TDD 実践入門

yamamoto kiyohiro

2021/10/25

研修の前に…

質問1

TDD（テスト駆動開発）の名前を聞いた事がある

[はい]　　[いいえ]

研修の前に…

質問2

TDD（テスト駆動開発）がどんな物か知っている

[はい]　　[いいえ]

研修の目的

- TDD のリズムについて知る
- TDD の思想について知る
- TDD について興味を持つ

参考書籍


Test-Driven Development
By Example
テスト駆動開発
Kent Beck 著
和田卓人 訳
テスト駆動開発（TDD）を
原点から学ぶ
本書は、Kent Beck, Test-Driven Development by Example
(Addison-Wesley, 2003)の新訳版です。
Ohmsha

今日の内容

- TDD とは
- ライブコーディング
- TDD のパターン振り返り

TDD とは

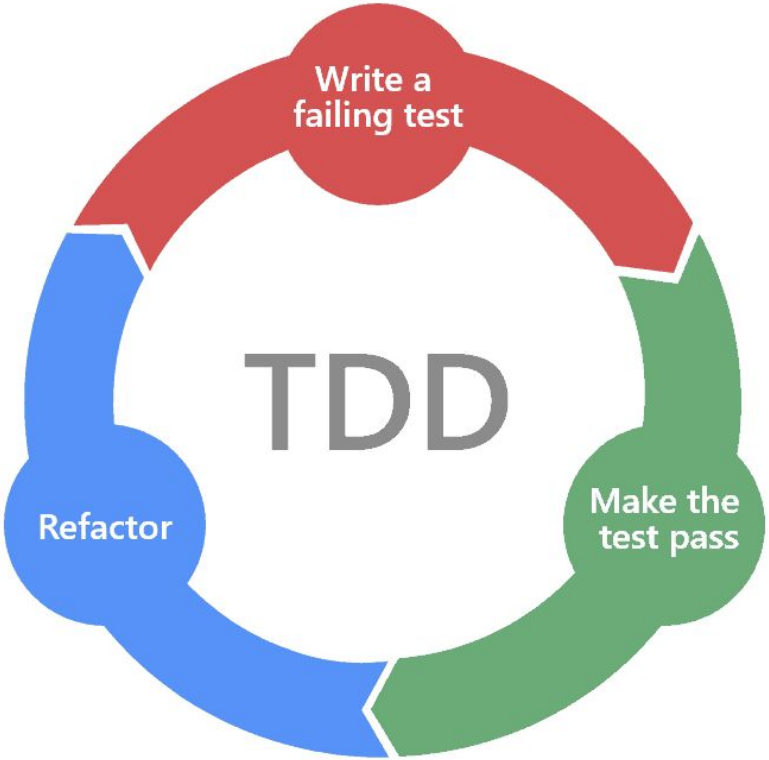
TDD とは

テスト駆動開発 (てすとくどうかいはつ、英: test-driven development; TDD) とは、プログラム開発手法の一種で、プログラムに必要な各機能について、最初にテストを書き（これを**テストファースト**と言う）、そのテストが動作する必要最低限な実装をとりあえず行なった後、コードを洗練させる、という短い工程を繰り返すスタイルである。多くのアジャイルソフトウェア開発手法、例えばエクストリーム・プログラミングにおいて強く推奨されている。近年[いつ?]はビヘイビア駆動開発へと発展を遂げている。

ウィキペディア
フリー百科事典

https://ja.wikipedia.org/wiki/テスト駆動開発

TDD とは

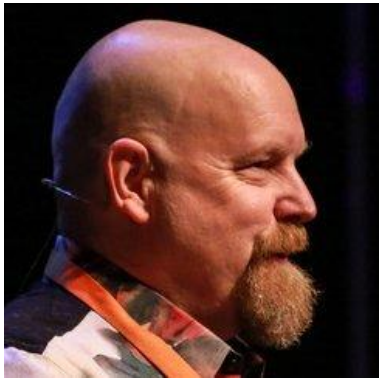


https://marsner.com/blog/why-test-driven-development-tdd/

TDD とは

テストを書く → 実装を書く → 洗練させる

のループを高速で回す開発手法

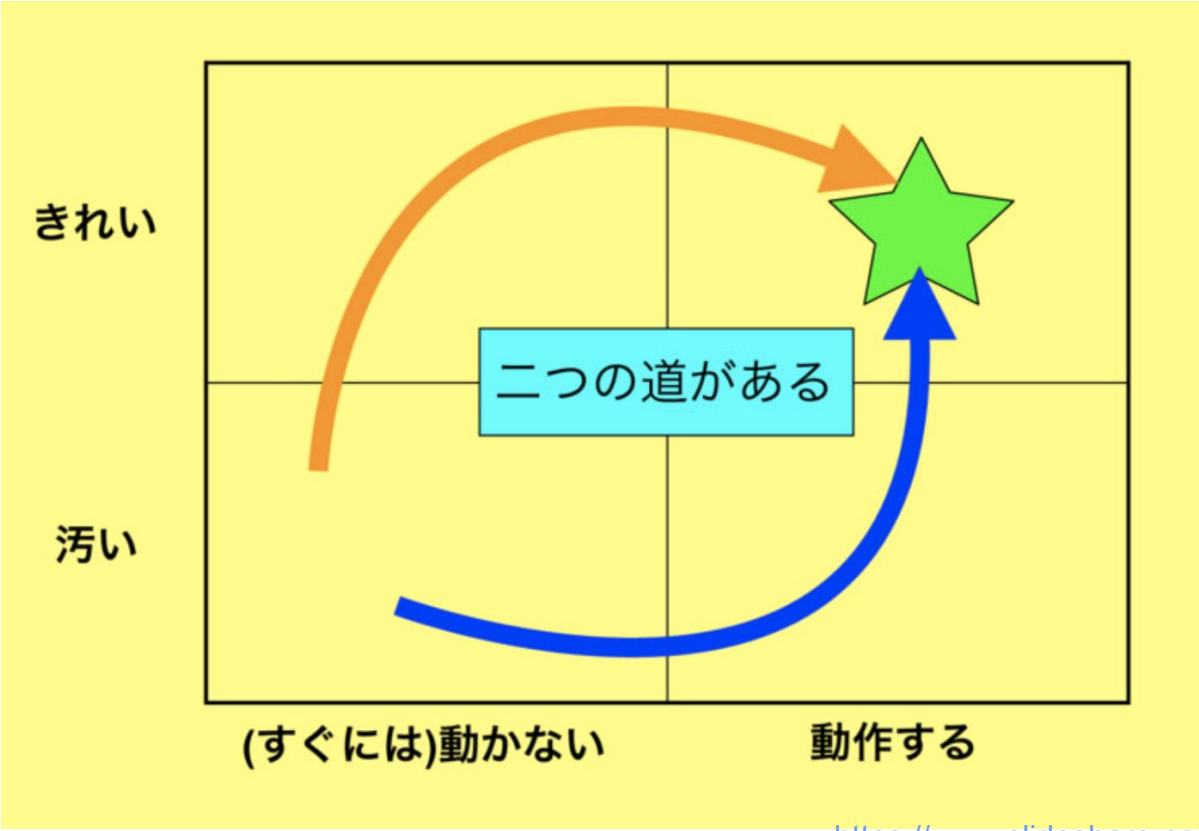


考案者は Kent Beck

- XP（Extreme Programming）考案者
- アジャイルマニフェスト起草者の1人
- JUnit 作者（エーリヒ・ガンマと共同）

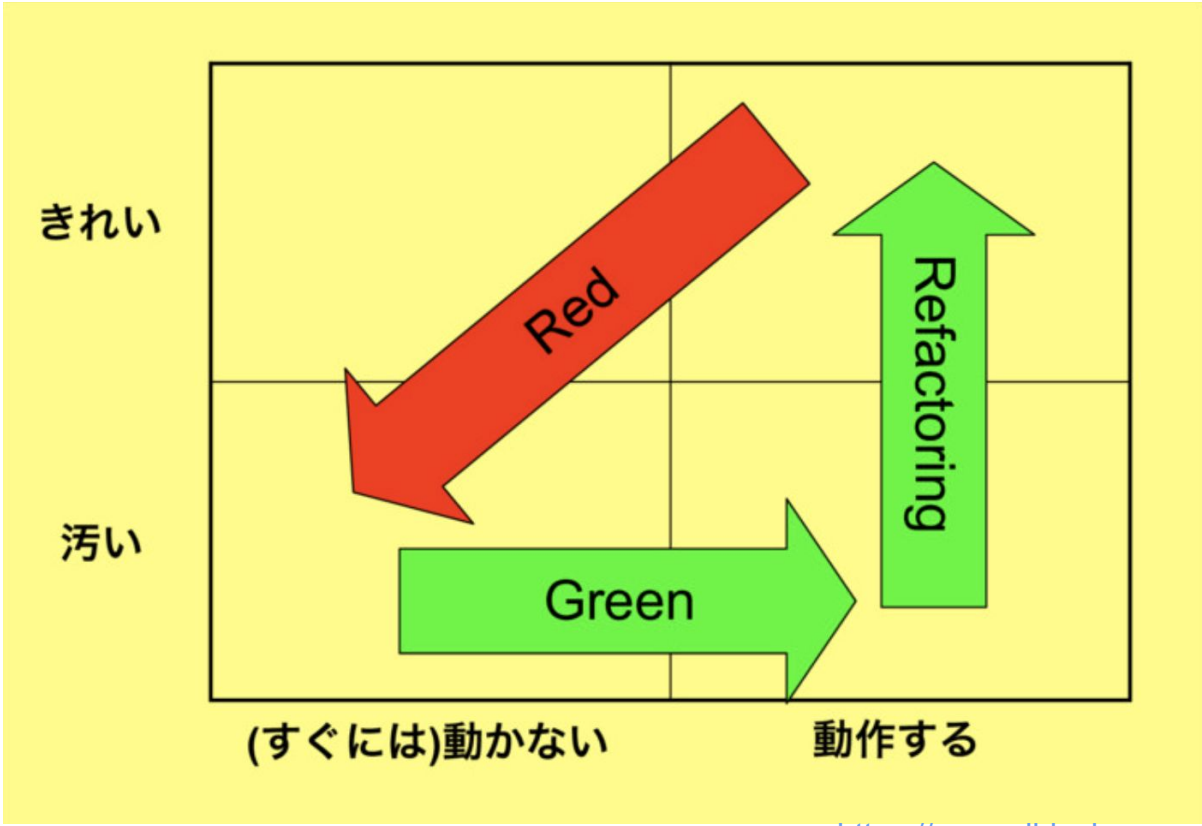
「動作するきれいなコード」がゴール

TDD とは：動作するきれいなコード



https://www.slideshare.net/t_wada/the-spirit-of-tdd

TDD とは：動作するきれいなコード



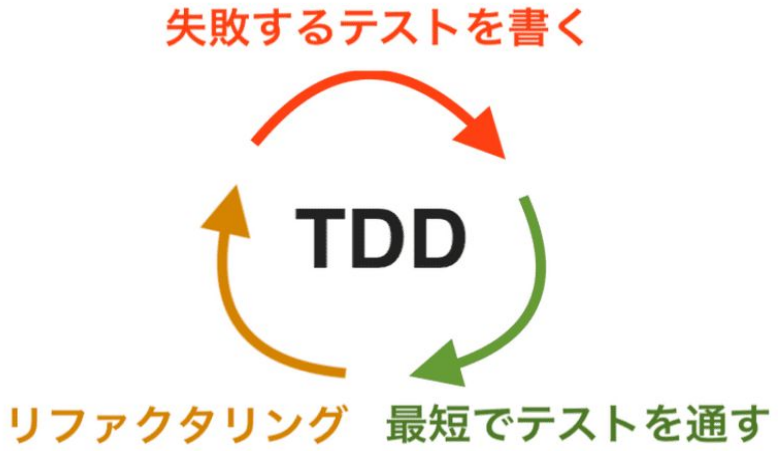
https://www.slideshare.net/t_wada/the-spirit-of-tdd

TDD とは：Red

1. 失敗するテストを書く

   欲しいと思うインタフェースを創造する

   当然コンパイルすら通らないけど、それでいい



https://panda-program.com/posts/test-driven-development

TDD とは：Green

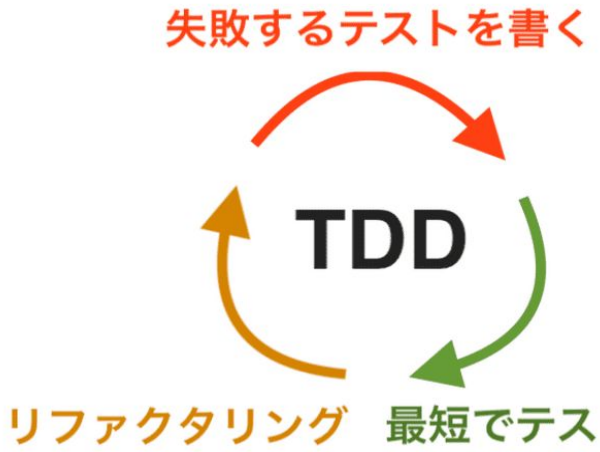
2. 最短でテストを通す

まずはコンパイルを通すところから

それが出来たら可能な限り早くテストを通す（グリーンにする）

どんな手を使ってもいい

→ 仮実装、明白な実装



https://panda-program.com/posts/test-driven-development

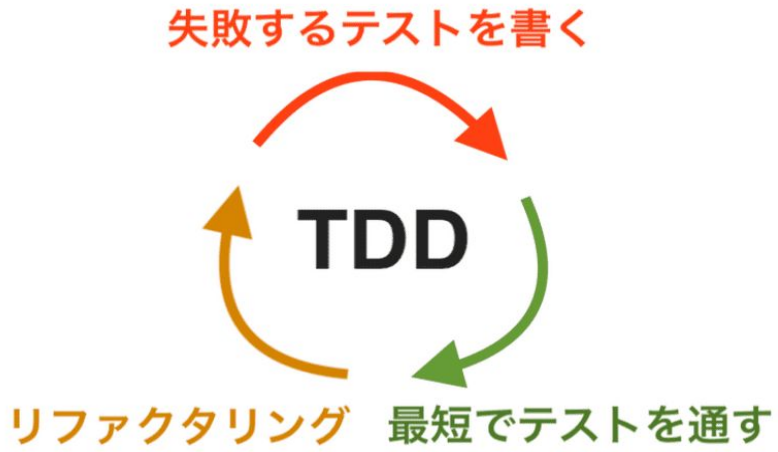
TDD とは：Refactoring

3. リファクタリング

グリーンである間は挙動が変わらないので内部実装をどう変えてもよい

修正は一歩ずつ行う

例: 引数の変更
→ 引数の追加、削除の2ステップに分解



https://panda-program.com/posts/test-driven-development

ライブコーディング

Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

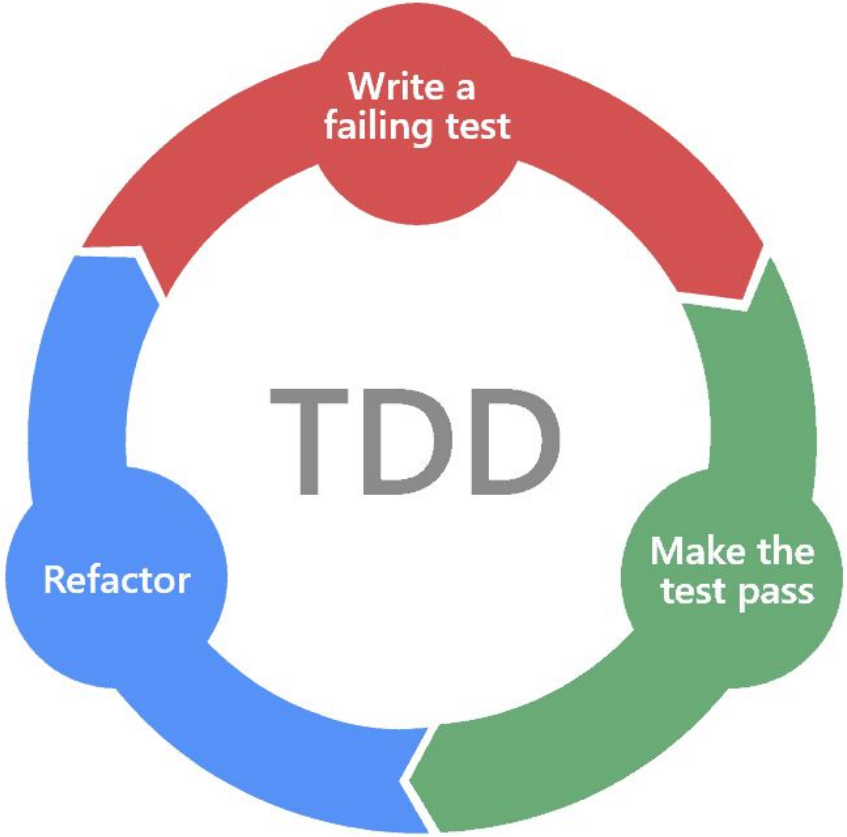
1から100までの数をプリントするプログラムを書け。ただし3の倍数のときは数の代わりに「Fizz」と、5の倍数のときは「Buzz」とプリントし、3と5両方の倍数の場合には「FizzBuzz」とプリントすること。

ライブコーディングまとめ

- 問題を小さく分割する
  - TDD のサイクルを高速に回すため
  - 1歩ずつ「動作するきれいなコードを目指す」

- 歩幅を調整する
  - スキルがつくほど高速に実装できるようになる
    テスト → 仮実装 → 三角測量 → 実装
    テスト → 仮実装 → 実装
    テスト → 明白な実装

- テストを構造化する
  - テストは実行できる仕様書



https://marsner.com/blog/why-test-driven-development-tdd/

研修の目的:再確認

- TDD のリズムについて知る
- TDD の思想について知る
- TDD について興味を持つ

TDD のパターン補足

TDD のパターン

独立したテスト

テストの実行は他のテストに絶対に影響を与えてはならない

問題を小さく直行した形に分解する必要がある（ここにスキルが必要）

凝集度が高く、かつ結合度が低い小さなオブジェクトを組み合わせた設計になる

TDD のパターン

TODO リスト

着手する前に、実装しなければならない振る舞いをリストに書き出す

頭に全部入れておくのは一般人には無理

今やっていることを見失わないよう、1つの作業に集中する

TDD のパターン

仮実装

心理的効果：すべてがグリーンならば、自信を持って一歩を踏み出せる

スコープ制御：本質と関係ない問題に気を取られなくなる

TDD のパターン

三角測量

- 仮実装だとたまたまテストが通っただけの場合がある
- テストケースを増やし、一般的な問題に対応出来ていることを確認する
- 正しい一般化が難しい場合に使う

TDD のパターン

明白な実装

仮実装や三角測量をするほどでもない場合はそのまま書けばよい

ただし、すべてを明白な実装にするのは無理（常に完璧な人間などいない）

「きれい」な「動く」コード

まず「動く」を倒すことを意識する