Université de Lille 1 SCIENCES ET TECHNOLOGIES

UFR IEEA
Formations en Informatique de Lille 1

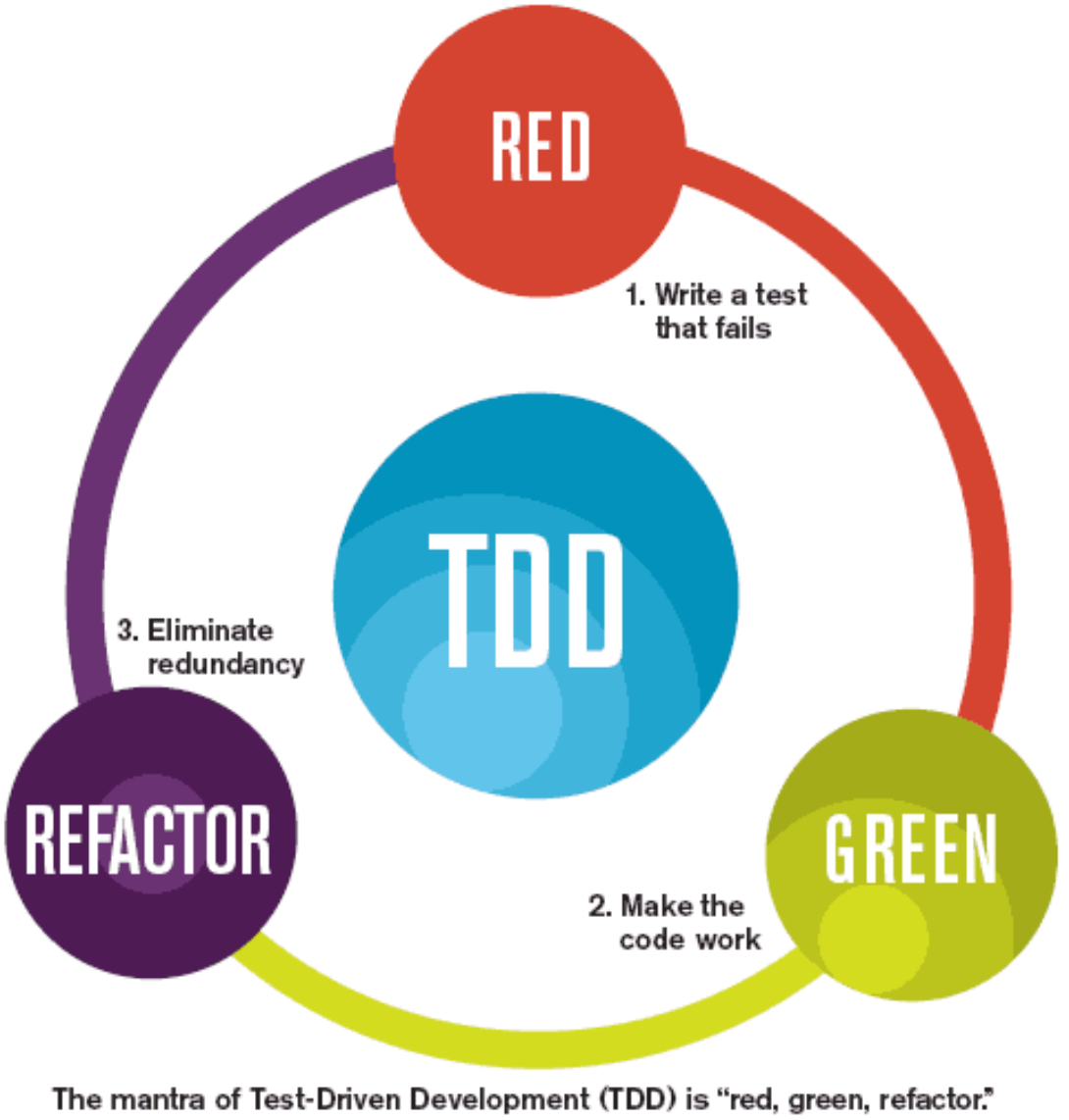
Conception Orientée Objet

Romain Rouvoy
Licence mention Informatique
Université Lille 1

Menu du jour

1. Rappels
2. Kata
3. Rigueur
4. XP
5. Yin et yang

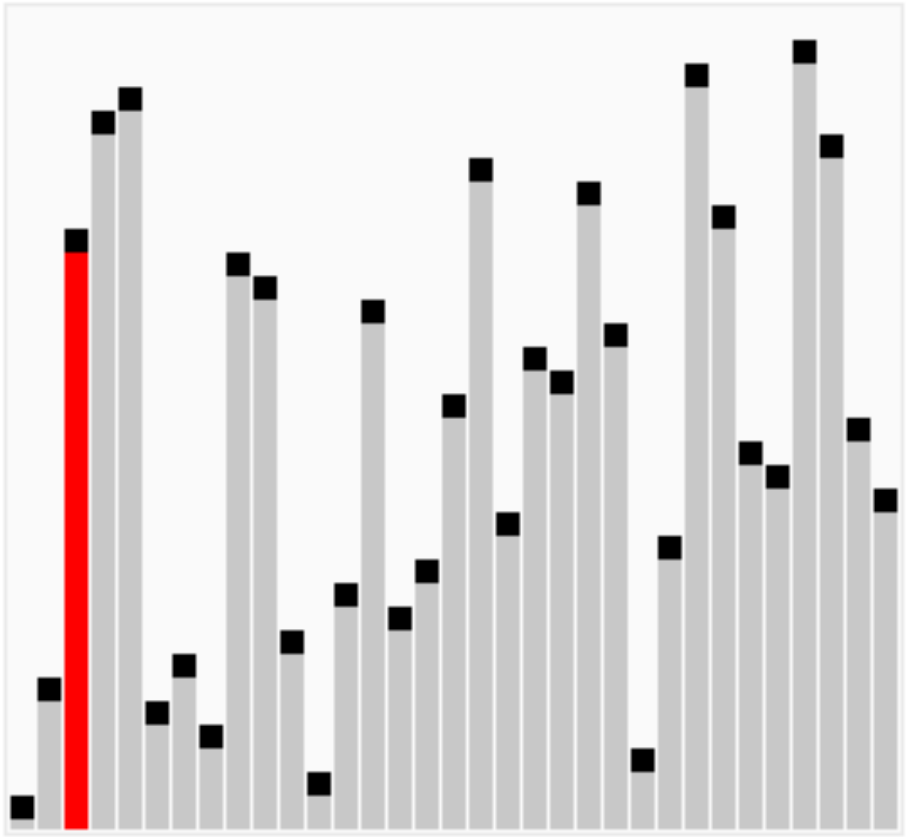
KISS
DRY
YAGNI



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

Bubble Sort Kata

- https://www.youtube.com/watch?v=AraZURpk_YA



Le bon message de commit

- Each commit message consists of a **header**, a **body** and a **footer**. The header has a special format that includes a **type**, a **scope** and a **subject**:

```
<type>(<scope>): <subject>
<BLANK LINE>
<body>
<BLANK LINE>
<footer>
```

- The **header** is mandatory and the **scope** of the header is optional
- Any line of the commit message cannot be longer 100 characters! This allows the message to be easier to read on GitHub as well as in various git tools

Type de commit

- **`feat`**: A new feature
- **`fix`**: A bug fix
- **`docs`**: Documentation only changes
- **`style`**: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)
- **`refactor`**: A code change that neither fixes a bug nor adds a feature
- **`perf`**: A code change that improves performance
- **`test`**: Adding missing tests
- **`chore`**: Changes to the build process or auxiliary tools and libraries such as documentation generation

Portée et sujet du commit

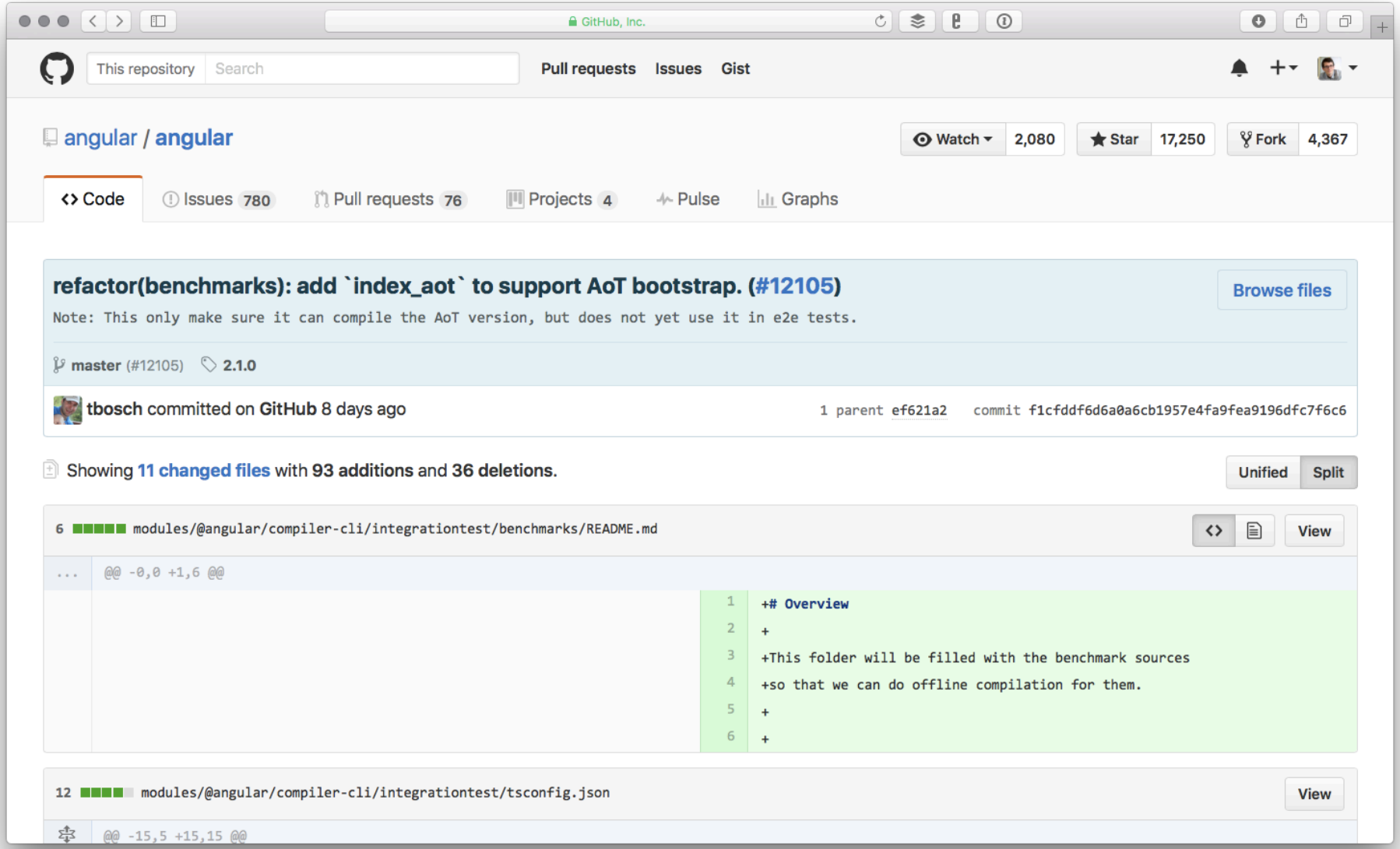
- The **scope** could be anything specifying place of the commit change. For example, `Division`, `Manager`, etc.

- The **subject** contains succinct description of the change:
  - use the imperative, present tense: "change" not "changed" nor "changes"
  - don't capitalize first letter
  - no dot (.) at the end

Corps et pieds du commit

- Just as in the **subject**, use the imperative, present tense: "change" not "changed" nor "changes". The body should include the motivation for the change and contrast this with previous behavior

- The footer should contain any information about **Breaking Changes** and is also the place to reference GitHub issues that this commit **Closes**
- **Breaking Changes** should start with the word BREAKING CHANGE: with a space or two newlines. The rest of the commit message is then used for this


GitHub, Inc.
This repository
Search
Pull requests
Issues
Gist
angular / angular
Watch 2,080
Star 17,250
Fork 4,367
Code
Issues 780
Pull requests 76
Projects 4
Pulse
Graphs
refactor(benchmarks): add `index_aot` to support AoT bootstrap. (#12105)
Note: This only make sure it can compile the AoT version, but does not yet use it in e2e tests.
Browse files
master (#12105)
2.1.0
tbosch committed on GitHub 8 days ago
1 parent ef621a2
commit f1cfddf6d6a0a6cb1957e4fa9fea9196dfc7f6c6
Showing 11 changed files with 93 additions and 36 deletions.
Unified
Split
6 modules/@angular/compiler-cli/integrationtest/benchmarks/README.md
View
@@ -0,0 +1,6 @@
+# Overview
+
+This folder will be filled with the benchmark sources
+so that we can do offline compilation for them.
+
+
12 modules/@angular/compiler-cli/integrationtest/tsconfig.json
View
@@ -15,5 +15,15 @@

eXtreme Programming

Harder, Better, Faster, Stronger

eXtreme Programming (XP)

- puisque la **revue de code** est une bonne pratique, elle sera faite en permanence (par un binôme)
- puisque les **tests** sont utiles, ils seront faits systématiquement avant chaque mise en œuvre
- puisque la **conception** est importante, elle sera faite tout au long du projet (*refactoring*)
- puisque la **simplicité** permet d'avancer plus vite, nous choisirons toujours la solution la plus simple
- puisque la **compréhension** est importante, nous définirons et ferons évoluer ensemble des métaphores
- puisque **l'intégration** des modifications est cruciale, nous l'effectuerons plusieurs fois par jour
- puisque les besoins évoluent vite, nous ferons des cycles de développement **très rapides** pour nous adapter au changement

Pair programming

- Facteurs d’échec
  - **Silence** : la programmation en binôme nécessite de programmer à voix haute et de partager son point de vue avec son partenaire. Un silence persistant indique un manque de collaboration
  - **Désengagement** : un des membres se désintéresse du projet et vaque à ses occupations
  - **Effacement** : lorsqu'un membre est plus expérimenté qu'un autre, le novice se contente d'observer l'expert réaliser la majorité des tâches de développement
  - **Problèmes relationnels** : les deux membres du binôme ne s'entendent pas et ne souhaitent pas travailler ensemble

The Pomodoro Technique

1. **Decide** on the task to be done.
2. Set the timer to **25 minutes**.
3. **Work** on the task until the timer rings.
4. Take a short 5 minute **break**.

repeat 4 times

5. Take a 15-30 minute **break**.

FRACTUS LEARNING

#bitesizePD

Vers l’intégration continue

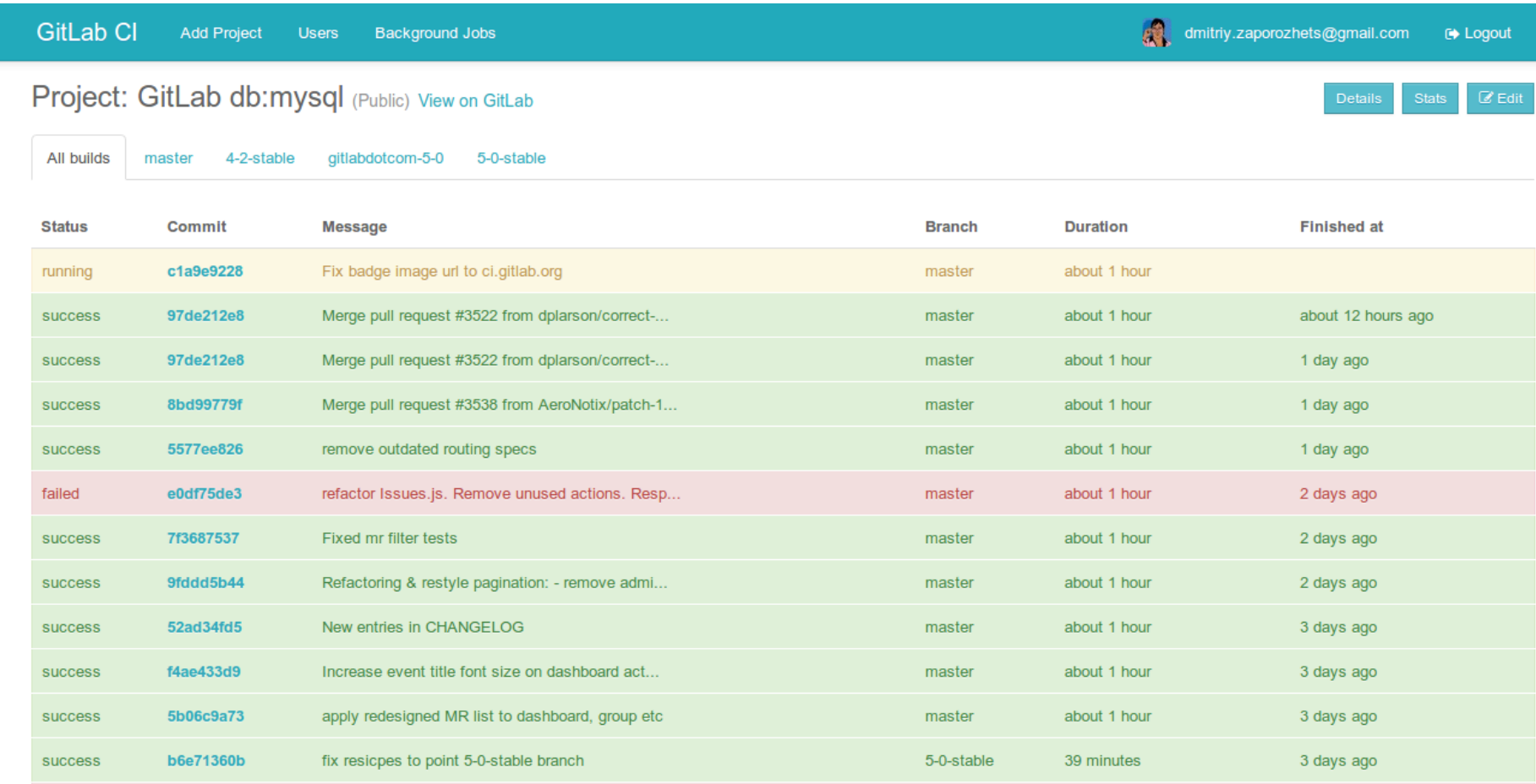
- Serveur tiers [gitlab-ci]
  - Récupère la dernière version du code [git]
  - Compile le code [maven]
  - Compile les tests [maven]
  - Exécute les tests sur le code [maven]
  - Si OK, package le code [maven]
  - (Déploie le code) [maven]

Vers l'intégration continue

- Maintain a **single source repository**
- **Automate** the build
- Make your build **self-testing**
- **Every commit should build** on an integration machine
- Keep the **build fast**
- Test in a **clone of the production** environment
- Make it **easy** for anyone to get the latest executable
- Everyone can **see what's happening**
- **Automate** deployment

Vers l’intégration continue

- Check in frequently
- Don’t check in broken code
- Don’t check in untested code
- Don’t check in when the build is broken
- Don’t go home after checking in until the system builds


GitLab CI
Add Project
Users
Background Jobs
dmitriy.zaporozhets@gmail.com
Logout
Project: GitLab db:mysql (Public) View on GitLab
Details
Stats
Edit
All builds
master
4-2-stable
gitlabdotcom-5-0
5-0-stable
Status
Commit
Message
Branch
Duration
Finished at
running
c1a9e9228
Fix badge image url to ci.gitlab.org
master
about 1 hour
success
97de212e8
Merge pull request #3522 from dplarson/correct-...
master
about 1 hour
about 12 hours ago
success
97de212e8
Merge pull request #3522 from dplarson/correct-...
master
about 1 hour
1 day ago
success
8bd99779f
Merge pull request #3538 from AeroNotix/patch-1...
master
about 1 hour
1 day ago
success
5577ee826
remove outdated routing specs
master
about 1 hour
1 day ago
failed
e0df75de3
refactor Issues.js. Remove unused actions. Resp...
master
about 1 hour
2 days ago
success
7f3687537
Fixed mr filter tests
master
about 1 hour
2 days ago
success
9fddd5b44
Refactoring & restyle pagination: - remove admi...
master
about 1 hour
2 days ago
success
52ad34fd5
New entries in CHANGELOG
master
about 1 hour
3 days ago
success
f4ae433d9
Increase event title font size on dashboard act...
master
about 1 hour
3 days ago
success
5b06c9a73
apply redesigned MR list to dashboard, group etc
master
about 1 hour
3 days ago
success
b6e71360b
fix resicpes to point 5-0-stable branch
5-0-stable
39 minutes
3 days ago

Code smells & Anti-patrons

- Mauvaises pratiques de développement
- Peuvent apparaître lors de la conception ou du dev.
- S'oppose aux *design patterns*
- Peuvent engendrer
  - Problèmes de performances
  - Problèmes de maintenance
  - Coût de développement accru
  - Comportements anormaux
  - Bugs

Interface de constantes

```
public interface Constantes {
  public static final String MY_NAME = "my_name";
  // etc.
}

public class MaClasse implements Constantes {
  public static final String MY_NAME = "my_class";

  private String name = MY_NAME;
  // etc.
}
```

- Quel est le problème ?
- Quelle est la solution ?

Interface de constantes

```
public final class Constantes {
  private Constantes() {
    throw new UnsupportedOperationException();
  }

  public static final String MY_NAME = "my_name";
  // etc.
}


public class MaClass {
   private String name = Constantes.MY_NAME;
   // etc.
}
```

Méthodes longues

- Méthode contenant trop d'instructions
  - Difficile à tester
  - Difficile à maintenir
- Mauvaise décomposition
- Correction : **KISS (décomposition)**

Méthodes longues

- Méthode contenant trop d'instructions
  - Difficile à tester
  - Difficile à maintenir

- Mauvaise décomposition

- Correction : **KISS (décomposition)**
  - Démo: https://youtu.be/y4_SJzNJnXU?t=38s

Classes larges

- Problème analogue à la méthode longue mais généralisé à la classe complète
- Problème apparaissant souvent dans les classes UI
- Correction : **KISS (décomposition)**

Ancre de bateau (*lava flow*)

- Aka *code mort*
  - Code rapidement mis en production
  - Code inutilisé mais conservé
  - Raison politique…

- Correction : **YAGNI**

Abstraction inverse

- Définition d'une interface inadéquate
  - `float divide(float, float)` pour une division entière
- Conduit à un usage plus complexe
  - `Math.floor(divide(5,2));`
- Correction : **KISS**

Code spaghetti

- Dérive d'un PoC (*quick and dirty*)
- Impossible de modifier la classe sans impacter d'autres classes du système
  - Beaucoup de méthodes sans paramètres
  - Couplage très fort avec d'autres classes
  - Utilisation de variables globales
- Correction : **abstractions + polymorphisme**

Objet divin (*God object – BLOB*)

- Classe qui accumule trop de fonctions essentielles
  - Contraire au principe OO : *diviser pour régner*
- Beaucoup trop de méthodes/attributs
  - *E.g.*, plus 60 attributs dans une seule classe !
- Défaut de conception
  - Faiblesse de la phase de spécification
- Correction : **KISS**

Feature Envy

- Méthode qui appelle beaucoup de getters d'une autre classe

```
public class Bateau {
  private SPI spi;
  private Reserve reserve;
  // ...
  public void preparer_bateau() {
    this.laver();
    this.verifier_coque();
    reserve.verifier_provisions();
    spi.regler_hauteur();
    spi.regler_largeur();
    spi.verifier_voile();
} }
```

- Correction : **Déplacer la méthode vers la classe appelée || Regrouper les appels en une seule méthode**

Classes Poltergeist

- Prolifération de classes
  - Classes sans état
  - Cycle de vie très court
  - Peu de responsabilités
  - Associations temporaires
- Impacte les performances
- Complexité excessive

- Correction : **KISS**

En résumé

- 3 principes clés
  - KISS, DRY, YAGNI (+ SOLID)
- Tester c’est douter
  - Tests fonctionnels, structurels, non-fonctionnels
  - TDD
- XP programming
  - Pair programming
- Anti-patrons
- PRACTICE, PRACTICE, PRACTICE !!!

Au prochain épisode…

- Comment mesure-t-on la qualité d'un code ?
- Quels outils pour nous aider à surveiller le code?
- Peut-on corriger la structure d'un code ?
- …