UE Conception Orientée Objet

Le facteur sonne toujours deux fois

La réalisation de ce sujet va se faire par la réalisation de « versions » successives. Chaque version doit être réalisée complètement avant de passer à ma suivante. Une bonne conception doit faciliter l'intégration des extensions apportées par les versions successives.

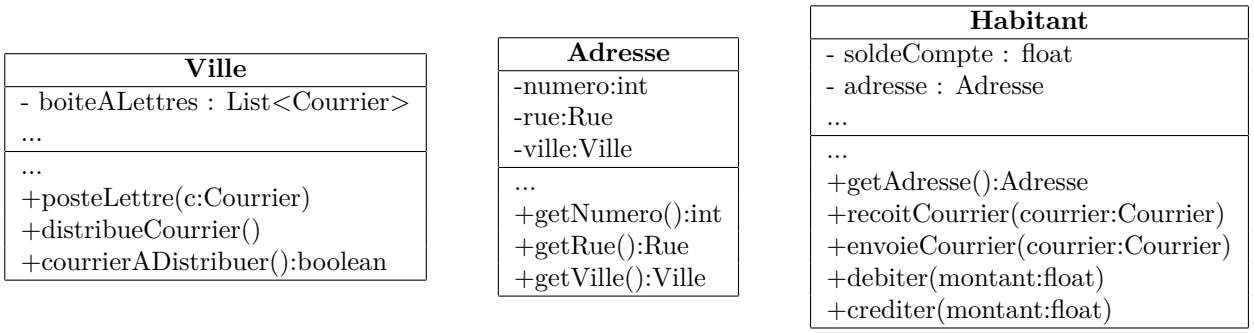
On se propose de modéliser la distribution du courrier : chaque jour, des personnes envoient des courriers à d'autres personnes, habitant la même ville ou une autre ville.
Elles mettent ces courriers dans des boîtes aux lettres qui sont relevées, et le courrier est distribué le lendemain.

Villes et habitants

Dans cet exercice, les **villes** sont considérées comme des bureaux distributeurs : elles réunissent toutes les lettres postées le jour même dans la ville, et les distribuent le lendemain. Il convient donc, comme dans la réalité, de distinguer la boîte aux lettres, où sont postées les courriers, de la sacoche du facteur, qui contient les courriers à distribuer.
Une ville possède un nom, des rues et des **habitants**.

- On peut ajouter un habitant. On précise alors son adresse (on ne fera pas de vérification sur la validité de l'adresse ou les éventuels conflits d'adresse).
- La méthode `posteLettre(Courrier c)` ajoute un courrier à l'ensemble des courriers à distribuer (dans la boîte aux lettres).
- La méthode `distribueCourrier()` s'occupe de la distribution du courrier (posté la veille et qui est, au moment de la distribution, dans la sacoche du facteur).

Les habitants sont caractérisés par un nom, une adresse (une rue, un numéro, une ville) et un compte en banque. On doit pouvoir créditer ou débiter ce compte d'un montant donné. Un habitant peut recevoir et envoyer des courriers à un autre habitant.
Voici des diagrammes UML indicatifs et partiels pour ces classes :



Les courriers

Un **courrier** comporte un expéditeur et un destinataire (deux personnes) et un contenu (a priori, n'importe quoi).
A tout courrier est associée une action, qui sera accomplie lorsque ce courrier est reçu par son destinataire.
De plus tout courrier a un coût.

Version 1.0

Dans cette première version, un courrier peut être :

- Une **lettre simple** : son contenu est un texte. Le coût d'une lettre simple est fixe (0,5 par exemple).
- Une **lettre de change** : elle contient une certaine somme d'argent, qui provient du compte de l'expéditeur, et sera versée au compte du destinataire. Le destinataire, reconnaissant, envoie alors une lettre de remerciement à son bienfaiteur. Le coût d'une lettre de change est de 1 + 1% de la somme transférée.

Q 1 . Modélisez les différents éléments décrits ci-dessus sous la forme de diagramme de classes UML.

Donnez en parallèle le code des méthodes `distribueCourrier` de `Ville` et `recoitCourrier` et `envoieCourrier` de `Habitant`

Version 2.0
Il doit maintenant être possible d'envoyer tout courrier **en recommandé**. Dans ce cas, **en plus** de l'effet lié au courrier, un accusé de réception est alors envoyé l'expéditeur en retour. Un recommmandé a un surcoût de 15% par rapport au courrier initial.

Q 2 . Modélisez les courriers recommandés, comment s'intègrent-ils à la version 1.0 ?

Version 3.0
On ajoute maintenant la prise en compte de **courriers urgents**. Tout courrier peut être "transformé" en courrier urgent (y compris les recommandés). L'action est inchangée et le coût est doublé.

Q 3 . Comment prendre en compte cette modification ? Faites une proposition qui s'intègre à la version 2.0 .

Version 4.0 – « Chaîne des naïfs » Chacun connaît les chaînes de courrier censées vous rendre millionnaire. Le fonctionnement est le suivant : vous recevez une lettre contenant une liste composée de quatre habitants et la lettre vous demande de :

- envoyer 5 euro à chacun des noms de la liste, sous la forme d'une « lettre de change » particulière ;
- supprimer le premier nom et mettre votre nom en dernière position ;
- envoyer la lettre (avec la nouvelle liste de noms) à 10 personnes, en leur expliquant le processus ;

En principe, vous devez tôt ou tard devenir riche...

Q 4 . Faites une proposition de conception pour prendre en compte ces nouveaux courriers. Intégrez là aux propositions précédentes.

Pour ne pas modifier la classe existante, vous pourrez créer une sous-classe d'`Habitant` qui aura la capacité de gérer les réponses ou non aux courriers pour naïfs.

Simulations (TP)

Q 5 . Créez une ville V et ses habitants (vous gèrerez aléatoirement les adresses, pour simplifier on considèrera que tous les habitants de la simulation habitent la ville V).

Chaque jour, pendant k jours, n_k habitants envoient un courrier à un autre habitant de la ville, choisi aléatoirement. Vous ferez varier les types des courriers envoyés. Eventuellement, ces courriers peuvent engendrer des réponses (accusés de réception, lettres de remerciement, etc.).

Simulez la collecte et la distribution du courrier tant qu'il reste des courriers à distribuer.

NB : Le caractère « urgent » des courriers n'est pas pris en compte dans cette simulation, urgent ou non un courrier arrive à destination le lendemain du jour où il est posté, simplement un courrier urgent coûte le double...

Q 6 . Créez une seconde simulation avec une « chaîne à naïfs » initiée par un seul courrier.

Dans cette simulation,

- il y a une certaine probabilité qu'un habitant donné réponde,
- un habitant dont le solde du compte est insuffisant ne répond pas,
- dès qu'un habitant ne répond pas une fois, il ne répondra plus jamais par la suite,
- un habitant qui aura répondu une première fois, répondra au moins les 5 fois suivantes, mais s'il ne reçoit jamais rien en retour il ne répondra plus à aucune courrier de ce type,
- un habitant qui a été bénéficiaire recommencera à répondre toujours avec espoir...

Vous ferez fontionner votre simulation pendant un certain nombre de jours, sauf s'il n'y a plus de courriers bien sûr... La simulation produit-elle des millionnaires ?