遗留代码重构&测试

入门篇

CAC@OPPO by 黄俊彬 & 覃宇

💡用 Code Inspection 寻找坏味道

- Sonar Lint
- Alibaba Guideline

？“先重构再测试”还是“先测试再重构”？

- 先**安全**重构再**单元**测试（随手就做）
- 先**集成**测试再**手动**重构（慎重计划）(进阶篇)

❓什么是安全重构?

可以用工具自动完成，小步前进。让工具帮我们写“八股文”代码，让工具帮我们检查。

> ❓Android Studio 最让人惊艳的功能？你最常用的快捷键是?

💡我最常用的快捷键

- Context Actions(Alt+Enter)
- Refactor This(Ctrl+T)
- Generate(Cmd+N)
- Live Template
- 使用 IDE 支持的 Marker Annotation(androidx.annotations.*)

💡 最大化 ROI

- 围绕最有价值的代码
  * 理解起来难（长方法、复杂条件）
  * 重复代码（避免散弹式修改，极有可能被重用）
- 投入相对低
  * 安全重构就可以完成
  * 尝试失败立即回滚
  * 可以使用 Robolectric 和 Mockito，不要用 PowerMock

❓如果特别有价值，但成本高怎么办?

记录下来，作为**技术债务**按优先级排进迭代计划，先**集成**测试再**手动**重构，最后结对完成。

💡小步前进

- 随时 Undo
- 多尝试几种自动重构
- 可以添加单元测试就立即添加
- 修改代码后立即执行测试

❓单元测试回顾

- ❓F.I.R.S.T原则?
- ❓测试类、测试方法命名规范?
- ❓“三段式”测试方法模板?

DEMO

FastHub示例介绍

https://github.com/k0shk0sh/FastHub

开源 GitHub 客户端

- 👍采用 MVP 模式
- 👍代码结构相对清晰
- 👍RxJava & RxAndroid
- 👍部分使用了 Kotlin
- 👎几乎没有注释
- 👎完全没有自动化测试

❤️准备活动

- Code Inspection 介绍
- Android Studio 快捷键介绍
- 增加单元测试 dependencies 和 config

💓轻松热身： FeedsPresenter

- 几乎不用自己写代码
- 单元测试不用 mock
- @VisibleForTesting 使用

@VisibleForTesting 在编译时的检查，官方使用 lint 来检查:https://developer.android.com/studio/write/annotations?hl=zh-cn。lint rule 的名字是 VisibleForTests：https://sites.google.com/a/android.com/tools/tips/lint-checks

💗强度升级：CreateIssuePresenter

- 用到了 Robolectric 和 Mockito

课后作业

至少完成5个单元测试方法的编写，上不封顶。可在上次作业重构代码基础上添加，或者重新挑选代码重构添加测试。

作业要求：

1. 要求至少完成 5 个有效的单元测试方法。
2. 必须保证测试的有效性和质量（测试有效性和质量由教练判断，并公开评语）。
3. 作业以文档形式提交到石墨（被测代码片段，测试代码，以及简单说明）。
4. 完成时间截止2020年3月26日晚上24点。
5. 作业可以分步提交（建议持续提交）。

积分规则：

1. 按时按照要求完成作业提交即获得 3 个基础积分。
2. 教练对作业进行 Review，评出最多数量奖前三名、最佳高质量奖前三名（名次允许并列，一人不会兼得两个奖项）。两个奖项的前三名分别获得额外7、5、3个积分。
3. 学员获得的总积分 10 分封顶。

获得大奖的秘籍：

1. 保证测试有效性
2. 将遗留代码重构后编写测试