Pipers Chris G39864 Aout 2017

Dévelloppement DEVG4a-ATL

Projet Poker

Année 2016-2017

ESI – Haute Ecole de Bruxelles

Directeur – Michel Willemse

Sommaire

Modification du projet 4
MVC 5
Modele : 5
View : 5
Controller : 6
Base de données 7
Contenu de la base de données : 7
Modèle de la base de données : 7

Modification du projet

J'ai du implémenter la fonctionnalité bounty.

Une partie de poker avec Bounty est une partie ou, en plus de l'apport du pot par le joueur celui-ci rajoute une somme d'argent en plus (ici dans le jeux le bounty à une certaines valeur dans game et le joueur à un facteur multiplicatif comme attribut appeler Bounty, celui-ci est un reel car si il y as trois finaliste en all in et que deux sont gagnant, les deux gagnant recupere la moitier du Bounty de l'éliminer). Le facteur multiplicatif permet de symplifier l'affichage. Le joueur en fin de partie ou si il est éliminer il gagne le montant (player bounty * valeur bounty).

Quand le joueur se fait éliminer il « donne » son Bounty au(x) gagnant(s) mais si il en à gagner préalablement , ceux la sont d'office garder, il ne perd que son propre Bounty du départ tout ceux gagner le sont définitivement.

Pour implement cela :

j ai ajouté à la classe Game un attribut entier (qui correspond à un facteur multiplicatif) Bounty qui est comme un « pot » contenant chaque fois le bounty si un joueur est all in ;

J'ai egalement ajouter un entier qui contient la valeur du bounty ;

j'ai ajouter un attribut double a la classe player pour stocker le facteur multiplicatif de son Bounty ;

j'ai ajouter à la classe Match l'attribut bounty qui est un entier , si un joueur fait all in cet attribut reçois +1.

La nuance entre le double pour le player et l'entier pour match réside dans le faite que le joueur all in ne met que sont bounty en jeu qui as et aura toujours la valeur 1, donc en jeu il y aura que des valeurs entières. Le joueur quant à lui à la valeur en double car si deux gangant se partage un bounty de valeur 3 chacun aura respectivement 1.5.

C est l'attibut bounty de match qui est modifier et qui est remonter avec un getter dans game qui lui peut le retourner avec également un getter.

MVC

Modele :

Le modele fournit à subit de légére modification noté dans le read me, à cela j ai ajouter le package Observer contenant les classes Observable et Observer au package model.

La classe Observable permet d'ajouter, de supprimer et de notifier les observateur(observer) inscrit dans une liste lors d'un changement du modele.

La classe Game l'implemente Observable ce qui lui permet de notifié chaque observateur chaque fois que le modele subit un changement, ca lui permet aussi d'ajouter ou de supprimer un observateur. La classe game contient une liste d'observer.

La classe Observer permet à chaque classe qui l'implemente de s'enregistrer en tant qu'observateur à un Observable. Elle contient la méthode update ,qui permet lorsque les observateur sont notifié par la methode qui notifie les observateur de la classe implementant Observable, qui contient les méthode metant a jour l'observeur.

Les interfaces des composant extends Observer ce qui permet a chaque composant hbox d implementer la méthode update et de l'adapter . Les composants TableVieuw, PlayerComponent, FlopComponent, ChoiceBoxPlayer et le controlleur du fichier FXML implement Observer .

View :

Pour la vue j'ai créer un fichier FXML ou j'ai ajouter une Hbox, qui sera le contenat de tout les autres composants, le controller FXMLViewController qui permet de faire le lien entre le vue et le modèle et la classe main qui permet de lancer le jeu.

Pour acceder a cette Hbox j'utilise le controller, FXMLViwController, avec @FXML.

Mes différents composants constituant la vue sont :

- cardComponent qui extends Parent et qui permet de construire un composant carte avec une valeur et une couleur.

- handPokerPlayer qui extends Hbox et qui permet de construire une main pour un joueur de poker constituer de deux cartes et donc de deux cardComponent.

- playerComponent qui extends Hbox et qui est constituer d'une handPokerPlayer, d une Vbox contenant les attribut d'un player c'est-

à-dire , son nom, la valeur de son pot et le facteur multiplicatif de son bounty.

- deckComponent qui extends Hbox et qui permet de construire un paque de carte avec en vue aérienne.

- flopComponent qui extends Hbox et qui permet de construire le flop du jeu mais aussi d afficher la valeur du pot en jeu et le nombre de bounty à prendre . Il es également composé d'un deckComponent qui est purement un détail graphique .

- choiceBoxPlayer qui extends GridPane qui reprend tout les boutons qui corresponde au choix possible du joueur, c'est-à-dire : call, all-in, check, raise (qui possede un textfield ou entrer la somme a raise) et fold. Chaque bouton à son handler qui permet de gerer le faite de clicquer dessus et d'effectuer l'action si cela est possible selon l'état du jeu.

- tableComponent qui extends GridPane qui reprend elle tout les composant et les positionnes selon , pour les joueurs, à la position définie par la classe PositionPlayer et pout les autres un position définie dans la classe elle-même. TableComponent contient entre 4 et 10 playerComponent, 1 FlopComponent et d'une ChoiceBoxPlayer.

- Image contient le package cards contenant les images des 52 cartes pour le jeu de poker et aussi l'image de fond de la table et la « photo » du joueur.

Controller :

Le FXMLViewController permet de faire le lien entre le modele et la vue. Il construit la table et l'ajoute au fichier FXMLView, permet de definir le vainqueur.

Le controller est sencer controller les actions du joueur, ici les action du joueur son controller en majeur partie par le composant ChoiceBoxPlayer qui via son affichage qui s'adapte au choix permis par le jeu permet d'évité de devoir controller chaque action du joueur.

Base de données

Contenu de la base de données :

Dans la base de données, j'enregistre chaque joueur, en plus des attributs des joueurs du modele il faut rajouter un ID au player qui lui sera unique. Ce qui permettra de tous les differenciers.

Modèle de la base de données :

Ma gestion de base de données est composées de 6 classes :

- BaseDeDonnées
  Cette classe permet d'ajouter des joueurs à la base de données.
- BaseDeDonnéesException
  Cette classe permet de gerer ses propre exception avec un message plus claire.
- DriverManagerP
  Celle classe permet si il n'y pas de connection possible d'avoir une connection par défaut.
- ManagementBaseDeDonnées
  Cette classe permet de gérer les composants de la base de données, ici étant des joueurs.
  Elle permet de recuperer une liste de joueurs, recuperer certaines valeurs et/ou de les mettre a jour.
- PlayerBDD
  Cette classe est le type de joueur que contient la base de données en plus des attributs du joueur de poker, ce type contient un identifiant unique nommé IDPlayer. Ceci permettant d'identifier chaque joueur.
- mainBaseDeDonnees
  Cette classe permet de se connecter à la base de données et d'ajouter eventuellement des joueurs à la base de données.