Université du Québec à Rimouski

300, allée des Ursulines, C. P. 3300, succ. A
Rimouski (Québec) G5L 3A1, CANADA
www.uqar.ca

Département de mathématiques, d'informatique et de génie

Programmation Orientée Objet II
Devoir 2 — Énoncé

SIGLE :	INF11207
TITRE :	Programmation orientée objet II
GROUPE :	06
PROFESSEUR :	Steven Pigeon
	K-212
	steven_pigeon@uqar.ca
DATE DE REMISE :	8 mars, avant minuit

Supposons que vous devez produire le logiciel de gestion des comptes pour une banque — dont nous ne considérerons ici qu'une version extrêmement simplifiée. Le logiciel maintient les comptes individuels et gère la sauvegarde/lecture de l'information au disque.

Les comptes ont les caractéristiques suivantes :

- Tous les comptes doivent avoir un numéro *unique*.
- Tous les comptes sont assignés à un client (modélisé par un nom et une date de naissance).
- Tous les comptes ont un solde, qui ne peut pas être négatif, mais qui n'est pas borné supérieurement.

Les comptes viennent par ailleurs en trois types distincts : les *comptes normaux*, les *comptes pour enfants*, et les *comptes d'épargne-retraite*, chacun obéissant à des règles différentes.

Par ailleurs :

— Tous les types de comptes permettent que l'on examine le solde.

— Les *comptes normaux* permettent de déposer de l'argent, d'en retirer sans contrainte (en autant qu'il y ait des fonds disponibles, évidemment).

— Les *comptes pour enfants* sont liés à un compte pour adulte, celui du parent responsable. Les comptes pour enfants permettent de déposer de l'argent, mais la limite de retrait est établie à 10$ par jour (possiblement composés de plusieurs retraits), jusqu'à concurrence de 50$ par mois, s'il reste des fonds. Les clients qui ont 10 ans et moins reçoivent automatiquement un compte spécial pour enfants, et le compte doit être lié à un *compte normal*, celui du parent responsable.

— Les *comptes d'épargne-retraite* permettent qu'on y dépose de l'argent comme les comptes normaux, mais les retraits, en plus d'un montant, nécessitent une autorisation spéciale (qui serait, dans la réalité, accordée par un officiel de la banque).

Donc

1. **(13 pts)**. Proposez et implémentez une hiérarchie de classes qui respecte l'énoncé, et, autour, un programme C++ qui devra pouvoir, à la console :

   (a) Offrir un « menu » des diverses opérations (en mode texte comme pour le premier devoir fera très bien l'affaire),

   (b) Lister les informations de tous les comptes, ou d'un compte en particulier,

   (c) Créer un compte, en demandant le nom et la date de naissance du client (et s'il est trop jeune, demander le numéro de compte du parent),

   (d) Effectuer les opérations sur le compte : dépôt, retrait et afficher le solde (en demandant la date de l'opération ou l'autorisation, selon le compte, de façon à respecter *toutes* les contraintes),

   (e) Lire l'information des comptes au démarrage et la sauvegarder quand on quitte.

   Pour le fichier, prenez le format CSV, où chaque champ est séparé par des virgules et chaque enregistrement terminé par une nouvelle ligne — vous pouvez négliger en toute quiétude les cas pathologiques où on aurait des virgules dans les noms et où les données sont invalides.

Vous devez pouvoir lire et manipuler le fichier fourni avec cet énoncé.

Pour l'autorisation, considérez que vous n'aurez besoin que d'une classe qui « fait semblant » d'aller chercher l'autorisation dans une autre base de données. Vous devrez donc quand même concevoir cette classe, même si elle ne fait pas grand-chose.

Indice : Vous aurez besoin du patron de conception *factory* pour créer vos objets à la lecture de la base de données.

Vous *devez* valider les entrées : les dates doivent être numériques, les menus ne doivent accepter que les choix valides, une entrée imprévue ne doit pas lancer le programme dans une boucle sans fin, etc.

2. **(2 pts)**. À la lumière du précédent exercice, expliquez comment vous remplaceriez la hiérarchie de classe de comptes que vous avez conçue par une seule classe paramétrisée (sans l'implémenter, mais donnez un squelette de la classe suffisant pour la spécifier). Comparez les deux solutions. La quelle vous paraît plus adéquate (clarté, facilité de maintenance, etc.). Pourquoi ?

Cette fois-ci, vous aurez droit à la librairie standard : <`vector`>, <`list`>, <`map`>, <`fstream`>, en plus de <`string`>, <`iostream`> et <`iomanip`>. Évitez les structures « à la C » ou « à la C# ».

*
* *

La qualité du français écrit est de mise, et jusqu'à 10% de la note finale pourraient être retranchés. N'oubliez pas de citer vos sources, mêmes celles dont vous vous inspirez plus ou moins vaguement. Enfin, vous vous exposez à 20% de pénalité par jour de retard, jusqu'à concurrence de quatre jours — sauf circonstances exceptionnelles.

Quant à la remise du devoir (la partie 1), préparez une archive (.zip) qui contiendra votre solution Visual Studio 2010 *au complet* (ou peu importe l'environnement de développement que vous utiliserez), et dont le nom sera *votrenom*`.devoir2.zip`. Cette archive contiendra aussi vos réponses à la deuxième partie (un simple `.txt` suffira).