Java @ Et3-info > projet

2021-22

L'objet de ce projet noté est de développer en langage Java une application simplifiée permettant la gestion de la collection de jeux vidéo d'un réseau social. Ce projet à réaliser individuellement et est à rendre selon les modalités indiquées à la fin de ce document.

I Description de l'application

L'application permet de représenter un ensemble de joueurs de jeux vidéo qui pourront être mis en relation afin de prendre part à des parties *multijoueurs*.

Un **joueur** humain est identifié par un pseudo, un email et une date de naissance. Il possède au moins une **machine de jeu** (ex. PC, PS3, GBA) et un ensemble de **jeux** (chacun pour une machine particulière). [1] Un **joueur** peut également avoir accepté d'autres joueurs comme **amis** avec lesquels il pourra prendre part à des parties *multijoueurs* s'ils possèdent tous les deux le même jeu. [2]

Un **joueur** a un profil, lequel est notamment caractérisé par des restrictions (cf. table 1) :
(a) le nombre maximal de jeux possédés
(b) la possibilité d'acquérir ou offrir des jeux
(c) le nombre d'amis
(d) la possibilité d'inviter des joueurs comme amis
(e) le nombre de parties *multijoueurs* au maximum dans une même journée

	(a)	(b)	(c)	(d)	(e)
standard	50	oui	100	oui (autres joueurs et bots) ses enfants uniquement	5
gold	illimité	oui	illimité	oui (autres joueurs et bots) ses enfants uniquement	10
enfant	30	non	10	non autres enfants uniquement	3
bot	illimité	non	illimité	non	illimité

FIGURE 1 – Profils de joueurs principaux

1. On suppose ici que tous les jeux disposent d'un mode *multijoueurs*, et qu'il est donc pertinent de chercher d'autres joueurs pour y jouer. De plus, afin de simplifier l'application, on considère qu'une partie *multijoueurs* ne permet qu'à 2 joueurs de prendre part simultanément à une partie *multijoueur*.
2. On suppose pour simplifier que les jeux sont compatibles entre eux pour le jeu en ligne quelle que soit leur machine de jeu.

Un joueur ayant le profil ***enfant*** ne peut donc pas acquérir de jeux ni inviter d'autres joueurs comme amis, sauf si ce sont des enfants. Seuls peuvent offrir des jeux à un enfant des membres qui sont son parent/tuteur. [3] L'inscription d'un enfant doit passer par la demande d'un membre qui est son parent. Un ***enfant*** et ses parents/tuteurs font nécessairement réciproquement partie de leurs amis.

Un joueur de profil ***bot*** correspond à un système informatique susceptible de jouer à l'aide d'un module d'intelligence artificielle pour un jeu. La notion de *jeu possédé* pour lui correspond simplement à un jeu pour lequel il dispose du module d'intelligence artificielle approprié. Ce type de joueur ne fait que répondre à des invitations comme ami (en les acceptant par défaut, sauf si elle dépasse le quota d'un joueur ***standard***). Il existe au plus un seul ***bot*** par jeu (indépendamment de la plate-forme) : on ne créera donc les ***bots*** que lorsque nécessaire. De plus, on suppose qu'il n'existe pas d'intelligence artificielle implémentée pour tous les jeux. [4]

Pour tous les joueurs hormis les ***bots***, on enregistrera la liste des parties effectivement jouées par jour unique (jeu, ami joueur).

L'affichage des informations des joueurs dépendra de leur profil : affichage sans restrictions pour les joueurs ***standard*** et ***gold***, affichage avec uniquement le pseudo et le nombre de parties sans leurs détails pour un joueur ***enfant***, affichage avec uniquement le pseudo et la liste de jeux pour un joueur ***bot***.

L'application permettra d'effectuer les actions suivantes :

1. Création d'un compte de joueur. [5]
2. Affichage du profil d'un joueur.
3. Invitation par un joueur d'un joueur comme ami par pseudo. [6]
4. Suppression d'un ami de la liste d'amis d'un joueur. [7]
5. Parcours de la liste des jeux par machine et catégorie.
6. Affichage des informations sur un jeu. [8]
7. Acheter un jeu pour un joueur.
8. Offrir un jeu pour un joueur ami.
9. Proposer une partie *multijoueurs* d'un jeu particulier à un ami possédant ce jeu. [9]
10. (*optionnel*) Recherche par un joueur d'un joueur à inviter comme ami par jeu. [10]
11. (*optionnel*) Affichage de statistiques sur le réseau social : classement des joueurs ayant le plus joué ce jour, classement des jeux les plus joués (nombre de parties *multijoueurs*), autres.
12. (*optionnel*) Passage d'un membre de profil ***enfant*** à l'âge adulte, vers un profil ***standard*** ou ***gold***.

3. On suppose, pour simplifier, qu'un enfant peut avoir un ou deux parents/tuteurs.

4. Pour simplifier, on considérera qu'il existera une intelligence artificielle pour tout jeu mis sur le marché à partir d'une année particulière (paramètre de l'application).

5. La création d'un compte ***enfant*** doit passer par le compte d'un adulte qui est son parent/tuteur. Cela peut être l'occasion de mentionner un possible autre parent/tuteur parmi les autres joueurs.

6. Par simplification, on considère que *toutes* les invitations possibles seront acceptées, à concurrence de la possibilité d'ajouter un nouvel ami pour les 2 joueurs concernés.

7. Action réciproque et acceptée sans confirmation.

8. Voir plus bas les informations accessibles dans le jeu de données disponible proposé.

9. Par simplification, on considère que *toutes* les invitations possibles seront acceptées, à concurrence de la possibilité de participer à une nouvelle partie pour les 2 joueurs concernés.

10. Le critère de sélection et de tri des joueurs potentiels pourra être adapté comme jugé utile : on pourra par exemple préférer des joueurs qui possèdent de nombreux jeux en commun et le moins d'amis déjà acceptés.

13. (*optionnel*) Simulation du résultat des parties *multijoueurs*, et conservation par joueur (sauf ***bots***) de la proportion de victoires/matchs nuls/défaites par jeu.
14. (*optionnel*) Permettre la recherche d'un jeu par mots-clés de son titre.

II Travail à réaliser

Vous devrez développer une application en langage `Java` permettant d'implémenter la description ci-dessus. Il faudra en particulier :

1. Proposer une interface humain-machine très simple en mode texte pour l'application.
2. Utiliser un jeu de données disponible pour la collection de jeux vidéos :
   `https://raw.githubusercontent.com/stef-aramp/video_games_sales/master/vgsales.csv`
   Cette collection est au format CSV (*comma-separated values*) selon le format suivant :

   ```
   Rank,Name,Platform,Year,Genre,Publisher,NA_Sales,EU_Sales,JP_Sales,Other_Sales,Global_Sales
   ```

   dont l'exemple suivant est un extrait :

   ```
   15467,Dino Dini's Kick Off Revival,PS4,2016,Sports,Avanquest,0,0.02,0,0,0.02
   ```
3. Identifier tous les cas d'erreurs possibles à l'exécution dans l'application, et proposer des traitements appropriés.
4. Documenter votre code avec `JavaDoc`, avec une documentation pour l'application au niveau de la classe principale.

III Rendu attendu

Ce travail est à réaliser et à rendre individuellement. Le programme sera écrit en langage Java en mettant en œuvre les principes étudiés lors du module. Une attention particulière devra être portée au modèle objet proposé, à l'encapsulation des données, à la gestion des erreurs, et à la documentation du code au moyen d'annotations JavaDoc.

Le rendu sera livré sous forme d'une archive JAR au nom de l'étudiant.e qui contiendra les répertoires suivants :
— `src` : l'ensemble des sources du projet (`.java`).
— `bin` : l'ensemble des classes compilées du projet (`.class`).
— `doc` : l'arborescence JavaDoc du projet.

Chaque archive sera à envoyer par email **pour la date mentionnée sur la page eCampus du module** à l'enseignant avec comme titre d'email :

```
[Polytech > Et3] rendu projet Java NOM_ETUDIANT.E
```