

P1 : UNO (Partie 1)

Ce projet a pour objectif de vous faire mettre en œuvre les différentes nouvelles notions (JavaBean, Observateur-Observé, MVC, ...) abordées dans ce cours. Ce travail, ainsi qu'un autre que nous aborderons ultérieurement, sera à présenter pour votre examen.

La lecture du « Guide du graphiste, Graphics2D » peut-être bénéfique pour les détails ... graphiques.

1 ÉNONCÉ

Nous allons écrire une application permettant de jouer au jeu de UNO.

Les règles du jeu sont disponibles

- ✓ [Wikipédia \(en\)](#)
- ✓ [Wikipedia \(fr\)](#)
- ✓ [Le site de Mattel](#)

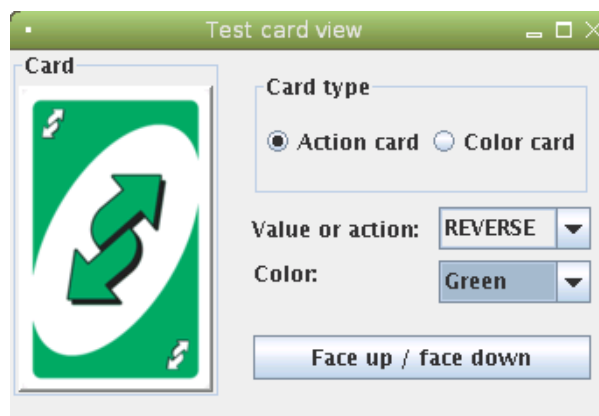


Votre travail consiste en l'écriture d'un GUI permettant de jouer au jeu. Nous vous fournissons le modèle sous la forme d'une bibliothèque (*library*) à inclure à votre projet ainsi que la javadoc du modèle. Vos vues devront s'inscrire comme écouteur du modèle.

2 BEAN, CARDVIEW

Écrivez un composant, `CardView`, permettant de représenter une carte du jeu. Les cartes ont la capacité d'être « cliquées » afin d'être sélectionnées/désélectionnées. Ce composant a comme propriétés: `card`, `selected`, `clickable`, et `faceUp` représentant respectivement la carte qu'il représente, la propriété sélectionnée ou pas, le fait qu'elle puisse être cliquée et une valeur booléenne précisant si on montre la carte ou pas¹.

Écrivez également une classe qui aura l'allure suivante, permettant de tester ce composant :



¹ On ne la montrera pas dans le cas de la pioche.

3 BEAN, HANDVIEW

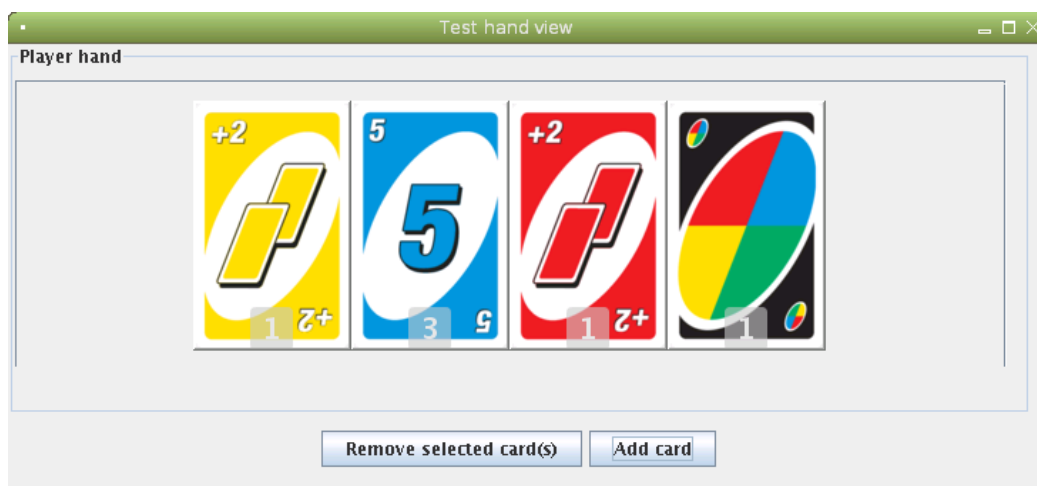
Les cartes d'un joueur sont rassemblées dans une main. Une main peut contenir de 1 à n cartes². Décidons d'afficher 6 cartes (par défaut, s'il y en a moins, le composant ne « rétrécira » pas) et d'ajouter une barre de défilement (*scrollbar*) lorsque ce nombre augmente³.

Écrivez un composant `HandView` représentant cette main de joueur. Ce composant rassemblera directement les cartes identiques (nous imposons donc de jouer toutes les cartes identiques). Pour représenter les cartes identiques vous écrirez un composant, héritant de `CardView` et affichant un nombre sur la carte (par exemple). Ce composant, `HandCardView`, aura une propriété supplémentaire par rapport à `CardView`, `nCards` représentant le nombre de cartes identiques.

Le composant `HandView`,

- x aura une propriété `selectedHandCardView` (et le *getter/setter* qui l'accompagne),
- x proposera les méthodes `add(Card)`, `add(List<Card>)` et `removeSelectedHandCardView()`

Écrivez également une classe permettant de tester ce composant qui aura l'allure suivante:



L'appui sur le bouton « *Add card* » ouvrira une **fenêtre modale** permettant de choisir une carte. Pour ce faire, vous réutiliserez votre frame de test du composant `CardView`⁴.

to be continued ...

² n est plus petit que 108 !

³ Vous devrez, pour ce faire, utiliser les classes `Container`, `JScrollBar`, ... n'hésitez pas à demander des informations complémentaires.

⁴ Il vous suffira de recopier la classe sous un autre nom (`DialogAddCard` par exemple) et de modifier son héritage. Cette classe doit maintenant hériter de `JDialog` plutôt que de `JFrame`.