Projet Empreinte Carbone

Introduction :

Le but de ce projet est de réaliser un calculateur d'empreinte carbone. Ce type de programme quantifie les émissions de Gaz à Effet de Serre (GES) d'un.e individu.e en tonnes de CO2 équivalent (TCO2eq) en fonction de son mode de vie. Pour ce faire, votre programme devra obtenir des informations sur l'utilisateur.rice du programme :

- *Logement* : Vit-il/elle dans un logement bien isolé ? Avec quelle superficie ?
- *Alimentation* : Quel pourcentage des repas de l'utilisateur.rice sont principalement à base de boeuf ? À base de viande ?
- *Transport* : L'utilisateur.rice se déplace-t-il/elle avec une voiture ? Un SUV ? Combien de kilomètres sont parcourus à l'année ? Combien de temps la voiture est elle conservée ?
- *Consommations diverses* : L'utilisateur.rice est-il/elle plutôt dépensier.ère ou sobre dans ses dépenses ?

À partir de toutes ces informations, votre programme devra donner à l'utilisateur.rice le détail de son empreinte carbone ainsi que des recommendations pour qu'il/elle puisse adopter un mode de vie plus soutenable.

Afin de réaliser ce sujet, j'ai utilisé le guide de l'animateur du jeu *Inventons nos vies bas carbone* rédigé par Claire et Gildas Véret, Arnaud Brulaire, François-Joseph Grimault et Mathieu Hestin. Vous trouverez ce guide ici et la page web du jeu ici.

Ce travail est à réaliser **en binôme**. En plus de votre code, vous devrez rendre un petit *rapport* (5 pages max) contenant :

- Une présentation de la structure de votre code.
- Une présentation de vos extensions éventuelles.
- Une discussion sur ce qui a été le plus dur à implémenter pour vous et sur le niveau de difficulté du projet.

Votre projet devra également contenir un fichier *ReadMe* présentant succinctement votre projet et détaillant les instructions à réaliser pour le compiler, l'exécuter, puis l'utiliser.

→ Les détails de la soumission de votre projet seront donnés sur l'espace moodle du cours.

Ce projet compte pour 40% de la note de l'UE. Il est donc souhaitable que la note corresponde au travail de votre groupe, et non aux conseils d'autres groupes, d'autres étudiants ou d'internet. Si vous utilisez des sources (articles de recherche, posts sur internet, etc...), vous devez mentionner vos sources dans le rapport.

Planning de travail :

À la fin du cours 1 :

- Constituer des binômes.

À la fin du cours 2 :

- Créer un *package* `consoCarbone`.
- Créer une *énumération* CE avec 7 instances nommées $\{A, B, C, D, E, F, G\}$ représentant les différentes classes énergétiques possibles d'un logement.
- Créer une classe `Logement` avec les attributs suivants :
  - **`int`** `superficie`, la superficie du logement en m^2 ;
  - CE `ce`, la classe énergétique du logement ;
  - **`double`** `impact`, l'impact du logement en termes d'émissions de GES en TCO2eq.

  Cet impact se calcule selon la formule suivante $\alpha_{\texttt{CE}} \times \texttt{superficie}$ où $\alpha_{\texttt{CE}}$ est un coéfficient multiplicatif dépendant de la classe énergétique du logement selon la correspondance suivante :

CE	A	B	C	D	E	F	G
$\alpha_{\texttt{CE}}$	0.005	0.01	0.02	0.035	0.055	0.08	0.1

- Créer une classe `Alimentation` avec les attributs suivants :
  - **`double`** `txBoeuf`, le taux de repas (une valeur entre 0 et 1) à base de boeuf (le type de viande le plus émissif) ;
  - **`double`** `txVege`, le taux de repas végétariens ;
  - **`double`** `impact`, l'impact de l'alimentation de l'utilisateur.rice en termes d'émissions de GES en TCO2eq.

  Cet impact est calculé selon la formule suivante $8 \times \texttt{txBoeuf} + 1.6 \times (1 - \texttt{txVege} - \texttt{txBoeuf}) + 0.9 \times \texttt{txVege}$ (selon l'hypothèse qu'un repas ni végétarien, ni à base de boeuf est à base de volaille). Les valeurs 8, 1.6 et 0.9 pourront être déclarées comme des constantes de la classe `Alimentation`.
- Réaliser des *getters*, des *setters*, et des *constructeurs*.
- Réaliser dans chacune des deux classes une *méthode statique* détaillant sur la console l'empreinte carbone moyenne d'un.e français.e vis-à-vis de ces deux postes de consommation. Vous trouverez les informations nécessaires ici page 10.
- Écrire une méthode `main` pour tester votre programme.

À la fin du cours 3 :

- Créer une classe `ConsoCarbone` représentant un poste de consommation carbone générique et faites en sorte que vos classes précédentes *héritent* de cette classe. Quels attributs pouvaient vous placer dans `ConsoCarbone`. Chaque instance de la classe `ConsoCarbone` aura un attribut **`int`** `id`, correspondant à un identifiant unique attribué à l'instance.
- Travailler de nouveau sur les *constructeurs*, *getters*, et *setters*, et sur la *visibilité* de vos attributs et méthodes.
- Créer une classe `BienConso` avec l'attribut suivant : **`double`** `montant`, le montant des dépenses annuelles de l'utilisateur.rice. Pour calculer l'impact de ces dépenses, on fera l'hypothèse simplificatrice qu'une tonne de CO2eq est équivalente à 1750€ de dépenses.
- Créer une énumération `Taille`, avec deux instances P et G correspondant à "petite voiture" et "grosse voiture". La production d'une petite voiture émet 4.2tCO2eq et celle d'une grosse voiture 19tCO2eq.
- Créer une classe `Transport` avec les attributs suivants :
  - **`boolean`** `possede`, un boolean indiquant si l'utilisateur.rice possède une voiture.

  - `Taille taille`, la taille du véhicule.
  - `int kilomAnnee`, nombre de kilomètres parcourus par an.
  - `int amortissement`, durée de conservation du véhicule.

  Comme hypothèse simplificatrice, l'impact des déplacements de l'utilisateur.rice sera de 0 si `possede` est `false` et $\texttt{kilomAnnee} \times 1.93 \times 10^{-4} + \texttt{fabrication}/\texttt{amortissement}$ sinon où `fabrication` correspond aux émissions nécessaires à la fabrication de la voiture.
- Réaliser le même travail et les mêmes méthodes pour les nouvelles classes `Transport` et `BienConso` que pour les classes `Alimentation` et `Logement`.
- Redéfinir la méthode `toString` de la classe `Object` dans vos différentes classes. À partir de cette méthode, tester le *polymorphisme* dans votre méthode `main`.

À la fin du cours 4 :
- Déterminer si certaines de vos classes devraient être *abstraites*.
- Faites en sorte que la classe `ConsoCarbone` implémente l'*interface* `Comparable<ConsoCarbone>`. Deux instances seront alors comparées en fonction de leurs impacts en termes d'émissions de GES.
- Créer une classe `ServicesPublics` représentant le poste de consommation carbone induit par les services publics (justice, police, éducation, santé, ...). Ce poste de consommation carbone à la particularité que tou.te.s les Français.e.s ont la même empreinte de 1.5TCO2eq pour ce poste de consommation. Quel *design pattern* pouvez vous alors utiliser pour rendre compte de cette particularité ?
- Dans un second package que vous créerez, réaliser une classe `Utilisateur` ou `Utilisatrice` avec les attributs suivants :
  - `Alimentation alimentation`, le poste de consommation carbone de l'utilisateur.rice concernant son alimentation.
  - `BienConso bienConso`, le poste de consommation carbone de l'utilisateur.rice concernant ses dépenses en biens de consommation.
  - `Logement logement`, le poste de consommation carbone de l'utilisateur.rice concernant son logement.
  - `Transport transport`, le poste de consommation carbone de l'utilisateur.rice concernant ses déplacements.
  - `ServicesPublics services`, le poste de consommation carbone de l'utilisateur.rice concernant son utilisation des services publics.
- Écrire une méthode `double calculerEmpreinte()` qui calcule l'empreinte carbone de l'utilisateur.rice.
- Écrire une méthode `void detaillerEmpreinte()` qui afficher sur la console une description détaillée de l'empreinte carbone de l'utilisateur.rice.

À la fin du cours 5 :
- Placer votre projet sur *github* afin de pouvoir *versionner* votre code pour la suite du projet.
- Étendre votre hiérarchie de classes du package `consoCarbone` en vous documentant sur un poste de consommation carbone de votre choix.

À la fin du cours 6 :
- Faire la *documentation Javadoc* de votre projet.

— Choisir deux classes de votre choix et créer deux classes de *tests* JUnit pour ces classes afin de réaliser des *tests unitaires*.
— Rajouter les *annotations* nécessaires à votre projet telles que les `@Override`.

À la fin du cours 7 :
— Réaliser une méthode qui *ordonne* les consommations carbone de l'utilisateur.rice dans une liste puis présente l'information obtenue à ce.tte dernier.e, puis fait des *recommendations* pour obtenir un mode de vie plus durable.
— Modifier la classe `Utilisateur` ou `Utilisatrice` pour qu'il/elle puisse avoir une collection de logements et/ou de voitures.
— À ce stade du projet, vous pouvez commencer à réfléchir à une extension du projet, par exemple en réalisant une interface graphique, ou en créant une classe `Population` consistant en une collection d'utilisateurs.ices. À partir de cette classe `Population` vous pourrez réaliser des simulations informatiques afin de tester (de manière très simplifiée) des mesures de politique publique, e.g., une méthode réduisant de manière stochastique la consommation de viande de la population, ou une méthode incitant à la rénovation énergétique... Vous pouvez également avoir d'autres idées !

À la fin du cours 8 :
— Créer un menu intéractif afin d'intéragir dans la console avec l'utlisateur.rice de votre programme.
— Écrire un constructeur de la classe `Utilisateur` afin de pouvoir initialiser un utilisateur ou une utilisatrice à partir d'un *fichier* texte.
— Gérer les différentes *exceptions* possibles de votre programme comme les exceptions générées par les flux d'entrées-sorties ou l'utilisation d'arguments inapropriées (par exemple, pour s'assurer que la superficie du logement soit positive ou bien que le taux de repas végétariens soit bien une valeur entre 0 et 1, etc).