OO Training

环境搭建

- 什么是测试框架
- gtest

OO Training Plan

- Tasking,TDD,Unit Testing, Git
- Value Object Pattern
- Interface Design,Strategy
- Composite Pattern
- Observer, Visitor

测试驱动开发

- 在实现之前充分了解需求
- 写出来的代码都是为了实现用户需求，没有无用代码
- 信心，写出来的代码都有测试覆盖
- 为后续读代码的人留下线索

Red—Green—Refactor

Tasking

- 节约开发成本，实现用户价值
- 降低任务耦合，提高工作并发
- 缩短反馈周期，快速相应变化
- 跟踪开发进度，降低产品风险

Walking Skeleton

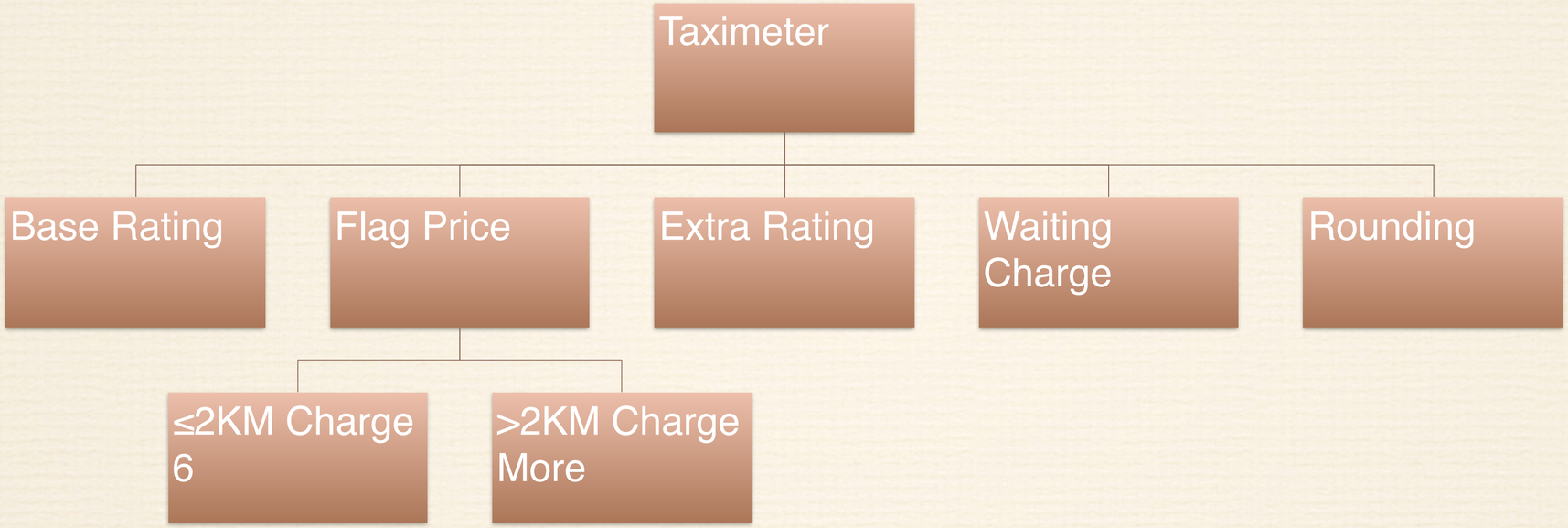
练习

我们考虑出租车计价问题，出租车的运价是每公里0.8元，八公里起会加收50%的每公里运价，起步价是两公里以内6块，停车等待时加收每分钟0.25元，最后计价的时候司机会四舍五入只收块块钱

Tasking

- 作为司机, 我可以通过计价器计算车费, 每公里0.8元
- 作为司机, 我可以通过计价器在八公里后每公里加收50%运价
- 作为司机, 我可以通过计价器在两公里内收取6元定额费用
- 作为司机, 我可以通过计价器在等待时收取每分钟0.25元费用
- 作为司机, 我可以通过计价器在收费时去掉零头, 四舍五入至元

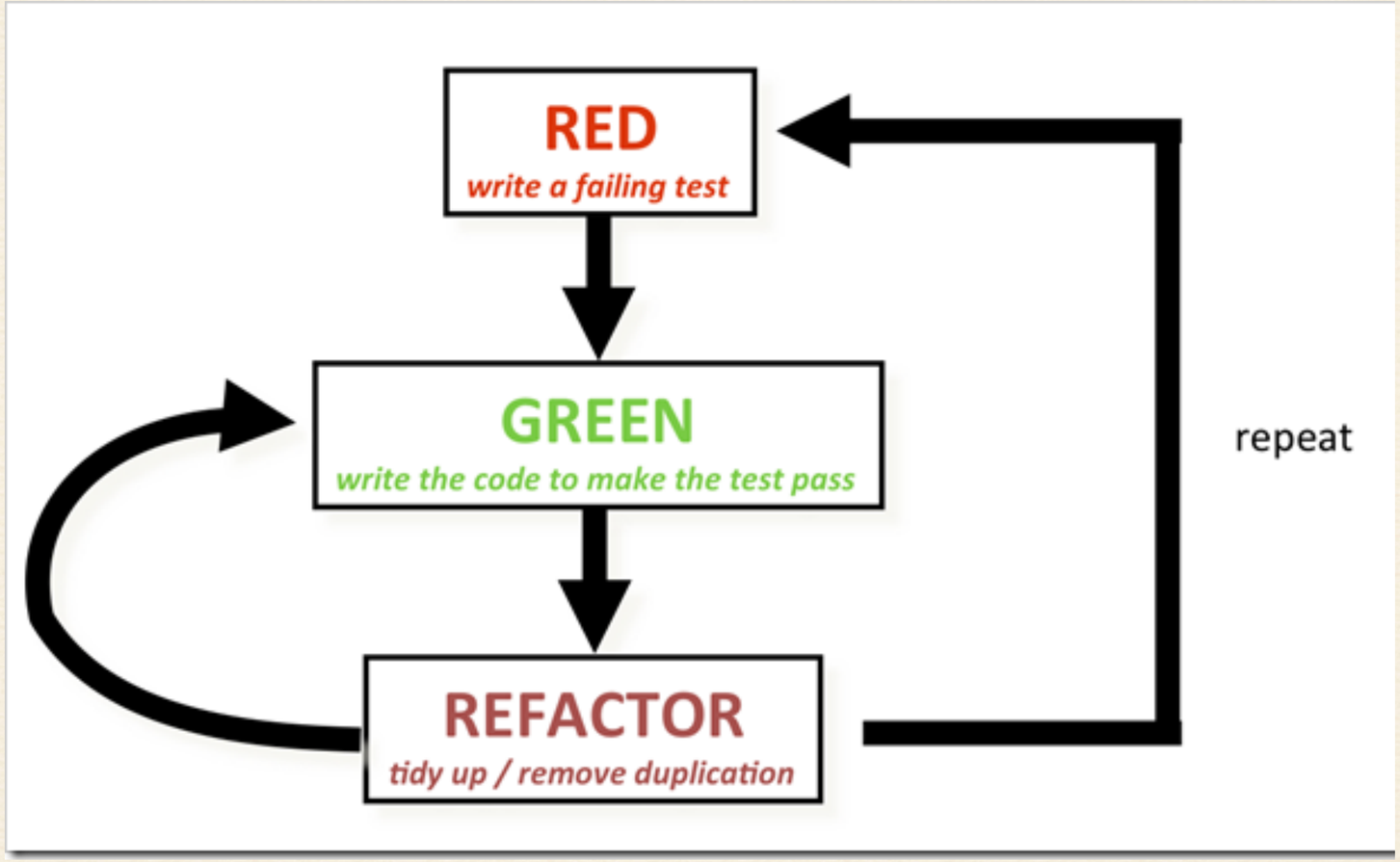
Tasking Tree



Tasking

- fee = distance * 0.8
- distance > 8 , fee = 0.8 * 50%
- base_rating = 6 , when distance <= 2
- waiting_charge = 0.25/min
- rounding

怎样TDD


RED
write a failing test
GREEN
write the code to make the test pass
REFACTOR
tidy up / remove duplication
repeat

Tasking

- **fee = distance * 0.8**
- distance > 8 , fee = 0.8 * 50%
- base_rating = 6 , when distance <= 2
- waiting_charge = 0.25/min
- rounding

原则

- 测试：一次只写一个刚好失败的测试，作为新加功能的描述；
- 实现：不写任何产品代码，除非它刚好能让失败着的测试通过
- 重构：只在测试全部通过的前提下，开始新加功能，或重构代码。

Tasking

- 分解问题：

  1. 任何子问题均可通过测试来验收
  2. 所有子问题域的集合恰好等价于原问题域
  3. 子问题域之间无交集

- 计划任务：

  能够递进地解决所有子问题

  其他任务与子问题放在同一列表中被计划

- 追踪进度：

  在列表中随时勾除完成任务或加入新增任务