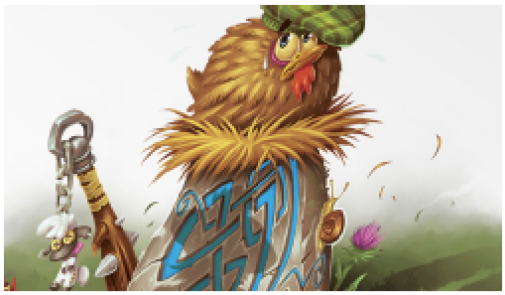
Projet Schotten-Totten en Java

PG220



Introduction

Dans ce projet, il s'agit de concevoir et développer une application permettant de jouer au jeu de société Schotten-Totten, créé par Reiner Knizia. Pour les 20 ans de ce jeu populaire, une nouvelle édition appelée Schotten-Totten-2 a été créée, ajoutant une dimension asymétrique au premier jeu. De plus, il existe plusieurs variantes pour jouer (normal, tactique, experts).

Dans la suite du sujet, pour simplifier, quand on parlera d'une édition du jeu, on entendra indifféremment une des éditions du jeu joué dans une des variantes. Les différentes éditions utilisent les mêmes mécaniques de base, mais diffèrent sur les cartes utilisées, le nombre de cartes distribuées, l'utilisation de cartes tactiques ou non, le nombre de bornes/murailles que l'on peut revendiquer à chaque tour, la présence ou non de certains éléments tactiques, etc.

Pour découvrir ces jeux, vous pouvez consulter par exemple les règles de Schotten-Totten et celles de Schotten-Totten-2, ou regarder des tutoriels (nombreux). Il est conseillé de jouer quelques parties (en commençant par la variante de base) pour bien comprendre les différents mécanismes du jeu.

1 Fonctionnalités attendues

L'application devra permettre :

- De jouer une partie dans une édition/variante choisie, en proposant au moins la variante de base et la variante tactique.
- De paramétrer les joueurs (nom, type de joueur humain ou IA, type d'IA, rôle).
- De vérifier les revendications des joueurs et de déterminer le gagnant.

2 Éléments d'interface

L'application doit inclure :

- Un moyen de choisir l'édition/variante du jeu.
- Une interface permettant de définir les paramètres des joueurs.
- Au moins un type d'IA, même basique (décisions aléatoires).

Remarque : L'interface peut être réalisée en mode console, avec des visuels textuels suffisants pour comprendre l'état du jeu.

3 Architecture du projet

L'architecture doit être modulaire et extensible, permettant d'ajouter facilement :

- De nouvelles IA pour les joueurs.
- De nouvelles éditions ou variantes du jeu.
- Des fonctionnalités supplémentaires sans impact majeur sur le code existant.

La conception devra démontrer l'extensibilité de l'application en expliquant comment intégrer ces éléments sans modification importante.

4 Conception orientée objet

Le projet doit utiliser les concepts fondamentaux de la programmation orientée objet :

- **Encapsulation** : chaque classe doit avoir un rôle bien défini.
- **Héritage** : exploiter les relations entre les classes pour la réutilisation de code.
- **Polymorphisme** : permettre aux différentes variantes d'utiliser des comportements spécifiques.

5 Organisation des classes

Les classes seront organisées en plusieurs packages :

- `com.schottenTotten.model` : contient les classes de base du modèle de jeu (`Carte`, `Joueur`, `Borne`, etc.).
- `com.schottenTotten.controller` : contient la logique du jeu et les règles (`Jeu`, gestion des tours, etc.).
- `com.schottenTotten.view` : gère l'affichage et les interactions avec l'utilisateur (interface console).
- `com.schottenTotten.ai` : implémente les stratégies d'intelligence artificielle pour les joueurs non humains.

6 Gestion des variantes

L'application devra être conçue de manière à supporter différentes variantes du jeu. Cela pourra être réalisé en utilisant un design pattern de type `Factory` pour créer les éditions du jeu, par exemple, une classe `JeuFactory` permettant d'instancier le jeu en fonction de la variante choisie.

7 Robustesse et gestion des erreurs

Il est important de prévoir la gestion des erreurs pour éviter les comportements indésirables :

- Validation des entrées utilisateur (par exemple, les indices de carte).
- Gestion des cas exceptionnels (par exemple, cartes insuffisantes dans le deck).

8 Tests unitaires

Les tests unitaires devront être réalisés à l'aide de JUnit pour valider les fonctionnalités du jeu :

- Tester les méthodes des classes du modèle (`Carte`, `Joueur`, etc.).
- Vérifier la gestion des revendications de bornes.
- Valider les conditions de victoire.

9 Évaluation

Ce projet est à réaliser par groupe de 2 (3 si besoin) étudiants. Vous serez évalués sur :

- Conception des classes et architecture logicielle.
- Modularité et extensibilité du code.
- Utilisation des concepts de la programmation orientée objet.
- Robustesse des fonctionnalités implémentées.
- Gestion des erreurs et des cas d'exception.

10 Conseils pour le développement

- Utiliser un logiciel de gestion de versions (comme Git) pour faciliter le travail collaboratif.
- Respecter les principes SOLID pour garantir une conception propre et extensible.
- Documenter le code source pour en faciliter la maintenance.
- Travailler en équipe en suivant les bonnes pratiques du développement agile.

11 Livrables

Les livrables attendus sont :

- Le code source complet de l'application, organisé selon les packages décrits.
- Un rapport décrivant la conception, les choix d'architecture, et les tests effectués.