

PLAN

- Présentation rapide
- Démonstration
- Premier exercice
- Points complémentaires
- Deuxième exercice
- Conclusion

TDD

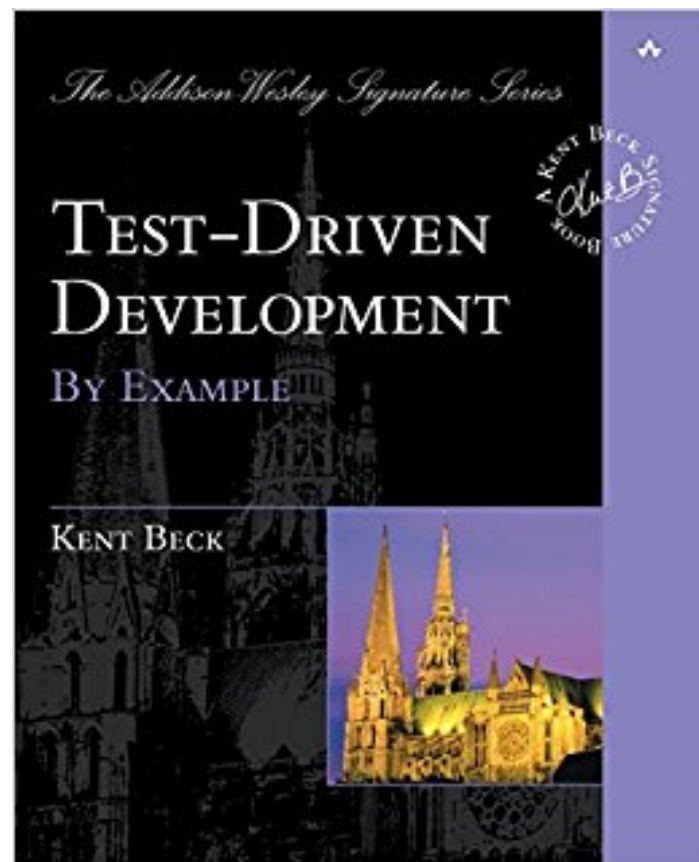
- Le sujet principal n'est pas le test
- C'est un process de développement guidé par les exemples
- L'objectif est de maintenir un rythme de développement

KENT BECK



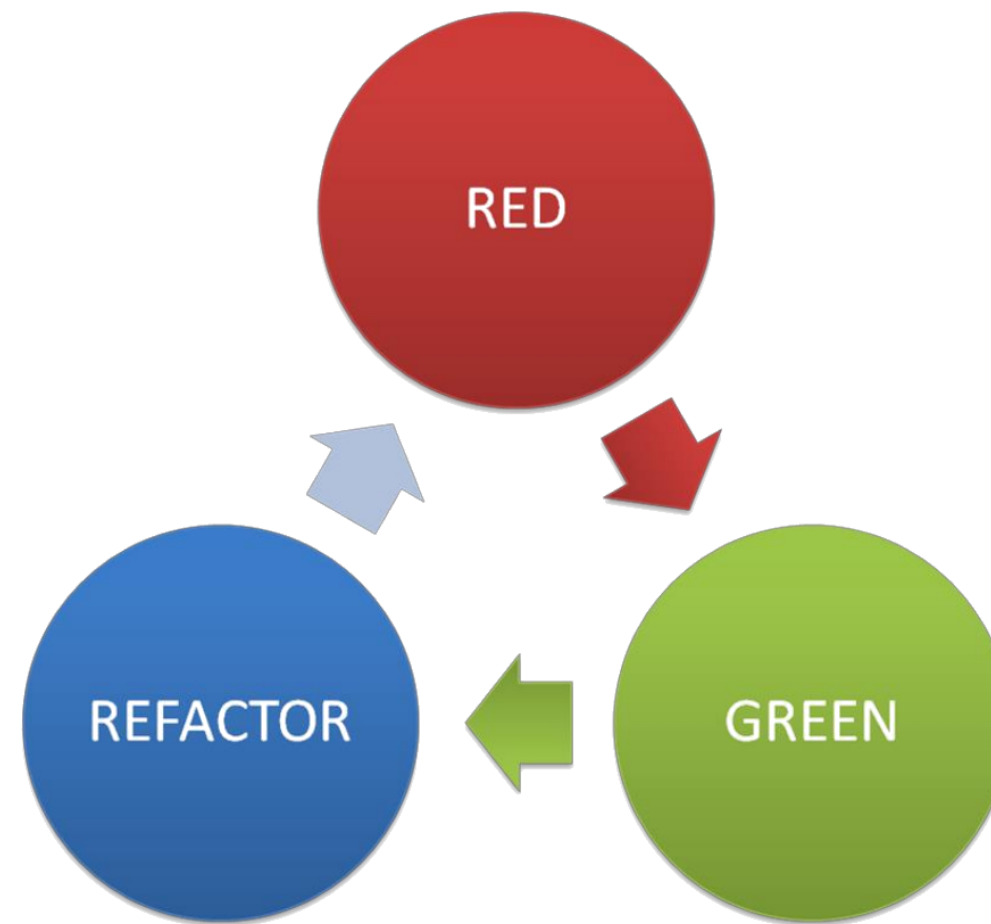
- Extreme programming
- Manifeste agile
- Test-driven development
- JUnit

PRÉFACE



- N'écrivez pas une ligne de code tant que vous n'avez pas d'abord un test automatique qui échoue
- Eliminez la duplication

TDD MANTRA



UN EXEMPLE

- Une calculatrice gérant l'addition de plusieurs entiers

CONCEPTS CLÉS

- Ecrire un test et s'assurer qu'il échoue
 - Les valeurs en « dur » orientent le test suivant
- Ecrire « du » code pour satisfaire le test
 - Passer rapidement au vert
 - Implémentation rapide
- Ecrire « le » code que l'on va garder
 - Renommage, restructuration
 - Changement d'implémentation

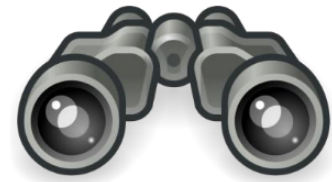
QUESTIONS



PREMIER EXERCICE - FIZZBUZZ

- Implémenter la fonction FizzBuzz qui prend en paramètre un nombre entre 1 et 100
- Par défaut, retourner le nombre sous forme de chaîne de caractères
- Si le nombre est un multiple de 3, retourner Fizz
- Si le nombre est un multiple de 5, retourner Buzz
- Pour les nombres multiple de 3 et de 5, retourner FizzBuzz

RÉTROSPECTIVE



POINTS D'ATTENTION

- Se focaliser sur le comportement/besoin et non la manière
- Penser utilisation avant implémentation
- Ne pas modifier le test et le code en même temps
- Prendre autant soin des tests que du code

BONNES PRATIQUES DE TEST

- Indépendance
- Rapidité d'exécution
- Reproductibilité
- Lisibilité
- Tester une seule chose à la fois
- Le nom du test indique l'objectif

OUTILLAGE

- Framework de tests: Junit, TestNG
- Outils de build: Maven, Ant
- IDE: Eclipse, IntelliJ, NetBean
- Intégration continue: Jenkins, Travis CI, GitLab CI
- Couverture de code: Cobertura, Emma
- Analyse de code: Sonar, Checkstyle, PMD, ...
- Plugins: MoreUnit, Inifinitest

SPÉCIFICATION / DOCUMENTATION

- Spécification exécutable
 - Code lisible
 - Indépendant de l'implémentation
- Documentation à jour
 - Exemple d'utilisation du code
 - Spécification du comportement

CONCEPTS AGILES

- KISS (Keep It simple, stupid)
 - On commence par une implémentation triviale
 - On restructure pour simplifier
- YAGNI (You Ain't Gonna Need It)
 - On ne développe que ce qui est nécessaire pour faire passer un test
 - On écrit un test que pour décrire un cas utilisateur

FEEDBACK

- Baby steps
 - Approche itérative très courte
 - Construction organique
- Feedback
 - Retour immédiat
 - Rythme de développement

MESURE DE LA COUVERTURE

- La couverture est assurée par construction
- On ne s'en préoccupe pas spécialement

QUALITÉ DE CODE

- Refactoring
 - Modification de l'implémentation sans changer le comportement
 - Elimination de la duplication
 - Amélioration de l'implémentation

BDD: BEHAVIOR DRIVEN DEVELOPMENT

- Continuité du TDD
- Encore plus orienté vers le métier
- Rédaction en collaboration avec le métier
- Syntaxe Gerkin: Given / When / Then

```
Scénario: Compléter toute ma todo liste  
  Etant donné que j'ai 2 tâches dans ma todo liste  
  Lorsque je complète toutes mes tâches  
  Alors ma todo liste est vide
```

SECOND EXERCICE

- Bowling Game
- Game of life
- How much water ?
- Tennis score

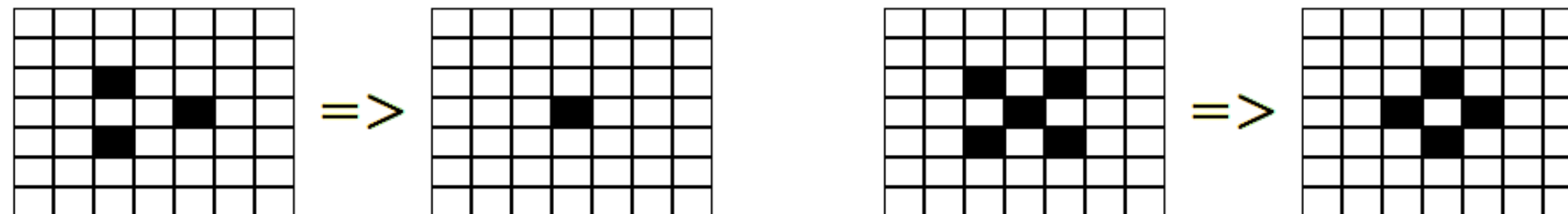
SCORE DE BOWLING

- La grille de bowling est constituée de 10 cadres.
- Pour chaque cadre, le joueur à deux lancers pour faire tomber les 10 quilles.
- Le score du cadre est le nombre de quilles tombées plus un bonus en cas de spare ou de strike.
- Il y a spare lorsque qu'un joueur fait tomber toutes les quilles en deux coups. Le bonus est le nombre de quilles tombées au coup suivant
- Il y a strike lorsque toutes les quilles tombent au premier essai. Le bonus est le score des deux coups suivants.

[illegible]

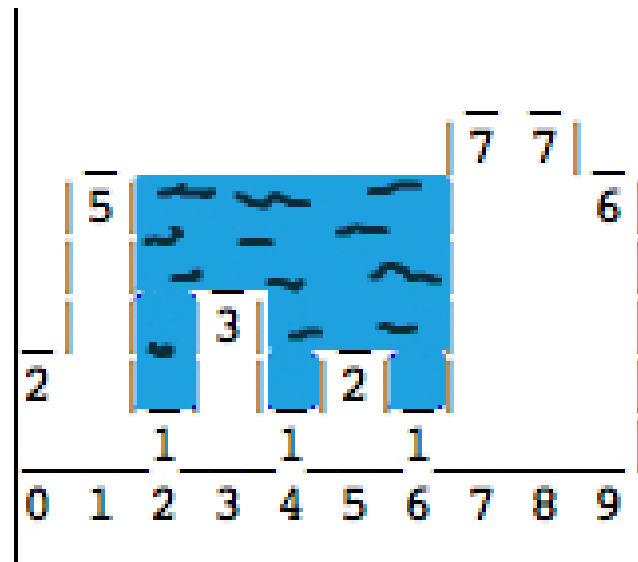
LE JEU DE LA VIE: JOHN CONWAY

- Pour un emplacement 'peuplé':
 - Une cellule avec un ou aucun voisin meurt de solitude.
 - Une cellule avec quatre voisins ou plus meurt de surpopulation.
 - Une cellule avec deux ou trois voisins survit.
- Pour un emplacement 'vide' ou 'non peuplé'
 - Une cellule avec trois voisins devient peuplée.



HOW MUCH WATER ?

- Etant donnée une liste d'entiers représentant les hauteurs de colonnes
- On cherche la quantité d'eau qui resterait prisonnière des cuvettes formées par les colonnes



- Exemple :
 - Valeurs: 2, 5, 1, 3, 1, 2, 1, 7, 7, 6
 - Résultat: 17

TENNIS SCORE

- Afficher le score d'un jeu au tennis à partir d'une liste indiquant qui a marqué chaque point
- Les points valent 15, 30, 40
- Au delà de 40,
 - le premier avec 2 points d'écart gagne le jeu: "Game [A ou B]"
 - s'il n'y a pas 2 points d'écart, on affiche "Advantage [A ou B]"
 - s'il y a égalité, on affiche "Deuce"

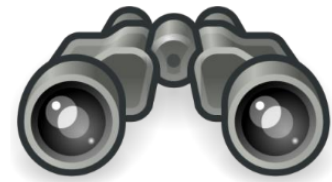
Exemples

AAAB \Rightarrow 40 - 15

BBBB \Rightarrow Game B

ABABABA \Rightarrow Advantage A

RÉTROSPECTIVE



LES POINTS DIFFICILES

- Les méthodes privées
- Les contributeurs
- Rester indépendant de l'implémentation
- Tester sur du code existant
- Conception émergente

BÉNÉFICES

- Composants prévus pour être testés
- Composants prévus pour être réutilisés
- Capacité à faire évoluer/modifier le code
- On sait ce qui marche ou pas
- Projet auto validé
- Rapidité d'analyse des défauts

BÉNÉFICES

- Le tests ne sont plus une option "lorsqu'il reste du temps"
- On ne perd pas du temps à écrire les tests, on gagne du temps pour écrire le code

RÉFÉRENCES

- *TDD*
Jason Gorman, 2016
- *Extreme programming explained: embrace change*
Kent Beck, Addison-Wesley, 1999
- *Test-Driven Development: By Example*
Kent Beck. Addison-Wesley, 2002
- *Test-Driven Development: A Practical Guide*
David Astels. Prentice Hall, 2003
- *Growing Object-Oriented Software, Guided by Tests*
Steve Freeman, Nat Pryce, 2009

SITES

- Cyber dojo: <http://cyber-dojو.org/>
- CodingDojo: <http://codingdojo.org/kata/>
- Yosethegame: <http://yosethegame.com/>
- Coding Game: <https://www.codinggame.com/start>

VIDÉOS

- [TDD : pour que votre code soit testable et testé!](#) - Xavier Nopre, 2019
- [Xavier Screen Cast #3, Premier test unitaire et intro TDD](#), présentation du TDD à partir de 6mn
- [Test-Driven Development \(TDD\) in Python #1 - The 3 Steps of TDD](#) - Jason Gorman

MEETUPS

- <https://www.coderetreat.org>
- <https://www.meetup.com/fr-FR/Craft-your-Skills/>
- <https://www.meetup.com/fr-FR/CARA-Dojo/>
- <https://www.meetup.com/fr-FR/Software-Craftsmanship-Lyon>

Pour apprendre le TDD,
il faut le pratiquer