

P1 : UNO (Partie 2)

Voici la suite du projet ...

1 ÉNONCÉ (SUITE)

Vous avez, à ce stade écrit quelques composants que vous allez maintenant mettre ensemble afin d'écrire un GUI `GameView` vous permettant de jouer à UNO.

Voici un exemple d'interface graphique. On y distingue la défausse (une `CardView`), la pioche (une `CardView`), la liste des joueurs, une zone de notifications (une manière d'informer de la couleur choisie lors de la pose d'une carte « wild » et « wild_draw4 »), la main du joueur, une « combo box » permettant de choisir la couleur que l'on associe à une carte « wild » et « wild_draw4 » et les boutons permettant de piocher ou de jouer une carte.

La dernière combo box permet de choisir de jouer directement deux cartes lorsqu'elles sont identiques (elles sont alors empilées).



Votre classe `GameView` devra implémenter `UnoObserver` afin d'exécuter la méthode `update`. Votre `observable` sera la classe `Game`¹.

Pour commencer une partie, il suffit de créer des joueurs (`Player`), un code à l'allure suivante devrait faire l'affaire

¹ Le second paramètre de type `Object` sera également un objet de type `Game`.

```
Deck deck = new Deck();
DiscardPile discardPile = new DiscardPile();
PlayerWheel playerWheel = new PlayerWheel(discardPile);
List<Player> players = new ArrayList<Player>();
player = new Player("Juste");
players.add(player);
playerWheel.setPlayers(players);
game = new Game(playerWheel, deck, 5);
game.addObserver(this);
```

Le constructeur de la classe Game configurera chacun des joueurs de PlayerWheel et complètera cette liste de joueurs avec des joueurs AI² afin de « faire le nombre ».

to be continued ...

² L'intelligence artificielle se résume à choisir la première carte (celle de plus petit indice) qui convient.