Programmation Fonctionnelle
TP N°2

Gestion d'une bibliothèque

Note : Les fonctions demandées doivent être purement fonctionnelles (sans affectation), ce qui ne veut pas dire qu'on ne s'autorise pas à les utiliser avec des références (*atom*) :

Supposons que vous deviez développer une application pour gérer une bibliothèque. La bibliothèque possède une collection de livres, chaque livre a un titre, un auteur, un éditeur, une date de publication, un nombre d'exemplaires disponibles et un numéro ISBN (qui peut servir de clef primaire). La bibliothèque a également des lecteurs, dont on connaît le nom, le prénom, l'adresse et la liste des emprunts, et un identifiant qui est le hash de la concaténation du nom et du prénom). Un lecteur peut emprunter un livre pour une période limitée, seulement si il est disponible (nombre d'exemplaires disponibles supérieur strictement à 0).

Pour représenter ces données dans votre application, vous décidez d'utiliser des enregistrements Clojure pour les livres et les lecteurs. La collection des livres et la liste des lecteurs seront intégrées dans des atoms clojure.

1. Définissez l'enregistrement Book avec les champs suivants: titre, auteur, éditeur, date-de-publication, isbn et exemplaires-disponibles.
2. Créez une fonction appelée create-book qui prend cinq arguments représentant les champs d'un enregistrement Book, et renvoie un nouvel enregistrement Book avec exemplaires-disponibles initialisé à 1.
3. Créez une fonction appelée list-books-by-author qui prend une liste d'enregistrements Book et un nom d'auteur en argument, et renvoie une nouvelle liste d'enregistrements Book qui ont l'auteur spécifié.
4. Créez une fonction appelée sort-books-by-date qui prend une liste d'enregistrements Book en argument et renvoie une nouvelle liste d'enregistrements Book triés par date de publication, du plus ancien au plus récent.
5. Créez une fonction appelée get-book-isbn qui prend un enregistrement Book en argument et renvoie le numéro ISBN du livre.
6. Créez une fonction appelée set-book-isbn qui prend un enregistrement Book et un nouveau numéro ISBN en argument, et renvoie un nouvel enregistrement Book avec le numéro ISBN mis à jour.
7. Créez une fonction appelée print-book qui prend un enregistrement Book en argument et imprime le titre, l'auteur, l'éditeur, la date de publication et le numéro ISBN du livre sous une forme lisible.
8. Définissez l'enregistrement Borrower avec les champs suivants: nom, prénom, adresse et livres-empruntés.
9. Créez une fonction appelée create-borrower qui prend quatre arguments représentant les champs d'un enregistrement Borrower, et renvoie un nouvel enregistrement Borrower avec une liste vide de livres-empruntés (pour le hashage, voir https://clojuredocs.org/clojure.core/hash)
10. Créez une fonction appelée borrow-book qui prend une clef d'emprunteur, l'ISBN d'un livre et une date de prêt en argument. La fonction doit ajouter le livre à la liste des livres-empruntés du lecteur et décrémenter le champ exemplaires-disponibles du livre, seulement si le nombre d'exemplaires disponibles est supérieur à 0.
11. Créez une fonction appelée return-book qui prend une clef d'emprunteur et l'ISBN d'un livre en argument. La fonction doit retirer le livre de la liste des livres-empruntés du lecteur, incrémenter le champ exemplaires-disponibles du livre.

12. Créer une fonction appelée late-return qui renvoie la liste des emprunts (livre et emprunteur) ayant datant de plus de 1 mois.
13. Ecrire une fonction appelée save-lib qui sauvegarde la liste des livres et la liste des emprunteurs dans des fichiers (https://clojuredocs.org/clojure.java.io/writer)
14. Ecrire une fonction appelée load-lib qui charge la liste des livres et la liste des emprunteurs depuis des fichiers (https://clojuredocs.org/clojure.java.io/reader)

Et pour finir, une petite interface utilisateur (en mode texte) pour accéder aux fonctionnalités de la bibliothèque.