Atelier Refactoring (TP1)

1 Introduction

Dans ce travail pratique, vous allez exercer diverses techniques de refactoring sur des fragments de code Python. Les exercices se trouvent dans le répertoire `TP1`.

2 Exercices

2.1 Explicit over Implicit

Fichier : `explicit_over_implicit/1.py`

Objectif : Le code utilise des noms de variables cryptiques et des nombres magiques. Refactore-le pour rendre l'intention explicite. Techniques :

- Rename Method/Variable
- Replace Magic Number with Symbolic Constant

2.2 Generalisation

Fichier : `generalisation/template_method.py`

Objectif : Les classes `ResidentialSale` et `LifelineSite` partagent une logique similaire. Utilisez le modèle de conception « Template Method » pour éviter la duplication. Techniques :

- Form Template Method

2.3 Organizing Data

Fichier : `organizing_data/replace_data_value_with_object.py`

Objectif : La classe `Order` conserve les données client sous forme de primitives. Extrayez une classe `Customer`. Techniques :

- Replace Data Value with Object

2.4 Simplify Conditions

Répertoire : `simplify_conditions/`

Objectif : Simplifier la logique conditionnelle dans les fichiers fournis.

1. `consolidate_conditional_expression.py` : Combiner les vérifications en une seule expression ou fonction.
2. `decompose_conditional.py` : Extraire des parties de la condition complexe dans des fonctions.
3. `introduce_assertion.py` : Rendre les suppositions explicites.
4. `replace_nested_conditional_with_guard_clauses.py` : Réduire l'imbrication en utilisant des clauses de garde (guard clauses).

2.5 Simplify Methods

Répertoire : `simplify_methods/`

Objectif : Refactorings spécifiques pour les méthodes.

1. `parameterize_method.py` : Fusionner des méthodes similaires en ajoutant un paramètre.
2. `preserve_whole_object.py` : Passer l'objet entier au lieu de champs individuels si raisonnable.
3. `replace_parameter_with_method_call.py` : Si un paramètre peut être dérivé, dérivez-le à l'intérieur de la méthode.