Citadelles - Équipe E

Bienvenue dans Citadelles !
Version anglaise

Ce projet a été développé dans le cadre d'un projet collaboratif en programmation. Il s'agit d'un programme en Java permettant à 4 robots de jouer à la première édition du jeu Citadelles.

Table des matières

Sujet

Ce projet vise à créer une version informatisée du jeu "Citadelles", en se basant sur la première édition, en intégrant les fonctionnalités suivantes :

Fonctionnalités :

- **Interface de jeu :**
  - Représentation des cartes et des villes de chaque joueur.
- **Moteur de jeu :**
  - Gestion de la mise en place, des tours, des actions, des interactions, de la progression et de la fin de la partie.
- **Développement de robots de jeu :**
  - Du niveau de base jusqu'à des stratégies sophistiquées démontrables.
- **Simulation de parties :**
  - Implémentation d'une simulation de partie entre 4 robots, avec calcul des points et établissement d'un classement à la fin.
- **Visualisation textuelle simplifiée :**
  - Affichage textuel de l'état actuel du jeu.

Membres de l'équipe

Ce projet a été réalisé par l'équipe Polytâche (E), composée de 4 étudiants de 3e année de la promotion 2023 de Polytech Nice Sophia en filière Sciences Informatiques (SI) :

- Darina Chan
- Océan Razafiarison
- Mathis Jullien
- Quentin Elleon

Instructions d'utilisation

Prérequis

Avant d'utiliser ce programme, assurez-vous d'avoir les éléments suivants installés sur votre machine :

- **Java 17** : Le projet est développé en Java 17. Vous pouvez le télécharger depuis le site officiel d'OpenJDK.
- **Git** : Si vous ne l'avez pas déjà, installez Git depuis le site officiel de Git.
- **Maven** : Utilisé pour compiler et exécuter le programme, vous pouvez le télécharger depuis le site officiel de Maven.

Installation

Pour installer le programme :

Ouvrez une console et placez-vous dans le répertoire où vous souhaitez télécharger le programme.

```
git clone https://github.com/pns-si3-projects/projet2-ps-23-24-citadels-2024-e.git
cd projet2-ps-23-24-citadels-2024-e
```

Exploration du Projet

- **'src/main/java/fr/cotedazur/univ/polytech/startingpoint'** : Contient le code source.
- **'src/test/java/fr/cotedazur/univ/polytech/startingpoint'** : Contient les tests unitaires.

Utilisation

Compiler et Exécuter

Assurez-vous d'être dans le répertoire racine du projet (**'projet2-ps-23-24-citadels-2024-e'**). Vous pouvez alors compiler et exécuter le programme principal avec la commande Maven suivante :

```
mvn exec:java
```

Exécution des Tests

Les tests unitaires sont situés dans le répertoire `src/test/java/fr/cotedazur/univ/polytech/startingpoint`. Pour les exécuter, utilisez la commande suivante :

```
mvn clean package
```

Assurez-vous que tous les tests passent sans erreurs, confirmant ainsi la solidité du programme.

Point d'avancement

Fonctionalités réalisées

`Toutes` les fonctionalités `basiques` du plateau de jeu ont étés réalisées.
Les seules règles n'ayant pas étés traitées sont celles des limitations techniques d'un jeu de société (nombre de pièces dans la banque).

Système de logs

Les logs ont étés réalisés à l'aide de la librairie de logging interne a java (java.util.logging).

- `Problème :` Nous avons besoin de régler les logs en deux parties différentes puisque nous ne devons pas avoir les logs d'une partie normale (description de chaque joueur, leurs choix...etc) lors de l'éxécution des 2x1000 parties.
- `Solution :` Pour pallier à ce problème, nous avons crée un second logger avec un niveau différent du premier, ce qui nous permet donc d'activer ou de désactiver le logger non voulu à notre guise.

Archive des statistiques sous forme de CSV

Lorsque nous mettons comme argument --csv, nous sauvegardons alors les parties jouées. 20 parties sont jouées et nous obtenons un CSV de la forme suivante :

	Nom du joueur	Algorithme du joueur	Score Moyen	Nombre de parties jouées	Nombre de 1ere places	Nombre de 2e places	Nombre de 3e places	Nombre de 4e p
1	Fifi	Richard	21	20	2	3	7	
2	Picsou	Einstein	28	20	8	8	3	
3	Riri	Richard	20	20	1	4	7	
4	Donald	Einstein	27	20	9	5	3	

Il est bon de noter que lorsque nous réexcutons la même commande avec --csv, les données sont agrégées et les scores moyens sont recalculés (scoreInitial + nouveauScore)/2
Explication rapide :

- *Main.java :* C'est ici que les nouvelles données sont stockées et calculées avant de les envoyer au programme les rajoutant dans le CSV. Elles sont stockées sous la forme de deux HashMap, les deux ayant pour clé le nom du joueur mais avec une différence dans leur valeur :
  `totalScores` associe son score à un joueur.
  `totalPlacements` associe un array de quatre entiers représentant la 1ère place, 2e place... à un joueur, donnant donc le nombre de fois où il a obtenu cette place.
- *Csv.java :* C'est ici que les données sont injectées dans le Csv. Son principe est simple et se divise en deux parties :
  `Vérification de l'existence du fichier` , s'il n'existe pas, en créer un avec les en-têtes de colonnes (resetStats).
  `Rajout de données` si le fichier existe, lire les données déjà existantes et les additionner (ou dans le cas des scores, faire une moyenne) avec celles que nous voulons rajouter. Cela est géré en regardant la colonne n°1 qui est le nom du joueur et en ajoutant ses valeurs a la liste envoyée par Main correspondante.
  `Si le joueur n'est pas déjà dans la base de données` mais que nous avons d'autres joueurs déjà présents, il sera alors rajouté à la fin.
  `Enfin` le fichier est reinitialisé, nous rajoutons les en-têtes et les nouvelles données.

Bot Richard

//Océan-Mathis

Pourquoi Einstein bat-il Richard ?

Einstein joue pour lui même : La principale différence vient du fait qu'Einstein joue pour lui-même et pour maximiser ses propres chances. Il va essayer de récupérer le plus d'argent possible et de construire ses bâtiment le plus vite possible. `A l'instar` de Richard qui lui essaie de déstabiliser ses adversaires et de les faire perdre. Dans une partie avec quatre joueurs, celui se priorisant sur les autres est voué à être plus performant sur le long terme en général.

Architecture du projet

//Océan

Processus

Le projet est divisé en plusieurs catégories toutes différentes les unes des autres :

- `Personnages :` Quentin et Darina.
- `Merveilles :` Mathis et Océan.
- `Planification et gestion de l'architecture du projet :` Océan.
- `Stratégie Richard :` Mathis et Océan.
- `Stratégie Einstein :` Quentin et Darina.
- `Implémentation Jcommander :` Darina.
- `2x1000 et csv :` Quentin.
- `Logger :` Mathis.
- `Calcul des scores :` Mathis.
- `Tests unitaires et Mocks :` Tout le monde mais surtout Darina.

Process de l'équipe :
//Mathis

Nous espérons que vous apprécierez l'utilisation de notre programme ! N'hésitez pas à lancer plusieurs parties pour découvrir les différentes possibilités.