Université de Bourgogne
Année universitaire 2019-2020
Info 2a — programmation objet
TD 4 — *Principe de délégation (suite)*

Objet : pratique du principe de délégation.
En programmation objet, le programme fonctionne via les instances qui sont créées, chacune d'entre-elles « offrant » ses propres « services ». C'est la méthode `main` de la classe principale (celle qui représente le programme) qui crée les premières instances et articule les services qu'elles apportent. Dans ce cadre, le principe de délégation consiste à ne faire effectuer par une instance d'une classe que ce qu'elle « sait faire » mieux que les instances d'autres classes. C'est notamment le cas des méthodes qui utilisent les valeurs de ses attributs. De même, elle ne doit pas empiéter sur les « compétences » des autres instances et doit leur déléguer ce qu'elles « savent faire » mieux qu'elle.

Objets géométriques

Terminez éventuellement le TD précédent.

Compte bancaire

1. Un compte bancaire comporte un type : compte courant (CC), plan/compte épargne logement (PEL/CEL), plan d'épargne populaire (PEP), livret A…, un numéro de compte, un solde, un titulaire — qui est un client de la banque — et on peut y attacher une carte de paiement. Il peut être créé vide (solde = 0) ou avec un versement initial d'une certaine somme. On peut y effectuer des dépôts et des retraits d'une somme précisée en paramètre, ou des virements vers un autre compte. Il peut aussi être clôturé. Il comporte la liste des opérations qui ont été effectuées sur lui (création du compte, virements, retraits, clôture) associées à leur date.
2. Un client est représenté par son nom, un prénom, une ou deux adresses, un numéro de téléphone à dix chiffres. Il peut posséder un ou plusieurs comptes et peut connaître la somme totale dont il dispose en sommant les soldes de tous ses comptes.
3. Une adresse comporte un nom de voie, un type de voie, un numéro dans la voie, un code postal et un nom de ville.
4. Une carte de paiement comporte un type (visa, visa international, gold,…), un numéro, le nom et le prénom du titulaire du compte correspondant, elle a aussi une date d'expiration.

Questions

1. Déterminez quelles classes sont nécessaires pour représenter cette situation.
2. Faites un diagramme indiquant les relations d'utilisation entre classes.
3. Décrivez les attributs (noms et types), ainsi que les signatures des méthodes et des constructeurs nécessaires (hors `toString` et `equals`), en indiquant bien à quelles classes ils appartiennent (n'écrivez pas pour l'instant les corps des méthodes) ;
4. Sans écrire les corps des méthodes, discutez des attributs que doivent prendre en compte les `equals` et les `toString` des classes précédentes.

Si tout est fini...

Complétez le travail sur les figures géométriques en traitant complètement le triangle, et notamment en créant des *accesseurs en écriture virtuels*[1]. Vous pouvez aussi étudier comment modéliser une droite, un vecteur, un polygone…

[1] C'est un `setNom(…)` où `nom` n'est pas le nom d'un attribut mais celui d'une propriété dérivée d'un objet dont la modification détermine celle d'un ou plusieurs attributs. Par exemple, même si l'aire d'un disque n'est pas représentée par un attribut, il est possible de définir `setAire(…)` qui modifie le rayon du disque en fonction de la nouvelle surface. De même, pour un triangle défini par ses trois sommets `A`, `B` et `C`, il est possible d'écrire l'accesseur virtuel en écriture `setAB(…)` (respectivement `setAC(…)` ou `setBC(…)`) qui modifie les sommets `A` et `B` (respectivement `A` et `C` ou `B` et `C`).