Projet Splendor – INFO1

Sujet : Réaliser le jeu Splendor en Java

Le but de ce projet est de réaliser une version PC offline d'un jeu de société : **Splendor**. Il s'agit d'un jeu de stratégie où plusieurs joueurs s'affrontent pour asseoir leur contrôle sur le commerce des pierres précieuses.

Le jeu

La règle détaillée du jeu Splendor est disponible **en ligne**. Nous vous invitons à la lire avec attention, puisqu'il s'agit du jeu que vous allez devoir programmer.

Conseils

- ▷ Lisez l'ensemble de l'énoncé avant même de commencer quoi que ce soit !
- ▷ Le but de ce projet est de nous montrer que vous savez programmer "objet". Vous serez donc pénalisés si vous n'exploitez pas suffisamment les notions vues en cours.
- ▷ Lisez bien les règles du jeu et prenez bien le temps de réfléchir à la meilleure façon de mettre en place tel ou tel mécanisme du jeu avant de commencer à coder. Si vous commencez à coder avant d'avoir suffisamment réfléchi, vous serez, en pratique, obligés de revenir en arrière, ce qui vous demandera beaucoup plus de temps et d'efforts.
- ▷ Le sujet est évolutif : respectez bien les phases de réalisation, mais gardez à l'esprit ce que vous devrez faire dans les phases suivantes lorsque vous faîtes des choix d'implantation !

Le programme à réaliser

Vous allez réaliser votre jeu en 4 phases. Chaque phase devra être terminée et fonctionnelle avant de passer à la phase suivante.

Phase 1 : La base

Dans un premier temps, vous devez réaliser une version simplifiée du jeu, avec uniquement un mode de jeu permettant à deux joueurs humains de jouer l'un contre l'autre, avec un affichage **en ligne de commande**. Dans cette version simplifiée,

- ▷ on a supprimé les nobles : les cartes sont donc le seul moyen d'acquérir des points de prestige ;
- ▷ on effectue l'affichage en ligne de commande ;
- ▷ on a supprimé la possibilité de réserver une carte, et donc les jetons joker or ;
- ▷ il y a un seul niveau de cartes : chaque carte coûte trois jetons de sa couleur et rapporte un point de prestige, et on a huit cartes de chaque couleur.

Phase 2 : Le jeu complet

Une fois la phase 1 terminée, vous devez réaliser le jeu complet, permettant de faire jouer de deux à quatre joueurs humains, et toujours un affichage **en ligne de commande** :

- ▷ la liste complète des $40 + 30 + 20$ cartes du jeu est disponible **ici** ;
- ▷ la liste complète des 10 nobles présents dans le jeu est disponible **ici**.

Vous devrez donc trouver une manière adaptée de laisser à l'utilisateur le choix entre la version de base du jeu et la version complète.

Phase 3 : Affichage graphique

Une fois la phase 2 terminée, il vous est demandé de mettre en place une interface graphique simple. Vous devrez trouver une manière adaptée de laisser à l'utilisateur le choix entre l'interface graphique et l'interface en ligne de commande.
Afin de réaliser cette interface graphique, vous devrez utiliser la bibliothèque d'interface graphique `zen` fournie avec ce sujet (fichier `zen5.jar`).
Pour ajouter un jar à un projet sous Eclipse, il faut :

- ▷ Rajouter un dossier `lib` dans le répertoire du projet et y placer le fichier `.jar`.
- ▷ Dans Eclipse, faire un clic droit sur le fichier `.jar` et choisir `Build Path > Add to Build Path`.

Phase 4 : Améliorations

Une fois la phase 3 terminée, deux améliorations vous seront demandées :

- ▷ ajouter la présence de joueurs simulés par l'ordinateur : nous laissons à chaque groupe le soin de mettre au point des stratégies pour les joueurs simulés ;
- ▷ proposer une extension des règles du jeu ; vous devrez trouver une manière adaptée de laisser à l'utilisateur le choix entre la version basique du jeu (issue de la phase 1), le jeu complet (issu des phases 2 et 3) et l'option étendue.

Toute autre amélioration sera considérée négativement tant que les améliorations ci-dessus n'ont pas toutes été mises en place.

Critères de notation

- ▷ Rémi Forax Junior, 9 ans, doit pouvoir jouer à votre jeu ;
- ▷ la propreté et la lisibilité du code auront un poids très important dans la note ;
- ▷ l'architecture que vous aurez définie (interfaces, classes, etc) devra être donnée dans document PDF et aura également un poids très important dans la note ; ainsi, votre code devra être modulable, de manière à ce qu'ajouter d'autres extensions (par exemple davantage de cartes, un niveau de plus, ...) soit aussi facile que possible ;
- ▷ votre code ne devra pas contenir de méthodes de plus de 20 lignes ;
- ▷ pas de duplication de code, et respect des principes de programmation objet ;
- ▷ pas de variable globale;
- ▷ pas de code inutile ;
- ▷ présence des différents rapports et, par conséquent, orthographe correcte !
- ▷ prise en considération des remarques faites lors de la soutenance β pour le rendu final.

Règles à respecter impérativement – Mort subite

Voici une liste de règles qu'il vous faudra respecter impérativement. Si vous ne respectez pas ne serait-ce qu'une seule de ces règles, la notre de votre projet sera 0, et celui-ci ne sera pas évalué.

- ▷ Le projet ne **devra pas** utiliser ou inclure de libraire externe autre que celles indiquées dans le sujet.
- ▷ Le projet ne **devra pas** contenir de code copié-collé du net. La présence d'un tel code sera interprétée comme une tentative de tricherie.
- ▷ Le projet ne **devra pas** utiliser de classes du package `java.io` autres que les classes `InputStream`/`OutputStream`, `BufferedReader`/`BufferedWriter` et l'exception `IOException`. En particulier, il ne **devra surtout pas** utiliser `java.io.File`.
- ▷ Le projet ne **devra pas** contenir de champ avec une visibilité autre que `private`, et toute méthode de visibilité `public` devra commencer par vérifier que ses arguments sont valides.

Références

1. **Ant Manual** pour la construction du fichier `build.xml`
2. **How to create an executable jar?**
3. **JavaDoc**
4. **Les entrées/sorties sur fichier**
5. La bibliothèque graphique **Zen 5** et sa **documentation**
6. Un **exemple de code** utilisant le modèle de développement Modèle-Vue-Contrôleur, mais qui ne respecte pas plusieurs des règles de mort subite (et vaut donc 0) : javadoc manquante pour quelques classes ou méthodes publiques, absence de contrôle des arguments dans certaines méthodes publiques, ...