R6.A06.D05.alt : Maintenance applicative
Feuille TD-TP n° 1

Refactoring sur IntelliJ – Prise en main

Objectifs :

1.- Découverte des outils de l'IDE destinés au Refactoring

Ressources à votre disposition :

- L'archive **`tutoRefactoring.zip`** : projet pour l'exercice 1
- L'archive **`carApp.zip`** : projet pour exercice 2 (application simple pour s'exercer à refactoriser)

- L'archive **`junit5-jupiter-starter-gradle.zip`**
  C'est un projet 'Modèle' minimal pour le développement en Java avec IntelliJ et gradle. La fonction main() de son unique clase Main affiche « Hello world ».
  Il pourra vous servir pour créer de nouveaux projets

Préparation du travail

1.- Création du dossier consacré aux TDs et TPs de cette ressource

Dans le dossier de votre espace réseau destiné à vos travaux pratiques, créer un sous-dossier destiné à recevoir tous les TPs de la présente ressource. Nous l'appellerons, par exemple : **`r6_AD`**

Dans ce dossier, créer un dossier **`tdtp1`**.

2.- Installation des fichiers de 2 projets

a. Télécharger les archives **`tutoRefactoring.zip`** et **`carApp.zip`** et les décompresser dans **`r6_AD\tdtp1`**
b. Supprimer les archives **`.zip`** téléchargées

3.- Création d'un dépôt git pour tous les travaux de cette ressource

a. Créer un dépôt **`git`** pour les remises de travaux de cette ressource.
b. Vous déposerez les travaux commités de chaque exercice de la ressource dans ce dépôt.
c. En fin de séance, vous déposerez sur eLearn le lien vers votre dépôt GitHub (public).

Vous êtes prêt à travailler.

Vous avez 2 exercices à traiter.

Penser à tracer l'évolution de votre code avec des commits intermédiaires..

Exercice 1 – Refactoring sur IntelliJ – Prise en main

Objectif de l'exercice :
Découverte accompagnée des fonctionnalités de IntelliJ pour le Refactoring.

Exercice 2 – Analyse et Refactoring de l'application carApp

Objectif de l'exercice :
Analyser l'application fournie et appliquer les modifications mises en évidence par les outils d'inspection de code utilisés (ceux de IntelliJ + plugin Checkstyle)

Cela suppose :

- S'approprier le code de l'application
- Installer le plugin CheckList
- Paramétrer les règles que CheckStyle devra vérifier : Google ou Sun, via File/Setings/Tools/CheckStyle
- Paramétrer l'inspecteur de code File/Settings/Edit/Inspections pour que l'inspection se focalise sur le CheckStyle et Java
- Inspecter l'application et catégoriser les types d'erreurs identifiées (par les outils ou par vous-mêmes)
- Vérifier que les tests existants suffisent à valider le refactoring que vous allez réaliser. Sinon, les aménager.
- Mettre en œuvre le refactoring
- Passer - les tests associés à ce refactoring
- Sauvegarder (commit) chaque refactorisation avant de passer à la technique suivante. Le message du commit documentera la technique de refactoring mise en œuvre
- A la fin, déposer le lien vers votre dépôt GitHub (public) sur eLearn, sur l'espace prévu à cet effet