Paulina Czyż, Julian Nowak
groupe 385 (Erasmus)

Projet noté

Programmation Orientée Objets

UNIVERSITÉ DE NANTES

Introduction

Le but était de créer un programme permettant de gérer un réseau d'échanges des services. Tous les membres d'un réseau peuvent effectuer des tâches aussi bien que recevoir des tâches réalisées par d'autres membres. La monnaie d'échange est le jeton et chaque membre a une somme fixe des jetons au départ.

Le code de notre projet est disponible sur le site *github.com* qu'on a utilisé pour partager le code entre nous: https://github.com/julnow/ProjetPCJN

Modélisation

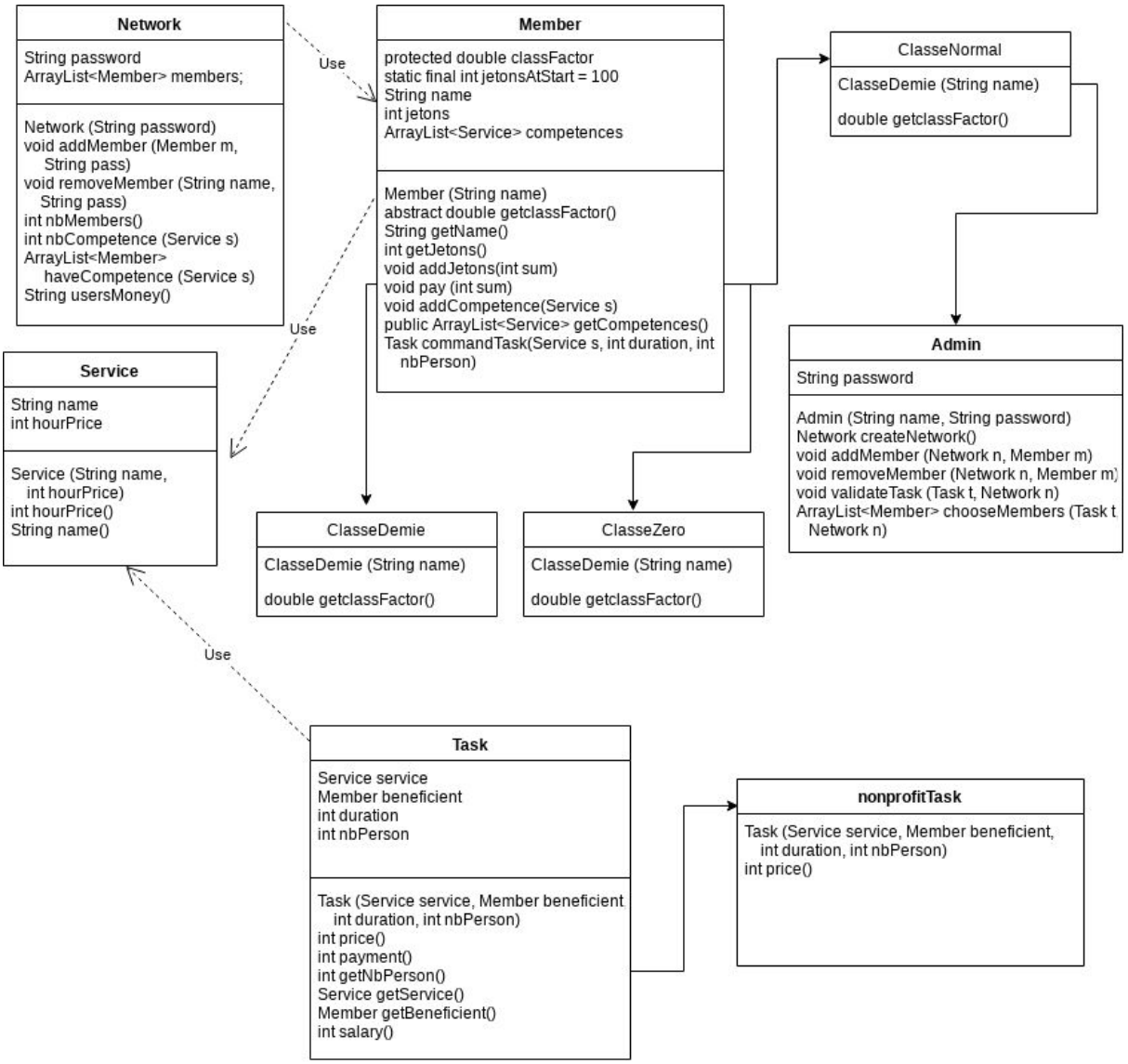

Fig. 1 - Diagramme de classes

Le réseau est réalisé par la classe *Network*, qui est composée des membres. Les membres sont réalisés par des sous-classes de classe *abstract Member*. La seule différence entre les *ClasseDemie, ClasseZero et ClasseNormal* est la valeur de *classFactor*: 0.5 pour *ClasseDemie,* 0.0 pour *ClasseZero* et 1.0 pour *ClasseNormal*. Classe *Admin* hérite de *ClasseNormal* et introduit de nouveaux fonctions pour gérer *Network*. Les tâches sont réalisés par la classe *Task*, qui utilise la classe décrivant des services (*Service*). Les compétences de membres sont aussi décrits par *Service*. Il existe aussi des tâches bénévoles - *nonprofitTask*, sous-classe de *Task*, pour lesquels la prix *price* retourne toujours 0. La classe Main est dans la classe *Program* (pas sur le diagramme).

Spécification des classes

Classe: ***Member***

Rôle: Permet de créer un membre du réseau donné. C'est une classe abstraite qui a des sous-classes *ClasseDemie, ClasseZero et ClasseNormal* qui diffèrent par son classFactor. Il existe une méthode abstraite *getClassFactor()* qui retourne sa valeur.

Méthodes:

Member (String name)	construit une instance de Member avec son nom, nombre de jetons défini au départ et un ArrayList contenant ses compétences
String getName()	retourne le nom du membre
int getJetons()	retourne le nombre de jetons du membre
void addJetons (int sum)	ajoute une somme aux jetons du membre
void pay (int sum)	retire une somme des jetons du membre afin de payer pour un tâche qu'il a commandé
void addCompetence (Service s)	ajoute un service à l'ArrayList de ses compétences
ArrayList<Service> getCompetences()	retourne un ArrayList avec toutes les compétences du membre
Task commandTask (Service s, int duration, int nbPerson)	permet de commander une tâche

Classe: ***Admin***

Rôle: Permet de créer un administrateur qui peut créer un réseau et ajouter/retirer ses membres. Il est aussi responsable de la validation des tâches.

Méthodes:

Admin (String name, String password)	construit une instance avec son nom et mot de passe
Network createNetwork()	construit un nouveau réseau
void addMember (Network n, Member m)	ajoute un membre au réseau donné
void removeMember (Network n, Member m)	retire le membre indiqué du réseau donné
void validateTask (Task t, Network n)	permet de valider une tâche qui doit être réalisé par les membres du réseau

Classe: ***Network***

Rôle: Permet de gérer un réseau. Contient un ArrayList avec tous ses membres. Elle est protégée avec un mot de passe qui est nécessaire pour effectuer des changements tels que addition/soustraction des membres et sa création.

Méthodes:

Network (String password)	construit une instance de Network avec un mot de passe et un ArrayList contenant ses membres
void addMember (Member m, String pass)	ajoute un membre au réseau si le mot de passe est correct
void removeMember (String name, String pass)	retire un membre du réseau si le mot de passe est correct
int nbMembers()	retourne le nombre de membres du réseau
int nbCompetence (Service s)	retourne le nombre de membres qui ont la compétence recherchée
ArrayList<Members> haveCompetence (Service s)	retourne un ArrayList avec tous les membres qui ont une compétence recherchée

String usersMoney()	retourne un String avec les noms des membres et le nombre de leurs jetons

Classe: ***Service***

Rôle: Permet de créer des services et décrire des compétences de membres

Méthodes:

Service (String name, int hourPrice)	construit une instance de Service avec son nom et coût horaire
int hourPrice()	retourne le coût horaire du service
String name()	retourne le nom du service

Classe: ***Task***

Rôle: Permet de créer des tâches effectuées au sein d'un réseau et calculer la salaire pour l'exécution de ces tâches.
Il existe aussi une sous-classe *nonprofitTask* qui décrit une tâche bénévole et pour laquelle la méthode *price()* retourne toujours 0.

Méthodes:

Task(Service service, Member beneficient, int duration, int nbPerson)	construit une instance de Task contenant sa durée, bénéficiaire, service et le nombre de personnes nécessaires pour le faire
int price()	calcule le prix du tâche et retourne son valeur
int payment()	calcule le paiement en tenant compte de la classe sociale du bénéficiaire
int getNbPerson()	retourne le nombre de personnes nécessaires pour effectuer la tâche
Service getService()	retourne le service du tâche
Member getBeneficient()	retourne le beneficient du tâche
int salary()	calcule le salaire des membres qui effectuent la tâche et retourne son valeur

Exemple d'exécution

L'exemple d'exécution est présenté dans la class *Program.java*. Comme la majorité de méthode *jette* d'exceptions, tous les objets sont déclarés avant la boucle *try* est définis dedans. Les autres commentaires sont introduits dans le code.

Résumé

Notre solution est extensible aux nouveaux types de tâches et classes sociales, comme tous les classes qui existent maintenant sont sous-classes d'une classe abstraite *Member*, et comme on a montré avec l'exemple de tâche bénévole *nonprofitTask*.

Il faut se demander si l'algorithme de choix aléatoire de travailleurs est une bonne méthode, peut-être il pourrait être remplacé par le choix volontaire dans le futur. Aussi, la personne qui commande une tâche peut être choisie pour sa réalisation si elle possède une compétence nécessaire, ce qui peut être pas désirable.

Aussi, la protection du réseau par un mot de passe qui est un *String* n'est pas trop secure. Il faudrait le remplacer par un mot de passe crypté.

Cependant, nous avons créé un prototype qui pourrait devenir, un jour, un vrai réseau d'échanges des services.