La règle du scout

"Leave the campground cleaner that you found it."

— Règle scout

"If we all checked-in our code a little cleaner than we checked it out, the code simply could not rot."

— Robert C. Martin : Clean Code

Comment l’appliquer ?

A chaque intervention sur le code :

- Améliorer sa lisibilité
- Corriger les "code smells"

Quand s’arrêter ?

- Si vous y passez trop de temps
- Si vous commencez un gros refactoring

Améliorer la lisibilité

Le nommage

- Donner un nom explicite
- Éviter la lourdeur
- Utiliser une convention de nommage

Donner un nom explicite

- Plus de 4 lettres
  - Exception : i,j,k sont tolérés pour l'index d'une boucle
- Ne pas mentionner le type

Donner un nom explicite

```
// Avant
void statement(List<Video> list) {
    for(int i; i < list.size(); i++) {
        append(list.get(i).price); // we add each video price into the report
    }
}
```

```
// Après
void processVideosInReport(Video[] videos) {
    for (Video video: videos) {
        addToReport(video.price);
    }
}
```

Eviter la lourdeur

```
// Avant
List<Video> videoList = getVideos();
boolean checkedBool = true;
```

```
// Après
List<Video> videos = getVideos();
boolean isChecked = true;
```

Utiliser une convention de nommage

```
VideoRepository video_repository = new VideoRepository();
VideoRepository reportGenerator = new Report_Generator();

reportGenerator.generate_report(video_repository.fetch_videos());
```

Les "code smell"

- les mauvaises pratiques
- le code redondant

Exemples

- Avoir des valeurs en dur
- Avoir du code mort
- Avoir du code dupliqué
- Avoir du code trop long

Avoir des valeurs en dur

```
if(status == 400) {
    return "REQ_ERR"
}

return "REQ_OK"
```

```
if(status == STATUS_REQUEST_ERROR) {
    return CODE_ERROR
}

return CODE_SUCCESS
```

Avoir des valeurs en dur

Les déplacer dans :

- une constante
- un fichier de configuration
- un fichier de traduction

Avoir du code mort

- on supprime ! (au pire il y a Git)
- moins il y a de code
- plus c’est facile à maintenir 😈

Avoir du code dupliqué

- On applique la règle des 3 fois

$\Rightarrow$ et on factorise

Avoir du code trop long

- pas plus de deux indentations
- pas plus de 20 lignes

$\Rightarrow$ découper en sous-fonctions

Outillage

- votre IDE
- les linters
- les analyseurs de code

Refactoring automatisé avec l'IDE

- renommer un token
- extraire une méthode
- "inliner" un token

Les linters

- éliminer les mauvaises pratiques
- forcer les conventions

Exemple de linter

- eslint
- tslint

L’analyse de code statique

- analyse plus fine du code
- détection de duplication
- détection de problèmes de sécurité

Exemple d’analyseur de code

- SonarQube

Pour aller plus loin

Exemples de style guides

- Mozilla (js) [1]
- Airbnb (js) [2]
- Google (java) [3]

1. https://developer.mozilla.org/en-US/docs/MDN/Writing_guidelines/Writing_style_guide/Code_style_guide/JavaScript
2. https://github.com/airbnb/javascript
3. https://google.github.io/styleguide/javaguide.html

Pour aller plus loin

Ressources sur les code smells

- un extrait inspiré de Clean Code [1]
- un article inspirant du blog JBrains.ca [2]
- liste de code smells [3]

---

1. https://gpcoder.gitbook.io/clean-code/the-key-principles-of-clean-code/meaningful-names
2. https://blog.jbrains.ca/permalink/the-four-elements-of-simple-design
3. https://refactoring.guru/refactoring/smells