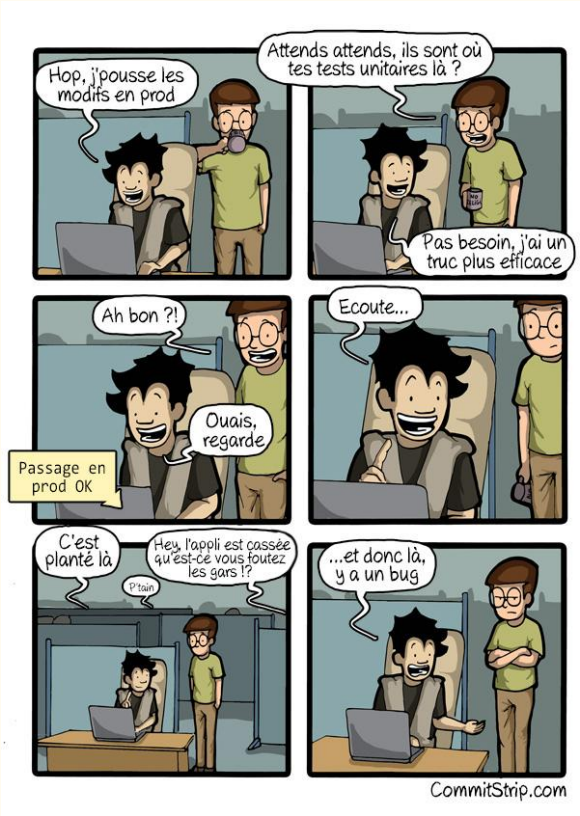
Améliorer son efficacité avec le TDD

—

Charlotte Cavalier & Christophe Pont

- Pourquoi vous ne testez pas
- Qu’est-ce qu’un test unitaire
- La bonne granularité
- Les bonnes pratiques
- Mise en pratique

Introduction



Top 5 des raisons pour ne pas tester :

- J’ai pas le temps
- C’est pas vendu
- Ça n'intéresse pas le client
- De toute façon ça sera pas maintenu
- Il reste toujours des bugs

How to test?

What to test:

- Public methods of the unit
- All branches of code execution within the unit
- For units that have direct user exposure (such as controllers in an MVC framework) unexpected inputs that a user may use to break the system

What not to test:

- Interaction between the tested unit and another unit
- Correct behavior of underlying libraries*
- Unmodified boilerplate code (including generated scaffolds)

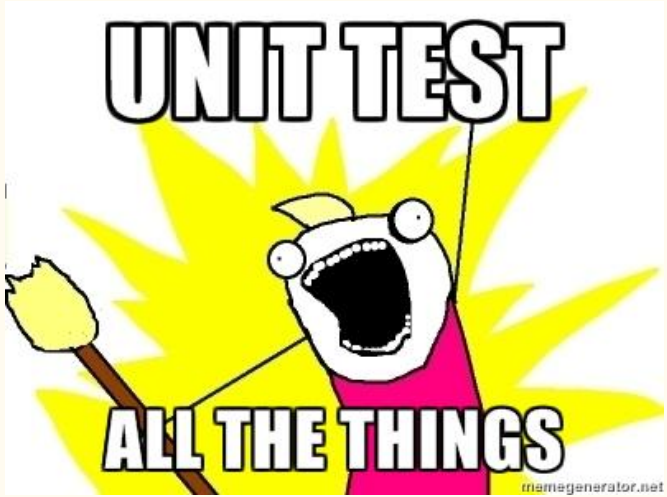
What to stub:

- Behaviors of other units (including external applications and libraries) that are necessary for this unit to function

Granularité

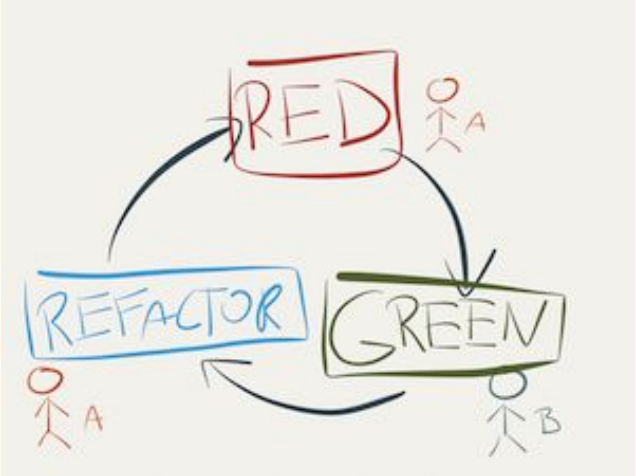
des tests UNITAIRES :

- Test d’une fonctionnalité
- Mock des dépendances
- Suffisamment petit



3 steps

- Red
  - Ecrire un test qui échoue
- Green
  - Le faire passer
- Refactor
  - L’améliorer



TDD


Red, green, look a squirrel! Push to production, 6 months later, refactor takes 3 days instead of 3 minutes.
- Byron Sommardahl

Uncle Bob’s 3 rules

1. You can't write any production code until you have first written a failing unit test.

2. You can't write more of a unit test than is sufficient to fail, and not compiling is failing.

3. You can't write more production code than is sufficient to pass the currently failing unit test.

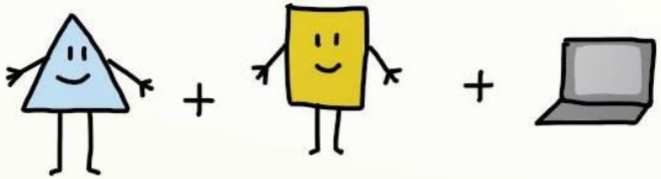
Coding Dojo: Kata

Fruit Shop

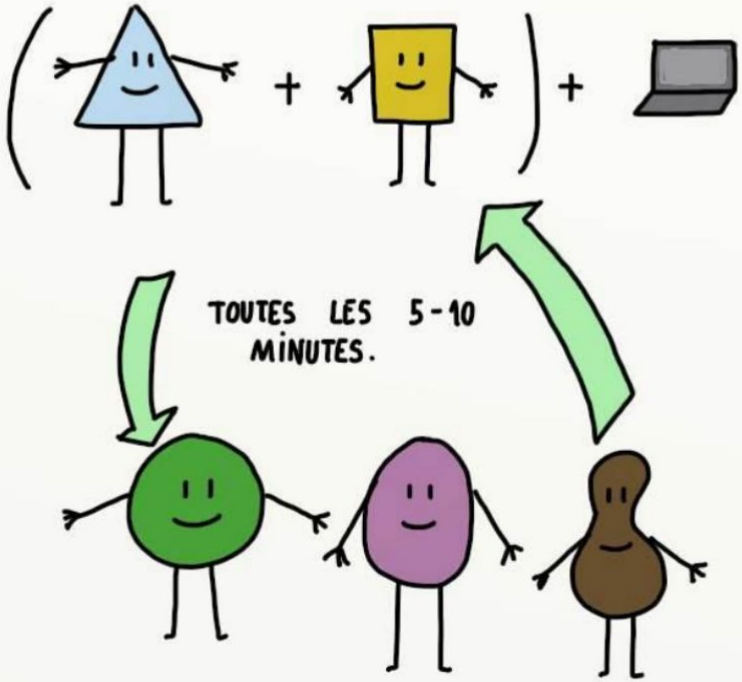
Coding Dojo

Inspiré du Coding Dojo des Duchesses (17/12/2016)

LE KATA



LE RANDORI*



*AUCUN RAPPORT AVEC UN PLAT INDIEN !

Atelier *Fruit Shop*

basé sur l’atelier de Jean-Laurent de Morlhon

Itération 0

- Un binôme : Christophe & Charlotte
- Une machine
- Un langage : Java :-)
- Un programme qui lit/écrit depuis l’entrée et la sortie standard

Itération 1

10 minutes !

- Faire une caisse enregistreuse simple
- avec 3 produits : Pomme (1€), Bananes (1,5€) & Cerise (0,75€)
- 1 article par ligne & afficher le total du panier en centimes à chaque fois

Tests:

```
Pommes > 100

Cerises > 175

Cerises > 250
```

Itération 1 - Vérification

```
Cerises > 75

Pommes > 175

Cerises > 250

Bananes > 400

Pommes > 500
```

Itération 2

10 minutes !

- On garde les mêmes entrées/sorties
- Appliquer des réductions:

  lots de cerises achetés, on a 20 centimes de réduction.

```
Pommes > 100

Cerises > 175

Cerises > 230
```

Itération 2 - Vérification

```
Cerises > 75

Pommes > 175

Cerises > 230

Bananes > 380

Cerises > 455

Cerises > 510

Pommes > 610
```

Itération 3

10 minutes !

- On supporte le format CSV:
  - plusieurs articles par entrée séparé par des virgules
- Même prix et réduction que l'itération 2

Tests:

```
Pommes, Cerises, Bananes > 325

Pommes > 425
```


BON... CHANGEMENT DE PROGRAMME... LES UTILISATEURS ONT DES NOUVELLES DEMANDES PLUS PRIORITAIRES...

Iteration 3’

10 minutes !

- Support du format CSV reporté
- On revient vers une entrée par ligne
- La réductio n pour les cerises passe à 30 centimes
- Un lot de banane acheté, le second est offert.

Tests:

```
Cerises > 75

Cerises > 120

Bananes > 270

Bananes > 270
```

Itération 3’ - Vérification

```
Cerises > 75

Pommes > 175

Cerises > 220

Bananes > 370

Pommes > 470

Bananes > 470

Cerises > 545
```

Itération 4

10 minutes !

- Support de la localisation:
  - On doit supporter les mots “Apples” et “Mele” pour “Pommes”
- La réduction pour les cerises repassent à 0,20€
- Tests :

```
Cerises > 75

Apples > 175

Cerises > 230

Bananes > 380

Bananes > 380
```

Itération 4 - Vérification

```
Cerises > 75

Apples > 175

Cerises > 230

Bananes > 380

Pommes > 480

Mele > 580
```

Itération 5

- Support de la localisation par article
- Le support du CSV est pour la prochaine itération
- 3 lots de “Apples” valent 2€ “Mele” valent 1,50€
- 2 lots de
- Tests:

```
Mele > 100
Apples > 200
Apples> 300
Pommes > 400
Apples > 400
Mele > 450
Cerises > 525
Cerises > 580
```

10 minutes !


BON... CHANGEMENT DE PROGRAMME... LES UTILISATEURS ONT DES NOUVELLES DEMANDES PLUS PRIORITAIRES...

Itération 5’

10 minutes !

- Support du format CSV
- 3 lots de “Apples” valent 2€
- 2 lots de “Mele” valent 1€

- Tests:

```
Mele, Apples, Apples, Pommes, Apples, Mele,
Cerises, Cerises, Bananes> 680
```

Itération 5’ - Vérification

```
Cerises, Apples > 175

Cerises > 230

Apples, Pommes, Bananes > 580

Apples, Pommes > 680

Mele > 780

Pommes > 880
```

Itération 6

10 minutes !

- Améliorer la qualité
- Bug: support du CSV et d'une entrée par ligne
  achetés, 1€ de réduction sur la facture globale
- 5 fruits achetées, 2€ de réduction
- Tests:

```
Mele, Apples, Apples, Mele > 200

Bananes > 150

Mele, Apples, Apples, Pommes, Mele > 150
```

Itération 6 - Vérification

```
Mele, Apples, Apples, Pommes,
Mele > 100

Bananes > 250
```

Références

- Source :
  - https://github.com/DuchessFrance/CodingDojoJava
- Solutions possibles :
  - https://github.com/jeanlaurent/cashregister
  - https://github.com/boucardbruno/FruitShop-Kata