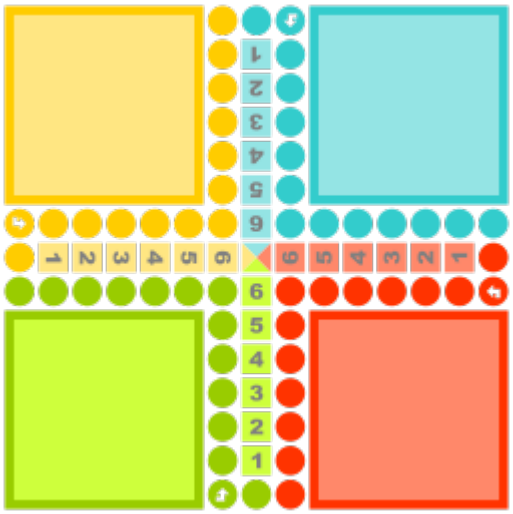
Rapport du projet algorithmique (M2103)



Plateau de jeu. (Source :https://fr.wikipedia.org/wiki/Jeu_des_petits_chevaux)

Rapport du projet algorithmique sur le jeu « Les Petits Chevaux » dans le contexte du module M2103. Réalisé en binôme et rendu le dimanche 19 mai 2019.

Projet réalisé par :

- MONTEIL Julien
- RICHARD Florian

Année 2018/2019

Préambule :

Ce projet a été réalisé en JAVA sous les IDE Eclipse et IntelliJ. ANSI, un module d'Eclipse a été utilisé pour faciliter la différenciation entre les joueurs et les pions. En effet, ce module permet l'interprétation de séquences d'échappement pour permettre à la console d'afficher des couleurs.(Lien d'installation d'ANSI :https://marketplace.eclipse.org/content/ansi-escape-console) De base, IntelliJ permet l'affichage des couleurs, sur Eclipse il est nécessaire d'ajouter ce module.

Diagrammes de classe :

Le diagramme de classe original et le final sont présents dans le dossier rendu avec le projet. (trop larges pour les intégrer au rapport) Le diagramme de classe final est un dérivé du diagramme de classe original que nous avons respecté le plus possible. Il a été généré grâce à un module de StarUml qui permet de créer des diagrammes depuis un projet JAVA. Le fichier du diagramme représente le paquetage domaine mais la totalité du projet est consultable grâce au fichier StarUml également fourni dans le dossier.

Au niveau des différences, une énumération EtatPion a été rajoutée. Elle possède 3 instances : ECURIE, PISTE et ECHELLE. Cette énumération nous a permis d'être plus précis quant à la position et au statut de chaque pion tout au long d'une partie. Ne pas l'ajouter nous aurait compliqué la tâche au niveau des conditions et des changements de types de case. (CaseDEcurie, CaseDEchelle et CaseDeChemin)

Dans l'énumération Couleur, un code couleur et son getter ont été rajoutés pour permettre l'affichage en couleur du plateau et du nom des joueurs.

Dans la classe Plateau, la méthode sortieEcurie a été rajoutée afin de nous permettre une séparation plus nette et une gestion plus organisée des différents mouvements liés aux pions. La sortie de l'écurie d'un pion nécessitait la gestion du nombre de pions encore à l'écurie, ainsi que sa sortie sur sa case dédiée selon son joueur propriétaire. Dans la classe Partie, une méthode vérificationTourComplet a été ajoutée.

Elle permet de vérifier si un pion a fait le tour du plateau et peut désormais monter à l'échelle. Ainsi, un attribut caseEntreeEchelle a été ajouté pour chaque joueur, accompagné de son setter et de son getter.

Les méthodes toString des différentes cases ont été redéfinies pour obtenir un affichage adapté et en couleur.La structure globale du diagramme de classes a été respectée, seuls quelques méthodes et attributs ont été rajoutés. En revanche, nous n'avons pas eu besoin d'implémenter les exceptions car nous avons programmé dans le but de les éviter dès le début. Par conséquent, aucune des exceptions spécifiées dans la diagramme original ne se produit. (sauf oubli de notre part)

Les fonctionnalités :

- Permettre aux 4 joueurs de choisir leur nom et leur couleur.
- Contrôles de saisie pour ne pas avoir de conflits de couleurs ou d'erreurs lors de la saisie.
- Initialiser un plateau de jeu avec les couleurs choisies par les joueurs pour leurs écuries et échelles respectives.
- Jouer un tour : choisir de sortir un pion de l'écurie si le lancé de dé est égal à 6 ou déplacer un pion déjà présent sur la piste.
- Affichage de l'état de chaque pion d'un joueur lors de son tour de jeu.
- Indiquer à chaque joueur si leurs pions sont déplaçables ou non selon leur situation.
- Manger un cheval si l'occasion se présente.
- Vérifier si un déplacement est valide et indiquer au joueur pourquoi le cas échéant.
- Vérifier la fin d'une partie et indiquer le gagnant.

L'organisation du travail et la répartition des tâches :

Concernant l'organisation du travail et la répartition des tâches, cela s'est fait assez facilement et naturellement car nous avons pris l'habitude de travailler en binôme.

Au début du projet, nous avons beaucoup travaillé ensemble afin d'établir toutes les classes, les paquetages et pour que nous soyons sur la même idée de conception du projet. Cela nous a permis de réfléchir de la même manière tout le long du travail. Au fil du temps, avec les contrôles et autres projets qui avaient lieu pendant la conception du projet, nous nous sommes répartis les tâches afin d'optimiser notre temps de travail. Cela nous a permis d'avancer sur le projet algorithmique et de nous concentrer en même temps sur les travaux et les contrôles.

Nous séparions le travail par rapport aux classes pour ne pas être bloqués pendant notre avancée. Nous estimions aussi quelles méthodes étaient primordiales par rapport à d'autres pour établir la base du projet. Ainsi, plus le projet avançait, plus les méthodes s'enrichissaient et plus le projet se précisait sans jamais être restreints par des méthodes non implémentées. Dès que l'un de nous deux terminait une partie de son travail, nous mettions en commun nos classes et mettions à jour le projet. De cette manière, il n'y a eu presque aucune attente d'un membre par rapport à l'autre, nous étions toujours en activité.

Bilan :

Ce projet fut très enrichissant au niveau programmation ; au niveau humain ce fut un renforcement de notre envie de travailler en groupe. L'idée de la programmation orientée objet est maintenant encore plus claire, le fait de travailler avec des méthodes spécifiques pour chaque objet est plus logique et plus compréhensible que pour la programmation séquentielle comme en C. De plus, travailler avec des ArrayLists pour presque chaque objet fut très intuitif, nous avons également fait en sorte d'utiliser le plus possible les méthodes liées aux ArrayLists pour rendre le projet modulable. (e.g. les boucles for allant de i à getJoueurs().size() et non pas de i à 4.) Le fait de réaliser le même projet une deuxième fois avec un langage de programmation différent nous a permis de voir les différences entre le JAVA et le C, et personnellement nous trouvons qu'en JAVA, ce fut beaucoup plus simple et agréable à réaliser. Encore une fois, nous fûmes surpris par la complexité des règles de ce jeu de plateau, le nombre de conditions dans nos méthodes justifie cette surprise.

Nous avons rencontré quelques difficultés lors de la durée du projet. Premièrement, l'affichage des couleurs fut assez complexe bien que cela puisse paraître futile, avoir des couleurs pour différencier les pions et avoir un affichage plus esthétique est très satisfaisant et nous a permis de progresser plus efficacement. Ensuite, toutes les conditions pour permettre ou non le déplacement d'un pion furent difficiles à mettre en place. De plus, le changement d'état de chaque pion, la gestion des ArrayLists selon la place des pions et ce tout le long de la partie dans chaque situation fut une tâche difficile. Enfin, nous avons rencontré quelques problèmes pour accéder aux objets que nous voulions selon les méthodes que nous utilisions avec l'accès ou non au joueur courant, aux échelles des joueurs, etc.

Comme dit précédemment, nous sommes d'accord par rapport au fait que la programmation en JAVA est bien plus agréable et naturelle qu'en C. La réalisation fut également plus facile et plus rapide car nous avions déjà connaissance des règles et des fonctionnalités à mettre en place. Pour la plus grande partie, nous avons réussi à améliorer notre code (moins important, plus modulable, plus lisible) mais nous avons quand même rencontré des problèmes déjà présents en C, tels que l'affichage du plateau ou les conditions à mettre en place pour que le programme respecte les règles du jeu et soit cohérent.

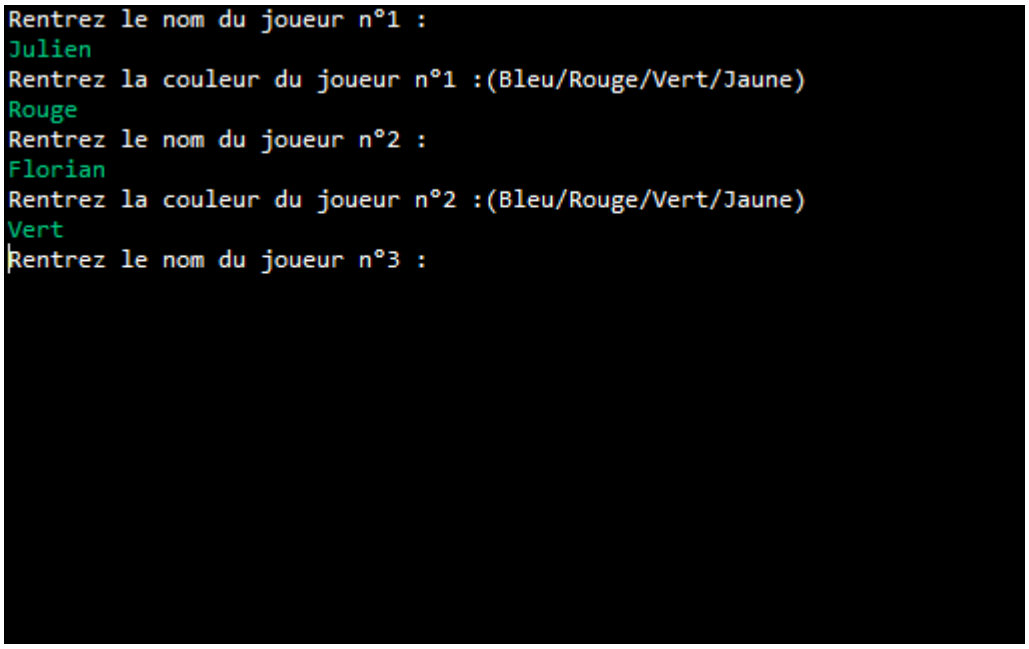
Finalement, la réalisation du projet fut plus naturelle et rapide qu'en C. Le fait d'avoir conservé le même binôme nous a beaucoup aidés.

Mode d’emploi :

Comme dit précédemment, afin que le programme soit le plus agréable à utiliser, il est conseillé de l’exécuter sous un IDE. (Eclipse avec ANSI ou IntelliJ)

Initialisation :

Chaque joueur choisit son nom et sa couleur.



Une fois les joueurs et couleurs initialisés, lors de chaque tour les noms des joueurs seront affichés dans la couleur qu’ils ont choisi. Leurs pions le seront aussi, tout comme leurs case d’échelles et celle d’écurie.

Premier tour :

Les tours de chaque joueur passent jusqu’à ce qu’un jour effectue un 6 au lancé de dé. Il peut choisir de sortir n’importe lequel de ses pions. (cf page suivante)

```
C'est au joueur Florian de jouer !
Lancé de dé. Vous avez fait : 3
Cheval n°1 est en ecurie.
Cheval n°2 est en ecurie.
Cheval n°3 est en ecurie.
Cheval n°4 est en ecurie.
Vous n'avez aucun cheval à déplacer, vous passez votre tour.
            0 0 0
            0 1 0
            0 2 0
      4     0 3 0     4
            0 4 0
            0 5 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 0
0 1 2 3 4 5 6   6 5 4 3 2 1 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 0
            0 5 0
            0 4 0
      4     0 3 0     4
            0 2 0
            0 1 0
            0 0 0
C'est au joueur Didier de jouer !
Lancé de dé. Vous avez fait : 6
Cheval n°1 est en ecurie.
Cheval n°2 est en ecurie.
Cheval n°3 est en ecurie.
Cheval n°4 est en ecurie.
Le cheval n°1 est à l'écurie, souhaitez-vous le sortir ? (o/n)
n
Le cheval n°2 est à l'écurie, souhaitez-vous le sortir ? (o/n)
o
Le pion est deplacé.
            0 0 0
            0 1 0
            0 2 0
      4     0 3 0     4
            0 4 0
            0 5 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 0
0 1 2 3 4 5 6   6 5 4 3 2 1 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 1
            0 5 0
            0 4 0
      4     0 3 0     3
            0 2 0
            0 1 0
            0 0 0
C'est au joueur Didier de jouer !
Lancé de dé. Vous avez fait : 6
Cheval n°1 est en ecurie.
Cheval n°2 est en piste.
Cheval n°3 est en ecurie.
Cheval n°4 est en ecurie.
Le cheval n°1 est à l'écurie, souhaitez-vous le sortir ? (o/n)
```

On voit que le joueur peut choisir de sortir son 2ème pion d'abord plutôt que le premier. De plus, le joueur a réalisé un 6 il peut donc sortir un pion et jouer de nouveau. Par contre, il ne pourra pas jouer plus de 2 fois à la suite. (cf règles)

Déroulement de la partie :

Ensuite, la partie suit son cours, les joueurs déplacent leur pion et se rapprochent de leur échelle.

Ici, les joueurs rouge et vert on sorti un pion, le joueur bleu en a sorti deux et le joueur jaune en a sorti un mais il a été mangé. (cf capture d'écran suivante)

```
          0 0 0
          0 1 0
          0 2 0
     4    0 3 0    4
          0 4 0
          0 5 0
1 0 0 0 0 0 1 6 0 0 0 0 0 0 0
0 1 2 3 4 5 6   6 5 4 3 2 1 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 1
          0 5 0
          0 4 0
     3    0 3 0    2
          0 2 0
          0 1 0
          0 0 0
C'est au joueur Didier de jouer !
Lancé de dé. Vous avez fait : 6
Cheval n°1 est en piste.
Cheval n°2 est en piste.
Cheval n°3 est en ecurie.
Cheval n°4 est en ecurie.
Le cheval n°1 est déplaçable, souhaitez-vous le déplacer ? (o/n)
o
Le pion est deplacé.
Vous avez mangé le cheval ou les chevaux adverses sur la case n°50 car vous vous êtes arrêtés sur la même case.
          0 0 0
          0 1 0
          0 2 0
     4    0 3 0    4
          0 4 0
          0 5 0
0 0 0 0 0 0 1 6 0 0 0 0 0 0 0
0 1 2 3 4 5 6   6 5 4 3 2 1 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 1
          0 5 0
          0 4 0
     4    0 3 0    2
          0 2 0
          0 1 0
          0 0 0
```

Ici, nous pouvons voir que le joueur bleu (Didier) a mangé le pion du joueur jaune (Michel). Il était 6 cases derrière lui et a réalisé un 6, ainsi il était en mesure de manger le ou les pions sur cette case. Malheureusement pour Michel, son pion était sur cette case. Il est donc retourné à l'écurie. (Case écurie jaune qui passe de 3 à 4)

Lors du déroulement de la partie, il est également possible qu'un cheval barre la route d'un autre. Dans ce cas là, si le joueur ne fait pas attention, son tour est consumé mais le pion n'est pas déplacé.

Fin de partie :

```
          0 0 0
          0 1 0
          0 2 0
     4    0 1 0    4
          0 1 0
          0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 1 2 3 4 5 6   6 5 4 3 2 1 0
0 0 0 0 0 0 0 6 0 0 0 0 0 0 0
          0 5 0
          0 4 0
     4    0 3 0    4
          0 2 0
          0 1 0
          0 0 0
Partie terminée. Le joueur Florian a gagné !
```

Dans ce cas là, le joueur Florian a réussi a placé ses 4 pions sur les cases 3, 4, 5 et 6 de son échelle. Par conséquent il remporte la partie et le programme prend fin. (La situation est mise en place par nous-mêmes ce qui explique l'absence de pions adverses sur le terrain)

Conclusion :

Pour conclure, nous pouvons dire que le projet nous a beaucoup apportés en termes techniques et scientifiques. En effet, la programmation orientée objet nous paraît beaucoup plus claire désormais. Les cours et exercices ne sont pas suffisants pour s'en rendre compte mais ce projet nous a permis de bien mieux nous familiariser avec la programmation orientée objet et JAVA. Malheureusement, nous avions choisi de programmer de manière à éviter les exceptions, par conséquent nous n'avons pas eu l'occasion de les utiliser même si nous aurions pu en forcer l'utilisation, nous ne trouvions pas cela nécessaire. Nous avons aussi fait en sorte de mieux commenter notre code, nous avons compris que c'était important de mieux commenter et documenter son code pour le rendre modulable et réutilisable par une personne extérieure. Au niveau humain, nous avons constaté une progression importante. Les projets de groupe étant récurrents, nous avons pris l'habitude de plus communiquer, d'échanger par rapport à nos problèmes et mêmes nos réussites. Ainsi, le projet s'est très bien déroulé, bien mieux qu'en C.

Nous sommes d'accord pour dire que ce projet est important et enrichissant au niveau informatique et humain. Il nous apprend à nous familiariser avec les langages de programmation, à travailler ensemble et à se serrer les coudes pour trouver les solutions à nos problèmes.