Rapport Projet Java
Youssef Martin - de Castellane Gaetan

I-Structure du code

Ce code est structuré en deux parties, un package consocarbone qui contient les classes représentant les différents postes de consommation carbone ainsi que quelques classes de test et un package utilisateur qui permet de créer et initialiser des profils utilisateurs.

Le package consocarbone contient les classes Alimentation, Avion, BienConso, Logement, ServicesPublics et Transport qui représentent chacune un poste de consommation carbone. La classe Avion à été ajoutée en plus de celles demandées par le sujet car nous trouvions que ce poste de consommation était manquant et important.

Toutes ces classes héritent de la classe mère abstraite ConsoCarbone qui permet ensuite de faire des collections de poste de consommation n'ayant pas le même type. On y implémente l'interface Comparable pour pouvoir trier les différents postes de consommation en fonction de la valeur de leur impact carbone.

La classe ServicesPublics utilise le design pattern Singleton pour garder une seule instance immuable partagée par les utilisateurs, pour représenter le fait que tout le monde est concerné par les dépenses publiques, sans pouvoir changer leur impact.

Chacune des ces classes contient une fonction toString et une fonction carboneMoyen. La première permet d'afficher une instance de la classe sur la console, la seconde détaille la consommation moyenne des français vis-à-vis de ce poste de consommation.

Le package contient également deux classes de test pour Alimentation et Logement, utilisant JUnit, ainsi que les énumération CE, Longueur, Taille utilisée pour caractériser respectivement la classe énergétique d'un logement, la longueur d'un trajet en avion et la taille d'un véhicule.

On y retrouve finalement une classe Main qui teste quelques constructeurs, setters et getters, et illustre le polymorphisme sur toString et carboneMoyen.

Le package utilisateur contient la classe Utilisateur, où à chaque instance est attribuée une collection de logements et de véhicules, une instance de BienConso et ServicesPublics et le nombre de trajet long, moyen et petit courrier en avion. Cette classe permet donc de créer un profil utilisateur complet contenant les informations nécessaires à l'évaluation de son empreinte carbone.

Elle contient donc également une méthode calculerEmpreinte qui calcul et renvoi la valeur de l'empreinte carbone associée à ce profil, une méthode detaillerEmpreinte qui détaille les impact respectifs des postes de consommation carbone et enfin une méthode recommandations qui stocke puis trie les instances de ConsoCarbone du profil utilisateur et fait une recommandation basée sur l'instance avec le plus grand impact pour rendre le mode de vie de l'utilisateur plus durable.

Le package utilisateur contient également une classe IO qui interagit avec l'utilisateur du programme depuis la console afin de lui faire des recommandations.

La méthode run affiche un menu grâce à la méthode menu puis lit le choix de l'utilisateur et appelle la fonction adaptée. L'utilisateur peut choisir d'entrer les valeurs à la main depuis la console ou peut les avoir inscrites dans un fichier .txt respectant la mise en forme précisée dans le fichier README. On a donc deux méthodes initialisationManuelle et intialisationFichier qui lisent les valeurs nécessaire à l'initialisation d'une instance de Utilisateur, depuis la console ou depuis un fichier.

Nous avons également implémenté une classe ErrFormatFichier qui hérite de Exception, qui est une exception jetée lors de la détection d'une erreur de format dans le fichier .txt fournit par l'utilisateur.

II-Les difficultés et enseignement du projet

- Sur la création de projet dans Visual Studio Code et GitHub et l'organisation des classes :

  Nous avons codé le projet en utilisant chacun Visual Studio Code sur notre ordinateur et en partageant notre travail sur GitHub, il n'a pas été facile de comprendre et de s'approprier tous leurs outils.

  Notamment le concept de projet *main* et de branches (fork) avec lequel un contributeur peut soumettre une requête de changement(s) à apporter au projet (*main*) après avoir fait un *commit* dans sa propre branche.

  Ou encore la création d'un projet (avec un workspace) dans Visual Code Studio afin de pouvoir réaliser des tests Junits pour nos deux classes tests Junit.

- Sur le code en lui-même :

  Nous avons, à travers la mise en œuvre de ce projet, compris l'intérêt de l'hérédité et la généricité en Java et plus généralement de la programmation orientée Objet.

  Par exemple, comprendre l'intérêt de regrouper des fichiers (qui représente une classe ou encore une énumération utilisée dans les classes) dans un même package et de pouvoir ensuite importer ses classes et méthodes dans une classe d'un autre package.
  Dans le cadre de notre projet, il s'agissait d'importer le package consocarbone dans la classe utilisateur, celle-ci ayant pour attributs des instances de type ConsoCarbone, la classe mère du package consocarbone.

Il a également fallu s'approprier le concept des classes tests qui héritent de TestCase et le module Junit et ses différentes méthodes;

- Sur le travail de groupe :

  Le projet nous a permis de nous rendre compte qu'il était possible de travailler efficacement à plusieurs en apprenant à utiliser de manière optimale les outils à notre disposition et en contribuant de manière complémentaire tout en s'informant sur les contributions de chacun.

- Sur la robustesse du code :

  Nous avons été amenés à réfléchir à la robustesse de notre code après le rendu du premier jalon. notre programme gère les exceptions pour des valeurs entrées par l'utilisateur : par exemple s'il entre un taux négatif un message affichera que le taux doit être compris entre 0 et 1. Mais il ne gère pas les erreurs rentrées manuellement par le développeur : par exemple si le développeur crée une instance alimentation = new Alimentation(-1, -0.5), le programme ne verra pas d'erreur.