# 【TDD-测试驱动开发实战营】

# 2019.10.20

<https://github.com/Choosue/tdd>

@熊节 Jeff Xiong 算上搭环境，一共用了 40 分钟……请熊老师指导一下，有什么问题？

①项目二Args，一个一般的开发者，完成时间是2-4小时，经过反复训练你会在27分钟内完成。

②实战营不是视频课，只有当你真正开始刻意训练才能收益。

③今天开始训练，我们会在最先的50条打卡记录里选出前3，赠送小米人工智能音响，打卡形式详见实战营。

TDD演示视频1、2: Python老司机实战FizzBuzz

<https://mp.weixin.qq.com/s/c45YeR6Re6FmsA4K7z2c-A>

<https://mp.weixin.qq.com/s/kMaGRKXy_cSVqe0zxQisoA>

是不是应该遵循"只允许写刚好让测试通过的代码"

　　　那个步子时而大，时而小，随性而为。但主体原则是按照TDD方式来做的。这个可以根据自己对业务的熟练度来调整，个人认为。

# Summary

## 【video】如何提升你的编程效率

<https://apppptz6pch9932.h5.xiaoeknow.com/content_page/eyJ0eXBlIjoxMiwicmVzb3VyY2VfdHlwZSI6NCwicmVzb3VyY2VfaWQiOiJsXzVkYTk5MDAwY2Y1MDlfTHFDa0lTTTciLCJwcm9kdWN0X2lkIjoiIiwiYXBwX2lkIjoiYXBwcHB0ejZQY0g5OTMyIiwiZXh0cmFfZGF0YSI6MH0=?entry=2&entry_type=2001&state=7c32d174260a68d4101ac9af72ff997a_v8H1Kp&app_id=apppptz6PcH9932>

## 【rule】TDD

**第一条规则：没有失败的测试就不能写代码**

**第二条规则：只允许写刚好让测试通过的代码**

**一定是自己练，练到一个关窍，想不明白，然后读到书里有一段恰好讲这个事，一下就懂了，记住了怎么都不会忘**

## 【rule】Agile

第一句？

第二句 **可工作的软件重于面面俱到的文档**

· 在软件生命周期的任何一分钟，整个软件是可用的，高质量的，符合需求描述的。任何人本地没有隔夜未提交的代码，任何代码提交之后软件仍然是可用、高质量、符合需求描述的状态。这才叫可工作的软件。

· 你可以在任何一天、任何一个时刻说，项目结束了，现在把软件拿出来发布。此时的软件是可用的，可以创造尽可能大的价值的。这才叫可工作的软件。

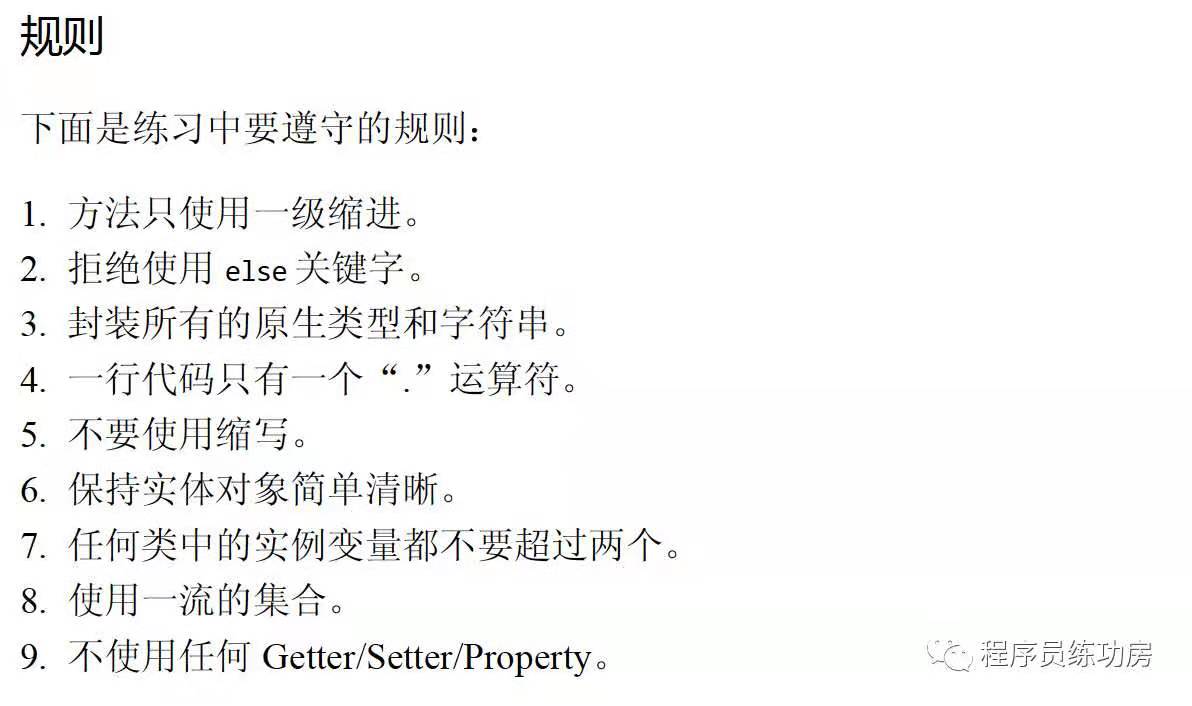
## 【rule】Code

Java对象节约内存

<https://mp.weixin.qq.com/s/twTkfoAF7hOa1xTK-87fRA>

**远离毫无依据的过早优化**

**学会写体面的面向对象代码**



Details:

[http://kaelzhang81.github.io/2018/06/10/%E5%AF%B9%E8%B1%A1%E5%81%A5%E8%BA%AB%E6%93%8D%E8%AF%A6%E8%A7%A3/?nsukey=fu8OQScFqRZt7WSeoRWL14bxRIMsIKqulQjVBQSyOAMl0gcVNv244%2BrrTE30fFeEambNG%2BxKSqcmfpx6fl8IdPKIGf8IeHQChrKXZBWqDsV6BBUCBsgzVj5TAgKDHHHRB9e4xzq4k1iDO64hpV07ZqO2caXaMW5awlZbTCW86%2BnRBC%2FeVHGOCt06Xjo6GJJzWZpNcu77eFjav41s4%2BpghA%3D%3D](http://kaelzhang81.github.io/2018/06/10/%E5%AF%B9%E8%B1%A1%E5%81%A5%E8%BA%AB%E6%93%8D%E8%AF%A6%E8%A7%A3/?nsukey=fu8OQScFqRZt7WSeoRWL14bxRIMsIKqulQjVBQSyOAMl0gcVNv244+rrTE30fFeEambNG+xKSqcmfpx6fl8IdPKIGf8IeHQChrKXZBWqDsV6BBUCBsgzVj5TAgKDHHHRB9e4xzq4k1iDO64hpV07ZqO2caXaMW5awlZbTCW86+nRBC/eVHGOCt06Xjo6GJJzWZpNcu77eFjav41s4+pghA==)

## 【conception】TDD

　　　1.测试驱动开发，是一种“开发方法”，不是测试方法；

　　　写代码前先写个测试，

　　　没有“开发方法”能“想到”你本来没有想到的事——这事交个测试就好了

## 【conception】Problem Grain

开发问题颗粒

　　　1）测试红色

　　　2）开发代码->绿色，（不超过2分钟）

　　　两者不超过五分钟；（问题的颗粒大小）

　　　3）重构（20秒）

　　　“颗粒”与“速度”紧密相关

　　　速度快，步子可以大一点；反之亦然；

## 【environment】Develop

　　　科学上网 + 英文作为主要工作语言

## 【reference】Book

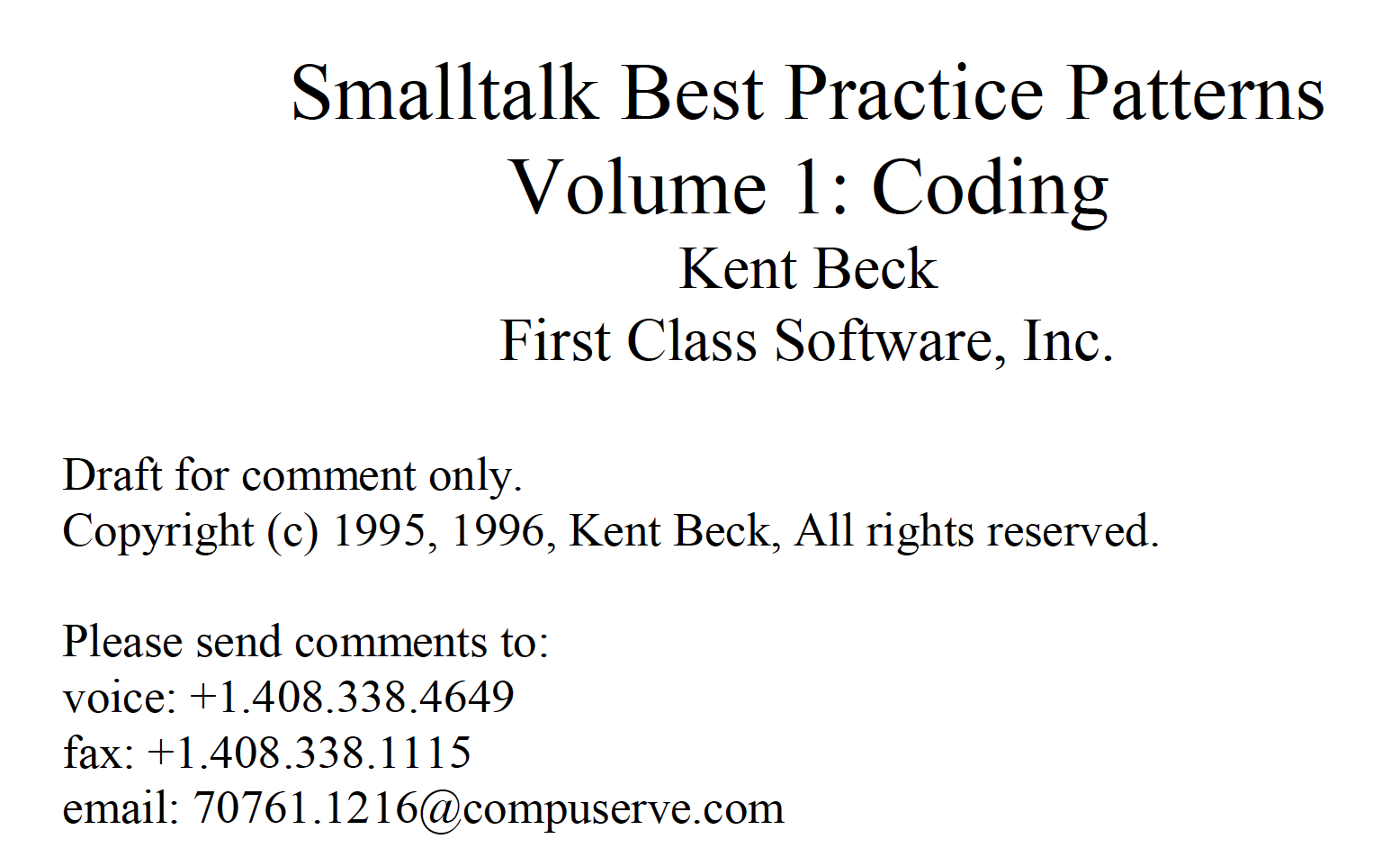
软件业三大必读经典

<https://mp.weixin.qq.com/s/0OgJxQ34Ijjd2Ch8KwvDTA>

《实现模式》、《重构》、《测试驱动开发》

重构与模式，设计模式，代码大全

实现模式的前身



## 【reference】article

<https://technologyconversations.com/2013/12/24/test-driven-development-tdd-best-practices-using-java-examples-2/>

## 【shortcut】IDEA





## 【关于领域模型和TDD】

· 极限编程有两个很重要实践

· 隐喻 和 CRC卡

需求用测试来表示，我感觉这一步就已经是拆分需求 和设计

任务拆解和领域模型不冲突，领域模型也是问题域拆解。

领域模型是对业务概念的抽象。包含了业务的属性和方法。是系统中最基本的组成。

· 画类图需要什么工具？

用rose

就是开始写代码前的思考

· 嗯，我用纸卡片。一般经验，**讨论20分钟的复杂度，大概开发三四天**

领域模型有部分作用也可以理解为，一种业务专家、架构师、用户、工程师、产品人员等都能理解的设计和架构方式。从业务角度抽象和聚焦对象、核心属性、行为、对象与对象的关系等。花点时间还是能交流清楚，就看愿不愿意。

就你们的话题，“沟通清楚（想明白、或者卡片等我觉得是编码之前的事）” 和 “下笔（写测试、编码）”有没有个大概的界限。比如拿到args这个题目，是不是理解用户，想到代码输入和输出，然后可以TDD了？就你们的话题，“沟通清楚（想明白、或者卡片等我觉得是编码之前的事）” 和 “下笔（写测试、编码）”有没有个大概的界限。比如拿到args这个题目，是不是理解用户，想到代码输入和输出，然后可以TDD了？

· 画一画也挺好

· 如果写测试你不知道该写 new Schema("l:boolean")，你更不可能画出这个类图。你上哪儿去想象出这几个对象来？

· 有个 "l:boolean d:string" 这样的字符串摆在面前，你都想不到要把它包装成一个对象，这个对象又包含n个小对象

· 打开一个test类，输入是什么就一股脑先建个对象。然后问这个对象，你想要的答案

老师，所以说“应该做个Schema,用来解析这个字符串”，于是写个测试类，有了输入，还能测输出？

· 是你首先看到 "l:boolean d:string" 这样一个字符串，你就应该问自己，这是个啥玩儿？

· 你要命名它，你说，哦，原来这货就是个schema

· 于是你就有了 new Schema("l:boolean d:string")

· 这句代码在哪儿写？在测试里

之前头一回看到args的实现，我眼睛看到印象很深的是，包好的这三个对象很融洽地相融了。我就在想，啥时候我也写多个对象，一下就融合到一起，之前都是用个本子写点东西，而不是把TDD驱动出的现成类融一起

· 因为你不需要发明什么，只是描述

· 很简单一个道理，你有N个用户，每个用户有类似 "l:boolean" 这样的schema，每个schema可以用来解析很多组参数。所以你把这些东西变成对象，他们肯定可以融在一起。这没有什么需要设计的，有啥你就写啥就完了

· 你把世界用对象描述出来，那么这些对象肯定能以原本的方式协作。如果不能协作，肯定是你的对象和世界不一样。因为世界是可以运作的。这也是为什么我说建新系统先在桌上把票据走通。

# Question:

## 【test】unfixed

没有输出的函数如何测试？

## 【test】fixed

Q：FizzBuzz项目在接到需求变更后，是修改之前的测试好呢？还是新增测试好呢？

熊：如不必要，最好新增测试，放在相关测试用例附近。原有业务逻辑还存在，所以原来的测试也原样存在，一般倾向于一个测试讲一件事，可以运行测试以后不看测试代码，只看测试报告，看是否说清楚了软件需求。

## 【git】

建议使用git的stage区，已经没问题的在stage区，正在写的在untracked

## 【requirement】

作为甲方，应该多想怎么改变，乙方来帮你想怎么落地的问题

# Task

## 【TDD】您未来21天的被虐清单如下：

[www.plus.jiker.com](http://www.plus.jiker.com)

----------------线上训练营以训练为主、学课为辅

一开始会不太适应

因为他需要您主动刻意练习

而不是被动等老师讲课

当您完成项目二

从2小时提高到40分钟

才能真正体会到TDD的用处

----------------

<http://codingdojo.org/kata/FizzBuzz/>

项目一：小试牛刀——FizzBuzz

任务数：4个

预计完成时间：7天

训练目标：

1.掌握测试驱动开发基本节奏

2.至少练习3遍FizzBuzz题目

3.能在10分钟内完成FizzBuzz题目

----------------

项目二：Args

任务数：11个

预计完成时间7天

训练目标：

1. 准确框定需求范围

2. 把需求拆解成明细的任务列表

3. 练熟测试驱动开发的节奏

----------------

项目三：MarsRover

预计完成时间：7天

任务数：3

训练目标：

1.反复练习测试驱动开发的节奏

2.练习识别和消除代码坏味道

需要您现在：

1.PC端登录训练营地址见微信群

（用手机号验证登陆，已绑定）

2.选择【测试驱动开发实战营】开始训练

3.有问题在群里随时和熊老师同步

4.7天后提交项目一作业

----------------

熊老师在群里全程答疑，请务必保持训练反馈

## 【TDD】reference

Kent Beck的《测试驱动开发》，可以只读第1章

## 【TDD+self】unit conversion

美国人习惯使用很古怪的英制度量单位。英制度量单位的换算经常不是十进制的，比如说：

\* 1英尺（foot）= 12英寸（inch）

\* 1码（yard）= 3英尺

\*\*任务：写一个程序，用于处理英寸、英尺、码之间的换算。\*\*

你的程序应该能很好地处理下面这些换算关系：

\* 1英尺 应该等于 12英寸

\* 1码 应该等于 3英尺

\* 1码 应该等于 36英寸

<https://www.bilibili.com/video/av73182203/>

## 【self】Kata

## 【self】split task

你可以找个dojo试试，让大家拆分任务，在纸上写下来

## 【self】the details of object calisthenics - Job Application domain 对象健身操详解

[http://kaelzhang81.github.io/2018/06/10/%E5%AF%B9%E8%B1%A1%E5%81%A5%E8%BA%AB%E6%93%8D%E8%AF%A6%E8%A7%A3/?nsukey=fu8OQScFqRZt7WSeoRWL14bxRIMsIKqulQjVBQSyOAMl0gcVNv244%2BrrTE30fFeEambNG%2BxKSqcmfpx6fl8IdPKIGf8IeHQChrKXZBWqDsV6BBUCBsgzVj5TAgKDHHHRB9e4xzq4k1iDO64hpV07ZqO2caXaMW5awlZbTCW86%2BnRBC%2FeVHGOCt06Xjo6GJJzWZpNcu77eFjav41s4%2BpghA%3D%3D](http://kaelzhang81.github.io/2018/06/10/%E5%AF%B9%E8%B1%A1%E5%81%A5%E8%BA%AB%E6%93%8D%E8%AF%A6%E8%A7%A3/?nsukey=fu8OQScFqRZt7WSeoRWL14bxRIMsIKqulQjVBQSyOAMl0gcVNv244+rrTE30fFeEambNG+xKSqcmfpx6fl8IdPKIGf8IeHQChrKXZBWqDsV6BBUCBsgzVj5TAgKDHHHRB9e4xzq4k1iDO64hpV07ZqO2caXaMW5awlZbTCW86+nRBC/eVHGOCt06Xjo6GJJzWZpNcu77eFjav41s4+pghA==)