Guillaume GAUJAC
I.U.T de CAEN – D.U.T Informatique

MODULE : 2103
MINI-PROJET JAVA – Le « SHING-SHANG »

Image tirée de : https://i.ytimg.com/vi/lkZARyU8XuY/hqdefault.jpg

Image tirée de :
http://jeuxstrategieter.free.fr/jeu_shing_shang/couvercle.jpg

Introduction :

Durant cette première année en DUT Informatique nous avons réalisé un jeu en langage JAVA, dénommé Shing-Shang. L'intérêt de ce projet est de mettre en pratique et d'approfondir les connaissances que nous avons apprises lors de notre second semestre.

Fonctionnement du jeu :

Le Shing-Shang fait appel à un plateau de 10x10 cases où se trouvent des cases inaccessibles sur la première et dernière colonne sauf sur la quatrième et la cinquième ligne.

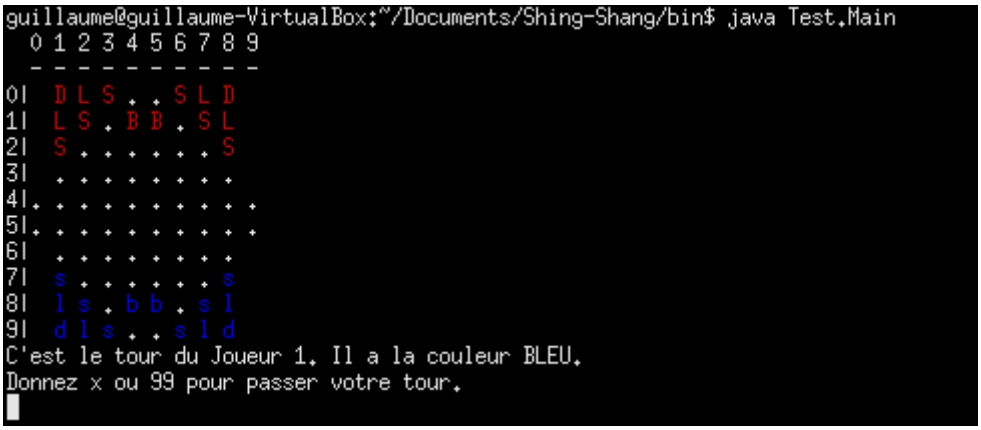
Sur ce plateau se trouvent quatre portails (ici représentés par des « B »/« b » pour BUT) se situant sur les lignes 1 et 8 et sur les colonnes 4 et 5.

Une partie se déroule à deux joueurs (le joueur rouge et le joueur bleu) ayant tous deux 12 pions : 2 dragons, 4 lions et 6 singes. Chaque type de pion doit respecter un déplacement qui lui est spécifique : le singe peut se déplacer d'une ou de deux cases dans toutes les directions, mais il ne peut pas arriver sur une case déjà occupée, par contre, il peut passer au-dessus des autres singes et les « manger » si ceux-ci appartiennent à l'équipe adverse. Le lion se déplace d'une case dans toutes les directions mais il ne peut pas arriver sur une case déjà occupée, sauf si c'est un autre lion ou un singe, auquel cas il peut le « manger » et avance alors dans la même direction d'une case supplémentaire. Le dragon ne se déplace que pour sauter sur un pion situé juste à côté qu'il peut alors « manger » si celui-ci est de l'équipe adverse ; il continue alors son déplacement jusqu'à la case immédiatement suivante dans la même direction.

Lorsqu'un pion saute sur un autre pion il gagne un tour jusqu'à ce qu'il ne puisse plus sauter. Lorsqu'on saute au-dessus d'un adversaire on doit rejouer mais on doit utiliser un autre pion. On effectue alors ce qu'on appelle un Shing-Shang.

La partie prend fin lorsqu'un des joueurs a mangé l'ensemble des pions adverses ou s'il réussit à se placer sur l'un des buts adverses.

J'ai respecté l'emplacement des différents pions selon le site : http://jeuxstrategieter.free.fr/Shing_shang_complet.php.



Disposition de l'échiquier

Mode d'emploi :

Pour compiler le programme, vous devez vous positionner dans le dossier « *bin* » est lancer :

« java Test.main »

À partir de là, le compilateur saura où trouver les différents fichiers objets pour permettre le bon fonctionnement du programme.

Une fois dans le programme, celui-ci vous demandera quelles cases initiales et finales vous souhaitez choisir.

Entrez donc les lignes et les colonnes qui vous intéressent. Si celles-ci sont fausses, le programme vous demandera d'autres coordonnées jusqu'à ce qu'elles soient valides.

Vous pouvez également entrer le chiffre : 99 (en x) dans le but de passer votre tour.

Si un joueur réussit à manger l'ensemble des pions, ou s'il réussit à positionner son pion sur un but adverse, la partie s’arrête et la console affiche le gagnant.

Les fonctionnalités implémentées :

Les fonctionnalités dont dispose l'utilisateur sont :

- Il peut saisir une case initiale et une case d'arrivée ;
- Il peut passer son tour (en tapant 99 lors du choix d'un x) ;
- Il peut redonner une case s'il s'est trompé.
- Il peut gagner une partie (celle-ci s’arrête et affiche son nom)
- Il ne peut pas accéder aux cases inaccessibles (celles inactives et en dehors du tableau)
- Il dispose d'une couleur pour se différencier.

Organisation du programme :

Mon programme s'organise de la manière suivante :

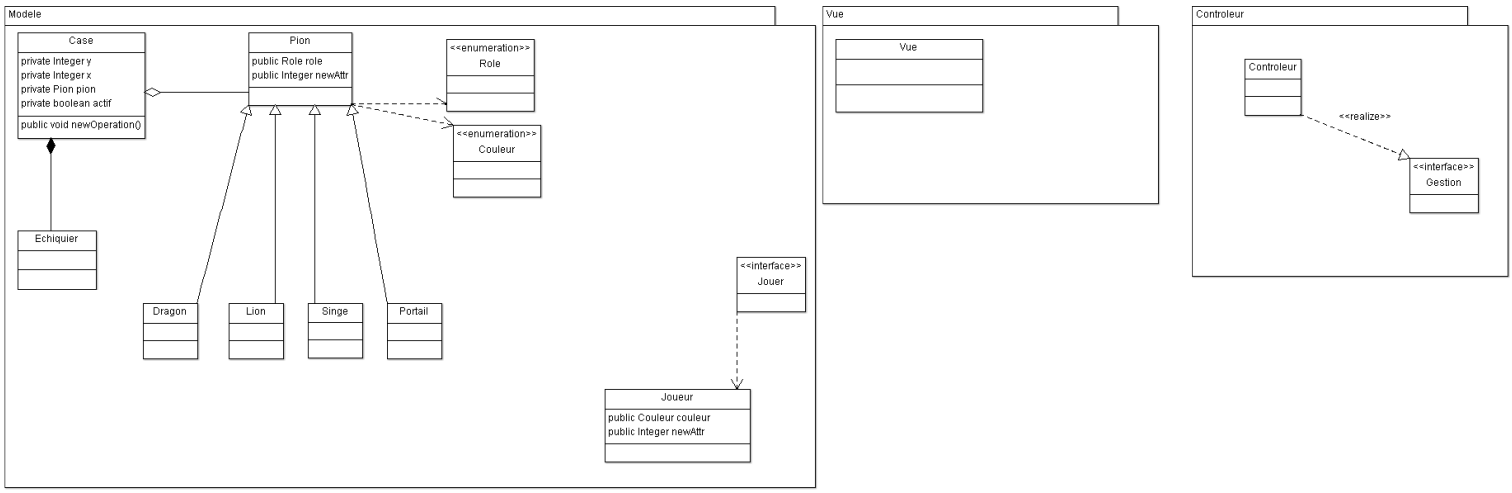
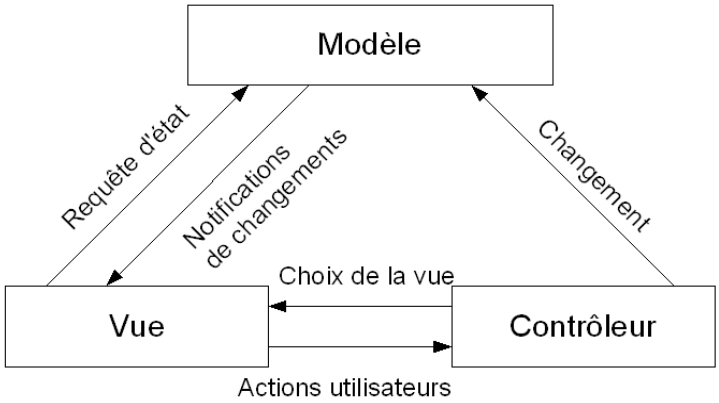


Diagramme de classes du projet : Shing-Shang

J'ai décidé d'organiser celui-ci de manière à ce qu'il respecte l'architecture MVC (Modèle-vue-contrôleur). Cette architecture permet une meilleure lisibilité du code en séparant les différentes taches présentes dans notre programme. Lorsqu'un utilisateur souhaite agir sur le plateau, celui-ci fait appel à un contrôleur qui va gérer l'ensemble des sollicitations transmises. Le contrôleur va ensuite pouvoir synchroniser le modèle, qui garde l'ensemble des informations liées au projet, et la vue, qui va permettre d'afficher nos informations.



Exemple de modèle MVC

Image tirée du site : http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/images/mvc.png

J'ai également fait le choix d'utiliser des énumérations (« enum ») pour lister l'ensemble des couleurs (Vide, rouge et bleue) et des rôles (Inaccessible, vide, singe, lion, dragon et but).

Je ne dispose pas de classe abstraite. J'ai décidé qu'une case devait obligatoirement se référer à un pion (même si celui-ci ne dispose d'aucune couleur ou d'aucun rôle) (par souci de facilité).

J'utilise une interface : « Gestion » qui me permet de définir l'ensemble des méthodes nécessaires pour gérer une partie (display(), estGagner(), deplacerContenusCase(), etc.). J'ai préféré utiliser une interface plutôt qu'une classe abstraite car celle-ci me parait plus lisible. De plus, il m'est possible d'implémenter plusieurs interfaces dans une même classe. En JAVA, nous sommes limités à une classe mère en ce qui concerne les classes abstraites. Je n'aurais donc pas pu hériter plusieurs classes abstraites.

Présentation des différentes normes ou conventions adoptées :

Mon programme respecte les conventions d'écriture propre au langage JAVA. L'indentation est appliquée par l'outil « save actions » présent dans les préférences de l'IDE Eclipse. La première lettre de mes classes et de mes interfaces sont toutes en majuscules. J'ai tenté d'éviter au maximum les acronymes. Mes méthodes disposent de nom explicite (elles reflètent une action) et leur première lettre est en minuscule. Mes variables disposent de nom court et descriptif. Les seules variables à une lettre sont mes compteurs internes et mes variables temporaires. Par souci de lisibilité, j'ai essayé de mettre l'ensemble de mes déclarations en début de bloc. En ce qui concerne les commentaires, j'ai décidé de signer l'intégralité de mes classes par mon prénom, mon nom ainsi que le numéro de version de la classe (visible dans la Javadoc).

Bilan qualitatif du travail :

La seule difficulté notable que j'ai pu rencontrer n'est autre que la gestion des exceptions cependant, après plusieurs recherches, j'ai su faire face à celles-ci.

Ce projet m'a permis de m'exercer et d'approfondir mes connaissances en JAVA.
Il m'a forcé à gérer mon temps, en m'inculquant une certaine méthode d'organisation.

Conclusion :

Ce projet fait donc office de suite logique aux différents exercices que nous avons pu rencontrer au cours de ce second semestre, à noter que ces exercices nous ont été d'une grande aide durant ces nombreuses semaines.