Nom : Florian VICTORIA DIT THOME

Thomas SCHAAL

Branche: *ISI*

Modélisation UML:

La bataille norvégienne

Résumé :

UV : *LO02*
Semestre : *Automne 2014*

INTRODUCTION

Afin de mettre en pratique ce qui a été appris durant le cours de « Principe et pratique de la programmation objets » un projet est proposé. Ce projet consiste à étudier et développer un jeu de bataille norvégienne. Les technologies utilisées dans ce projet sont le langage JAVA et la modélisation UML, Github pour le partage de fichier ainsi que la gestion de version. Nous effectuons ce projet en binôme, formé par Florian VICTORIA DIT THOME et Thomas SCHAAL. Dans l'optique de mener à bien ce projet, il a d'abord fallu comprendre le fonctionnement du jeu de la bataille norvégienne. Ce point va être détaillé ci-dessous.

Une partie est jouable par 2 à 11 joueurs et commence par la distribution des cartes. Chaque joueur reçoit 9 cartes. Les trois premières sont posées devant lui en ligne face cachée. Les trois suivantes sont à poser sur les trois précédentes, mais cette fois-ci face visible. Les trois dernières sont à prendre en main à l'abri du regard des autres joueurs. Les cartes non distribuées vont servir de pioches pour le jeu, il faut donc les poser sur le tapis. Chaque joueur peut échanger les cartes qu'il a en main avec celle face visible qui sont sur le tapis. Cette règle est valable uniquement avant le début d'une partie.
Ensuite, le jeu commence par la personne qui est à gauche du joueur qui a distribué. Les joueurs posent une ou plusieurs carte(s) sur le tapis un après l'autre. Chaque joueur ne peut que poser une carte dont la valeur est supérieure à celle sur le tapis (la couleur ou le signe ne compte pas pour ce jeu). Le joueur peut poser plusieurs cartes de la même valeur en même temps. Quand le joueur a posé une ou plusieurs cartes, il doit piocher des cartes pour maintenir 3 cartes dans sa main. Quand un joueur ne peut plus poser de carte d'une valeur supérieure la dernière carte visible jouée sur le tapis, alors ce joueur ramasse le paquet (bêche) et le met dans sa main.

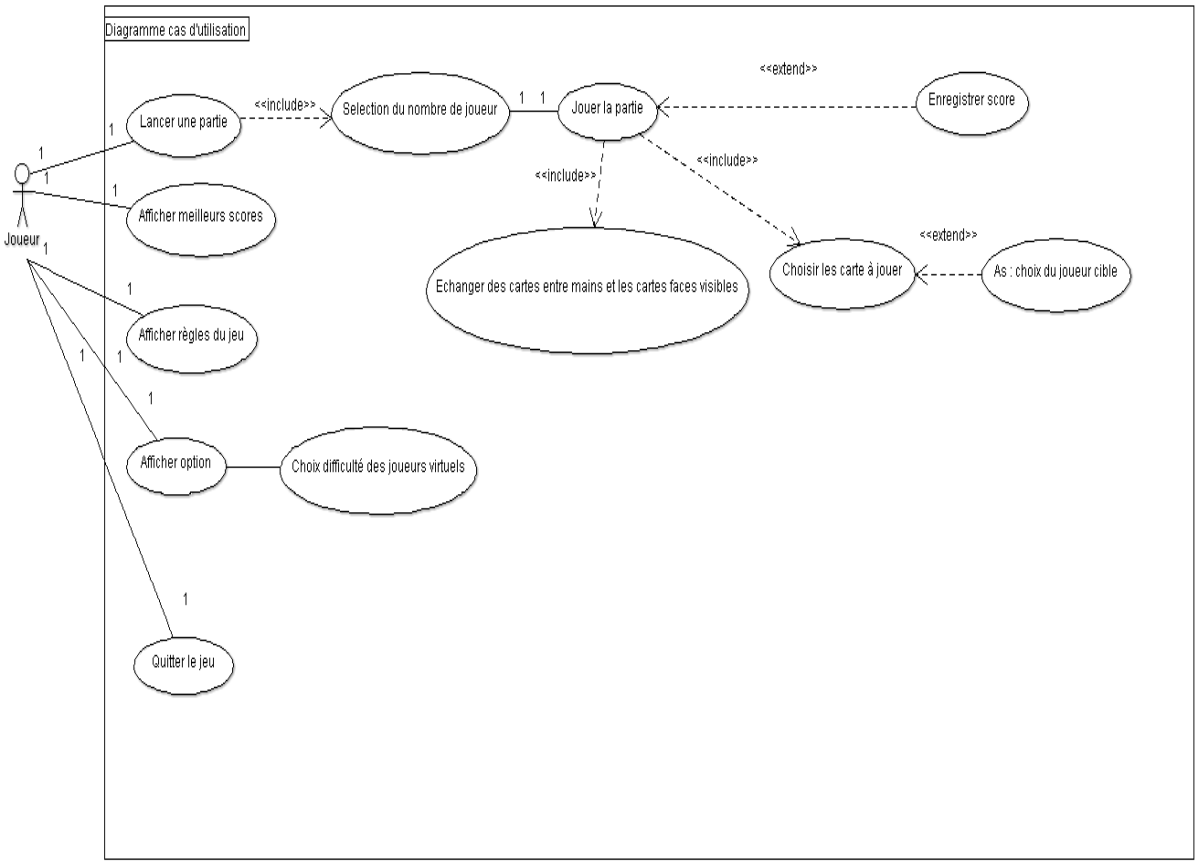
Toutes les cartes ne sont pas équivalentes. En effet, il existe des cartes spéciales, qui ont un effet bien précis, et il faut les utiliser de manière stratégique pour remporter la victoire et être « Danish ».

- Le 2 est une carte spéciale qui est supérieure aux autres, quand on l'a joué, le jeu repart à la valeur 2.
- Le 7 est une carte spéciale qui oblige le joueur suivant à jouer une carte d'une valeur inférieure ou égale à 7.
- Le 8 est une carte spéciale qui oblige le joueur suivant à passer son tour, le joueur suivant peut poser uniquement un 8, ce qui obligera le joueur suivant à passer son tour ou lui-même poser un autre 8. Et ainsi de suite.
- Le 10 est une carte spéciale qui banni les cartes présentes sont le tapis de manière définitive.
- L'As est une carte spéciale qui permet à celui qui pose la carte de choisir le joueur de son choix et lui donner le paquet de jeu.

Ce livrable comporte une analyse du diagramme des cas d'utilisation suivie par l'étude du diagramme de classes et du diagramme de séquence. Enfin nous conclurons en nous interrogeant sur les aspects sûrs de la modélisation et ceux qui seront voués à évoluer.

TABLE DES MATIERES

I. DIAGRAMME DE CAS D'UTILISATION



Le cas d'utilisation montre un système, du point de vue de l'utilisateur, sous l'angle des actions qu'il peut mener. Nous avons détaillé ici cinq cas d'utilisation.

Premier Cas :

· Objectif: Permettre à l'utilisateur de jouer une partie de bataille norvégienne

· Acteur concerné : Personne (Joueur)

· Pré condition : La personne a lancé correctement l'application.

· Scénario nominal : jouer la partie

1→ Le système affiche le menu principal.

2→ La personne clique sur « Lancer une partie ».

3→Le système affiche une liste déroulante avec les nombres de joueurs envisageables.

4→ La personne sélectionnée un nombre de joueurs.

5→ Le système affiche un écran de jeu et initialise la partie.

6→La personne entre dans un tour de jeu et a le choix d'échanger des cartes entre sa main et celles posées devant lui face visible. Puis il peut jouer une ou plusieurs cartes de même valeurs.

7→Le système vérifie si la valeur de la ou les carte(s) choisie(s) est jouable, c'est-à-dire supérieure à la dernière carte déposée sur le tapis de jeu. Si la dernière carte est un 7, une carte de valeur inférieure doit momentanément être utilisée par le joueur suivant. De plus, il est possible de jouer la carte 2 à tout moment au fil de la partie. Dans le cas où le joueur ne pourrait jouer aucune carte, il doit ramasser l'intégralité de la pile de cartes du tapis et les placer dans sa main.

8→Une fois qu'une carte a été jouée, le système vérifie que le joueur dispose au minimum de 3 cartes dans sa main sinon il sera invité à piocher le nombre de cartes nécessaire dans la pioche pour y parvenir. Il arrivera un moment ou la pioche sera épuisée et il n'aura plus aucune carte en main, auquel cas dans un premier temps on ramassera les trois cartes visibles situées en face du joueur. Dans un second temps ou la main du joueur sera à nouveau vide, on piochera au hasard une carte des trois cartes faces cachées. La partie se terminant au premier joueur qui n'a plus de cartes faces cachées et que sa main soit aussi vide.

Second Cas :

· Objectif: Permettre à l'utilisateur d'afficher les meilleurs scores obtenus lors des parties antérieures.

· Acteur concerné : Personne (Joueur)

· Pré condition : La personne a lancé correctement l'application.

· Scénario nominal : Afficher les meilleurs scores

1→ Le système affiche le menu principal.

2→ La personne clique sur « Afficher meilleurs scores ».

3→ Le système affiche un tableau contenant le nom des 10 meilleurs joueurs et le nombre de tours nécessaires à la victoire.

Troisième Cas :

· Objectif: Permettre à l'utilisateur des règles de jeu de la bataille norvégienne

· Acteur concerné : Personne (Joueur)

· Pré condition : La personne a lancé correctement l'application.

· Scénario nominal : prendre connaissance des règles du jeu

1→ Le système affiche le menu principal.

2→La personne clique sur « Afficher règles du jeu ».

3→ Le système affiche les règles du jeu et les renseignements du déroulement d'une partie.

Quatrième Cas :

· Objectif : Permettre à l'utilisateur de modifier les options de jeu

· Acteur concerné : Personne (Joueur)

· Pré condition : La personne a lancé correctement l'application.

· Scénario nominal : changer le niveau de difficulté des joueurs virtuels en mode offensif.

1→ Le système affiche le menu principal.

2→ La personne clique sur «Afficher option».

3→ Le système affiche un sous-menu.

4→La personne clique sur « Choix de difficulté des joueurs virtuels ».

5→ Le système affiche une liste déroulante.

6→ La personne sélectionne « offensif » ou « défensif ».

Cinquième Cas:

· Objectif: Permettre à l'utilisateur de quitter le jeu

· Acteur concerné : Personne (Joueur)

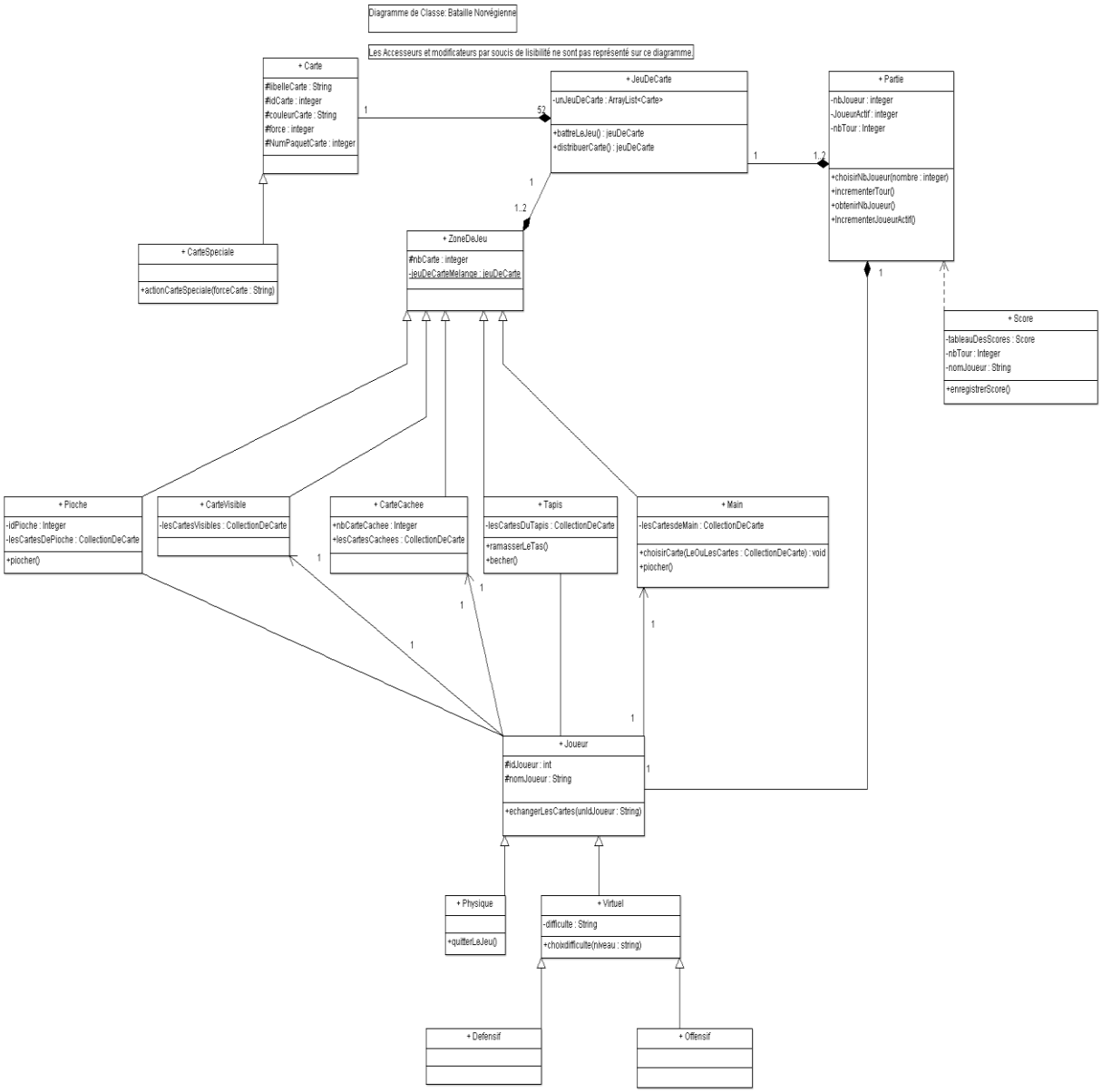
· Pré condition : La personne a lancé correctement l'application.

· Scénario nominal : quitter le jeu.

1→ Le système affiche le menu principal.

2→ La personne clique sur «Quitter le jeu».

II. DIAGRAMME DE CLASSES



Le diagramme de classes nous renseigne sur la structure interne du système. En effet, il permet notamment de visualiser les interactions entre les objets du système pour réaliser les cas d'utilisation. Nous pourrons donc voir les relations entre les classes, lesquelles se constituant d'attributs, de méthodes et d'interfaces.

La classe centrale du système est la classe Partie qui contiendra la méthode main () spécifiant où doit débuter l'exécution du programme. La classe Partie contient des informations sur le nombre de joueurs, joueurActif quel joueur est en train de jouer. nbTours sert à connaître le nombre de tours joués, un tour est effectué lorsque tous les joueurs ont posé une ou plusieurs cartes. La méthode incrementerJoueurActif() est utilisée après que chaque joueur est joué son coup et permet de connaître qui est en train de jouer.

Une Partie est composée de joueurs. La classe Joueur dispose de la méthode echangerLesCartes(pos1 :integer, pos2 :integer) permet d'échanger des cartes entre la main et la zone CarteVisible pour un joueur donné. La classe Physique et Virtuel héritent de Joueur. Les classes Defensif et Offensif héritent de la classe Virtuel et utilisent une stratégie. Nous aurons donc ici deux stratégies : offensive et défensive (Nous reviendrons sur le patron de conception juste après.). Une partie est composée d'un ou deux JeuDecarte.

La classe Score apparaît rattachée à Partie avec une flèche en pointillé, car c'est une classe que nous rajoutons optionnellement. Son but est de stocker les meilleurs scores, c'est-à-dire le nom du joueur et le nombre de tours qu'il a fallu réaliser pour décrocher la victoire et ceci grâce à la méthode enregistrerScore().

La classe JeuDeCarte comporte un attribut unJeuDeCarte qui est une collection (ArrayList), à l'image d'un jeu de cartes rangé par couleur. La méthode battreLeJeu() retourne une collection de jeuDeCarte ou les cartes ont été rangées de manière aléatoire. La méthode distribuerCarte() répartie les cartes dans les différentes zones de jeu. La classe JeuDeCarte est un singleton et composée de 52 cartes. Une carte appartient à un seul JeuDeCarte.

La classe Carte comporte les attributs protégés suivants : libelleCarte, idCarte, couleurCarte, NumPaquetCarte et enfin force de la carte. Dans le jeu de cartes, on peut attribuer une puissance à chaque carte. Par exemple, la carte 3 est de force 3, la carte 4 de force 4 et le roi de force 13.

La classe CarteSpeciale hérite de Carte et dispose de la méthode actionCarteSpéciale(forceCarte) qui en fonction de la carte (Deux, Sept, Huit, As, Dix) passée en paramètre va exécuter un bloc d'instruction différent correspondant à l'effet propre de cette carte. Ensuite, la classe ZoneDeJeu est composé de 1 à 2 JeuDeCarte et comportant un attribut protected nbCarte renseignant le nombre de carte contenues par une zone. Ainsi que l'attribut statique jeuDeCarteMelange de type jeuDeCarte. Les classes Pioche, CarteVisible, CarteCachees,

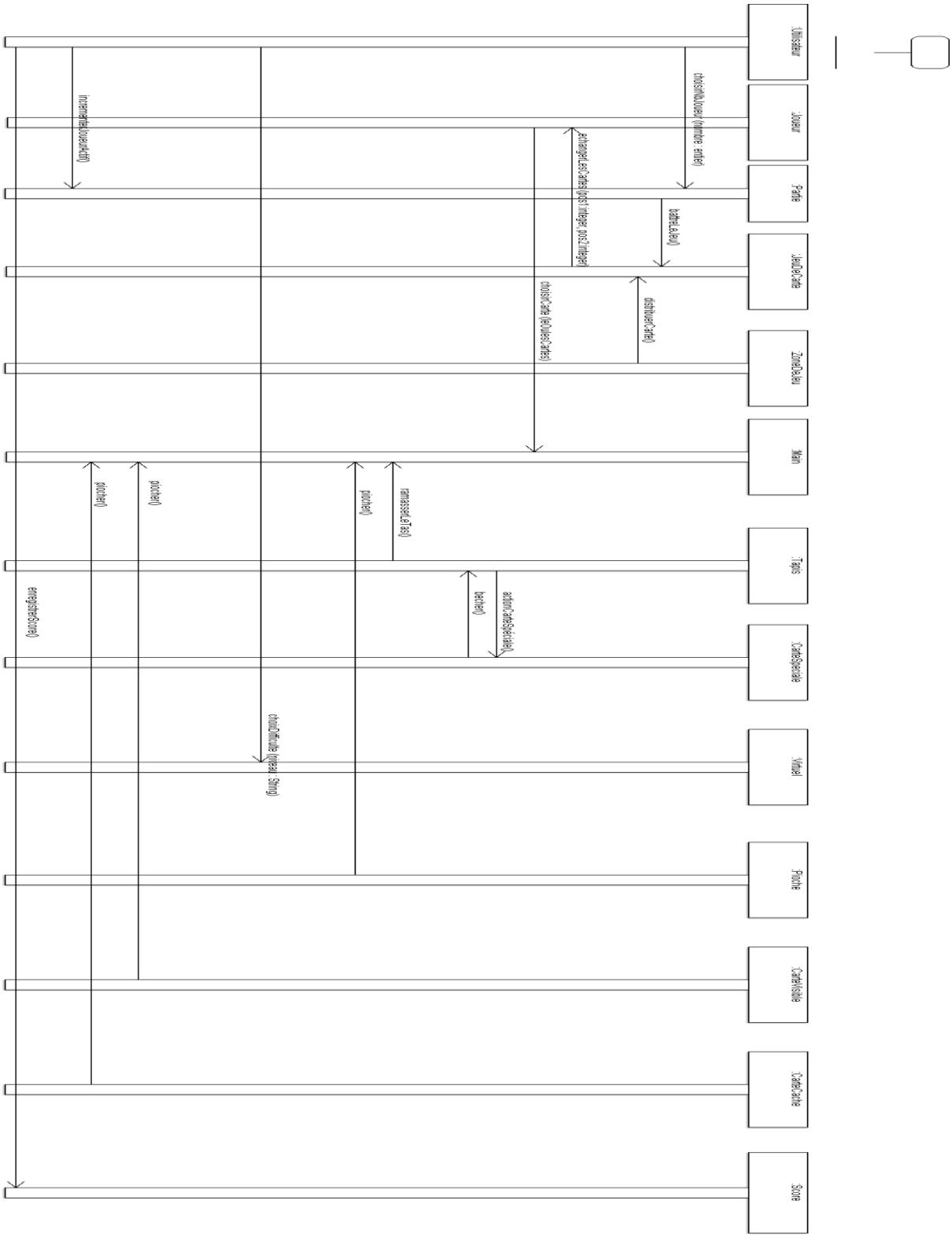
Tapis, Main héritent toutes de ZoneDeJeu. Ces classes ont toutes été redéfinies concernant la collection de cartes.

La classe Main est reliée à la classe Joueur de telle sorte qu'une main corresponde à un et un seul joueur et un joueur n'a qu'une et une seule main. Elle possède la méthode choisirCarte(LeOuLesCartes :Collection de Carte) qui permet de jouer une ou plusieurs cartes de mêmes valeurs.

Le Patron de conception Strategy :

Dans l'optique de séparer les algorithmes de la classe joueur virtuel tout en continuant de jouer normalement les parties, le patron de conception Strategy est donc la solution à adopter.
Il permettra au joueur virtuel de posséder deux types d'attitudes : La première aura tendance à utiliser les cartes spéciales dès qu'il le pourra tandis que l'autre tentera de les conserver pour les utiliser dans des circonstances précises. En clair, cela permet d'adapter à une situation donnée, les actions du joueur virtuel elles-mêmes adaptées à cette situation. Nous disposerons donc d'une interface Stratégie comportant une méthode d'exécution comme effectuer() ; les classes qui vont implémenter cette interface redéfiniront alors la méthode selon le mode du joueur virtuel choisi. On peut même imaginer un changement de mode au cours d'une partie pour un joueur virtuel.

III. DIAGRAMME DE SEQUENCE

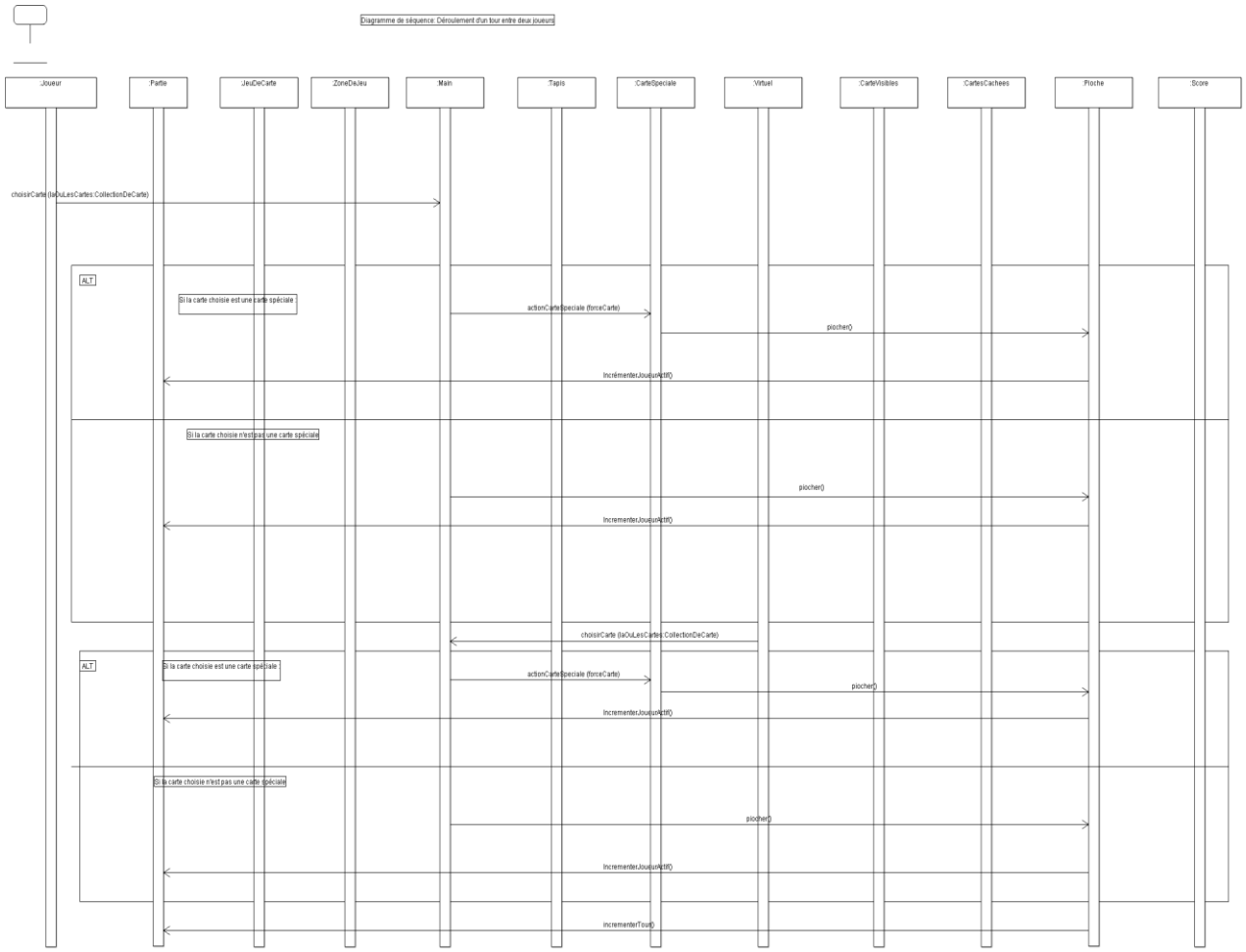


Le diagramme de séquence permet de savoir comment fonctionne l'application. Par rapport aux autres diagrammes, il ajoute la notion de temporalité au diagramme. Il est composé des classes et des méthodes qui font partie des actions directes de l'application. Dans cette partie, nous avons fait deux diagrammes de séquence. Le premier contient l'ensemble de l'application et montre les liaisons entre les classes et le deuxième contient le déroulement d'un tour entre deux joueurs avec des cas alternatif quand un joueur joue une carte spéciale ou non.

Dans le premier diagramme de séquence, nous voulons montrer qui exécute les méthodes dans les classes. Donc le plus important est la source et la destination de chaque méthode.

- Au début de la partie, le joueur choisit le nombre de joueurs virtuels avec qui il a envie de jouer. Cette action part donc du joueur et va ensuite vers la classe partie.
- Ensuite, le système choisit le nombre de jeu de carte en fonction du nombre de joueurs. Si il y a plus de 5 joueurs. Après le jeu de cartes (où les jeux de cartes sont) est mélangé(s). C'est la partie qui mélange la collection qui est dans JeuDeCarte. Puis le jeu de cartes est distribué(s). Chaque zone de jeu de chaque joueur reçoit trois cartes sauf la pioche qui reçoit le reste du tas de cartes.
- Avant le début de la partie, le joueur peut échanger ses cartes avec celles qui sont visibles devant lui. C'est le joueur qui échange ses cartes entre sa main et la carte visible.
- Le joueur choisit ensuite une carte pour la jouer et en pioche une dans la pioche.
- Les actions des cartes spéciales n'interviennent uniquement si le joueur a utilisé une carte spéciale.
    - L'action de bécher s'effectue par le tapis
    - L'action de ramasser le tas est exécutée par le tapis qui envoie les cartes dans la main d'un joueur.
    - Les autres actions sont incluses dans la méthode actionCarteSpeciale().
- A la fin de la partie, quand le joueur n'a plus de carte en main et qu'il n'y a plus de carte dans la pioche, le joueur ramasse les cartes visibles devant lui. Puis quand il a épuisé ses cartes, il ramasse celles qui sont cachées. Quand la partie est terminée, un score est enregistré qui comporte le nombre de tours de la partie et le nom du joueur.
- Durant toute la durée de la partie, quand un joueur a fini son tour une incrémentation est effectuée pour passer au joueur suivant (sauf action cartes spéciales).

Si le joueur va dans le menu option du menu principal, il peut modifier le niveau du joueur virtuel.



Le deuxième graphe est une représentation du déroulement du jeu entre deux joueurs pendant un tour. Il est composé de deux scénarios alternatifs, un quand le joueur choisi une carte spéciale et le deuxième quand ce n'est pas une carte spéciale. Quand le joueur choisit une carte spéciale, cela ajoute l'action de cette carte spéciale au processus de jeu.

Le jeu des deux joueurs est séparé, mais leurs actions sont semblables.

CONCLUSION

En conclusion, nous avons bien cerné le sujet de la bataille norvégienne et les développements principaux qui s'imposent. La réalisation du diagramme de cas d'utilisation nous a permis de nous glisser dans la peau l'utilisateur du jeu. Le diagramme de classes a apporté une vision sur les interactions entre les différents objets du système. Et plus précisément sur la manière dont ils interagissent. Nous avons donc pu dégager de manière sûre la présence des classes : partie, cartes, cartes spéciales, joueur. Cependant, notre point de vue actuel sera sûrement amené à évoluer. La gestion du patron Strategy, les classes de zones et la manière dont elles se lient sera sûrement modifiées par la suite, notamment pour certains attributs et méthodes, certain vont devoir passer en «protected » pour les besoins de l'application. La fonctionnalité de pouvoir mémoriser les meilleurs scores (mémorisation du nom joueur et son nombre de tours pour parvenir à la victoire) est un plus que nous mettrons en œuvre en dernier lieu.