

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 classe `Main` affiche « Hello world ».
Il pourra vous servir pour créer de nouveaux projets

Préparation du travail

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

Dans le dossier de votre espace réseau destiné à vos travaux pratiques, créer un sous-dossier destiné à recevoir tous les TP 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. A la fin de l'exercice 2, vous déposerez sur eLearn le lien vers votre dépôt GitHub (public) de cet exercice.

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 qui mettent en œuvre les actions de Refactoring chirurgical vues en cours (chapitre 2).

Exercice 2 – Analyse et Refactoring de l'application carApp

Objectif de l'exercice :

Analyser l'application fournie et appliquer les modifications identifiées par vous-mêmes et/ou par les outils d'inspection de code utilisés (ceux de IntelliJ + plugin Checkstyle)

Rappel : Sauvegarder (commit) chaque refactorisation avant de passer à la technique suivante. Le message du commit documentera la technique de refactoring mise en œuvre

Travail à faire

- 1.- Tout en parcourant – analysant l'application, **lister rapidement** sur un bloc-notes les défauts de qualité du code que vous identifiez sur cette application
- 2.- Installer le plugin CheckStyle
- 3.- Paramétrer l'outil d'inspection de code
 - Les règles que CheckStyle devra vérifier : Google ou Sun, via File/Settings/Tools/CheckStyle
 - Paramétrer l'inspecteur de code File/Settings/Editor/Inspections pour que l'inspection se focalise sur le CheckStyle et Java
- 4.- Inspecter l'application avec CheckStyle et catégoriser les types d'erreurs identifiées, vérifier que cette analyse concorde avec votre analyse
- 5.- Vérifier que les tests existants suffisent à valider le refactoring que vous allez réaliser. Sinon, les aménager.
- 6.- Mettre en œuvre le refactoring.
 - Sauvegarder (commit) chaque famille de refactorisation avant de passer à la technique suivante. Le message du commit documentera la technique de refactoring mise en œuvre
 - Passer - les tests associés à ce refactoring
- 7.- A la fin, déposer le lien vers votre dépôt GitHub (public) sur eLearn, sur l'espace prévu à cet effet
- 8.- Discussion sur la structure de l'application
 - Avez-vous des remarques sur les classes Voiture ? Conducteur ?