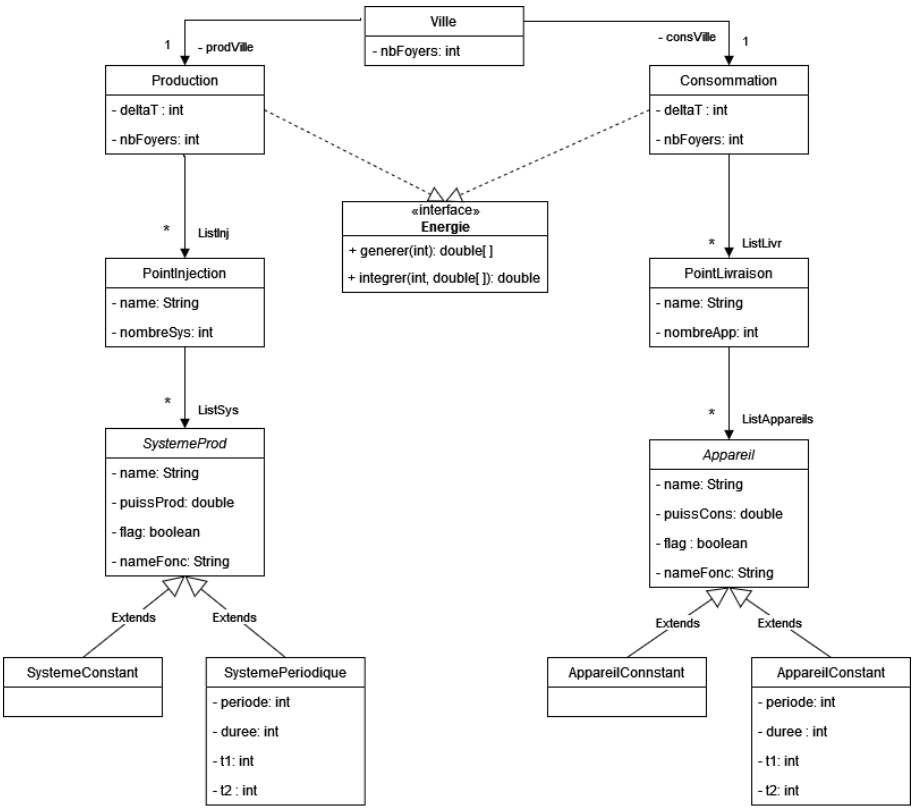
Voici le diagramme UML du projet :



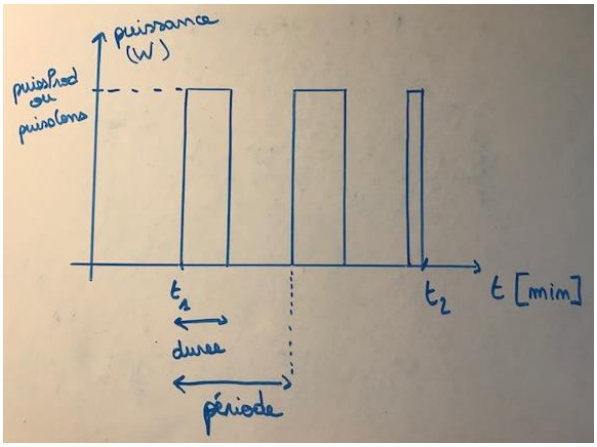
En particulier les classes SystemeProd et Appareil sont des classes abstraites comportant deux méthodes abstraites qui sont ensuite implémentée dans leurs sous-classes respectives. Toutes les listes sont des ArrayList des types considérés.

Vous trouverez dans le projet trois classes exécutables, une dans chaque paquetage. Ces classes permettent de réaliser des tests sur le fonctionnement des différentes méthodes des paquetages respectifs, hormis VilleMain qui elle teste la globalité du projet.

Les tests globaux menés dans VilleMain sont assez lents dû à de multiples parcours de listes de listes… Je n'ai pas réussi à concevoir de façon moins complexe.

1) Modèles de production et de consommation

Les deux modèles sont codés de la même façon qu'ils soient producteurs ou consommateurs d'énergie. Le modèle constant est basique mais ci-contre voici un graph explicitant les attributs du modèle périodique. t1 et t2 correspondent aux instants de début et fin de production des modèles périodiques et ce indépendamment du fait qu'ils soient en train de produire ou non. Pour en outre modéliser les fluctuations de production/conso sur l'année, je multiplie par une fonction de forme la puissance en fonction du nom de cette fonction de forme (nameFonc).



2) Paquetage Production

Dans le paquetage Production, vous trouverez toute la partie relative à la production d'énergie. Elle dépend entièrement du nombre de foyers dans la ville. Les méthodes d'affichage sont des displays dans la console des différents Points d'Injection et de leurs attributs, des tableaux de puissance produite, etc... De plus, j'affiche les graphs de la puissance produite à chaque minute pour chacun des deux modèles de production, ainsi que la puissance moyenne générée chaque jour de l'année. Voici les méthodes de construction :

- generer(int j) : double[ ] : cette méthode permet de générer le tableau rassemblant la totalité des productions de tous les systèmes de production. Elle fait appel à addProd
- addProd(double[ ] prod, int j) : double[ ] : cette méthode prend en argument un tableau de production et le modifie en ajoutant la production d'un système de Production et en tenant compte du jour de l'année

- integrer(int duree, double[ ] prod) : double : cette méthode fait une intégrale discrète (somme) des puissances produites sur la durée duree et renvoie l'énergie ainsi produite
- creerSysProd(int nbFoyers) : listeInjection : cette méthode est utilisée pour créer de façon automatique et pseudo-aléatoire le système de production d'une ville en fonction du nombre de foyers dans cette ville. Elle retourne une ArrayList de Points d'Injection

3) Paquetage Consommation

Ce paquetage est symétrique au paquetage Production. Vous y trouverez toute la partie relative à la consommation d'énergie. Elle dépend entièrement du nombre de foyers dans la ville. Les méthodes d'affichage sont des displays dans la console des différents Points de Livraison et de leurs attributs, des tableaux de puissance consommée etc.... De plus, j'affiche les graphs de la puissance consommée à chaque minute pour chacun des deux modèles de consommation, ainsi que la puissance moyenne consommée chaque jour de l'année. Voici les méthodes de construction :

- generer(int j) : double[ ] : cette méthode permet de générer le tableau rassemblant la totalité des consommations de tous les appareils. Elle fait appel à addCons
- addCons(double[ ] cons, int j) : double[ ] : cette méthode prend en argument un tableau de consommation et le modifie en ajoutant la consommation d'un appareil et en tenant compte du jour de l'année
- integrer(int duree, double[ ] cons) : double : cette méthode fait une intégrale discrète (somme) des puissances consommées sur la durée duree et renvoie l'énergie ainsi consommée sur cette durée
- creerSysConso(int nbFoyers) : listeLivraison : cette méthode est utilisée pour créer de façon automatique et pseudo-aléatoire le système de consommation d'une ville en fonction du nombre de foyers dans cette ville. Elle retourne une ArrayList de Points de Livraison

4) Paquetage Ville

Ce paquetage est au-dessus des autres d'un point de vue hiérarchique : il rassemble les deux pour effectuer les simulations à l'échelle de la ville. Vous y trouverez l'interface Energie qui lie les classes Production et Consommation ainsi que la classe Ville et sa classe exécutable associée. De même que précédemment, il y a des méthodes d'affichage dans le terminal mais en plus on y trouve les méthodes d'affichage au format CSV des puissances et énergies :

- displayCSVJour(int j) affiche la minute m, la puissance instantanée consommée, produite et l'énergie consommée ce jour, l'énergie produite ce jour pour chaque minute du jour J.
- displayCSVAn() affiche le jour j, la puissance moyenne consommée et la puissance moyenne produite sur ce jour ainsi que l'énergie consommée et produite depuis le début de l'année.

Enfin, il y a la méthode compare(int j) qui vérifie que la consommation n'excède pas la production d'énergie à chaque minute du jour J. Sinon un affichage console s'effectue.

Dans la classe exécutable s'effectue tous les tests globaux pour une ville de 6000 foyers et se tracent les graphs de production et consommation du jour 1 ainsi que les consommation et production moyenne sur l'année. Pour effectuer d'autres tests globaux, le constructeur Ville(int nbFoyers) permet de générer rapidement une nouvelle ville. Je ne vous conseille pas de dépasser 15000 foyers car après c'est vraiment long (~3min de calcul surtout à cause des graphs finaux). Néanmoins, la production codée est suffisante pour des villes allant jusqu'à 100000 foyers au moins.