IN201 : 12 – GUI 2

Solution

Author : Christophe Garion <garion@isae.fr>
Public : SUPAERO 2A
Date :

Résumé

Le but de ce TP est de construire une interface graphique permettant de jouer à un jeu de morpion dont le modèle (au sens du patron de conception MVC) est fourni.

1 Contenu

Ce document est un corrigé succint du TP portant sur le jeu de morpion. Tous les fichiers source sont disponibles sur le site `http://www.tofgarion.net/lectures/IN201`.

2 La classe `VueMorpion`

Pas de problèmes particuliers pour cette classe, il fallait faire attention lorsque l'on plaçait les instances de `JLabel` dans le `GridLayout` pour que les lignes et les colonnes correspondent bien (j'ai choisi dans mon corrigé de représenter les colonnes par la première « dimension » de la matrice). On peut remarquer également l'importance du test pour vérifier que les icônes s'affichent bien sur les bonnes cases en utilisant `setCase`.
J'utilise également la méthode statique `ClassLoader.getSystemResource` pour pouvoir facilement charger les fichiers images. Il suffit alors de les placer dans le `CLASSPATH` (directement ou via un JAR).

3 Le modèle du morpion

Le modèle du morpion n'est pas très bien conçu. Il aurait mieux valu, plutôt que d'utiliser des entiers pour représenter les états du morpion, utiliser une énumération ou le patron de conception État. Ces solutions auraient également permis de gérer plus facilement la vue.

4 Conception du contrôleur

J'avais choisi d'utiliser des classes internes qui permettait d'avoir accès directement à l'instance de `VueMorpion` et en particulier de son attribut `morpion` pour pouvoir modifier le modèle. On aurait pu se demander si l'utilisation d'une classe anonyme pour remplacer `ListenerCase` n'aurait pas été judicieuse. En effet, les instance de `ListenerCase` ne sont pas partagées. Malheureusement, `ListenerCase` utilise les valeurs des variables locales `i` et `j` dans la boucle d'inscription du *listener* pour initialiser les valeurs de ses attributs. Or, si une classe anonyme veut utiliser la valeur d'une variable locale, cette dernière doit être déclarée **`final`** (principalement pour des problèmes de *multithreading*). On ne pouvait donc pas utiliser une classe anonyme pour les *listeners* sur les cases.
On remarquera également qu'il aurait été beaucoup plus judicieux de spécialiser `JLabel` en une classe possédant deux attributs représentant les coordonnées du `JLabel` sur l'aire de jeu. Cela aurait grandement simplifié la gestion de la vue (pas de tableau à gérer par exemple).

5 Implantation

Pas de difficulté particulière ici. Vous trouverez sur le site les sources de toutes les classes. J'ai fait une petite classe de test avec SwingUnit, mais celle-ci n'est pas complète. Je teste simplement que lorsque l'on clique sur une ou plusieurs cases, la bonne image apparaît. Il aurait fallu tester également que le jeu se finit bien etc.