三、单元测试

1. 【强制】好的单元测试必须遵守 AIR 原则。
   说明：单元测试在线上运行时，感觉像空气（AIR）一样并不存在，但在测试质量的保障上，却是非常关键的。好的单元测试宏观上来说，具有自动化、独立性、可重复执行的特点。
   - A：Automatic（自动化）
   - I：Independent（独立性）
   - R：Repeatable（可重复）
2. 【强制】单元测试应该是全自动执行的，并且非交互式的。测试框架通常是定期执行的，执行过程必须完全自动化才有意义。输出结果需要人工检查的测试不是一个好的单元测试。单元测试中不准使用 System.out 来进行人肉验证，必须使用 assert 来验证。
3. 【强制】保持单元测试的独立性。为了保证单元测试稳定可靠且便于维护，单元测试用例之间决不能互相调用，也不能依赖执行的先后次序。
   反例：method2 需要依赖 method1 的执行，将执行结果做为 method2 的输入。
4. 【强制】单元测试是可以重复执行的，不能受到外界环境的影响。
   说明：单元测试通常会被放到持续集成中，每次有代码 check in 时单元测试都会被执行。如果单测对外部环境（网络、服务、中间件等）有依赖，容易导致持续集成机制的不可用。
   正例：为了不受外界环境影响，要求设计代码时就把 SUT 的依赖改成注入，在测试时用 spring 这样的 DI 框架注入一个本地（内存）实现或者 Mock 实现。
5. 【强制】对于单元测试，要保证测试粒度足够小，有助于精确定位问题。单测粒度至多是类级别，一般是方法级别。
   说明：只有测试粒度小才能在出错时尽快定位到出错位置。单测不负责检查跨类或者跨系统的交互逻辑，那是集成测试的领域。
6. 【强制】核心业务、核心应用、核心模块的增量代码确保单元测试通过。
   说明：新增代码及时补充单元测试，如果新增代码影响了原有单元测试，请及时修正。
7. 【强制】单元测试代码必须写在如下工程目录：src/test/java，不允许写在业务代码目录下。
   说明：源码构建时会跳过此目录，而单元测试框架默认是扫描此目录。
8. 【推荐】单元测试的基本目标：语句覆盖率达到 70%；核心模块的语句覆盖率和分支覆盖率都要达到 100%
   说明：在工程规约的应用分层中提到的 DAO 层，Manager 层，可重用度高的 Service，都应该进行单元测试。

9．【推荐】编写单元测试代码遵守 BCDE 原则，以保证被测试模块的交付质量。

- B：Border，边界值测试，包括循环边界、特殊取值、特殊时间点、数据顺序等。
- C：Correct，正确的输入，并得到预期的结果。
- D：Design，与设计文档相结合，来编写单元测试。
- E：Error，强制错误信息输入（如：非法数据、异常流程、非业务允许输入等），并得到预期的结果。

10．【推荐】对于数据库相关的查询，更新，删除等操作，不能假设数据库里的数据是存在的，或者直接操作数据库把数据插入进去，请使用程序插入或者导入数据的方式来准备数据。

反例：删除某一行数据的单元测试，在数据库中，先直接手动增加一行作为删除目标，但是这一行新增数据并不符合业务插入规则，导致测试结果异常。

11．【推荐】和数据库相关的单元测试，可以设定自动回滚机制，不给数据库造成脏数据。或者对单元测试产生的数据有明确的前后缀标识。

正例：在 RDC 内部单元测试中，使用 RDC_UNIT_TEST_的前缀标识数据。

12．【推荐】对于不可测的代码建议做必要的重构，使代码变得可测，避免为了达到测试要求而书写不规范测试代码。

13．【推荐】在设计评审阶段，开发人员需要和测试人员一起确定单元测试范围，单元测试最好覆盖所有测试用例（UC）。

14．【推荐】单元测试作为一种质量保障手段，不建议项目发布后补充单元测试用例，建议在项目提测前完成单元测试。

15．【参考】为了更方便地进行单元测试，业务代码应避免以下情况：

- 构造方法中做的事情过多。
- 存在过多的全局变量和静态方法。
- 存在过多的外部依赖。
- 存在过多的条件语句。

  说明：多层条件语句建议使用卫语句、策略模式、状态模式等方式重构。

16．【参考】不要对单元测试存在如下误解：

- 那是测试同学干的事情。本文是开发手册，凡是本文内容都是与开发同学强相关的。
- 单元测试代码是多余的。汽车的整体功能与各单元部件的测试正常与否是强相关的。
- 单元测试代码不需要维护。一年半载后，那么单元测试几乎处于废弃状态。
- 单元测试与线上故障没有辩证关系。好的单元测试能够最大限度地规避线上故障。