**Projet Programmation Orienté Objet**

Projet1 : Développer un Système de Gestion de Citoyens

Membre du binôme : Sekularac Elvir et Ngo David (Cao tien)

Professeur du TP : Mr. Melliti

**Diagramme UML**

**Une image contenant texte, carte

Description générée automatiquement**

Dans ce diagramme, réalisé avec StarUML, nous avons 6 classes chacune ayant ses propres attributs ainsi que les liens qu’elles ont entre elles.

Et on précise que ce diagramme est progressiste, ce qui veut dire que des citoyens de même sexe peuvent se marier ensemble, l’avantage est d’alléger le diagramme ainsi que le code.

**Programmation**

Une fois le code Java généré par StarUML obtenu, nous avons commencé la programmation à l’aide du logiciel Eclipse IDE et à faire un menu dans le Main.

Le menu est constitué de l’affichage dans la console des 7 choix : Mariage, Divorce, Naissance, Etat d’une personne, Affichage de la liste des personnes, Saisie des personnes, Quitter le programme. Chaque condition « if » correspond à chacun des choix. À l’intérieur de ces conditions, on fait ensuite des appels de méthodes. On a commencé par le menu car plus pratique pour tester les méthodes en les appelant directement et voir s’il n’y a pas de problème dès le commencement de la partie programmation. Ensuite on a créé les associations des 6 classes générées par StarUML grâce à ce lien : [**https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes#L3-6-2-g**](https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes#L3-6-2-g)**.**

### On crée aussi un constructeur avec en paramètre les attributs pour chaque classe, et chaque constructeur sert à créer une instance d’un objet. Ainsi que les méthodes pour recueillir les informations, les modifier ou les stocker et qui seront ensuite appelées dans le Main. Chaque objet créé est stocké dans des ArrayList car ce sont des tableaux dynamiques, donc pratique car on n’a pas besoin de définir une taille pour chaque tableau. Ces tableaux sont notre base de données et sont enfin sauvegardé au choix 7 du menu, lorsque l’on souhaite quitter le programme, une sauvegarde des tableaux est donc générée dans un fichier « sauvegarde » et le programme s’arrête. On a donc réalisé une sérialisationd'objets.

**Fonctionnalités**

Le programme est capable :

* D'enregistrer des citoyens avec le choix 6 du menu, on demande à l’utilisateur d’entrer son numéro d’identifiant (si le numéro est déjà utilisé, il y a un message d’erreur), son sexe (homme ou femme sinon message d'erreur), son Nom de naissance, son Prénom, sa date de naissance, le nom de la mairie où la personne est née, le code postal de la mairie et on répond à la question qui demande si la personne est encore en vie par oui ou par non (sinon message d’erreur). À la fin, 2 objets sont créés, l’un concernant le citoyen est enregistré dans un tableau ArrayList « listecitoyen » et l'autre concernant la mairie (nom et code postal) est enregistré dans « listemairie ». Un troisième objet est créé si le citoyen est décédé (l’objet est constitué des données telles que le numéro ID et la date de décès de la personne) et est enregistré dans « listedeces ».
* De marier 2 citoyens avec choix 1 du menu, pour cela on doit entrer leur numéro d’identifiant (message d’erreur si la personne est déjà mariée, pas dans la base de données ou si le numéro du second citoyen est le même que celui du premier) et la date du mariage. À la fin, l’objet créé est enregistré dans un tableau ArrayList « listemariage ».
* De divorcer 2 citoyens avec le choix 2 du menu, pour cela on doit entrer leur numéro d’identifiant (message d’erreur si la personne n’est pas mariée, si elle est déjà divorcée, pas dans la base de données ou si le numéro du second citoyen est le même que celui du premier) et la date du divorce. À la fin, l’objet créé est enregistré dans un tableau ArrayList « listedivorce » et on enlève aussi de « listemariage » les 2 époux qui viennent de divorcer.
* De mettre par défaut l’état civil « célibataire » et peut changer en « marié(e) » après un mariage, en « divorcé(e) » après un divorce, en « veuf(ve) » si le conjoint est décédé ou ne change pas, « décédé(e) », lorsque la personne est décédée.
* D’afficher la liste de tous les citoyens présents dans la base de données avec le choix 5 du menu, les informations affichées sont : Numéro d’identifiant, sexe, Nom, Prénom, Date de naissance, Etat civil. Le nom du conjoint est affiché sous les personnes qui en ont un.
* Le choix 4 du menu est semblable au choix 5 mais la différence est qu’on doit entrer le numéro d’identifiant d’une personne (message d’erreur si le citoyen n’est pas dans la liste de la mairie) et ainsi on a les informations qui s’affichent pour le citoyen qu’on recherche.
* D’enregistrer un nouveau-né avec choix 3 du menu. On entre le numéro d’identifiant du père biologique et de la mère biologique (message d’erreur si ce n’est pas un homme/une femme ou si pas dans la base de données) et la date de naissance. L’objet créé est enregistré dans un tableau ArrayList « listenaissance ». Le nouveau-né devient un nouveau citoyen, on lui demandera les mêmes informations qu’en choix 6 sauf pour la date de naissance qu’on récupérera, évitant ainsi de redemander la date de naissance. À la fin, l’objet créé est enregistré dans un tableau ArrayList « listecitoyen ».
* Le choix 7 permet de quitter le programme et grâce à la sérialisation d’objets, on peut sauvegarder dans un fichier « sauvegarde » la base de données, c’est-à-dire les 6 tableaux dynamiques (ArrayList).
* Dans le menu, on doit entrer un numéro et si on entre un numéro autre que de 1 à 7 alors un message d’erreur apparaît et on retourne au menu.
* À chaque message d'erreur, on indique pourquoi et on retourne au menu pour pouvoir recommencer.

**Bilan du Projet**

Nous avons réussi à faire marcher le programme mais nous avons eu des problèmes pour optimiser parfaitement celui-ci (méthodes répétitifs). Ce dernier marche sur du court terme, mais n’est pas proposable pour un programme pouvant fonctionner tout le temps dans une vraie mairie sans une dernière fonctionnalité : nous n’avons pas réussi à intégrer le fait de pouvoir réutiliser les informations sur les citoyens. Nous avons réussi à programmer le fait de sauvegarder les données mais nous ne savons pas comment les réutiliser en désérialisant.