TP Message chains et Middle man

Message Chains

Description

Cet anti-pattern se produit quand on effectue une chaine d'appel de méthode (ex :**a.getB().getC().calculate()**). Ceci est un symptôme qu'une classe pour exécuter des traitements doit connaitre la structure interne de l'application et traduit souvent un manque d'abstraction de celle-ci.

Penser à la Loi de Demeter : « Ne parler qu'à ces voisins connus »

Énoncé

Le projet du TP qui se trouve dans le package **message.chain.tp** représente une gestion de calcule de prix d'importation de produits en fonctions qu'ils proviennent d'Europe ou non.

Une fois que le projet Maven importer dans Eclipse :

- Identifiez cet anti-pattern dans la structure suivante.
- Proposez plusieurs choix pour corriger ce pattern
- Réaliser celui que vous pensez le plus pertinent
- Vérifiez le bon fonctionnement de la nouvelle application à l'aide des tests unitaires

Le Middle man

Description

Le middle man est un anti-pattern opposé où on a poussé la loi de Demeter à l'extrême en produisant des classes qui ne font aucun métier par eux-mêmes, mais se contente de déléguer. Une telle classe est un « middle man » et on peut se poser des questions sur la nécessité d'avoir une telle classe dans son application.

Énoncé

Le projet du TP qui se trouve dans le package **middle.man.tp** représente une gestion simple d'employés dans une entreprise.

- Identifiez cet anti-pattern dans la structure suivante.
- Proposez plusieurs choix pour corriger ce pattern
- Réaliser celui que vous pensez le plus pertinent
- Vérifiez le bon fonctionnement de la nouvelle application à l'aide des tests unitaires