Vannes
Université Bretagne Sud

Informatique

Projet de programmation
Lilian DAMIEN – 1A
Année 2020 – 2021

M2107 - Projet de programmation

Rendu Final

Les Bâtisseurs : Moyen-Âge

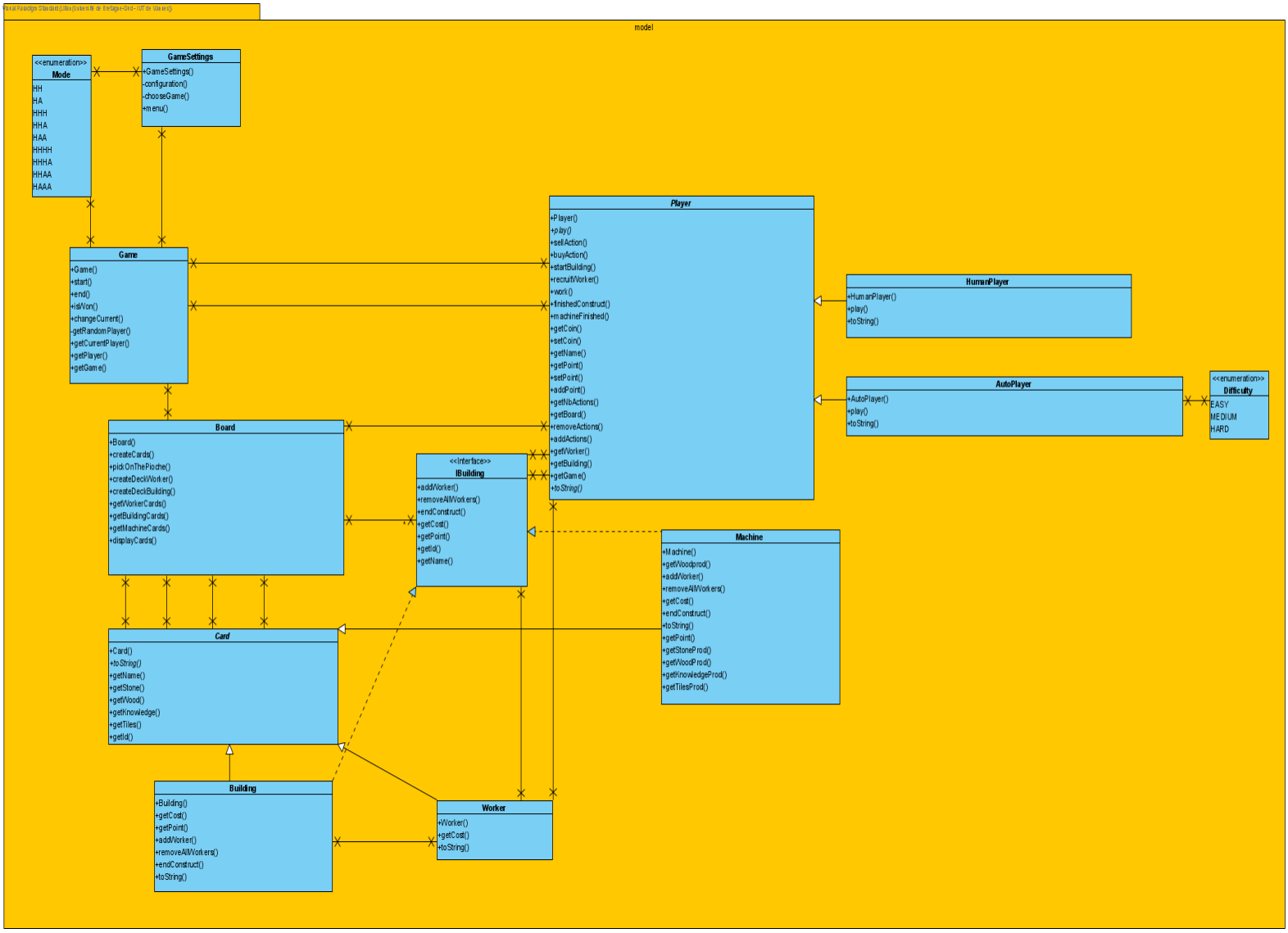
IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr

Université Bretagne Sud

Tables des matières

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr

1) Diagramme de classes d'analyse

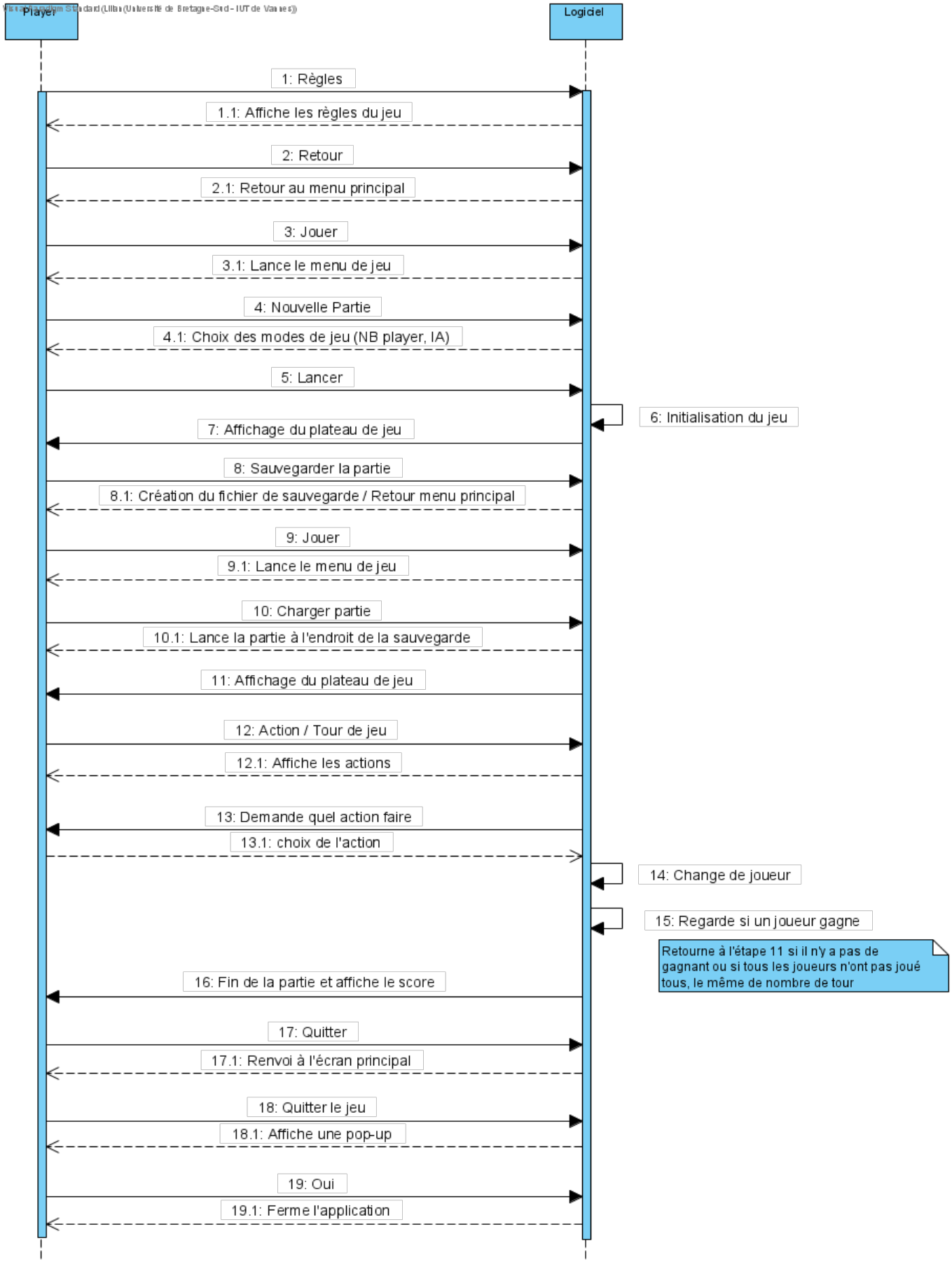


Les changements majeurs apportés au diagramme de la classe est l'ajout d'une interface Ibuilding qui va nous permettre de regrouper les machines et les building ensembles. Il y aussi l'ajout de plusieurs méthodes dans les classes Players et Board principalement.

Diagramme de séquence de boîte noire

Le diagramme de séquence n'a pas reçu de changement, en effet le diagramme illustre bien le fonctionnement de l'application que ce soit pour la partie textuelle ou graphique.

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr



IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr

2) Descriptions des choix techniques

J'ai tout d'abord décider de stocker les noms des joueurs dans une ArrayList, le nombre de joueur dépendait du mode de jeu choisit et proposé par l'énumération Mode.

Lors de la création de la partie un joueur parmi la liste est choisit au hasard pour jouer en premier, la boucle de jeu se situe dans la méthode start(), la boucle de jeu continue tant que la méthode isWon() qui va vérifier dans la liste de joueur après chaque changement de tour si l'un des joueurs à obtenu 17 points minimum renvoie faux. Si cette méthode renvoie vrai alors la boucle de jeu est terminé et viens le déclenchement de la méthode end() qui va faire le calcul des points de chaque joueurs et afficher le vainqueur.

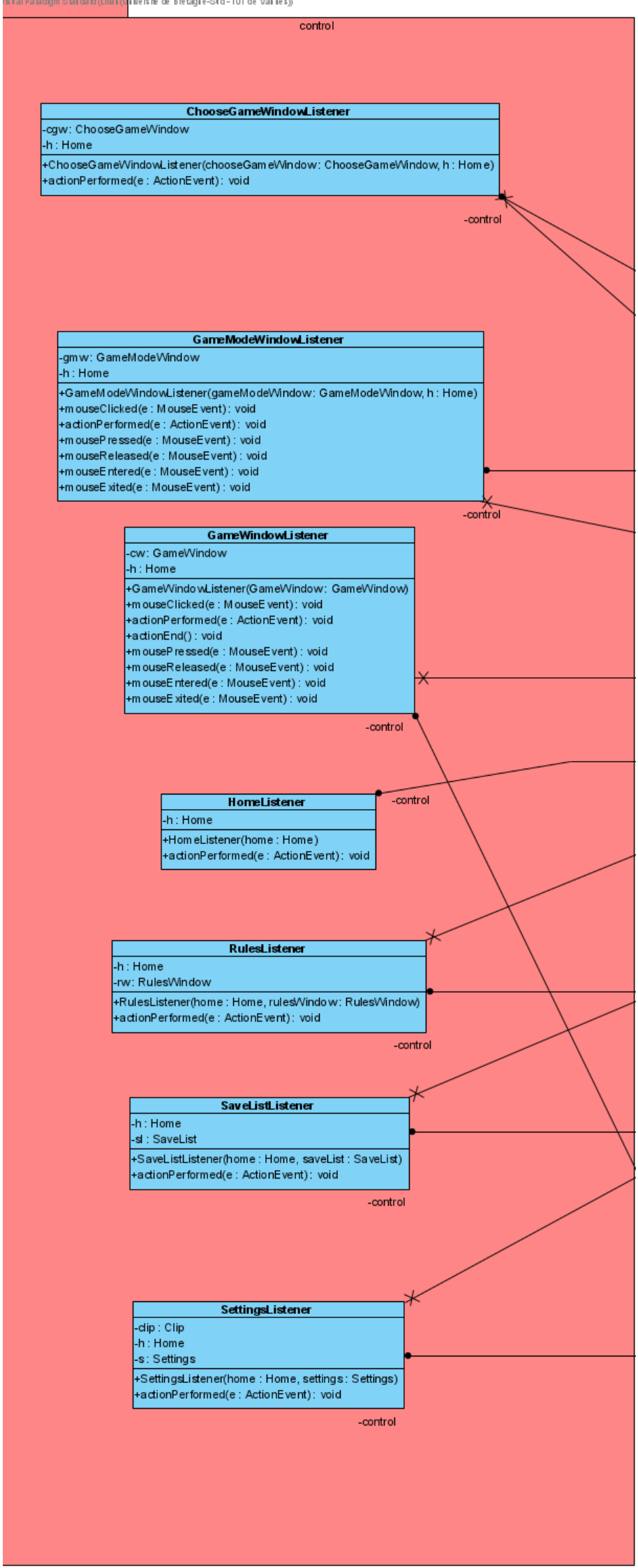
La méthode play() de HumanPlayer est une boucle de jeu qui s'arrête lorsque le nombre d'actions du joueurs est égale à 0. Elle affiche un menu de proposition d'actions possible, et demande au joueur qu'elle action il souhaite faire avec l'utilisation d'un scanner. Ensuite la valeur rentrée est comparé à plusieurs scénarios correspondant aux actions, puis ensuite déclenche le scénario choisit en vérifiant si l'action souhaitée est possible comme la vérifications du nombre d'écus, si la carte que l'on veut recruter, ouvrir, construire est bien dans notre deck de 5 cartes ou dans notre main de jeu. Pour cela j'ai décidé de créer des méthodes pour chaque actions vérifiant les paramètres et la validité de l'action, puis l'application en elle même de l'action et les répercussions comme la suppressions d'une action, ou l'ajout dans notre main d'une carte ou le paiement en écu d'une carte, etc.

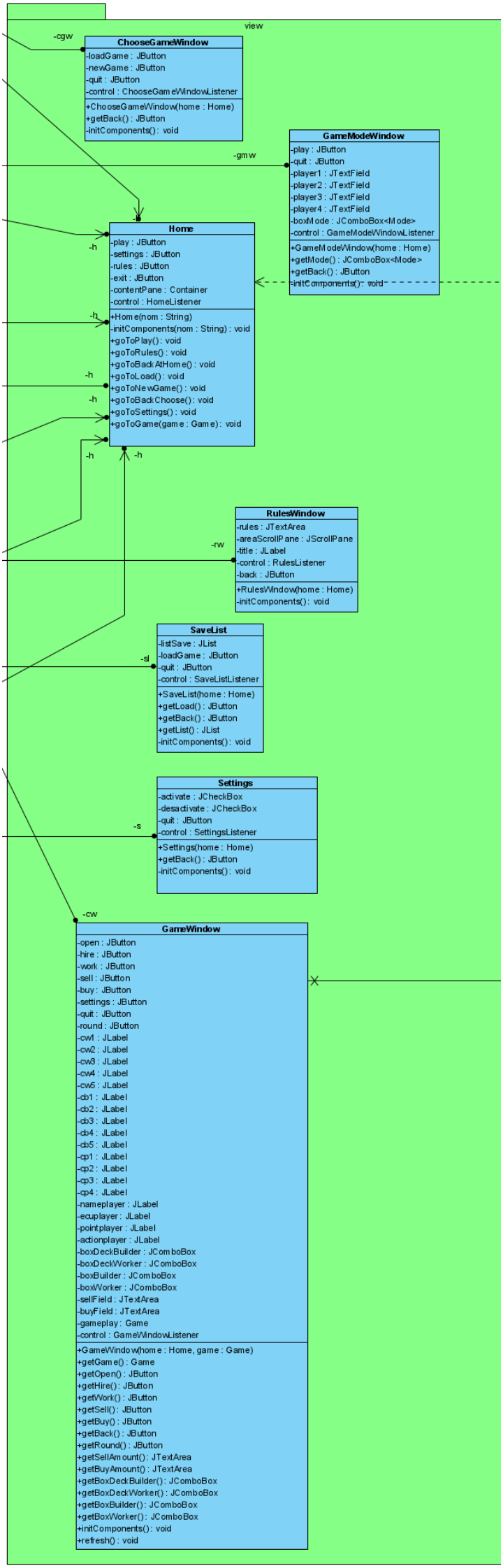
La méthode play() de AutoPlayer est assez similaire que celle de HumanPlayer cependant à la place d'un scanner j'utilise un Math.Random avec en maximum le nombre de proposition, la valeur retourné correspond à l'identifiant d'une action et si tout est valide l'action est réalisé. J'ai essayé de faire en sorte que dès que l'IA avait au minimum un bâtiment et un ouvrier libre dans sa main c'était l'action d'envoyer travailler l'ouvrier qui été déclencher.

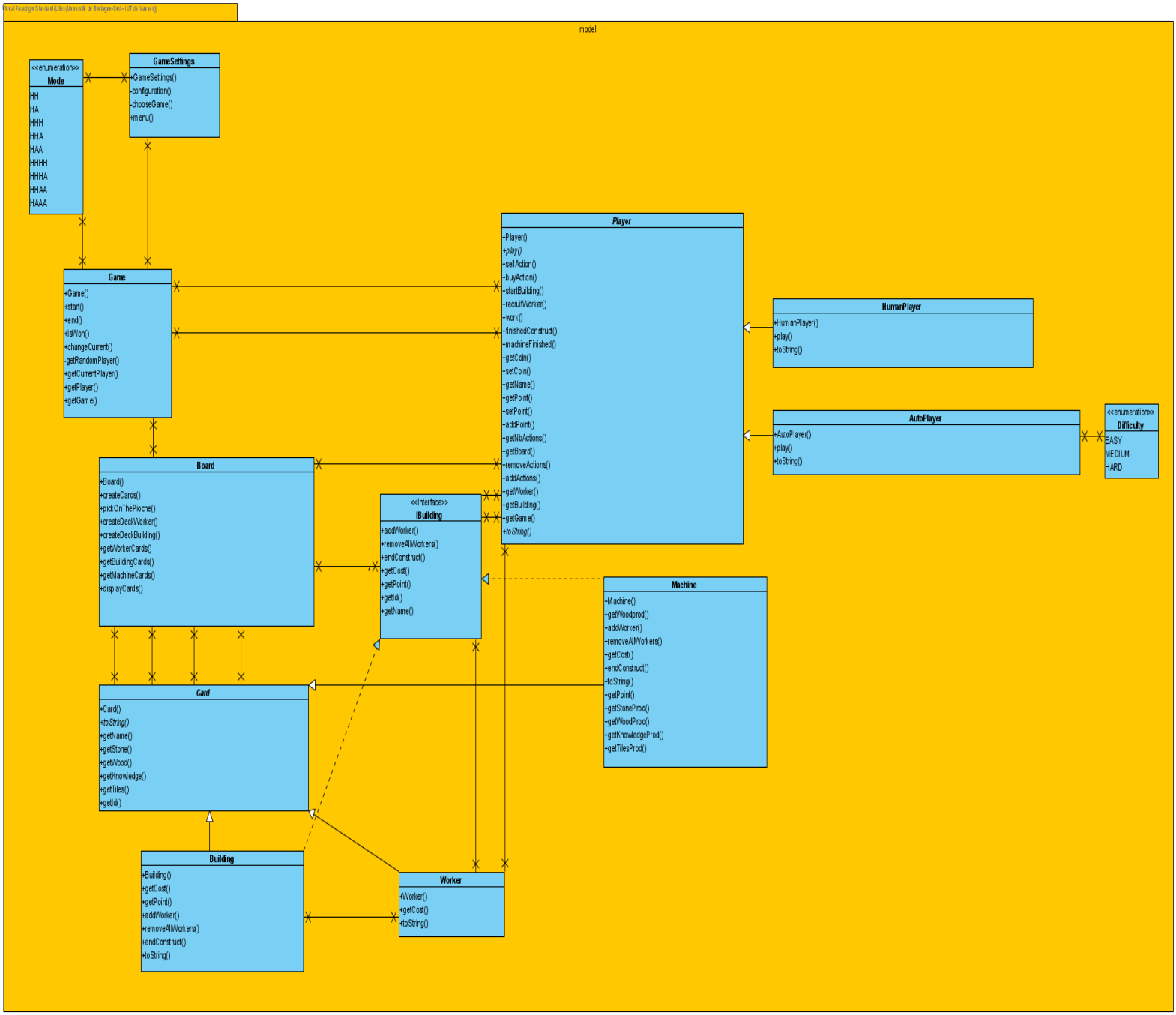
La méthode createCards() dans la classe Board lit les fichiers se situant dans data est crée un à un les objets. Ensuite j'ai décidé de mettre ses cartes dans les ArrayList de leur type respectifs, afin ensuite de créer deux méthodes l'une pour les ouvriers une autres pour les bâtiments qui vont créer les deux decks de 5 cartes respectifs. Pour remplir c'est deck, il y'a une boucle qui ne s'arrête que quand le deck est composé de 5 cartes, si il ne l'est pas alors sa fait appelle à une méthode pickOnThePioche qui va suivant le type de carte passé en paramètre prendre une carte aléatoirement du type pour le mettre dans le deck. Ensuite nous avons deux boucles à l'intérieur d'une méthodes displayCards() qui affiche les deux decks.

3) Diagramme de classe d'implementation

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr


control
ChooseGameWindowListener
-cgw: ChooseGameWindow
-h : Home
+ChooseGameWindowListener(chooseGameWindow: ChooseGameWindow, h : Home)
+actionPerformed(e : ActionEvent): void
-control
GameModeWindowListener
-gmw: GameModeWindow
-h : Home
+GameModeWindowListener(gameModeWindow: GameModeWindow, h : Home)
+mouseClicked(e : MouseEvent): void
+actionPerformed(e : ActionEvent): void
+mousePressed(e : MouseEvent) : void
+mouseReleased(e : MouseEvent) : void
+mouseEntered(e : MouseEvent) : void
+mouseExited(e : MouseEvent) : void
-control
GameWindowListener
-cw: GameWindow
-h : Home
+GameWindowListener(GameWindow: GameWindow)
+mouseClicked(e : MouseEvent): void
+actionPerformed(e : ActionEvent): void
+actionEnd() : void
+mousePressed(e : MouseEvent) : void
+mouseReleased(e : MouseEvent) : void
+mouseEntered(e : MouseEvent) : void
+mouseExited(e : MouseEvent) : void
-control
HomeListener
-h : Home
+HomeListener(home : Home)
+actionPerformed(e : ActionEvent): void
-control
RulesListener
-h : Home
-rw: RulesWindow
+RulesListener(home : Home, rulesWindow: RulesWindow)
+actionPerformed(e : ActionEvent): void
-control
SaveListListener
-h : Home
-sl : SaveList
+SaveListListener(home : Home, saveList : SaveList)
+actionPerformed(e : ActionEvent): void
-control
SettingsListener
-clip : Clip
-h : Home
-s : Settings
+SettingsListener(home : Home, settings : Settings)
+actionPerformed(e : ActionEvent): void
-control


view
-cgw
ChooseGameWindow
-loadGame : JButton
-newGame : JButton
-quit : JButton
-control : ChooseGameWindowListener
+ChooseGameWindow(home : Home)
+getBack() : JButton
-initComponents(): void
-gmw
GameModeWindow
-play : JButton
-quit : JButton
-player1 : JTextField
-player2 : JTextField
-player3 : JTextField
-player4 : JTextField
-boxMode : JComboBox<Mode>
-control : GameModeWindowListener
+GameModeWindow(home : Home)
+getMode() : JComboBox<Mode>
+getBack() : JButton
-initComponents(): void
-h
Home
-play : JButton
-settings : JButton
-rules : JButton
-exit : JButton
-contentPane : Container
-control : HomeListener
+Home(nom : String)
-initComponents(nom : String) : void
+goToPlay() : void
+goToRules() : void
+goToBackAtHome() : void
+goToLoad() : void
+goToNewGame() : void
+goToBackChoose() : void
+goToSettings() : void
+goToGame(game : Game): void
-rw
RulesWindow
-rules : JTextArea
-areaScrollPane : JScrollPane
-title : JLabel
-control : RulesListener
-back : JButton
+RulesWindow(home : Home)
-initComponents(): void
-sl
SaveList
-listSave : JList
-loadGame : JButton
-quit : JButton
-control : SaveListListener
+SaveList(home : Home)
+getLoad() : JButton
+getBack() : JButton
+getList() : JList
-initComponents(): void
-s
Settings
-activate : JCheckBox
-desactivate : JCheckBox
-quit : JButton
-control : SettingsListener
+Settings(home : Home)
+getBack() : JButton
-initComponents(): void
-cw
GameWindow
-open : JButton
-hire : JButton
-work : JButton
-sell : JButton
-buy : JButton
-settings : JButton
-quit : JButton
-round : JButton
-cw1 : JLabel
-cw2 : JLabel
-cw3 : JLabel
-cw4 : JLabel
-cw5 : JLabel
-cb1 : JLabel
-cb2 : JLabel
-cb3 : JLabel
-cb4 : JLabel
-cb5 : JLabel
-cp1 : JLabel
-cp2 : JLabel
-cp3 : JLabel
-cp4 : JLabel
-nameplayer : JLabel
-ecuplayer : JLabel
-pointplayer : JLabel
-actionplayer : JLabel
-boxDeckBuilder : JComboBox
-boxDeckWorker : JComboBox
-boxBuilder : JComboBox
-boxWorker : JComboBox
-sellField : JTextArea
-buyField : JTextArea
-gameplay : Game
-control : GameWindowListener
+GameWindow(home : Home, game : Game)
+getGame() : Game
+getOpen() : JButton
+getHire() : JButton
+getWork() : JButton
+getSell() : JButton
+getBuy() : JButton
+getBack() : JButton
+getRound() : JButton
+getSellAmount() : JTextArea
+getBuyAmount() : JTextArea
+getBoxDeckBuilder() : JComboBox
+getBoxDeckWorker() : JComboBox
+getBoxBuilder() : JComboBox
+getBoxWorker() : JComboBox
+initComponents() : void
+refresh() : void



Le gameLauncher demande au joueur sur quelle version il veut jouer si il choisit la version textuelle alors une instance de gameSettings est crée, sinon c'est l'instance Home qui a plusieurs méthodes. Tout d'abord c'est la première page de présention qui

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr

propose les 4 options du menu (play, settings, rules, exit), ensuite comme j'ai décidé de tout mettre dans un Container, j'ai crée des méthodes pour chaques pages en effet lorsque l'un des boutons d'une page est activé et détecté par le biais du controller de la window en question, il y'a l'appel d'un des méthodes qui a pour but de supprimer le contenu du container, de créer une nouvelle instance de la page en question et de l'ajouter au container.
La view Settings contient deux JcheckBox qui permet d'activer ou de désactiver la musique, elle contient aussi un bouton retour au menu principal.

La view Rules a pour objectif d'afficher avec un JtextArea, les règles du jeu et un bouton retour au menu principal.

La view ChooseGameWindow est une view avec 3 Jbutton qui permet de choisir entre reprendre une partie ou en commencé une nouvelle ou retourner au menu.

La view SaveList est la view regroupant toutes les sauvegardes de parties dans une Jlist il suffit de cliquer sur le nom de la sauvegarde puis le bouton load est la partie est reprise en déclenchant la méthode pour afficher la view GameWindow.

La view GameWindow représente le plateau de jeu elle contient plusieurs composant comme toutes les actions qui sont définit par un Jbutton, les deux decks représenté par des Jlabel contenant l'icon de la carte par rapport à son id, des Jpanel pour les informations comme les points du joueurs, etc. Puis des JcomboBox ou JtextField pour les actions afin de rentrer ou de sélectionner les valeurs pour réaliser l'action.

4) Tests

Le test de la version graphique permet bien de répondre aux scénarios de tests proposés dans le cahier des charges, avec le résultat attendu qui est correct.

Pour les test JUnit j'ai décidé de tester toutes les méthodes des classes Worker, Building, Machine afin de m'assurer du fonctionnement des objets représentant les cartes, et ensuite j'ai testé les méthodes de la classes HumanPlayer afin de tester les méthodes de Player.

Voici le rapport des test :

PS C:\Users\Lilian\Desktop\batisseurs> ant test
Buildfile: C:\Users\Lilian\Desktop\batisseurs\build.xml

init:

compile:

test-compile:

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr

```
    [javac] Compiling 4 source files to C:\Users\Lilian\Desktop\batisseurs\build\test

test:
    [junit] Running test.BuildingTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,013 sec
    [junit] Running test.HumanPlayerTest
    [junit] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,031 sec
    [junit] Running test.MachineTest
    [junit] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,015 sec
    [junit] Running test.WorkerTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,013 sec

BUILD SUCCESSFUL
Total time: 1 second
--- Summary ---
Errors:0, Info:9, Verbose:5, Debug:0
```

5) Avancement / Difficultés rencontrées

Pour la partie avancement, toutes les fonctionnalités ont été implémentes ils peut cependant resté quelques inexactitudes, j'ai implémenter en partie une fonctionnalité de musique. Sinon une version textuelle et graphique ont bien été réalisés.

Pour ce qui est de la partie model, les problèmes que j'ai rencontrée sont principalement du au fait que je n'ai pas pensé tout d'abord à utiliser une interface pour pouvoir regrouper les bâtiments et les machines.
J'ai rencontrée aussi des difficultés lors du codage de mon IA en effet j'ai remarquer que l'IA était forcément perdante et al plupart du temps ouvrer un chantier ou recruter un ouvrier ce qui faisait que le stock de carte diminué énormément, c'est donc pour sa que j'ai fait en sorte d'essayer de forcer l'IA a choisir d'envoyer un travailler un ouvrier dès qu'elle le peut.
J'ai aussi prévu que si on rajouté des modes où seuls des IA jouer, de créer une nouvelle carte de chaque type se nommant « Stop » est avec tout ses attributs à 0 qui serait créer quand la pioche serait vide et donc dès que cette carte se trouverait en première position dans le deck donc la dernière carte, la boucle de jeu s'arrêterait obligatoirement et déclare un vainqueur .

Pour la partie de sauvegarde j'ai rencontré une erreur lié à la sérialisation d'un attribut scanner qui est impossible.

Pour la partie view / controller, les difficultés que j'ai rencontrée c'est l'affichage des cartes correspondantes au cartes des decks ou des mains de joueurs.

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr

6) Bilan personnel

Pour ce qui s'agit de mon bilan personnel, j'ai bien aimé travaillé sur ce projet qui est l'un de mes premiers « grand » projet d'informatique, cela ma permis aussi de constater l'importante des phases et notamment celle de la conception où je n'avais pas pensé à énormément de choses dont je me suis rendu compte lors du codage.
Je suis satisfait du travail que j'ai réalisé que ce soit la version textuelle ou graphique.

J'ai trouvé cependant que la durée de la phase de codage était un peut courte, une durée un plus importante m'aurais permis de m'occuper plus de l'aspect graphique et textuelle (les cartes affichés) et l'aspect communication avec le client manqués.

IUT Vannes
8 Rue Montaigne - 56017 VANNES Cedex
02 97 62 64 64 - www.iutvannes.fr