Dauphine | PSL
UNIVERSITÉ PARIS

Rapport sur le Projet dont vous êtes le Héros

Binôme: Hiba KEZIBRI, Bilal EL HIRCHI

— Une présentation de la structure du code:

Le projet comporte deux packages.

1. univers
2. representation
3. play: ce package contient la classe saveGame pour sauvegarder et la classe Main pour exécuter le code.
4. tests: ce package contient les classes de tests JUnit.

1. univers: ce package contient les classes représentant l'univers dans lequel va évoluer le.la joueur.euse.

- **PersonnageDeBase (classe abstraite) :** Représente le personnage de base avec des attributs communs tels que le nom, le niveau et les points de vie. Contient des méthodes génériques pour afficher les caractéristiques du personnage.
- **Aventurier, Chevalier, Alchimiste (classes) :** Représentent différents types de personnages du jeu, étendant la classe PersonnageDeBase et implémentant l'interface Combattant. Chacun a ses propres spécificités.
- **Player (classe) :** Représente le joueur avec la possibilité de choisir un personnage parmi Chevalier, Aventurier, et Alchimiste. Implémente les méthodes spécifiques au joueur telles que choisir une arme.
- **Combattant (interface) :** Définit les méthodes communes pour les personnages capables de combattre, telles que subir des dégâts, attaquer, défendre, et vérifier si le personnage est vivant.
- **Ennemi (classe) :** Représente un ennemi du jeu, implémente l'interface Combattant. Possède des attributs tels que la force, la défense, et les points de vie. Les méthodes permettent d'attaquer, subir des dégâts, défendre, etc.
- **Arme (énumération) :** Définit différents types d'armes avec des noms et des forces associées.
- **PersonnageCollection (classe) :** Gère la collection de joueurs.

2. representation: ce package contient les classes représentant le graphe des états possibles de notre jeu.

- **Node(classe abstraite), InnerNode, DecisionNode, TerminalNode (classes) :** Implémentation d'un arbre de décision pour gérer le scénario du jeu. Chaque nœud représente une étape de l'histoire.
- **Event (interface) :** Cette interface définit le contrat pour les événements spéciaux dans le jeu.

- **NodeDecorator (classe abstraite) :** La classe NodeDecorator est une classe abstraite servant de base pour les décorateurs de nœuds de l'arbre de décision.
- **ImageNode (classe) :** La classe ImageNode étend NodeDecorator et représente un nœud de l'arbre de décision associé à l'affichage d'une image spécifique dans le jeu.
- **SoundNode (classe) :** La classe SoundNode étend NodeDecorator et est utilisée pour déclencher des effets sonores dans le jeu.

— Extensions Éventuelles

1. **Système de Combat :** Ajout d'un système de combat avec des ennemis où le joueur peut attaquer, se défendre, et subir des dégâts.
2. **Armes (enum):** Intégration d'une énumération d'armes pour améliorer les capacités du joueur.

— Une discussion sur ce qui a été le plus dur à implémenter pour vous et sur le niveau de difficulté du projet.

L'interface graphique est une partie qui a été difficile à implémenter. Initialement, nous avons tenté d'utiliser une bibliothèque graphique existante pour simplifier ce processus. Cependant, nous avons rapidement constaté que cette approche prendra plus du temps pour avoir une interface graphique digne.

Malgré l'ajout de la bibliothèque pour JUnit, une erreur persiste. Nous avons essayé de suivre la documentation et des tutoriels mais nous n'avons pas pu détecter le problème.

— Lien Github
https://github.com/HKezibri/Game

.