修改错误时
- 评审代码时

何时不能重构?

- 现有代码根本无法运行
  - 折衷办法：重构为封装良好的小型组件
- 项目接近最后期限

重构的基本技巧

- 小步前进、频繁测试

重构的格式

- 名称
- 概要：适用情景以及解决的问题
- 动机：为什么需要重构，何时不该使用这个重构
- 作法：介绍操作的步骤
- 范例：举例说明如何操作

如何阅读《重构》一书?

- 复习Java及面向对象设计的相关内容
- 完整阅读前4章，针对自己的问题选择性阅读重构内容，边看书边回顾自己的代码问题，尝试重构代码
  - 重新组织方法
  - 在对象之间搬移特性
  - 重新组织数据
  - 简化条件表达式
  - 简化函数调用
  - 处理概况关系
  - 大型重构（适用于研究生）
  - 重构工具
- 整理阅读体会为PPT