Rappels POO : Ludothèque

TP01 du Module 01 – Rappels POO

Objectif : Révision des concepts POO

Durée estimée
3-4 heures

Énoncé

Nous sommes une association qui gère une ludothèque. La ludothèque a besoin d'un logiciel de gestion des adhésions et d'emprunt des jeux.

Les personnes qui adhèrent sont appelées des membres, pour chaque membre nous avons besoin de leur nom, prénom, numéro de téléphone. Nous avons aussi besoin de savoir à quel moment ils ont adhéré, la durée de l'abonnement est de 1 an à partir de la date de l'adhésion.

Nous souhaitons conserver la date du début de l'emprunt ainsi que la date de restitution. Les jeux ne doivent pas être emprunté plus de 1 mois même si souvent les jeux sont retournés avec du retard. Les membres ne peuvent emprunter que 2 jeux en même temps.

La ludothèque gère des jeux de 2 types.

- Les jouets des enfants ;
- Les jeux de sociétés qui peuvent être aussi bien pour les enfants que pour les adultes.

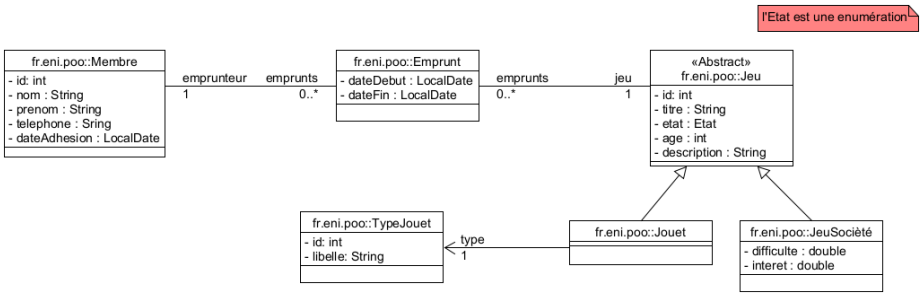
Dans les 2 cas on notera l'état du jeu ainsi que l'âge recommandé. Il est nécessaire d'avoir une description du jeu pour lister notamment les éléments contenus dans le jeu.

En outre nous devons connaitre pour les jouets le type de jouets (Playmobil, Légo, Construction, Puzzle), nous pouvons à tout moment créer un nouveau type.

Pour les jeux de société nous allons nous récupérer des informations sur le site de jeu BoardGameGeek et nous allons y récupérer la durée moyenne d'une partie ainsi que 2 notes :

- Une note de difficulté du jeu (réel entre 0 et 5) (appelée weight sur le site)
- Une note d'intérêt du jeu (réel entre 0 et 10) (appelée average rating sur le site)

A travers le texte précédent, Nous avons défini le diagramme de classe suivant.



I. Création des classes

Si vous vous sentez à l'aise les 2 premières parties peuvent être réalisés en parallèle.

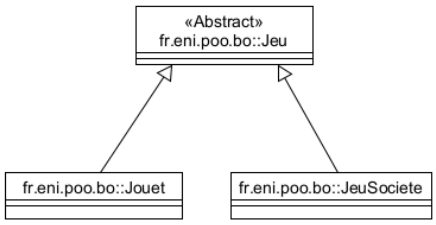
Réaliser les éléments suivants en accord avec le diagramme de classe.

Les classes de ce TP sont toutes créées dans un package **fr.eni.poo**

1. Créer les classes **Membre**, **Emprunt, TypeJouet**. L'état des jeux est une énumération qui peut prendre les valeurs suivantes :
   - Bon ;
   - Correct ;
   - Mauvais.

2. Ajouter à l'aide de l'IDE un constructeur par défaut, un constructeur avec tous les champs, les getter/setter.

3. Créer la classe abstraite **Jeu,** pour le moment la classe ne contient aucune méthode.



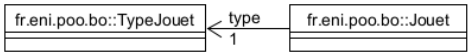
4. Créer les classes **Jouet** et **JeuSociete** comme des classes qui héritent de la classe **Jeu**. Pour ces 3 classes, créer un constructeur par défaut, un constructeur avec tous les champs, les getter/setter.

5. Utiliser la classe **TestPremierePartie**, dans le dossier test, pour tester la création des objets

```
-------------- Membre ------------------
ID: 1, Nom: Doe, Prénom: John, Téléphone: 0666666666, Date
d'inscription: 2024-06-12
ID: 2, Nom: Doe, Prénom: Jane, Téléphone: 0777777777, Date
d'inscription: 2023-12-25
-------------- TypeJouet ------------------
ID: 1, Nom: Jouets éducatifs
ID: 2, Nom: Puzzle
-------------- Jeu ------------------
ID : 1, Titre : scrabble, Etat: BON, à partir de 7 ans
ID : 2, Titre : Carte du monde, Etat: CORRECT, à partir de 3 ans
ID : 3, Titre : Les Colons de Catane, Etat: MAUVAIS, à partir de 10
ans
ID : 4, Titre : Terraforming Mars, Etat: CORRECT, à partir de 12 ans
-------------- Emprunt ------------------
Date d'emprunt: 2024-06-12, Date de retour: 2023-08-21
```

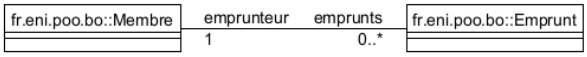
II. Lien entre les classes

1. Traduire en programmation l'association bidirectionnelle entre le **Jouet** et le **TypeJouet**.



   a. Associer chaque Jouet à un TypeJouet à l'aide de la méthode **setType().**
   b. Ajouter un constructeur à la classe Jouet qui permet de prendre en compte le type de jouet.

2. Traduire en programmation l'association bidirectionnelle entre le Membre et l'Emprunt.



   a. Associer plusieurs emprunts à un membre spécifique à l'aide d'une méthode **ajouterEmprunt(Emprunt emprunt)** dans la classe Membre.

3. Traduire en programmation l'association bidirectionnelle entre le Jeu et l'Emprunt

   a. Associer plusieurs emprunts à un jeu spécifique à l'aide d'une méthode **ajouterEmprunt(Emprunt emprunt)** dans la classe jeu.

4. Créer un nouveau constructeur pour l'emprunt :

```
public Emprunt(LocalDate dateDebut, LocalDate dateFin, Jeu jeu, Membre
emprunteur)
```

Ce dernier permet d'y associer l'emprunteur et le jeu emprunté. Les fonctions **ajouterEmprunt** créer précédemment sont utilisées lors de la création de l'emprunt pour y associer le membre et le jeu.

5. Utiliser la classe **TestDeuxiemePartie**, dans le dossier test, pour tester la création des associations.

```
--------------- Membre -------------------
Nom: Dupont, Prénom: Jean, Téléphone: 0123456789, Date d'inscription:
2023-01-10
Emprunt de Jean
de 2024-01-21 à 2023-02-21 : Terraforming Mars
de 2024-02-21 à 2023-03-21 : Les Colons de Catane

Nom: Martin, Prénom: Claire, Téléphone: 0987654321, Date
d'inscription: 2022-06-15
Emprunt de Martin
de 2024-03-10 à 2023-04-10 : Teddy bear
de 2024-04-10 à 2023-05-10 : puzzle paris

--------------- Jeu -------------------
Titre: Terraforming Mars, Etat: CORRECT, à partir de 12 ans
Emprunt du jeu Terraforming Mars
de 2024-01-21 à 2023-02-21 : Dupont Jean

Titre: Les Colons de Catane, Etat: MAUVAIS, à partir de 10 ans
Emprunt du jeu Les Colons de Catane
de 2024-02-21 à 2023-03-21 : Dupont Jean

Titre: Teddy bear, Etat: MAUVAIS, à partir de 3 ans
Emprunt du jeu Teddy bear
de 2024-03-10 à 2023-04-10 : Martin Claire

Titre: puzzle paris, Etat: BON, à partir de 5 ans
Emprunt du jeu puzzle paris
de 2024-04-10 à 2023-05-10 : Martin Claire
```

III. Ajout de fonctionnalités (optionnel)

Nous allons créer une classe Ludothèque dans laquelle nous allons implémenter les méthodes qui vont nous permettre de gérer notre ludothèque.
Cette classe possède une liste de **Jeu** (ArrayList) et une liste de **Membre** (ArrayList). L'ensemble des fonctions de la classe ludothèque va permettre de gérer les emprunts des jeux contenus dans la liste de jeux par les membres de la ludothèque. Ces membres sont répertoriés grâce à la liste de membres.
Précision, la date de fin n'est renseignée que lorsque que le jeu est restitué. Lorsque la date de fin n'est pas renseignée alors l'emprunt n'est pas terminé, on considère que l'emprunt ne dois pas dépasser un mois sinon on considère que le membre est en retard.

1. Dans la classe **Jeu** ajouté une fonction abstraite **affichageInformation** créer les implémentations de cette fonction dans **Jouet** et **JeuSociete** qui affiche les informations pour les jouets et les jeux de sociétés.
2. Méthode **inscrireMembre(Membre membre)** : Inscrit un nouveau membre à la ludothèque en ajoutant le membre à la liste des membres.
3. Méthode **ajouterJeu(Jeu jeu)** : Ajoute un nouveau jeu à la ludothèque en ajoutant le jeu à la liste des jeux disponibles.
4. Méthode **emprunterJeu (Membre membre, Jeu jeu, LocalDate dateDebut)** : Permet à un membre d'emprunter un jeu à partir d'une date spécifiée. Vérifie d'abord si le jeu est disponible (pas encore emprunté). Si le jeu est disponible, un nouvel emprunt est créé. Lorsqu'un emprunt est créé, si la date de début n'est pas donnée alors cette date est positionnée à la date du jour.
5. Méthode **retournerJeu(Membre membre, Jeu jeu)** : Permet à un membre de retourner un jeu emprunté. Trouve l'emprunt correspondant et met à jour la date de fin de l'emprunt.
6. Méthode **afficherEmpruntsMembre(Membre membre)** : Affiche tous les emprunts effectués par un membre donné.
7. Méthode **afficherJeuxEnRetard()** : Affiche tous les jeux en retard, c'est-à-dire les jeux qui n'ont pas été retournés après un délai d'un mois depuis la date d'emprunt.

Indications

Proposez des solutions les plus « orientées objet » possible

Solution

Une solution est proposée pour ce TP sous la forme de 3 projets à intégrer dans Eclipse.

Les 3 projets représentent une partie du TP.

Commenté [JM1]: À placer à la fin de chaque énoncé, la phrase peut être différente.