Master 1 à distance — PFA — Devoir n° 1

À rendre au plus tard le mardi 6 décembre 2016

Ce devoir est analogue, dans son principe, à l'exercice 1.16 (voir le chapitre 1).

1 Comptes en banque

Sur le modèle des compteurs (voir le chapitre 1), nous souhaitons définir une structure permettant la gestion d'un compte en banque. Voici quelles sont les informations — comparables aux méthodes privées dans les langages à objets — qu'il est préférable de gérer par une clôture lexicale :

— le solde (positif, négatif ou nul) du compte courant,

— le montant du découvert autorisé (c'est bien évidemment un nombre négatif ou nul).

Un compte bancaire est créé par l'évaluation d'une expression telle que :

`(define an-account (create-account` *init*`))`

où *init* est le solde initial (c'est un nombre positif ou nul) et `create-account` une fonction que vous avez à écrire. De même que les compteurs vus dans le chapitre 1, les comptes bancaires peuvent être gérés sous la forme d'une liste de fonctions partageant le même environnement lexical, soit par une fonction recevant des messages (c'est cette réalisation en *message-passing* qui conseillée). Voici quelles sont les fonctionnalités à implanter :

— consulter le solde courant ;

— effectuer un versement dont on précise le montant, puis retourner le nouveau solde ;

— effectuer un retrait dont on précise le montant, à condition que cette opération ne mette pas le solde en deçà du découvert autorisé ; si le retrait est possible, retourner le nouveau solde, sinon retourner la valeur `#f` ;

— effectuer un virement dont on précise le montant et le destinataire : les *deux comptes* doivent être mis à jour ; si cette opération est possible, retourner le nouveau solde du débiteur, sinon retourner la valeur `#f` ;

— modifier le découvert autorisé en la valeur donnée en argument et retourner cette valeur (à la création d'un compte, le découvert autorisé est de -300 euros).

2 Central de banque

À présent, ce n'est plus un compte qui est à gérer, mais *plusieurs* comptes, regroupés dans une liste d'associations de la forme :

$$\texttt{((}symb_0\ \texttt{ . }\ account_0\texttt{) (}symb_1\ \texttt{ . }\ account_1\texttt{) ...)}$$

— où $account_0, account_1, \ldots$ sont des comptes bancaires tels que nous les avons décrits au § 1, $symb_0, symb_1, \ldots$ sont des symboles désignant les titulaires des comptes correspondants — cette liste d'associations n'étant pas directement accessible, mais en clôture lexicale. De même que les comptes en banque, le central peut être réalisé par une liste de fonctions ou par une seule fonction recevant des messages.

Voici les fonctionnalités que doit réaliser le central de banque :

— retourner la liste des titulaires des comptes gérés par le central [1] ;

— ajouter un compte dont on fournit le nom et le solde de départ : cette opération n'est possible que si le nom ne correspond pas déjà à un titulaire [2] ; retourner le solde du compte créé ou `#f` si cette opération n'est pas possible ;

— fermer un compte dont on fournit le nom du titulaire : si le compte existe, effectuer cette opération et retourner le solde de ce compte au moment de sa suppression, sinon retourner `#f` ;

— consulter le solde d'un compte dont on donne le nom du titulaire ; retourner `#f` si le titulaire n'existe pas ;

— effectuer un versement (le titulaire et le montant étant précisés), puis retourner le nouveau solde ;

— effectuer un retrait (titulaire et montant précisés), à condition que cette opération ne mette pas le solde en deçà du découvert autorisé ; si le retrait est possible, retourner le nouveau solde, sinon retourner `#f` ;

— effectuer un virement dont on précise l'expéditeur, le montant et le destinataire ; si cette opération est possible, retourner le nouveau solde du débiteur, sinon retourner `#f` ;

— modifier le découvert autorisé (le titulaire est précisé, ainsi que le découvert accordé) et retourner la nouvelle valeur.

[1] On pourra utiliser la fonction prédéfinie `map`.
[2] On pourra utiliser les fonctions prédéfinies `assq` ou `assoc`.