TDD & Ping-Pong programming

Coding-Dojo

Julien Lafont & Vivian Pennel

Le programme

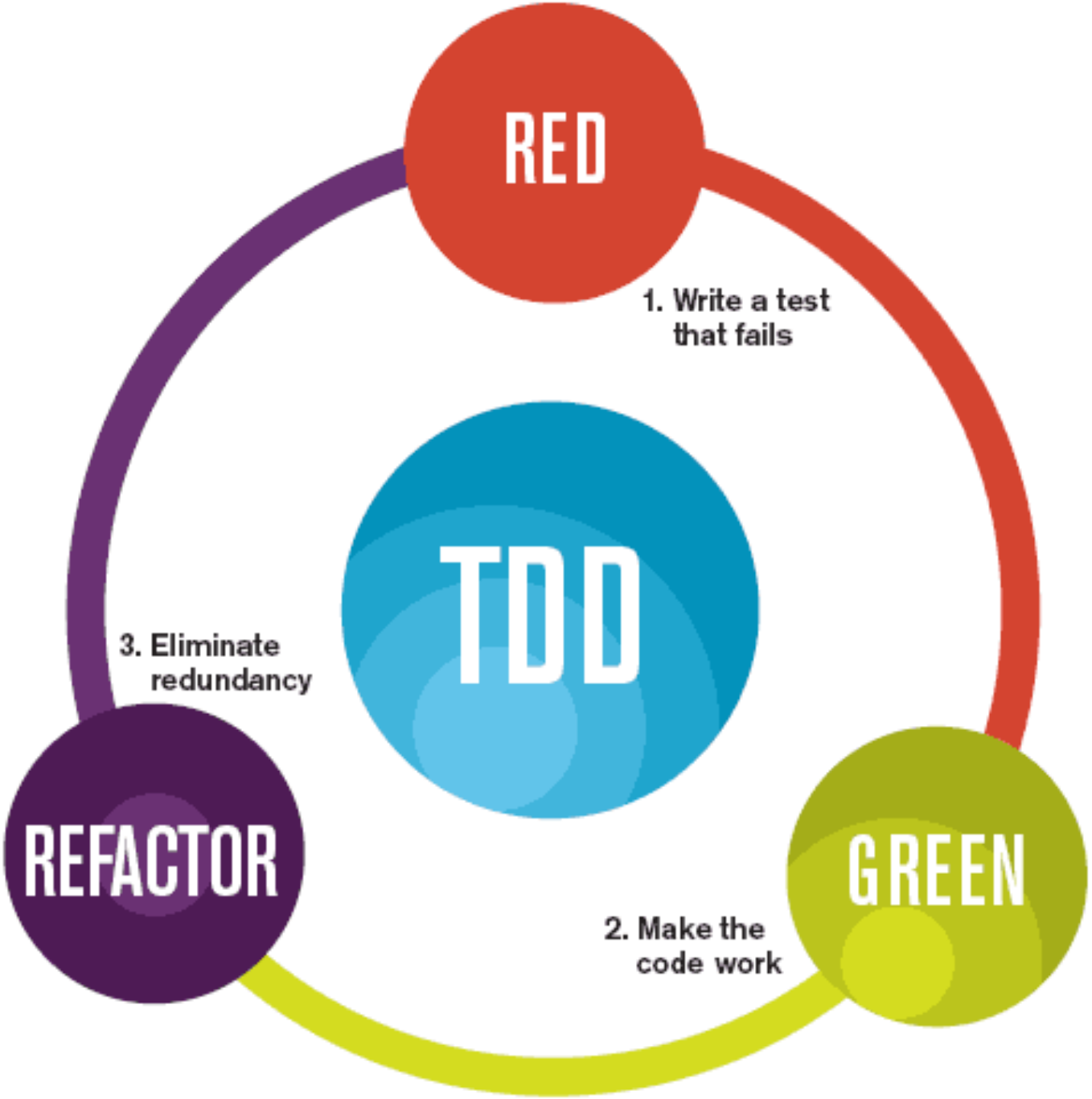
- Présentation rapide TDD et Ping-Pong Programming
- Explication des règles du jeu
- A vos claviers !
- Tour de table

TDD : Késako ?

Développement piloté par les tests : technique qui préconise d'écrire les tests unitaires avant d'écrire le code source d'un logiciel.

Le cycle préconisé par TDD comporte cinq étapes :

• Ecrire un premier test
• Vérifier qu'il échoue (car le code qu'il teste n'existe pas), afin de vérifier que le test est valide
• Ecrire juste le code suffisant pour passer le test ;
• Vérifier que le test passe
• Puis refactoriser le code, c'est-à-dire l'améliorer tout en gardant les mêmes fonctionnalités.



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Ping-Pong Programming

Une des nombreuses méthodes pour travailler en binôme.

- Le pilote écrit le code pour répondre au problème.
- Le copilote est là pour aider en le guidant, suggérant de nouvelle possibilités ou en décelant d'éventuels problèmes.

Ping-Pong Programming en pratique

while(! isComplete()) {

- A écrit un nouveau test qui échoue
- B implémente le code **minimum** permettant de passer le test
- B écrit le prochain test et vérifie qu'il échoue
- A implémente le code minimum pour passer le test

}

Kata du Jour : KataTennis

Vous avez 1 heure !

Kata du Jour : KataTennis

Objectif

Implémenter le calcul du **score** dans une partie de tennis.

On se focalise sur un seul **jeu** (jeu/set/match)

Les règles du jeu

- Les points sont gagnés suivant cette séquence : 0 → 15 → 30 → 40 → Victoire
- Marquer avec un score de 40 et au moins 1 balle d'avance → Victoire

- Si les deux joueurs ont 40 points → Egalité
- Marquer avec un Egalité donne l'Avantage
- Marquer avec l'Avantage → Victoire
- L'avantage se perd si le joueur adverse gagne le point (on revient à Egalité)

Spécifications

- Les joueurs peuvent marquer un point
- Une partie se termine avec un gagnant
- Pouvoir afficher le score après chaque point gagné (0-0, 0-15, 15-15, ...)
- Bonus : Gérer le cas du Deuce

Par où commencer ?

Ecrire un premier test

```
class TennisTest {

  @Test
  public testInitialisationNouveauJeu() {
    new Jeu();
  }

}
```

... qui échoue => **Ping** (on échange le clavier)

Et maintenant ?

Faire passer le test de la manière la plus simple possible

```
class Jeu {

}
```

Le test passe *(pfiou c'était dur !)*

Prochaine étape : Ecrire un nouveau test

Phase 2 : Test

```
class TennisTest {

  @Test public testInitialisationNouveauJeu() {..}

  @Test
  public testAfficherScoreDebutDePartie() {
    Jeu jeu = new Jeu();
    assertThat(jeu.score(), equalTo("0-0"));
  }
}
```

... qui échoue => **Pong**

Phase 2 : implémentation

```
class Jeu {
    private int[] scores = {0,0};

    public String score() {
       StringBuilder sb = new StringBuilder();
       sb.append(this.scores[0]).append("-")
          .append(this.scores[1]);
       return sb.toString();
}
```

```
class Jeu {
    public String score() {
         return "0-0";
    }
}
```

Retour aux règles

```
describe("Système de calcul du score au tennis", function() {

    it("Une partie est jouée entre exactement deux joueurs", function() {
        var game = tennis.game("Julien", "Marc");
        (game.players[0].name).should.equal("Julien");
        (game.players[1].name).should.equal("Marc");
    })

    it("Au début de la partie, le score est de 0-0", function() {
        newTestGame().forEachPlayer(function(player) {
            (player.score).should.equal(0);
        })
    });

    it("Quand un joueur gagne la première balle, son score est de 15", function() {
        var game = newTestGame();
        game.playerWin(player1);
        (game.players[player1].score).should.equal(15);
    })

    it("Quand un joueur gagne 4x d'affilé, le match est terminé et il est le gagnant",
fun
        var game = newTestGame();
        for (var i in fourTimes) game.playerWin(player1);
        (game.winner).should.equal(player1);
    })
```

Système de calcul du score au tennis

- ✓ Une partie est jouée entre exactement deux joueurs
- ✓ Au début de la partie, le score est de 0–0
- ✓ Quand un joueur gagne la première balle, son score est de 15
- ✓ Quand un joueur gagne 4x d'affilé, le match est terminé et il est le gagnant
- ✓ Quand un joueur gagne 3 balles et son adversaire 4, le match n'est pas terminé
- ✓ .. Et le second second joueur a l'Avantage
- ✓ Quand un joueur qui a l'avantage marque un nouveau point, il gagne le match
- ✓ Quand le joueur qui a l'avantage perd le point, le score revient à 'Egalité'
- ✓ Plusieur situations 'Egalité' peuvent s'enchainer sans que le match soit terminé
- ✓ Il n'est pas possible de marquer de point une fois le match terminé

✔ 10 tests complete (6ms)

4 rules of Simple Design

- Run (and pass!) all the tests
- Contains no dupplication
- Expresses the intent of the programmer

"Code that explains itself without the need of extraneous documentation"

- Has no superflous part

"Always implement things when you actually need them, never when you just foresee that you need them."