

Année universitaire : 2023-2024

Examen semestre 2

Matière : informatique

Classe : MP2, PC2, PT2

Nombre de pages : 4

Date : mai 2024

Durée : 2h

Problème : gestion d'une bibliothèque

L'objectif de ce problème est de développer un système de gestion de bibliothèque en utilisant le langage de programmation Python en se basant sur la notion de classes et de base de données dans la conception du code. Le programme doit permettre de gérer les livres, les emprunteurs ainsi que les opérations d'emprunt et de retour des livres dans la bibliothèque. Une pile sera utilisée pour gérer les emprunts et une file pour les listes d'attente.

Partie A : Pile, File et Orienté Objet

1. Créez une classe **Pile** ayant les méthodes :

- 1.a. **__init__(self)** : permettant d'initialiser un attribut d'instance **Items** par une liste vide pour contenir les éléments à empiler.
- 1.b. **push(self, item)** : permettant d'empiler l'élément item (un livre) dans la pile.
- 1.c. **pop(self)** : permettant de dépiler et renvoyer l'élément en haut de la pile (le dernier livre emprunté).
- 1.d. **is_empty(self)** : Vérifie si la pile est vide.
- 1.e. **sommet(self)** : Affiche le sommet de la pile sans le dépiler.
- 1.f. **display(self)** : Affiche le contenu de la pile sans le perdre.

!!! Les commandes sur les listes ne sont pas autorisées. Vous pouvez utiliser une pile auxiliaire.

2. Créez une classe **File** ayant les méthodes :

- 2.a. **__init__(self)** : permettant d'initialiser un attribut d'instance **Items** par une liste vide pour contenir les éléments à enfiler (des instances de la classe emprunteur).
- 2.b. **enqueue(self, item)** : Enfile un élément (un emprunteur en attente).
- 2.c. **dequeue(self)** : Défile et renvoie l'élément en tête de la file (l'emprunteur en tête de la liste d'attente).
- 2.d. **is_empty(self)** : Vérifie si la file est vide sans le perdre.
- 2.e. **sommet(self)** : Affiche l'élément en tête de la file sans le défiler.
- 2.f. **display(self)** : Affiche le contenu de la file sans le perdre.

!!! Les commandes sur les listes ne sont pas autorisées. Vous pouvez utiliser une file auxiliaire.

3. Sachant qu'un livre a comme propriétés son titre, son auteur, sa date de publication et son emprunteur (None par défaut), Créez une classe **Livre** ayant les méthodes :

- 3.a. **__init__(self, titre, auteur, date_publication)** : permettant d'initialiser un livre avec ses détails)

3.b. `__str__(self)` : Renvoie une représentation sous forme de chaîne du livre.

Exemple d'utilisation
<pre># Création des instances de livres et affichage. >>> livre1 = Livre("titre1", "Auteur 1", "2022") >>> livre2 = Livre("titre2", "Auteur 2", "2021") >>> livre3 = Livre("titre3", "Auteur 3", "2023") >>> livre4 = Livre("titre4", "Auteur 1", "2023") >>> print(livre1) titre1 de Auteur 1 publié en 2022</pre>

4. Sachant qu'un emprunteur a comme propriétés : nom, numero_adherent et emprunts_pile (une pile pour suivre les livres empruntés), Créez une classe **Emprunteur** avec les méthodes :

4.a. `__init__(self, nom, numero_adherent)` : permettant d'initialiser un emprunteur avec ses détails.

4.b. `__str__(self)` : Renvoie une représentation sous forme de chaîne d'un emprunteur.

Exemple d'utilisation
<pre># Création d'instances d'emprunteurs et affichage. >>> emprunteur1 = Emprunteur("nom_Emprunteur1", "A12345") >>> emprunteur2 = Emprunteur("nom_Emprunteur2", "B67890") >>> print(emprunteur1) Adhérent : nom_Emprunteur1 identifiant : A12345</pre>

5. Définir une classe **Bibliothèque** permettant de gérer les emprunts.

5.a. Le constructeur `__init__` de cette classe initialise deux attributs :

- *collection* : une liste vide pour contenir les livres de la bibliothèque,
- *liste_attente* : un dictionnaire qui associe un titre de livre (clé) à une file d'attente d'emprunteurs (valeur).

Les autres méthodes de la classe sont :

5.b. `ajouter_livre(self, livre)` : Ajoute un livre à la collection de la bibliothèque.

5.c. `emprunter_livre(self, emprunteur, titre_livre)` : Permet à un emprunteur d'emprunter un livre s'il est disponible. Si le livre est déjà emprunté, l'ajoute à la liste d'attente.

5.d. `rendre_livre(self, emprunteur, titre_livre)` : Permet à un emprunteur de retourner un livre. Si d'autres personnes attendent ce livre, le prête à la première personne de la liste d'attente.

5.e. `afficher_livres_disponibles(self)` : Affiche la liste des livres disponibles dans la bibliothèque.

5.f. `afficher_liste_attente(self, titre_livre)`: Affiche la liste d'attente pour un livre donné.

Exemple d'utilisation
<pre># Création d'une instance de la bibliothèque. >>> biblio = Bibliothèque() # Ajoutez des livres à la biblio. >>> biblio.ajouter_livre(livre1) >>> biblio.ajouter_livre(livre2) >>> biblio.ajouter_livre(livre3) >>> biblio.ajouter_livre(livre4) # Demande d'emprunts (livre1 est demandé par deux emprunteurs)</pre>

```

>>> biblio.emprunter_livre(emprunteur1, "titre1")
>>> biblio.emprunter_livre(emprunteur2, "titre2")
>>> biblio.emprunter_livre(emprunteur2, "titre1")
# Affichage des livres disponibles dans la bibliothèque.
>>> biblio.afficher_livres_disponibles()
titre3 de Auteur 3 publié en 2023
titre4 de Auteur 1 publié en 2023
# Affichage de la liste d'attente d'un livre donné.
>>> biblio.afficher_liste_attente("titre1")
Adhérent : nom_Emprunteur2 identifiant : B67890
# Rendre un livre.
>>> biblio.rendre_livre(emprunteur1, "titre1")
# Après retour de titre1, celui est automatiquement accordé à
emprunteur2 (le premier dans la file d'attente)

```

Partie B : Orienté Objet et SQLite

Pour permettre une gestion efficace et persistante de la bibliothèque, une nouvelle approche consiste à stocker les données sur les livres, les emprunteurs et les transactions d'emprunt dans une base de données.

6. Créez une classe **BibliothèqueDB** permettant de gérer les livres et les emprunts en utilisant une base de données SQLite.

6.a. **__init__** : Le constructeur de cette classe initialise une connexion à la base de données avec les tables **livres**, **emprunteurs**, **transactions** suivantes :

- **Livres** (titre (TEXT), auteur (TEXT), date_publication (TEXT), emprunteur(TEXT))
- **emprunteurs** (nom (TEXT), numero_adherent (INTEGER))
- **transactions** (emprunteur (TEXT), titre_livre (TEXT), date_emprunt (TEXT), date_retour (TEXT))

L'attribut **emprunteur** de la table **Livres** représente le nom de l'emprunteur actuel du livre, peut être NULL si le livre n'est pas emprunté.

La table **transactions** sert à enregistrer les transactions d'emprunt.

L'attribut **date_retour** de la table **transactions** contient la date de retour prévue du livre, peut être NULL si le livre n'a pas encore été retourné.

Les autres méthodes de la classe **BibliothèqueDB** sont les suivantes :

- 6.b. **ajouter_livreDB(self, livre)**: Ajoute un livre à la collection de la bibliothèque.
- 6.c. **ajouter_livreDB(self, livre)**: Ajoute un livre à la collection de la bibliothèque.
- 6.d. **emprunter_livreDB(self, emprunteur, titre_livre)**: Permet à un emprunteur d'emprunter un livre s'il est disponible. Si le livre est déjà emprunté, l'ajoute à la liste d'attente.
- 6.e. **rendre_livreDB(self, emprunteur, titre_livre)**: Permet à un emprunteur de retourner un livre. Si d'autres personnes attendent ce livre, le prête à la première personne de la liste d'attente.
- 6.f. **afficher_livres_disponiblesDB(self)**: Affiche la liste des livres disponibles dans la bibliothèque.
- 6.g. **afficher_liste_attenteDB(self, titre_livre)**: Affiche la liste d'attente pour un livre donné.
- 6.h. **__del__(self)** : ferme la connexion.

Exemple d'utilisation

```
# Création d'une instance de la bibliothèque avec le nom de la base de données
biblio_db = BibliothèqueBD("ma_bibliotheque.db")

# Ajout de livres à la bibliothèque
livre1 = Livre("Harry Potter", "J.K. Rowling", "1997")
livre2 = Livre("Le Seigneur des Anneaux", "J.R.R. Tolkien", "1954")
livre3 = Livre("1984", "George Orwell", "1949")

biblio_db.ajouter_livreBD(livre1)
biblio_db.ajouter_livreBD(livre2)
biblio_db.ajouter_livreBD(livre3)

# Emprunt d'un livre
emprunteur1 = Emprunteur("John Doe", "JD123")
biblio_db.emprunter_livreBD(emprunteur1, "Harry Potter")

# Affichage des livres disponibles
print("Livres disponibles dans la bibliothèque:")
biblio_db.afficher_livres_disponiblesBD()

# Affichage de la liste d'attente pour un livre
print("Liste d'attente pour le livre 'Harry Potter':")
biblio_db.afficher_liste_attenteBD("Harry Potter")

# Rendre un livre
biblio_db.rendre_livreBD(emprunteur1, "Harry Potter")

# Affichage des livres disponibles après retour
print("Livres disponibles après retour de 'Harry Potter':")
biblio_db.afficher_livres_disponiblesBD()

# Suppression de l'instance de la bibliothèque (fermeture de la connexion à la base de données)
del biblio_db
```

Partie C : SQL

Ecrire les requêtes SQL permettant de

7. Fournir la liste des livres actuellement empruntés par des emprunteurs dont le numéro d'adhérent est supérieur à 10000, avec les détails des emprunteurs et les dates d'emprunt et de retour prévues?
8. Donner le nombre de livres empruntés par chaque emprunteur dans la bibliothèque.
9. sélectionner les emprunteurs ayant emprunté plus de 2 livres dans la bibliothèque.
10. Sélectionnez les emprunteurs ayant emprunté plus de 2 livres écrits par l'auteur 'George Orwell' dans la bibliothèque, ainsi que les détails de ces emprunteurs.