**Exercice Le refactoring**

**Amélioration d’un code difficile**

Etape 1 : Supprimer le switch dans statement de Customer, remplacer par une méthode. On utilise la règle « Extract Method ».

Remarque sur cette étape : toujours refactoriser par petites étapes.

Etape 2 : Renommer des variables utilisées dans amountFor. On utlise la règle « Rename ».

Remarque sur cette étape : n’importe qui peut écrire du code compréhensible par la machine, seuls les bons programmeurs écrivent du code lisible par les humains.

Etape 3 : La méthode amountFor utilise les données de Rental, la déplacer vers Rental. On utilise la règle « Move »… et garder l’appel à la nouvelle méthode.

Etape 4 : Simplifier les appels à amountFor et les remplacer par getCharge(). On utilise la règle « Inline ».

Etape 5 : Remplacer les variables temporaires par des appels directs

Etape 6 : Pour éliminer le code manipulant frequentRenterPoints dans statement. Appliquer les règles : Extract Method, puis Move, puis Inline.

Etape 7 : Remplacer des variables locales. Dans statement, totalAmount sera supprimée. . La variable thisAmount va disparaître. Idem pour frequentRenterPoints.

Remarque : à cette étape, on augmente le code. De même, la performance peut être incriminée.

Etape 8 : Le problème du switch dans Rental : la logique conditionnelle sur les attributs d’un autre objet est une mauvaise idée. Déplacer getCharge dans Movie. Idem logique de getFrequentRenterPoints.

Etape 9 : Pour éviter le switch, il faudra introduire un héritage (Règle appliquer un Strategy ou State).Créer une famille de prix, et Movie pointe sur un prix (plutôt qu’un int). Puis getCharge sera déplacée sur Price. Et donc répartie sur les classes dérivées de Price. Même procédure avec getFrequentRenterPoints.