CARLA NACHMAN

SHIRELLE AZUELOS

L3 IM2D

PROJET EMPREINTE CARBONE

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 individu en tonnes de CO2 équivalent (TCO2eq) en fonction de son mode de vie.

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

Notre projet est constitué d'un dossier contenant le code en Java ainsi que d'un fichier texte README.TXT qui est « un mode d'emploi » du projet.

Ce projet comporte 2 packages *: consoCarbone* et *utilisateurOuUtilisatrice*.

- *consoCarbone* contient l'ensemble des fonctions qui permettent de faire les calculs des impacts.

La classe ConsoCarbone nous permet de donner à chaque sous-classe un id unique, qui s'incrémente automatiquement.

On a 6 fonctions qui héritent de la classe ConsoCarbone : Alimentation, BienConso, Logement, Transport, ServicesPublics et Numerique.

L'énumération CE nous permet d'obtenir la classe énergétique des logements. L'énumération a sept constantes : A, B, C, D, E, F et G. Chaque constante représente une classe énergétique différente pour un logement ou une résidence.

Avion, FretEtMessagerie, TrainsEtBus ainsi que Voiture héritent elles-mêmes de Transport.

Nous les avons créées lors de l'extension de la hiérarchisation des classes.

- **Avion:** Il s'agit d'une classe Java qui représente l'impact carbone du vol en avion.

La classe implémente aussi la méthode impact(), qui calcule l'impact carbone du vol en fonction du nombre de kilomètres parcourus par an. La méthode impact() utilise un ensemble d'instructions if imbriquées pour calculer l'impact carbone en fonction du nombre de kilomètres parcourus par an.

Si le nombre de kilomètres parcourus est inférieur ou égal à 1 000, l'impact est calculé comme suit : 2,3 x 0,0001 x nombre de kilomètres parcourus. Si le nombre de kilomètres parcourus est supérieur à 1 000 mais inférieur ou égal à 3 500, l'impact est calculé comme suit : 1,7 x 0,0001 x nombre de kilomètres parcourus. Si le nombre de kilomètres parcourus est supérieur à 3 500, l'impact est calculé comme suit : 1,5 x 0,0001 x nombre de kilomètres parcourus.

- **FretEtMessagerie:** Il s'agit d'une classe Java qui représente l'impact carbone de l'utilisation des services de fret et de messagerie pour le transport. La classe calcule l'impact carbone de

l'utilisation des services de fret et de messagerie en fonction du nombre de kilomètres parcourus et du type de marchandises transportées.

La classe a un marchand de champ privé qui représente le type de marchandises transportées et un impact de champ privé qui représente l'impact carbone de l'utilisation des services de fret et de messagerie. La méthode impact() calcule l'impact carbone comme le nombre de kilomètres parcourus multiplié par la quantité de CO2 émise par le type de marchandise transportée.

- **TrainEtBus:** Il s'agit d'une classe Java qui représente l'impact carbone de prendre le train ou le bus comme moyen de transport.

La classe implémente la méthode impact(), qui calcule l'impact carbone de prendre des trains ou des bus en fonction du nombre de kilomètres parcourus et du type de train ou de bus. La classe a un train sur le terrain privé qui représente le type de train ou de bus, et un impact sur le terrain privé qui représente l'impact carbone de prendre des trains ou des bus. L'impact carbone de chaque type de train ou de bus est calculé en fonction de la quantité de CO2 émise par le train ou le bus.

- **Voiture:** Il s'agit d'une classe Java qui représente l'impact carbone de l'utilisation d'une voiture. La classe implémente la méthode impact(), qui calcule l'impact carbone de la conduite en fonction du nombre de kilomètres parcourus par an, de la taille de la voiture, de la durée de possession de la voiture et du fait que la personne possède ou non une voiture ou non. La méthode impact() utilise une instruction if pour calculer l'impact carbone selon que la personne possède une voiture ou non. Si la personne ne possède pas de voiture, l'impact est fixé à 0. Si la personne possède une voiture, l'impact est calculé comme le nombre de kilomètres parcourus par an multiplié par 1,93 x 0,0001, plus la taille de la voiture divisée par la durée de possession de la voiture.

On a une **énumération Marchandises** (conteneur, camion, avioncargo) qui nous ait utile dans la classe FretEtMessagerie. Chaque type de marchandise est calculé en fonction de la quantité de CO2 émise lors du transport de la marchandise.

On a une **énumération TypesTrains** (TGV, Tram, Metro, RER, BusElect, RER, BusTherm) qui nous ait utile dans la classe TrainsEtBus.

L'énumération Taille nous permet d'obtenir la taille des voitures (P ou G), ainsi que sa valeur associée de type double, la quantité de carbone émise pour sa production. L'énumération Taille a également une méthode appelée getProduction qui renvoie la valeur du champ de production pour la constante particulière sur laquelle elle est appelée. Par exemple, si on appelle Taille.P.getProduction(), la méthode renverrait la valeur 4.2.

Nous avons créé la **classe Numerique** lors de l'extension de la hiérarchie des classes. En effet, selon nous, elle fait aujourd'hui partie intégrante de nos vies, et utilise une grande partie de notre empreinte carbone.

Pour chacune de ces classes qui héritent de ConsoCarbone,

Alimentation a 2 attributs : txBoeuf et TxVege, les taux consommés de Bœuf et de nourriture Végétarienne. La classe a également trois variables finales CSTE1, CSTE2 et CSTE3 qui sont utilisées dans le calcul de l'empreinte carbone.

BienConso possède l'attribut montant qui représente le montant total des biens consommés.

La classe logement a 2 attributs : la superficie du logement (superficie) ainsi que sa classe énergétique (ce).

Transport a pour toutes ses classes filles, un attribut kilomAnnee (le nombre de Kilomètres par ans effectués selon les moyens de transport utilisés); FretEtMessagerie a en plus le type de Marchandises, TrainsEtBus a en plus TypesTrains. Et voiture prend de surcroit, possede (pour savoir si l'utilisateur/trice possède ou non une voiture), la taille de la voiture (grâce à l'enumeration Taille), ainsi que son amortissement.

Pour **Servicespublics**, étant donné que tous les français ont le même impact (= 1.5), il n'y a aucun attribut que l'utilisateur doit rentrer.

La classe Numerique représente l'impact carbone de l'utilisation d'appareils numériques. La classe étend la classe ConsoCarbone et implémente la méthode impact() également, qui calcule l'impact carbone de l'utilisation d'appareils numériques en fonction de la présence ou de l'absence de différents types d'appareils numériques. La classe a un impact de champ privé qui représente l'impact carbone de l'utilisation d'appareils numériques, et un ensemble de 6 champs booléens privés qui indiquent la présence ou l'absence de différents types d'appareils numériques : possmontreco, posssmartphone, possconsole, possordiportable, posstv et possordifixe.

La méthode utilise un ensemble d'instructions if pour vérifier la présence ou l'absence de chaque type d'appareil numérique, et ajoute une certaine quantité à l'impact carbone en fonction du type d'appareil que l'utilisateur/trice possède . L'impact carbone de chaque type d'appareil est calculé en fonction de la quantité de CO2 émise lors de la fabrication de l'appareil

Et enfin, il y a la méthode **Main** où nous avons créer des instances de chacune des classes. (Nous utiliserons cette fonction Main pour tout le projet et non pas seulement pour le package consoCarbone). C'est la principale méthode d'un programme qui calcule l'empreinte carbone d'un individu en fonction de ses habitudes de consommation. La méthode principale crée des instances de différentes classes qui représentent différents aspects de l'empreinte carbone d'un individu. C'est le point d'entrée du programme.

Dans la méthode principale, plusieurs objets sont créés et leurs valeurs sont imprimées sur la console à l'aide de la méthode System.out. println .

Dans toutes ces classes, nous avons une methode impact( ) qui nous permet de calculer l'impact pour chaque attribut d'un utilisateur/trice, et que nous avons décidé de mettre en absract.

De plus, nous avons créé 2 tests JUnit : un pour Avion et l'autre pour BienConso, afin de tester si quand on met certaines valeurs, et quand on les calcule à la main, cela renvoie le même résultat ou non.

- *utilisateurOuUtilisatrice* possède 3 classes : UtilisateurOuUtilisatrice, Population, Menu.

La classe UtilisateurOuUtilisatrice créée des Utilisateurs ou des Utilisatrices avec les attributs alimentation, bienconso, logement, transport, servicesPublics, et numerique.

Mais au final, nous avons décidé de ne pas avoir un logement/ un transport pour un utilisateur/trice, mais une collection. En effet, il est plus judicieux d'avoir plusieurs logements (ou du moins plusieurs appartements), et un utilisateur/trice peut utiliser plusieurs transports. C'est pourquoi nous avons créés des listes pour ces 2 classes.

Dans cette classe UtilisateurOuUtilisatrice, nous calculons l'empreinte totale, c'est-à-dire la somme des impacts de tous les attributs. On compare également pour chaque attribut, l'impact calculé avec l'impact moyen des français. Nous détaillons également l'empreinte carbone de chaque utilisateur/trice. Puis, on met le tout dans une liste pour pouvoir l'ordonner, et puis la renvoyer. On finit par faire des recommandations aux utilisateurs pour limiter leur empreinte carbone.

Pour la **classe Population**, on crée une liste d'utilisateurs/trices. Elle a un champ LinkedList appelé population qui est utilisé pour stocker les objets UtilisateurOuUtilisatrice.

Et, on fait des simulations pour le gouvernement, pour montrer qu'après certaines décisions , les impacts étaient largement modifiés.

La classe Menu a un certain nombre de méthodes qui permettent à l'utilisateur/trice d'entrer ses données sur ses habitudes de consommation, y compris sa consommation alimentaire, son logement, ses biens de consommation, ses transports, son utilisation des services publics et du numérique.

Il y a 3 méthodes, une qui affiche le menu, une qui permet d'écrire un fichier selon les règles établies dans notre projet, et une qui nous permet de lire un fichier.

Nous avons ajoutées dans les docs un fichier : « Fichier UtilisateurOuUtilisatrice exemple » pour pouvoir tester cette methode.

Dans les classes, on met pour quasiment chacun des attributs des getters(pour pouvoir accéder aux valeurs des attributs), et des setters (pour modifier les valeurs). Quand nous n'avons pas besoin de modifier les valeurs, nous ne faisons pas de setters.

Nous avons également réaliser la JavaDoc pour l'entièreté du projet.

Difficultés rencontrées et ressenti du sujet :

- Nous avons choisi de créer une classe Population, trouvant cela plus pertinent que l'interface graphique. Lorsqu'il a été question de créer les simulations informatiques de cette classe, pour des décisions gouvernementales, nous avons dû passer beaucoup de temps pour comprendre exactement comment il fallait s'y prendre.
  En retravaillant les tds vus en classe, cela nous a permis de coder cette classe en s'inspirant des ArrayList du td7. Suite à cela nous avons pu adapter notre code et ainsi crée notre classe Population.

- Pour ce qui est d'étendre la hiérarchie nous avons eu du mal à comprendre exactement ce qu'il fallait faire. On a donc rajouter certaines classes qui dérivaient des principales. Par exemple, nous avons créé une classe avion qui représente l'impact carbone du vol en avion. La classe étend la classe Transport. Cela nous permet ainsi d'être plus précis sur l'empreinte carbone de l'utilisateur.

- De plus, ce qui a été pour nous le plus difficile à implémenter est le menu du package utilisateurOuUtilisatrice. Nous ne savions pas comment écrire nos fichiers exemples, ni même le coder dans la classe Menu.

- Nous n'avons pas trouvé dans l'ensemble, le sujet difficile en lui-même, il était bien guidé, et nous savions vers quoi il fallait aboutir pour chaque partie. Mais comme nous n'avons pas eu de cours de Travaux Pratiques dès le début du semestre, nous avions du apprendre à ouvrir et à utiliser Eclipse, avant de comprendre comment était utilisé Java. Ce qui nous a pris le plus de temps sont l'invention et l'implémentation de certaines méthodes.

Sites référés :

Pour obtenir les données en fonction du nombre de kilomètres par ans, nous avons utilisées le site internet avenirclimatique.org.

Pour écrire les recommandations, nous nous sommes basées sur le guide donné en énoncé : *Guide de l'animateur INVBC "Inventons Nos Vies Bas Carbone" .*