GIRAUD-RENARD Benjamin

Département INFORMATIQUE
1ère année - Groupe D

Année 2020 - 2021

M2107 Projet de programmation 2021

Rendu final

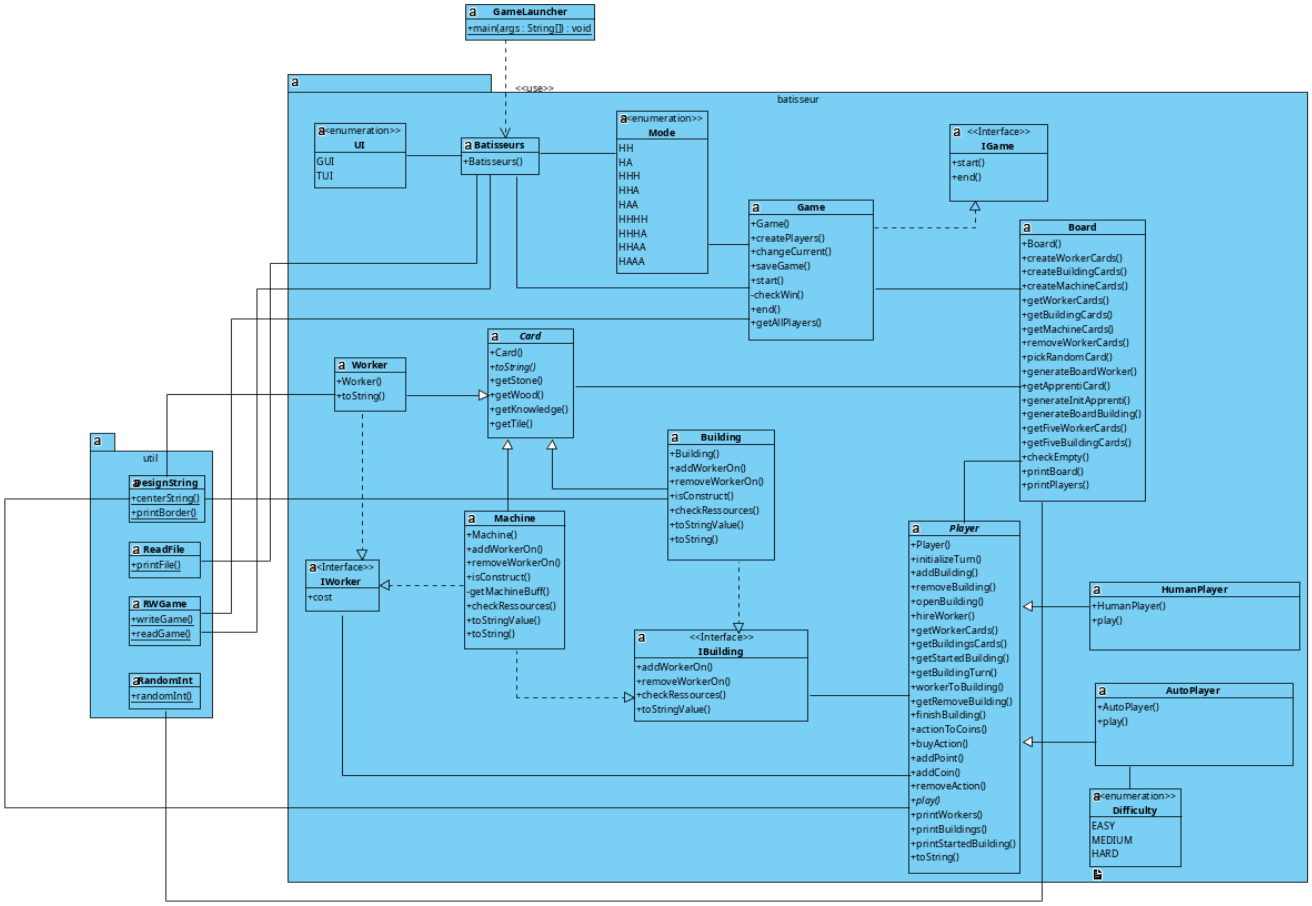
Création d'une version numérique du jeu de société "Les bâtisseurs : Moyen-Age" pour le compte de M. Sébastien Lefèvre

Destinataire
M. Sébastien Lefèvre
Enseignant-chercheur en informatique

Table des matières

Mise à jour des diagrammes

Diagramme de classe d'analyse

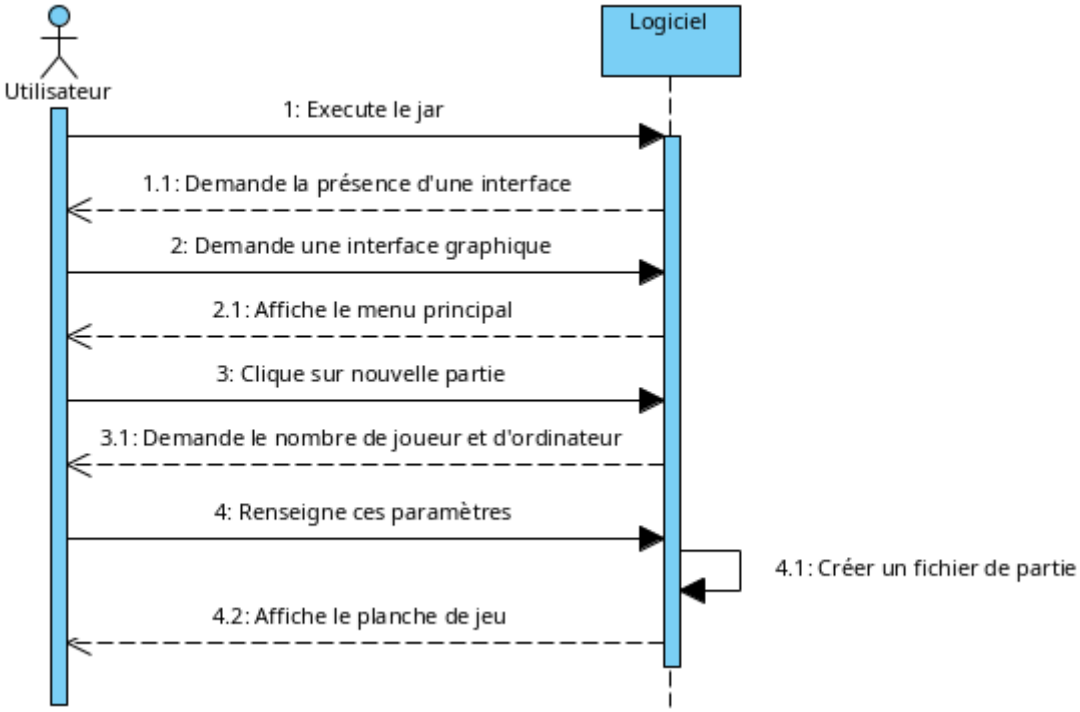


On remarque la présence de nombreuse méthode supplémentaires comparé au diagramme d'analyse précédent. De plus, le package util permettant de réaliser des opérations est présent.

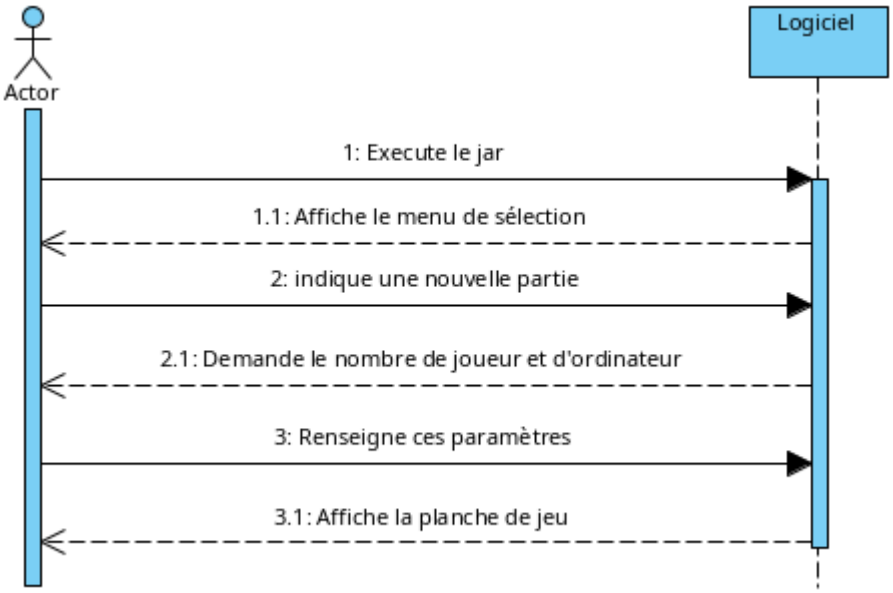
Cependant nous pouvons remarquer qu'il n'y a pas beaucoup de nouvelles classes, démontrant une bonne conception au préalable.

Diagramme de séquence

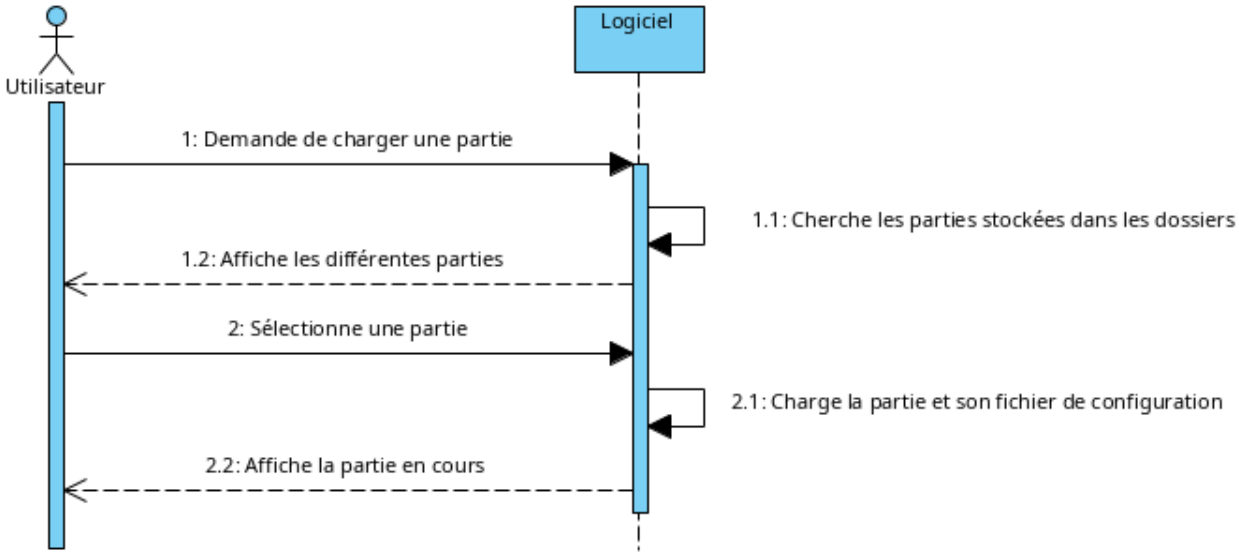
Lancer une partie en GUI n'a pas changé



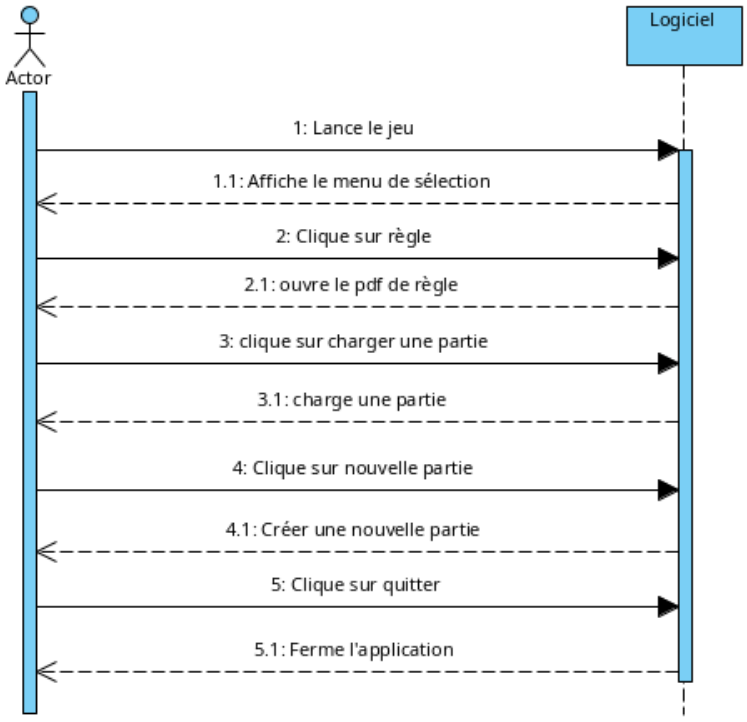
Lancer une partie en TUI (terminal) a légèrement changé. En effet ce mode est par défaut, pas besoin de demander la présence de l'interface ou non.



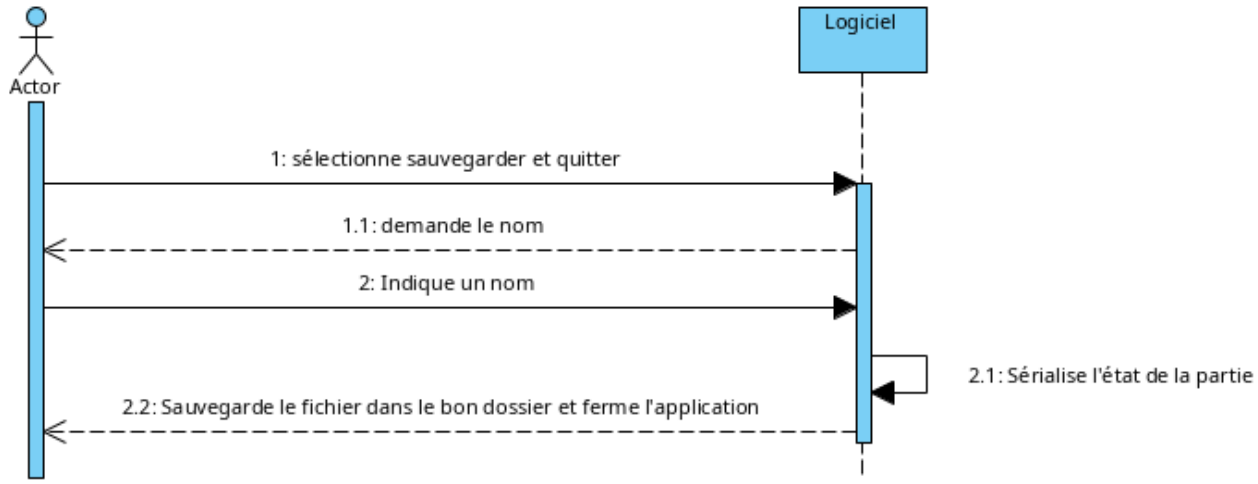
Charger une partie n'a pas été modifié



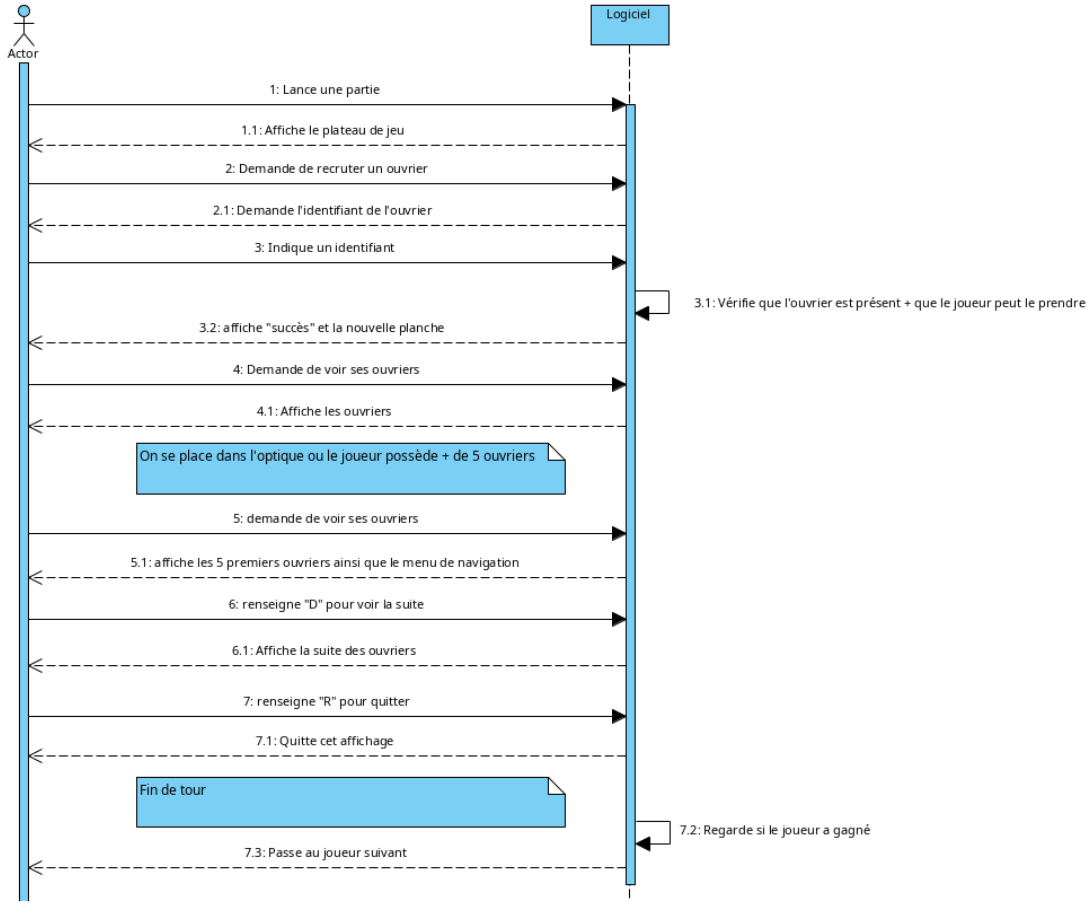
La navigation dans le menu principal a un peu changé. Il n'y a plus de paramètre et en cliquant sur les règles, le logiciel ouvre le pdf.



Dans l'idéal d'une interface graphique, sauvegarder une partie n'a pas changé. Le fonctionnement en console est cependant un peu différent



Le déroulement d'une partie console :



L'ajout d'un bâtiment correspond au même schéma. Faire travailler un ouvrier sur un bâtiment demande simplement deux identifiants. Vous pouvez par la suite voir l'avancement de vos bâtiments de la même manière que pour les ouvriers

Description des choix algorithmiques

Durant le développement, j'ai dû utiliser des méthodes pour éviter les répétitions et simplifier au maximum le programme pour me simplifier la programmation.

Le premier choix important provient de la conception des cartes. De l'architecture nécessaire afin de faire fonctionner les machines comme un ouvrier ou comme un bâtiment. Pour cela j'ai utilisé une super-classe Card permettant de décrire les interactions communes aux trois type de cartes. Pour permettre de différencier les cartes, deux interfaces IWorker et IBuilding avec Machine implémentant les deux. Permettant ainsi de passer en paramètre une carte sans problème de compatibilité.

Par manque de temps l'algorithme du joueur automatique n'est pas très intelligent mais intègre un minimum de réflexion.

En effet, le joueur automatique se base sur des actions aléatoires.

16 % de chance d'ouvrir un bâtiment en priorisant ceux rapportant le maximum de point ou les machines.

33 % de chance d'ajouter un ouvrier à sa main en priorisant celui le moins cher. Si l'action est impossible, le joueur décide d'échanger des actions contre des écus.

50 % de chance d'envoyer un ouvrier sur un bâtiment. Si le joueur possède des bâtiments en cours de construction, il priorise celui avec le plus d'ouvriers en train de travailler dessus. Il sélectionne un de ses ouvriers au hasard et a une faible chance d'échanger des actions contre des écus.

Même si le joueur possède des bâtiments en cours de construction, il a 50 % de chance de choisir un de ses bâtiments au hasard.

Au final, ces interactions bien que simple donnent un minimum d'intelligence à ce robot.

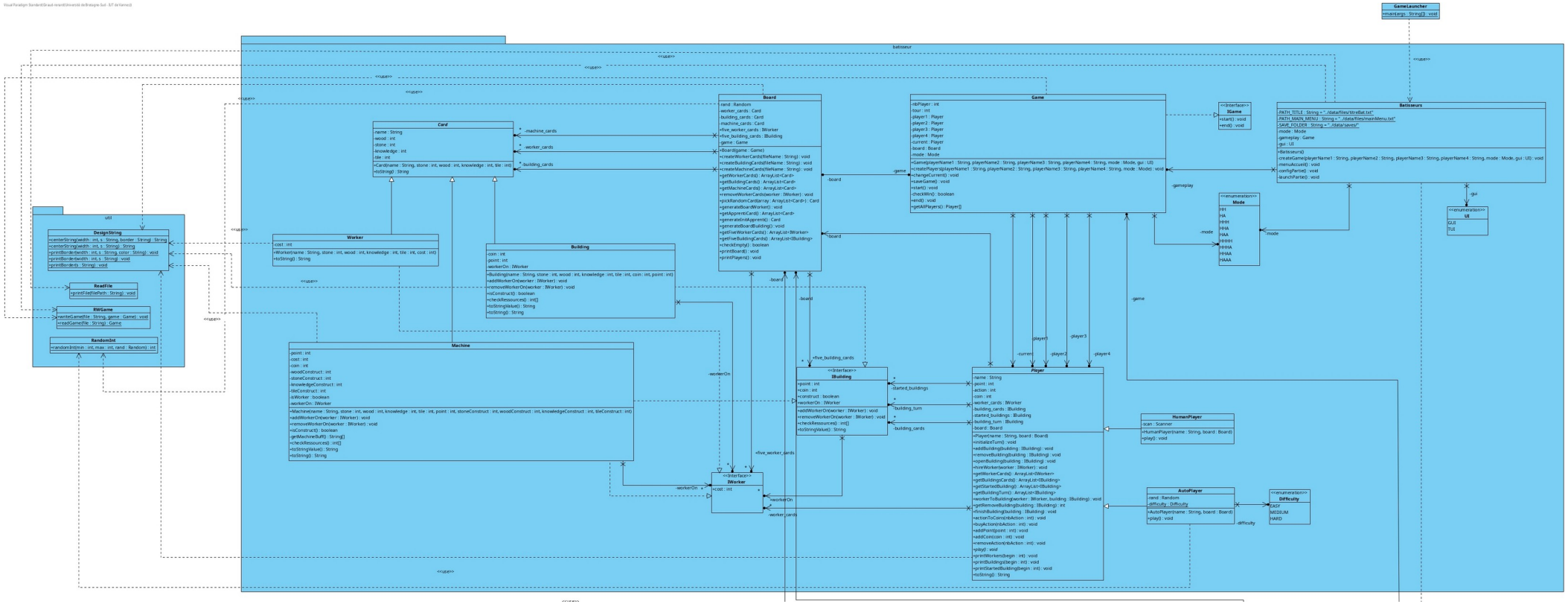
Réussir à trouver le gagnant n'est pas non plus une chose simple. Pour correctement avoir le classement, j'ai dû trier plusieurs fois un tableau des scores tout en conservant l'identité des joueurs. Avec un peu de réflexion derrière, le résultat est convainquant.

Concernant l'affichage terminal, afficher plusieurs éléments alignés n'est pas une mince affaire. Pour cela j'ai opté par un affichage ligne par ligne. Affichant ainsi seulement une partie de la carte totale pour permettre un bel affichage.

Couplé à une méthode permettant de centrer les strings entre deux bordures, l'affichage est une des choses dont je suis le plus content dans ce projet. L'affichage console est par ailleurs beaucoup plus développé que ce qui était initialement prévu dans le cahier des charges.

Diagramme de classe de conception fin de projet

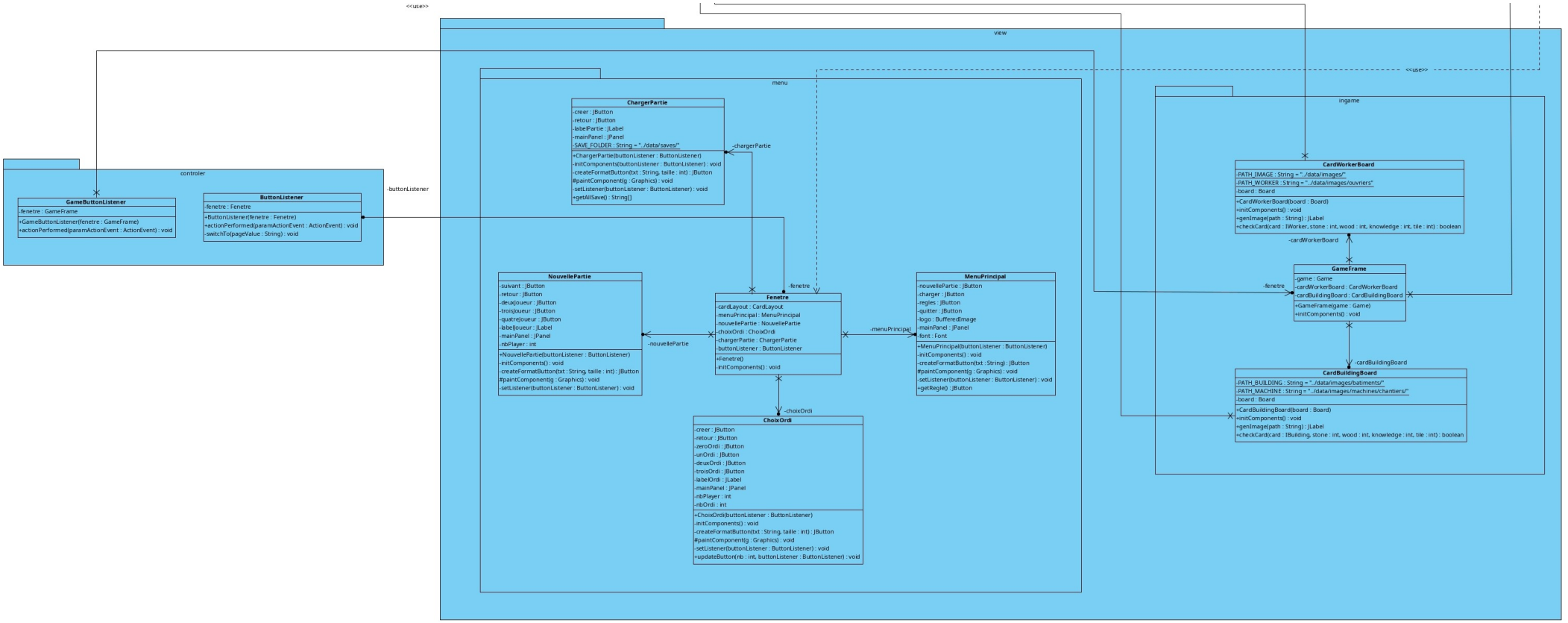
Diagramme de conception – Model + util



Ce diagramme de conception a un peu changé depuis la prévision du cahier d'analyse. La plupart des modifications apparaissent sur des méthodes n'étant initialement prévues ou certaines fonctionnalités non pensées durant la conception.

Également la liaison pour le package view ainsi que toute la partie textuelle n'était pas présente durant la conception du précédent rendu.

Diagramme de conception – View + controler



Je n'ai pas pu finir la partie view. Servant initialement à la visualisation graphique du jeu. Comme nous pouvons le voir dans ce diagramme, la partie menu est fonctionnelle. Il manque la partie visuelle et controler en jeu. Au vu de la taille du diagramme, l'image complète est disponible dans l'archive.

Campagne de tests effectuée

La liste des tests dans le cahier des charges ont été effectués afin de s'assurer du bon déroulement de l'exécution des méthodes.

Les tests permettent de s'assurer qu'une méthode fonctionne correctement. Pour cela j'ai décidé de tester les méthodes de la majorité des classes du projet.

Voici le résultat du build ant.

```
$ ant test
Buildfile: /hdd/IUT/semestre 2/partie 2/projet/Code/build.xml

init:

compile:

test-compile:
    [javac] Compiling 6 source files to /hdd/IUT/semestre 2/partie 2/projet/Code/build/test

test:
    [junit] Running test.AutoPlayerTest
    [junit] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,762 sec
    [junit] Running test.BoardTest
    [junit] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,56 sec
    [junit] Running test.BuildingTest
    [junit] Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,044 sec
    [junit] Running test.GameTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,386 sec
    [junit] Running test.MachineTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,047 sec
    [junit] Running test.WorkerTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,045 sec

BUILD SUCCESSFUL
Total time: 7 seconds
```

Avancement réel du développement

Pour cette partie nous allons comparer les fonctionnalités demandées dans le cahier des charges et celles réalisées.

Fonctionnalités principales

• Le jeu doit être jouable en version console et graphique

La version proposée ici est entièrement jouable en console mais ne possède pas les interactions utilisateur en graphique par faute de temps.

• Possibilité de jouer en multijoueur local (2, 3, 4 personnes)

Implémenté.

• Possibilité d'affronter un ordinateur prenant des décisions de base

Implémenté. L'ordinateur possède des actions basiques expliquées en détail dans Description des choix algorithmiques.

• Possibilité de sauvegarder le jeu et de reprendre la partie

Implémenté. Vous pouvez avoir un nombre infini de sauvegarde et les charger grâce à leur nom.

• Gérer les victoires et les égalités.

Implémenté

• Le joueur doit pouvoir voir la main des autres

Après réflexion, j'ai décidé de laisser le joueur voir les caractéristiques du joueur et non sa main entière. On considère que la main du joueur est cachée pour plus d'enjeu

• Les règles fidèles au jeu doivent être respectées :

• Chaque joueur reçoit un apprenti au début de partie

• Implémenté

•Chaque joueur reçoit 10 pièces au début de partie

• Implémenté

• Le premier joueur est désigné aléatoirement

- Implémenté

- Le joueur possède des actions qu'il peut vendre contre des écus
    - Implémenté
- Les actions ont un coût, et envoyer un ouvrier sur un chantier est variable.
    - Implémenté

Fonctionnalités supplémentaires

Comme expliqué dans le cahier des charges, des fonctionnalités supplémentaires peuvent avoir lieu.

- Ajouter des niveaux de difficultés à l'ordinateur
    - Non implémenté
- Ajouter un tutoriel
    - Non implémenté
- Ajouter des sons
    - Non implémenté
- Gérer plusieurs sauvegardes
    - Implémenté
- Avoir la possibilité de créer des nouvelles cartes
    - Non implémenté
- Pouvoir nommer son joueur
    - Implémenté dans la version console. Par défaut dans la version graphique.

Comme nous pouvons le voir, les fonctionnalités nécessaires au jeu sont implémentées. Il manque le fonctionnement complet de la partie graphique. Avec un peu plus de temps, j'aurais pu l'implémenter.

Synthèses des difficultés rencontrées et solutions apportées

Durant ce projet, j'ai pu être confronté à plusieurs difficultés.

Tout d'abord, le principal problème reste le temps. Gérer son temps limité est compliqué, on se retrouve très rapidement à faire des journées entières de programmation pour ne pas être en retard.

Je n'ai pas vraiment rencontré de difficulté technique. Évidemment j'ai dû apprendre à utiliser des nouveaux outils (la sérialisation par exemple), mais en recherchant un peu sur la javadoc et différent exemple, on arrive rapidement à mettre au point ce que l'on souhaite.

L'organisation du programme est un point compliqué à mettre en place. Nous devons séparer au maximum tout en simplifiant la programmation avec des méthodes intéressantes. Suivre à la lettre le modèle MVC est donc un point compliqué dans ce projet. Concernant l'architecture des classes, la conception effectuée au préalable a été utile pour structurer le projet correctement et pouvoir se lancer efficacement dans le développement.

Bilan

Au final, ce projet m'a permis d'utiliser des outils que je ne connaissais pas. Développer entièrement en autonomie sans aucune base permet également de se rendre compte de l'importance de la structure du projet.

Ne pas pouvoir terminer dans les temps la partie graphique est un peu décevant, le manque de temps est au final le seul point négatif du projet.