
Rapport - Projet POO

GELINAUD Clément
GHITA Alessandro
Benmoussati Souhaïl

SOMMAIRE

I - Documentation utilisateur	3
1.1 - Installation du jeu	3
1.2 - Comment jouer	3
1.3 - Listes des commandes disponibles	4
1.4 - Listes des mini-jeux disponibles	6
II - Documentation développeur	8
2.1 - Les différents packages	8
2.2 - Diagramme de classe	13
III - Organisation du groupe	16
3.1 - La préconception	16
3.2 - Travail de groupe et organisation du travail	16
3.3 - Répartition des tâches	16
3.3.1 - Ghita Alessandro	16
3.3.2 - Benmoussati Souhail	17
3.3.3 - Gelinaud Clement	17

I - Documentation utilisateur

1.1 - Installation du jeu

Pour installer le jeu, il faudra dans un premier temps télécharger et installer sur votre ordinateur :

- Java : <https://www.java.com/fr/download/>

Vous pourrez ensuite lancer le jeu avec un interpréteur de commande, pour cela vous devez :

- Se déplacer dans le répertoire contenant le fichier “spg.jar”
- Lancer le jeu avec la commande “java -jar “spg.jar ”

1.2 - Comment jouer

Quand vous lancez le jeu, vous arrivez directement dans la salle principale où vous pouvez vous déplacer vers un mini-jeu. Il y a 6 mini-jeux en tout. Quand vous êtes dans la salle principale, vous avez accès à des commandes. Celles-ci sont rappelées avec la commande “help”.

Au démarrage du jeu, vous posséderez directement l’item couteaux. Au fur et à mesure de la partie, le joueur reçoit un chargeur puis un pistolet.

Pour résumer, le but du jeu est de le terminer sans perdre un seul mini-jeu.

Bonne chance !

1.3 - Listes des commandes disponibles

Pour interagir avec le jeu, vous êtes obligé d'utiliser des commandes. Vous pouvez trouver ci-dessous la liste des commandes disponibles au cours de votre aventure :

- go <place>

Cette commande permet au joueur de se rendre dans la place voulue. Si la commande est tapée sans arguments, un message vous invitera à utiliser la commande “help go” afin d'avoir un exemple d'utilisation de la commande go. Également, si la commande est utilisée avec un nom de place inaccessible, ou avec une faute de frappe dans l'écriture de la place, la liste des sorties accessibles s'affiche à l'écran et vous serez invité à corriger votre entrée. Autrement, vous serez transporté à la place voulue.

- help <commande>

Si une commande est spécifiée après la commande help, alors vous aurez l'aide relative à la commande. Si aucune commande n'est spécifiée alors vous aurez la liste de toutes les commandes disponibles.

- inventory

Cette commande permet d'afficher le contenu de l'inventaire du joueur.

-
- look <item>

Si la commande look est suivie d'un item, vous aurez la description de l'item. Si aucun item n'est spécifié, vous obtiendrez alors la description du lieu dans lequel se trouve le joueur.

- play

Si le joueur est dans la salle d'un mini-jeu et qu'il utilise la commande play, alors il lancera le mini-jeu.

- quit

Quand cette commande est utilisée, en dehors des mini-jeux, elle quitte le jeu.

- take <item>

Il est obligatoire de spécifier un item, si un objet est récupérable par le joueur, alors il est ajouté dans l'inventaire du joueur.

- use <item> *Item*

Un item doit être spécifié, si l'item peut être utilisé alors l'effet de l'item sera appliqué et l'item sera retiré de l'inventaire du joueur. Il est possible de spécifier deux items afin de créer une combinaison d'item (exemple : use gun mag, cette commande chargera le gun avec le mag).

1.4 - Listes des mini-jeux disponibles

TicTacToe

Ce jeu, également appelé Morpion, consiste à aligner trois symboles identiques horizontalement, verticalement ou en diagonale sur une grille de 3x3 cases. On propose d'abord au joueur de choisir son symbole ('X' ou 'O'). L'ordinateur commence à jouer, puis c'est au joueur d'indiquer le numéro de case où il souhaite jouer (les cases sont numérotées de 1 à 9 horizontalement).

Find Number

Vous jouez contre un “Square Guard” qui pense à un nombre entre 0 et 999, et vous avez 10 essais pour le trouver.

Pour jouer, vous devez écrire un nombre compris entre 0 et 999. Le garde vous indiquera si le nombre que vous avez écrit est plus grand ou plus petit que celui auquel il pense.

RockPapersScissors

Vous jouez contre un “Circle Guard” au pierre-feuille-ciseaux en trois manches gagnantes. Pour jouer, vous devez écrire “rock” pour la pierre, “paper” pour le papier et “scissors” pour les ciseaux.

Pour rappel, la pierre bat les ciseaux, les ciseaux battent la feuille et la feuille bat la pierre.

GlassBridge

Ce jeu modélise un pont constitué de dalles en verre. Le but du jeu est de traverser la totalité du pont sans tomber sur une dalle ne supportant pas le poids du joueur. En effet, le pont est constitué de deux dalles en verre. A chaque étape, pour un total de cinq étapes, le joueur doit éviter de se déplacer sur une fausse dalle. Si le joueur marche malencontreusement sur une fausse dalle, cette dernière tombe et le joueur meurt. Le programme demande ainsi au joueur de choisir entre “l”et “r”, respectivement gauche et droite. Tant que le joueur ne marche pas sur une fausse dalle, le chemin parcouru est rappelé et on demande à nouveau le choix du joueur jusqu’à la fin du pont.

MarblesGame

Vous jouez contre le personnage non joueur “Marble Master”. Le dit personnage génère un nombre aléatoire compris entre 1 et 10. L’objectif du jeu est ainsi de deviner, en répondant par “yes” ou “no”, si le nombre généré par le “Marble Master” est pair ou impair. A savoir que vous débutez tous les deux avec dix billes et que le premier de vous deux à ne plus en avoir, a perdu. Chaque bonne réponse vous fait gagner un nombre de billes équivalent au nombre généré par le personnage non joueur, pendant que lui, perd ce même nombre de billes. Inversement, chaque mauvaise réponse vous fait perdre ce nombre de billes et les fait gagner au personnage non joueur.

GuessMyWord

Dans ce jeu, le personnage non joueur “Big Brain” choisit un mot dans une des cinq listes qui lui sont mises à disposition et cela à trois reprises. Chaque liste de mots contient un total de cinq mots dedans. Les mots contenus dans la liste choisie par le personnage non joueur seront affichés à l’écran. Il faut réussir à obtenir une bonne réponse a minima, en piochant parmi les mots affichés à l’écran, afin de réussir à ce mini jeu. Ce mini-jeu a été conçu de manière à éviter toute redondance dans le choix des listes de mots, ainsi, dans chaque partie, les trois listes seront strictement différentes.

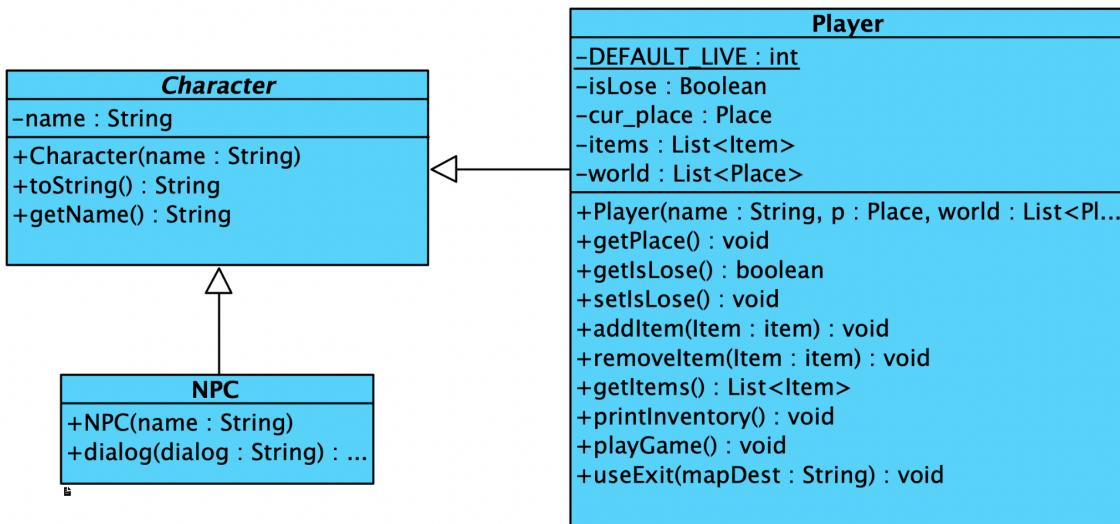
II - Documentation développeur

2.1 - Les différents packages

- squid.character :

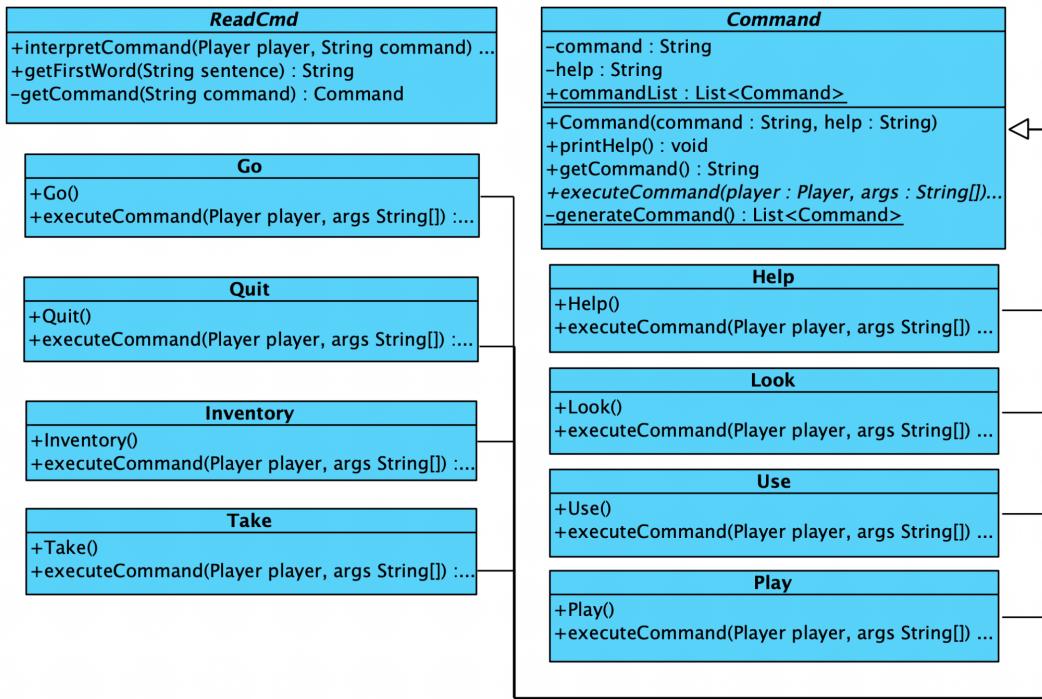
Le package character contient 3 classes, la classe abstraite Character contient toutes les informations de base des personnages (Joueur et NPC). La classe NPC possède la méthode dialog() qui permet à un NPC d'afficher son nom et une phrase. Puis la classe Player, une des classes le plus importantes du projet car elle permet de gérer toutes les actions relatives au joueur.

Les classes Player et NPC héritent de la classe Character. Dans le jeu il n'y aura qu'une seule instance de Player alors que chaque Place sera liée à une instance d'un NPC.



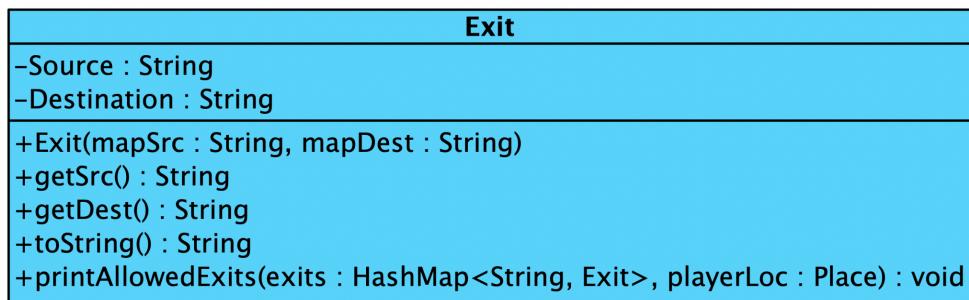
- squid.command :

Le package command contient 10 classes, la classe abstraite command contient la méthode qui permet d'exécuter une commande. On utilise une méthode abstraite car elle sera définie par chaque sous-classe de command.



- squid.exit :

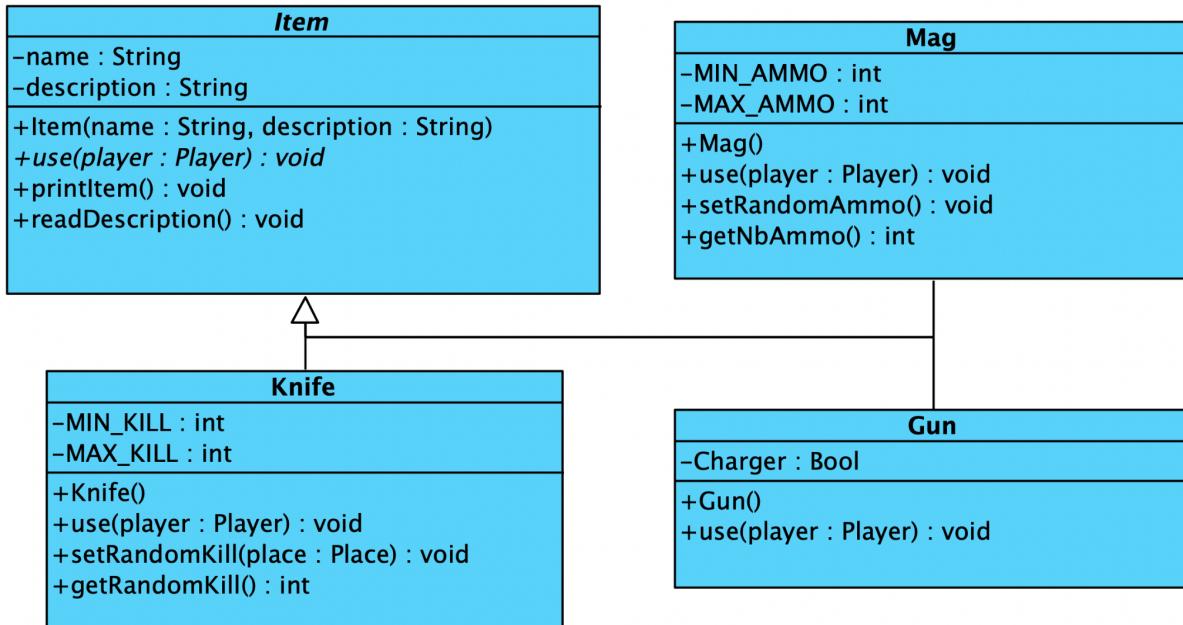
Le package exit contient une classe explicite définissant la conception des sorties dans le jeu. Le package inclut un ensemble de fonctions de traitement afin de permettre l'interaction avec le modèle imposé de gestion des sorties d'un lieu.



- squid.item :

Le package item contient un ensemble de cinq classes. Une classe abstraite, Item, définissant le principe de conception propre à tous les items ainsi que toutes les méthodes permettant la gestion ainsi que les interactions avec les autres classes de

ladite classe. Les quatre autres classes du package, définies de façon explicite, représentent les quatres possibilités d'items décidés à terme de la conception de notre jeu. Les dits items sont les suivants : Knife, Gun, Mag et Empty.



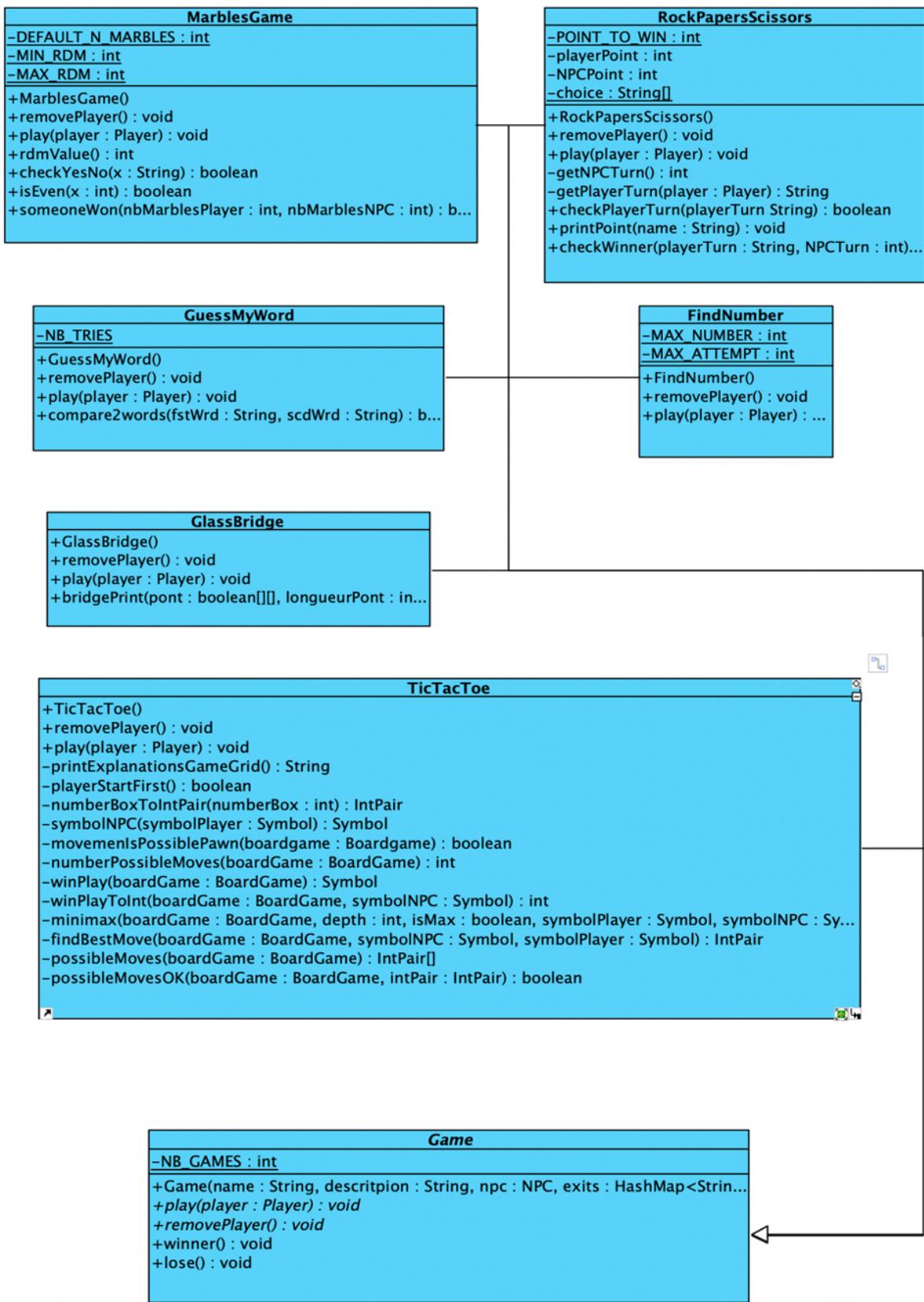
- squid.place :

Le package place est la pièce maîtresse du jeu. Elle inclut le package squid.place.game, définis ci-dessous, la classe explicite Place ainsi que la classe abstraite Game. La classe Place définit la conception de tous les lieux du jeu et inclut les fonctions d'interaction, de gestion et celles permettant ainsi de générer l'univers du jeu dans son ensemble. La classe Game quant à elle a été conçue de manière à établir une distinction concrète entre la “Main Room”, lieu principal sans mini jeu, et le reste des lieux accessibles au joueur, avec des minis jeux. Le contenu de cette dernière est détaillée davantage ci-dessous. Cet ensemble d’éléments de conception permettent ainsi au joueur d’entrer dans un environnement préconçu selon la volonté des développeurs.

Place
<pre> -nbPlayer : int -name : String -description : String -npc : NPC -exits : HashMap<String, Exit> +Place(placeName : String, placeDescription : String, myNPC : NPC, exits : HashMap<String, Exit>) +getName() : String +getDescription() : void +getNpc() : NPC +getNbPlayer() : int +setNbPlayer(nbPlayer : int) : void +genAllPlaces() : List<Place> +findPlace(placeName : String, map : List<Place>) +placeExits(placeName : String, exits : HashMap<String, Exit>) +getExits() : HashMap<String, Exit> +allExits() : Collection<Exit> +genExit2Ways(exits : HashMap<String, Exit>) </pre>

- squid.place.game :

Le package place.game contient tous les mini-jeux qui héritent chacun de la classe Place. Elle inclut le package TicTacToe qui a nécessité plusieurs sous-classes ainsi que le package GuessMyWord également. Si vous voulez ajouter un jeu, il faut juste créer une sous-classe de Game, définir la méthode play et de l'ajouter à la méthode genAllPlaces() de la classe Place.



- squid :

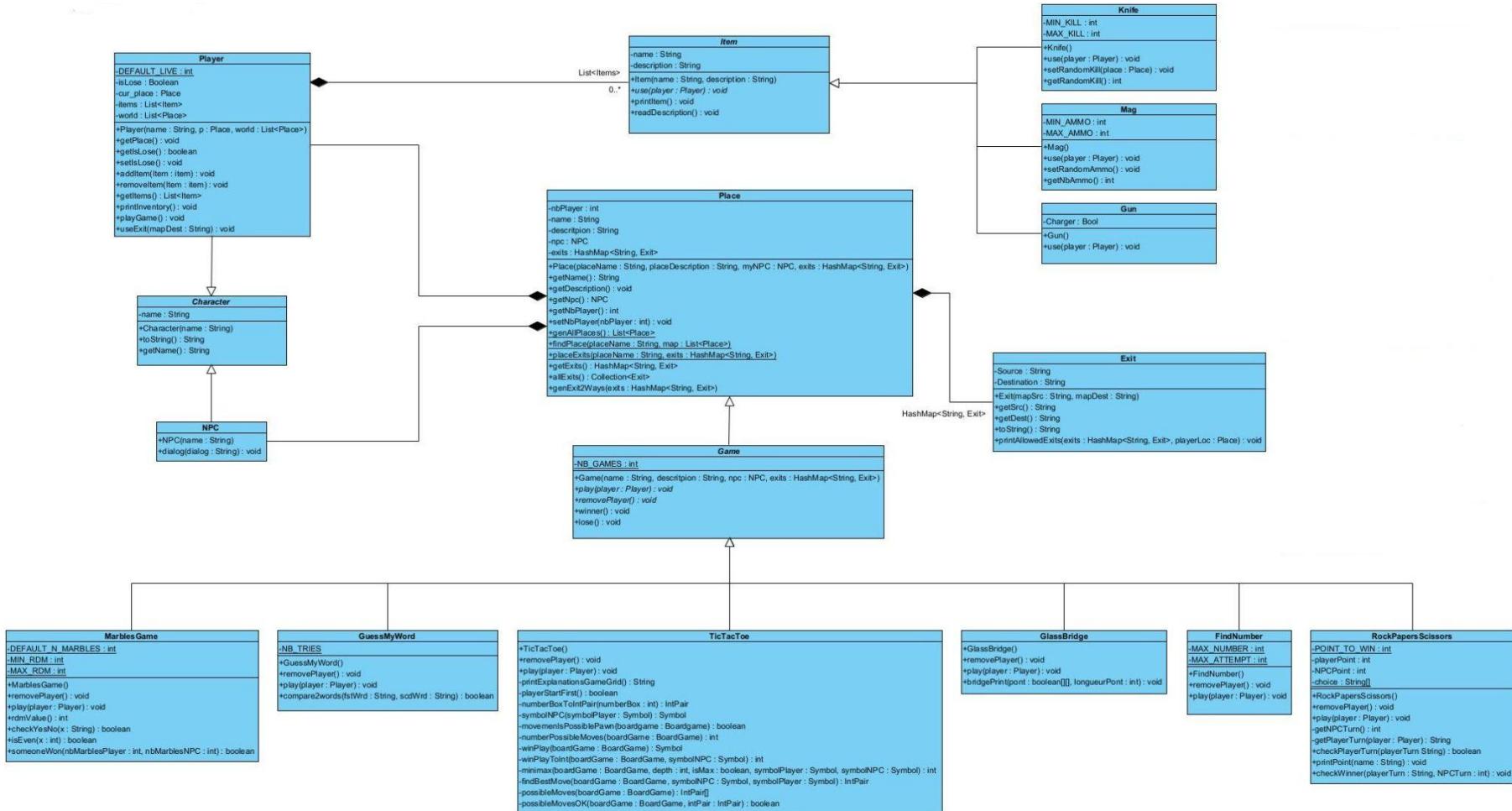
Le package contient la classe principale Play, cette classe possède une méthode main() qui est appelée quand le programme est lancé. La classe possède aussi un attribut qui est une instance de Scanner sur l'entrée standard.

- test :

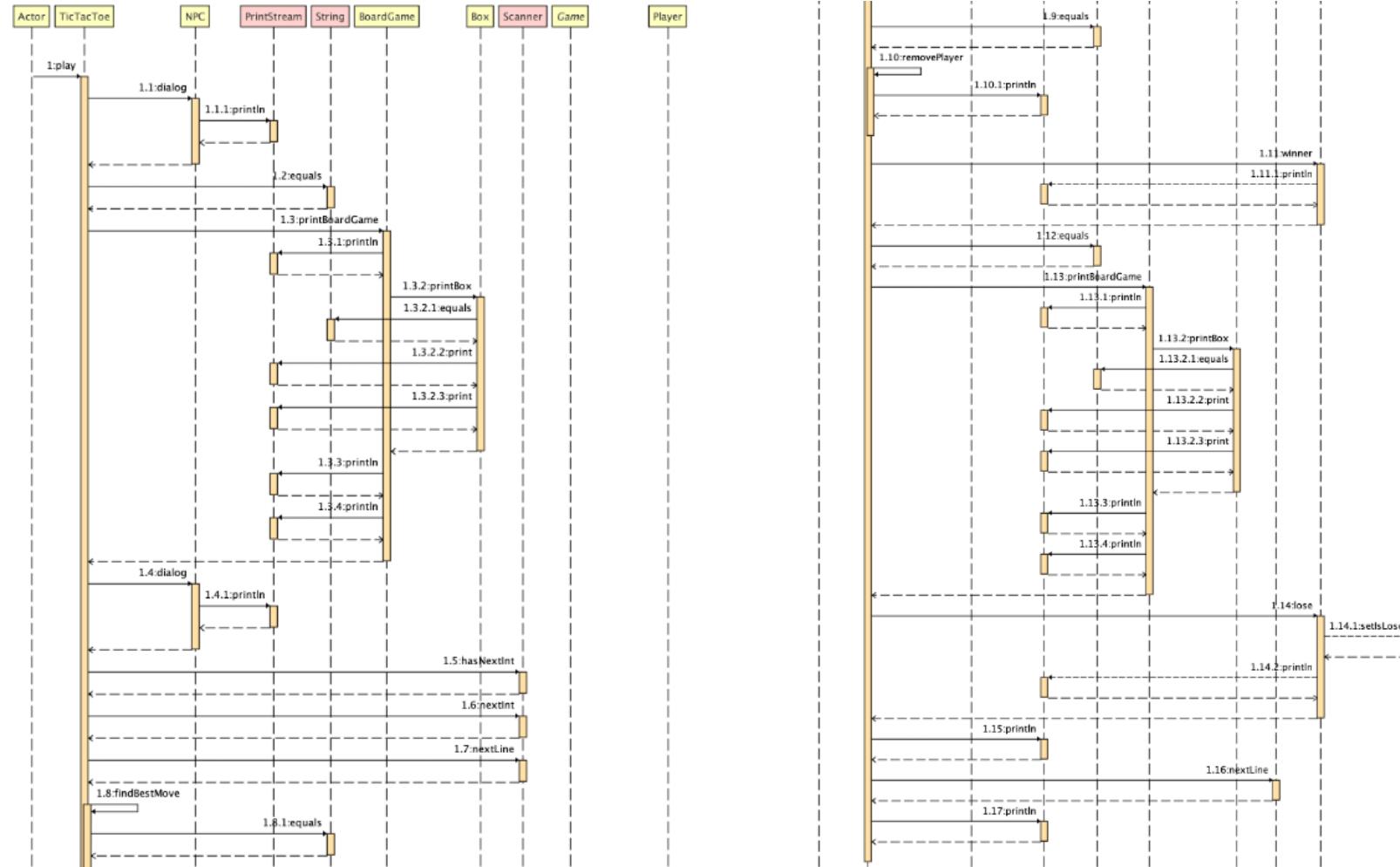
Ce package contient les différents tests unitaires qui ont été effectués sur certaines classes. Les tests sont rangés dans des sous-packages qui correspondent aux sous-packages des classes testées. Ainsi, on peut exécuter un test unitaire sur une fonction en particulier ou alors en tester plusieurs à la fois.

2.2 - Diagramme de classe

La préconception du projet a été faite en premier lieu. Cependant, au fur et à mesure du développement, cette conception a été modifiée afin de la faire correspondre aux nouvelles nécessités rencontrées par l'implantation de nouveaux éléments de jeu dans notre projet. De manière à avoir une vision globale de la conception du jeu, vous trouverez ci-dessous, l'ensemble du diagramme de classe, avec les différents liens entre les packages.



2.3 - Diagramme d'états et de séquences



Ce diagramme de séquence correspond à la commande `Play()` du jeu TicTacToe.

III - Organisation du groupe

3.1 La préconception

Le groupe initial était constitué de Clément et Alessandro. Ce n'est que par la suite que Souhaïl s'est joint à nous. Afin d'avoir une première idée sur le travail à réaliser pour mener à bout notre projet, nous nous sommes premièrement accordés sur la conception du jeu. L'idée, proposée par Clément, a immédiatement fait l'unanimité. Une fois le concept de jeu décidé, nous avons formulé les axes principaux de fonctionnement du jeu.

3.2 - Travail de groupe et organisation du travail

De manière à identifier la charge de travail nécessaire à la réalisation du projet, nous nous sommes concertés afin d'établir les différentes nécessités pour la réalisation du jeu. En découle alors de cet entretien, un premier diagramme de classe. Ce diagramme de classe nous a permis d'avoir une première idée des différents éléments de conception nécessaires pour mener à bien notre projet. Ayant connaissances des différents éléments intrinsèques à la conception de notre jeu, la répartition des tâches s'est alors faite de façon naturelle et équivalente.

Afin de mener à bien ce projet, nous avons décidé de travailler sur le même environnement de développement, à savoir IntelliJ Idea. Pour la communication, nous avons utilisé l'outil Discord. En ce qui concerne les fichiers de code, nous avons utilisé GitHub. Par ailleurs, le fait que IntelliJ intègre nativement des fonctionnalités de Git nous a facilité la mise en commun de nos codes.

3.3 - Répartition des tâches

3.3.1 - Ghita Alessandro

Alessandro s'est occupé initialement de la création de la classe Place, puis après de deux mini-jeux : Glass Bridge et TicTacToe. Il a, par la suite, pris en charge la conception du package Item dans son entiereté. Il a ainsi conçu l'ensemble des "Items"

disponibles dans le jeu, a l'exception du Empty. Alessandro a également pris en charge la conception de la commande Use.

3.3.2 - Benmoussati Souhail

Souhaïl s'est aussi initialement occupé de la création de la classe Place, avant de prendre en charge la création de la classe Exit. S'en est suivi de l'implantation de la Hashmap faisant la jonction entre les Exits et les Places et des fonctions d'utilisations de cette hashmap. Il a par la suite conçu deux minis-jeux : Marbles Game et Guess My Word, avant de prendre en charge la conception des commandes Go et Take, avec l'implantation de l'Item Empty.

3.3.3 - Gelinaud Clément

Clément s'est occupé d'une part de la conception des jeux FindNumber et RockPaperScissors, mais aussi d'autre part de la conception de l'ensemble du package Character, comportant à la fois la classe Player et la classe NPC. Il s'est par la suite chargé d'écrire les codes du restant des commandes non pris en charge par ses collaborateurs, à savoir Help, Inventory, Look, Play ainsi que Quit.

- Conclusion

Pour conclure, au sein de ce groupe, chacun y a mis du sien. Nous nous sommes tous aidés les uns les autres pour résoudre l'ensemble des problèmes rencontrés. Il y a eu un vrai travail de communication au sein du groupe dont la résultante est le fait que la répartition de la charge de travail au sein du groupe était au final équilibrée. Chaque décision impliquant une modification du modèle de conception propre au jeu, s'est faite après concertation et accord de l'ensemble des membres du groupe. Nous avons utilisé nos connaissances et expériences respectives afin de mener à bien ce projet. C'est ainsi avec plaisir que nous vous présentons notre projet de groupe qui représente, par la même occasion, la conception de notre premier jeu. Nous osons espérer que vous saurez apprécier les efforts réalisés pour mener à bout ce projet.

Nous vous souhaitons ainsi une agréable expérience de jeu.