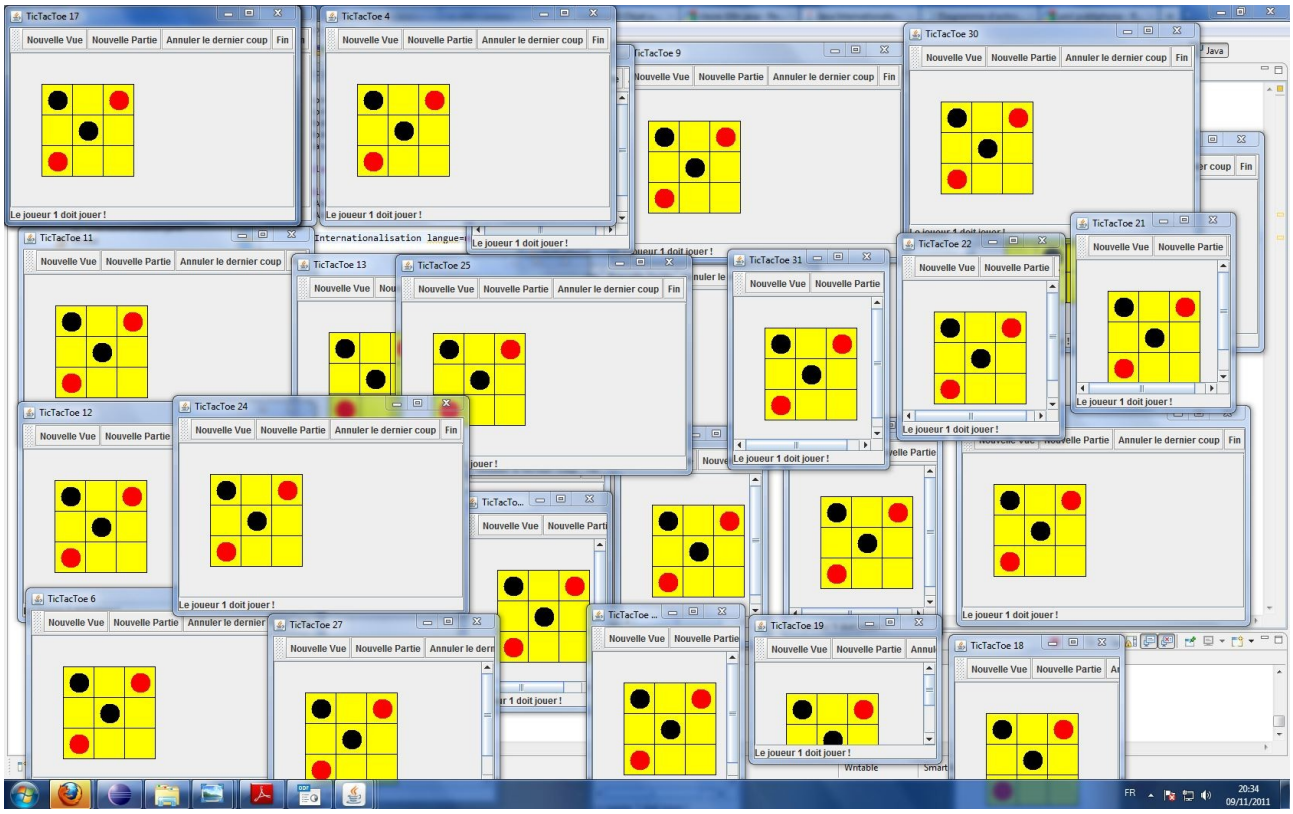
Cyril MONSIEUX
Master informatique

# Documentation tic-tac-toe



## I) Le pattern Modèle-Vue-Contrôleur (MVC)

Le but principal de cette méthode de conception est de séparer l'interface graphique (la vue) des données (le modèle). Cette mise en œuvre est plus longue, introduit plus de complexité mais permet finalement d'avoir une application plus facile à maintenir.

### I.1) Le modèle (package Model)

Le modèle sera chargé de stocker les données du jeu et d'avoir le statut de la partie.
Le modèle devra savoir :

– jouer un coup
– vérifier le plateau
– annuler un coup
– implémenter le pattern observé.

Afin de prévoir un changement éventuel de modèle, celui-ci est créé à partir d'un super-type de modèle (« AbstractTicTacToeModel »).
Les classes filles peuvent ainsi implémenter les méthodes de façon polymorphe.
Nous pourrions imaginer un modèle de données qui implémente un plateau de jeu avec 4 rangées.

### I.2) Le contrôleur (package Controler)

Le contrôleur sera chargé de faire le lien entre la vue et le modèle.
Afin de garder un maximum de souplesse, un contrôleur abstrait a été représenté, bien que dans notre cas il ne soit pas indispensable.

Le contrôleur devra savoir :

– agir lors d'un clic sur « Nouvelle vue », « Annuler le coup », « Fin », …
– accéder au modèle, afin de le prévenir des événements sur la vue.

Étant donné que le contrôleur doit accéder aux méthodes du modèle, ce dernier devra avoir une instance du modèle.

### I.3) La vue (package View)

La vue permet de représenter graphiquement l'application. Elle s'inscrit comme « observateur » sur le modèle et communique avec lui par le biais du contrôleur.
Il lui faudra donc une instance du contrôleur.

## II) Internationalisation

L'internationalisation d'un logiciel consiste à préparer son adaptation à des langues différentes. Afin d'aider à l'internationalisation d'un logiciel, Java propose un système basique de traduction de messages, et ce par le biais de deux composants : la Locale et le ResourceBundle.

En pratique, une Locale est une classe qui se base sur 2 codes, un code **langue** et un code **pays**, afin d'identifier un pays, une langue ou un dialecte.

```
// Locale française
Locale france = new Locale("fr", "FR");
```

Le deuxième composant i18n de Java est le ResourceBundle. Celui-ci est responsable de la récupération pour une locale Locale donnée.

Cependant, le ResourceBundle est une classe abstraite, c'est donc une implémentation concrète de RessourceBundle qui est utilisée, PropertyResourceBundle. Cette implémentation s'appuie sur un nom de fichier properties de base, et va rechercher pour une Locale donnée, si une traduction existe.
Et pour cela, elle va vérifier l'existence d'un fichier properties dont le nom est : <nom du fichier de base>_<le code langue>_<le code pays>

Et si ce n'est pas suffisant, elle va utiliser le fichier de base pour récupérer une traduction.

Exemple : MessagesBundle.properties

**Afin de modifier la langue de l'application, il faudra simplement modifier la variable « *langue* » du fichier « lang.java » (ligne 7).**