单元测试及代码覆盖率提升方案

概述

本文会先说明单元测试的针对重点和需要注意的方面，随后会分析我项目中原本的单元测试设计的不足之处，然后尝试用新的方案进行修改

单元测试的意义和目的

单元测试

单元测试是指对实现某功能对程序模块进行正确性检验的测试工作。

测试有很多种类型，包括单元测试、集成测试和端到端测试等等。 // TODO

我的理解中，单元测试不关心具体的业务逻辑，是对模块的具体功能的和边界条件进行检测的过程。而且，单元测试应该是开发流程中的一个部分，是在设计阶段根据需求进行分析和编写，而非代码实现结束后再补上（TDD）。

在我的理解中，单元测试应该考虑到两个层次：一、是否所有的逻辑和边界条件都考虑到了；二、是否所有的业务代码都被测试到了。其中前者可以根据需求进行拆解和分析，考虑各种边界条件和场景，可以使用等价类划分法、边界值分析法和错误推测法等方法进行分析和设计；后者则更像是一个检测机制，保证测试覆盖了所有代码部分，可以用代码覆盖率来表示。

我对TDD开发的理解是，第一步是对需求进行分解和梳理，将大的需求分解成小的部分，也就是每一个具体的功能模块；第二步是对这几个模块进行测试用例的设计和编写；最后一步才是对每个功能模块对代码实现，这一步是代码逻辑编写对过程。其中第二和第三步是需要相互弥补的，但是在编写时两者的重点和思考的出发点应该是不同的。

代码覆盖率

代码覆盖率可以分为几个层级：行覆盖率（Line Coverage）、函数覆盖率（Function Coverage）和分支覆盖率（Branch Coverage）。

在我的理解中，代码覆盖率是一个用来判断测试用例和优化代码逻辑的指标。就像测试结果标志着代码的健壮性，代码覆盖率一定程度上可以发现测试用例的不足之处。但是就像前文所说的，单元测试的编写和代码逻辑的编写的关注点是不同的，不应该为了通过测试而改写代码，或者为了提高覆盖率而刻意的添加测试用例。所以我认为代码覆盖率是一个用于参考的指标和用来增强测试质量的途径，但并不是说覆盖率高的测试就更好。

单元测试修改

原本方案不足

在原本编写单元测试中，我的编写有几个问题：

1. 在主逻辑编写完成后才编写测试用例。虽然在编写测试用例的时候确实是从功能逻辑的角度出发进行设计，但是还是会受到一部分代码逻辑思维的影响；

2. 因为没有采用TDD的方案，因此原本在代码模块的分级和测试分级上有些划分不是很明确的部分；

3. // TODO 我跑了一遍测试覆盖率，结果如下：



### 修改后的方案

### 测试用例编写

### 覆盖率测试

### 代码的重构

## 结果

## 后续

### 不足与思考

## 参考

1. https://martinfowler.com/bliki/TestPyramid.html

1. https://www.jianshu.com/p/62f16cd4fef3

1.