

とある道場の
乱取稽古
TDD Dojo

TDD 道場

～ みんな TDD やってみよう! ～



わんくま同盟 名古屋勉強会 - TDD 道場

TDD 道場 ～ 乱取り

- プロジェクトに接続された 1台の PC でコーディングする。
- ペアでコーディングする。
- **5 ～ 10分間隔でペアの片方を交代していく。**
聴衆 ⇒ ナビ、ナビ ⇒ ドライバー、ドライバー ⇒ 抜ける
- **コーディングを担当**しているときは、自分が何をしているのかを**説明しながらキーボードをタイプ**する。こうすることで聴衆も、何が起きているのかを理解できる。
- **聴衆**は、テストが **GREEN** の場合にだけ、設計について意見を述べてもよい。テストが **RED** の状態では、設計については質問しかできない。（**RED** の時に助け舟を出していいのは、アドバイザーのみ）
- **聴衆**が今行われている作業について混乱してきたら、コーディングしている人は手を止め、今やっていることを説明する。



ペアプログラミング

- ドライバー
 - キーボードを打つ
 - 目の前に集中
- ナビゲーター
 - 先を見る / 周囲に気を配る
- ペア プロの極意
 - 会話し続けること !!

ローカルルール

- 交代は5分を目安
 - テストケースを1つ書き、RED⇒GREENまで
 - 時間が余ったら、リファクタリング
- 最初に口頭でテストケースを宣言する
 - 「〇〇という入力的时候、△△になるケースを書きます」と宣言してから、テストケースを書き始める。

今回のルール

- リファクタリングだけやりましょう
 - 必要になったら、テストケースを追加してもいいです。
 - また、会場からも、どんどんアドバイスを出してください。

名古屋#14 の お題

- **CSVファイル読み書き** by C# & NUnit
 - ひとつとおり出来ている。(テストケースと製品コード)
 - **CsvConverter** クラスが**カッコ悪い!**
特に **ToDataTable()** メソッド (次スライド)
 - 現状: 2つのループ内で **Split()** メソッドを呼び出している ⇒ **1箇所にできないか?**

※ ちなみに。このサンプルコード書くためには、TDD三原則を少し破ることと、テストの粒度を大きくすることが必要でした。

(普段なら、ToDataTable() より先に、ToListList() メソッドを作ってる。)



わんくま同盟 名古屋勉強会 - TDD 道場

```
public static DataTable ToDataTable(IEnumerable<string> lines)
{
    DataTable dt = new DataTable();

    // テーブルに、必要な分だけのカラムを用意する
    int columnCount = 0;
    foreach (string s in lines)
    {
        IList<string> items = Split(s);
        columnCount = Math.Max(columnCount, items.Count);
    }
    for (int i = 0; i < columnCount; i++)
        dt.Columns.Add(new DataColumn());

    // 一行をカンマで切り分けて、テーブルの行として追加していく
    foreach (string s in lines)
    {
        DataRow r = dt.NewRow();
        r.ItemArray = Split(s).ToArray();
        dt.Rows.Add(r);
    }

    dt.AcceptChanges();
    return dt;
}
```



リファクタリング

- **GREEN**: ちゃんと動いていることを確認。
- **リファクター**: コードを綺麗に直す。
 - ※ ちょっとずつ進む!
 - ※ 分岐が増える? ⇒ テスト追加!
- **GREEN**: リファクターで壊していないね!

附録: リファクタリング カタログ

<http://www.refactoring.com/catalog/index.html>

■メソッドの構成

- ・メソッドの抽出 / インライン化
- ・一時変数のインライン化 / 説明用変数の導入
- ・メソッドによる一時変数の置き換え
- ・一時変数の分離 (スコープ分離)
- ・引数への代入の除去 (一時変数の導入)

■オブジェクト間での特性の移動

- ・メソッド / フィールド / プロパティの移動
- ・クラスの抽出 / インライン化
- ・外部メソッド (拡張メソッド) の導入

■データの再編成

- ・フィールドのプロパティ化
- ・オブジェクトによる複数データ値の置き換え
- ・単方向関連の双方向への変更 (あるいは逆)
- ・Enum や定数によるマジックナンバーの置き換え

■条件記述の単純化

- 条件記述のメソッドへの分解
- 条件記述の統合
- 制御フラグの削除 (中途脱出を使うことも考慮)
- ガード節による入れ子条件記述の置き換え
- ポリモーフィズムによる条件記述の置き換え
- ヌルオブジェクトの導入

■メソッド呼び出しの単純化

- メソッド名の変更
- 引数の追加 / 削除
- 問い合わせと更新の分離
- 引数オブジェクトの導入
- メソッドの隠蔽
- Factory Method によるコンストラクタの置き換え

…まだまだ沢山あるよ!



わんくま同盟 名古屋勉強会 - TDD 道場

道場 スタート

- ちょっと直したら、すぐテスト!
- ループを別メソッドに切り出すのは有効な手段だが … 今回ののはどうかな?
- 本来は、ひと区切りついたらチェックイン
※ 今回はリポジトリが無いので、出来ないけど。



附録: TDD 三原則

<http://yattom.jp/trac/public/wiki/TDDByUncleBobMartin>

- RED ⇒ GREEN

失敗するユニットテストを成功させるためにしか、プロダクトコードを書いてはならない。

- 失敗する(と思われる)テストケースだけ

失敗させるためにしか、ユニットテストを書いてはならない。コンパイルエラーは失敗に数える。

- テストをギリギリ通るだけ

ユニットテストを1つだけ成功させる以上に、プロダクトコードを書いてはならない。





わんくま同盟 名古屋勉強会 - TDD 道場