# Code et Design smells

# Introduction

- Affuter son nez et ressentir les "mauvaises odeurs" dans le code et la conception
- Identifier des signes de problèmes de conception
- Appliquer les heuristiques adaptées (SOLID, YAGNI, DRY, etc)

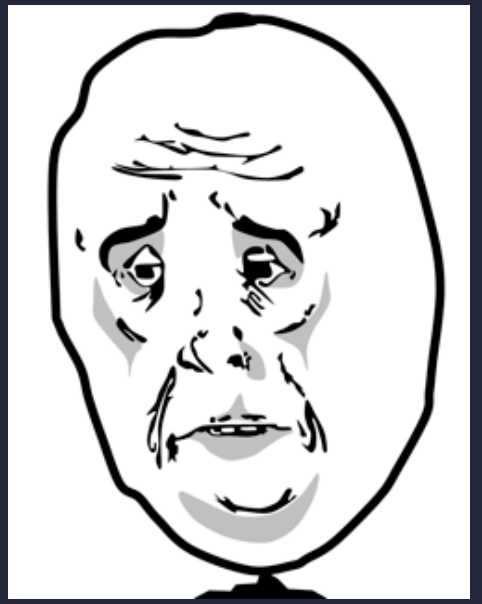
# Design smells

- Rigidité
- Fragilité
- Immobilité
- Viscosité
- Complexité inutile
- Répétition inutile
- Opacité

# Design smells

- Design smells mettent parfois du temps à se manifester et souvent trop tard

⇒ On s’arrête et on réfléchit sur les décisions prises dans le passé



# Code smells

- Plus granulaires que les Design smells
- Plus faciles et moins coûteuses à réparer

# 5 catégories

- **Bloaters** : entité devenue trop grosse et difficile à manipuler
- **Couplers** : cause un couplage excessif entre différents modules
- **Object-Orientation Abusers** : cas où la solution n’exploite pas le plein potentiel de la conception orientée objet
- **Change preventers** : freinent les changements du logiciel
- **Dispensables** : choses non-nécessaires qui devraient être retirées du code source

# Bloaters

- Long method
- Large class
- Primitive obsession
- Long Parameter List
- Data Clumps

# Couplers

- Feature Envy
- Inappropriate Intimacy
- Message Chains
- Middleman

# Object-Orientation Abusers

- Switch Statements
- Temporary Field
- Refused Bequest
- Alternative Classes with Different Interfaces

# Change preventers

- Divergent Change
- Shotgun Surgery
- Parallel Inheritance Hierarchies

# Dispensables

- Lazy Class
- Data Class
- Duplicated Code
- Dead Code
- Speculative Generality
- Comments

# Codes smells prioritaires

A addresser en priorité car elles ont tendance à entrainer les autres

- Duplication
- Primitive Obsession
- Feature Envy / Inappropriate Intimacy
- Message Chains

# Toujours éradiquer les code smells ?

- Les smells ne sont pas des **erreurs**
- **Symptôme** d’un problème **potentiel**
- Parfois il faut en accepter en compromis conscient

# Merci de votre attention