[POO] - Projet
Gypsy’s Carnival
--
Rapport

Documentation utilisateur

Comment installer et lancer le jeu ?

Pour lancer le jeu, il faudra au préalable télécharger et installer sur votre ordinateur :

- Java disponible à l'adresse ci-dessous : https://www.java.com/fr/download/help/download_options.html

- Java SE disponible à l'adresse ci-dessous : https://www.oracle.com/java/technologies/java-se-glance.html

Ensuite, il faudra lancer le jeu avec un interpréteur de commande, pour cela :

- Déplacez-vous dans le répertoire possédant le fichier "gypsy's_carnival.jar"

- Une fois dans ce répertoire lancer la commande "java -jar gypsy's_carnival.jar"

Comment joueur ?

Quand vous lancez le jeu, vous arrivez dans le carnaval et vous pouvez vous déplacer vers une place de cuivre, d'or ou de platine où se trouve respectivement des mini-jeux du niveau de la place.
Il y a donc 3 niveaux qui représentent la difficulté des mini-jeux présents dans ces places, le niveau de cuivre étant le plus facile et le niveau de platine étant le plus difficile, celui d'or est donc entre les deux.

Quand vous êtes dans le carnaval vous pouvez aussi accéder à deux magasins, l'un vendant de la nourriture et l'autre vendant des clefs. Ces deux types d'objets sont indispensables pendant votre aventure.
En effet, chaque mini-jeu est au début verrouillé et il vous faudra acheter une clef correspondant au niveau du mini-jeu pour déverrouiller celui-ci. Le prix de ces clefs

équivaut au niveau de celle-ci, ainsi les clefs de cuivres sont moins chères que les clefs d'or qui sont elles-mêmes moins chères que les clefs de platine.
Enfin, la nourriture vous permettra de regagner des calories qui représentent votre niveau de santé.

Au début du jeu vous possédez 50 pièces de monnaie et 100 calories, chaque mini-jeu vous fait gagner une certaine quantité de pièces aléatoire qui récompense le joueur selon le niveau du mini-jeu, vous obtenez donc plus d'argent en gagnant un mini-jeu d'or plutôt qu'un mini-jeu de cuivre.
Cependant, attention à ne pas perdre à un de ces mini-jeu, puisque si cela arrive, vous perdez des calories, et si par malheur vous n'avez plus de calories, le jeu se termine et vous devez recommencer depuis le début.

Un dernier lieu accessible à partir du carnaval est celui de la caravane scintillante qui est verrouillée durant toute votre aventure. La seule façon de déverrouiller ce lieu est de finir chacun des neuf mini-jeux au moins une fois.
Ce dernier lieu est le but de votre aventure, ainsi une fois à l'intérieur, vous pourrez finir le jeu.

Pour résumer le but du jeu est de finir tous les mini-jeux disponibles tout en gérant vos calories qui descendent si vous perdez un de ces mini-jeu.

Bonne chance !

Quelles sont les commandes ?

Ce jeu est en ligne de commande, vous devez interagir avec lui à l'aide de la liste de commandes que vous pouvez trouver ci-dessous (aucune commande n'est sensible à la casse, ainsi vous pouvez aussi bien écrire en majuscules qu'en minuscules) :

- go *location*

Si le lieu *location* est spécifié, alors cette commande vous permettra de rentrer dans le lieu si celui-ci est accessible.

Si aucun lieu n'est spécifié, alors cette commande affiche la liste des lieux qui sont voisins au lieu actuel du joueur.

- help *command*

Si la commande *command* est spécifiée, alors une description de la commande spécifiée est affichée à l'écran.
Si aucune commande n'est spécifiée, alors une liste de toutes les commandes disponibles est affichée.

- look *object*

Si l'objet *object* est spécifié, alors une description de l'objet est affichée à l'écran.
Si aucun objet n'est spécifié, alors une description du lieu où se trouve le joueur est affichée à l'écran.

- take *object*

Un objet *object* doit être spécifié, si cet objet peut être pris (le joueur est dans un magasin et possède assez d'argent) alors l'objet est ajouté dans l'inventaire du joueur et l'argent que coûte cet objet est retiré au joueur.

- quit

Quitte le jeu.

- use *object*

Un objet *object* doit être spécifié, si cet objet peut être utilisé alors l'objet est retiré de l'inventaire du joueur.

- play

Si le joueur est dans un lieu qui contient un mini-jeu et qu'il lance cette commande alors il lance le jeu. Durant la partie où le joueur joue à un mini-jeu, les commandes normales ne sont plus accessibles et seules les commandes spécifiques au jeu peuvent être lancées. Ces commandes spécifiques sont indiquées en début de partie de chaque mini-jeu.

- inventory

Affiche l'inventaire du joueur, ainsi que son nombre de calories, sa quantité d'argent et le nombre de jeux qu'il a terminé au moins une fois.

- unlock *location*

Un lieu *location* doit être spécifié, si ce lieu n'est pas déjà déverrouillé et que le joueur a de quoi le déverrouiller, alors le lieu se déverrouille et devient accessible.

Comment jouer à un mini-jeu ?

Une fois que vous avez déverrouillé un mini-jeu, vous pouvez rentrer dans le lieu du mini-jeu où le personnage qui tient le stand vous attend pour vous expliquer les règles du jeu et à l'affronter ou l'aider. Il y a neuf mini-jeux disponibles et chacun à ses spécificités expliqués ci-dessous :

- Trouve le nombre (niveau de cuivre)

Vous êtes en face de Vincent Faygaf qui pense à un nombre entre 0 et 999, vous avez 10 essais pour le trouver.
Pour jouer, vous devez écrire un nombre entre 0 et 999 et Vincent vous dira si le nombre qu'il a pensé est plus grand ou plus petit que celui que vous venez d'écrire.

- Chifoumi (niveau de cuivre)

Vous défiez Pierre Dupuis au chifoumi (pierre-feuille-ciseaux) en 3 manches gagnantes. Pour rappel, la pierre bat les ciseaux , les ciseaux battent la feuille et la feuille bat la pierre.
Pour jouer, vous devez juste taper “rock” pour pierre, “paper” pour feuille et "scissors" pour ciseaux.

- La bonne touche (niveau de cuivre)

Ethoufet Kwallah est un rappeur connu pour sa rapidité, il vous défie dans une joute verbale en 3 phrases pour tester votre rapidité.
Pour jouer, vous devez recopier la phrase qu’a dite Ethoufet après un compte à rebours de 3 secondes. Vous devez faire cela en un temps imparti, c’est-à-dire 20 secondes pour la première phrase, 15 secondes pour la deuxième et 10 pour la dernière.

- La tour d’Hanoï (niveau d’or)

Le vétéran Edwardo Nald est bien embêté devant ce défi qu’est la tour d’Hanoï, il a besoin de votre aide pour résoudre ce problème. Le principe est de déplacer trois disques du pilier A vers le pilier C en vous aidant du pilier B, cependant vous ne devez pas poser un disque plus grand sur un disque plus petit (le disque 3 étant le plus grand et le disque 1 le plus petit).
Pour jouer, vous devez écrire le pilier de départ du disque que vous voulez déplacer puis le pilier d'arriver ou vous voulez poser le disque. Par exemple, pour déplacer le disque en haut du pilier A vers le pilier C, vous devez écrire “a c”.

- L’énigme (niveau d’or)

Le sage et vieillot Jean-Pierre Fougas veut vous piéger par delà ces énigmes. Il vous donne trois chances pour arriver au bout d’une de ses énigmes.

Pour jouer, vous devez simplement écrire la réponse que vous pensez être correct.

- Le morpion (niveau d'or)

Le morpion est la spécialité de Sagrat Opubice qui vous défie le temps d'une partie. Le jeu se joue sur une matrice carrée de taille trois et le gagnant est celui qui arrive à aligner ses trois pions soit à la verticale, soit à l'horizontale ou en diagonale.
Pour jouer, vous devez écrire deux nombres entre 1 et 3, le premier représente la ligne et le deuxième la colonne. La première ligne et première colonne se situe en haut à gauche de la matrice, ainsi si vous écrivez "1 1" le pion se met en haut à gauche.

- Le pendu (niveau de platine)

Marina Lependu, grande amatrice de fruits et légumes, veut jouer au pendu avec vous. Le principe du jeu est de trouver, lettre par lettre, en cinq tentatives le fruit ou légumes auquel elle a pensé.

Pour jouer, vous devez simplement écrire la lettre à laquelle vous pensez, si celle-ci est bonne, sa position est affichée dans le mot, sinon vous perdez une tentative.

- Les questions (niveau de platine)

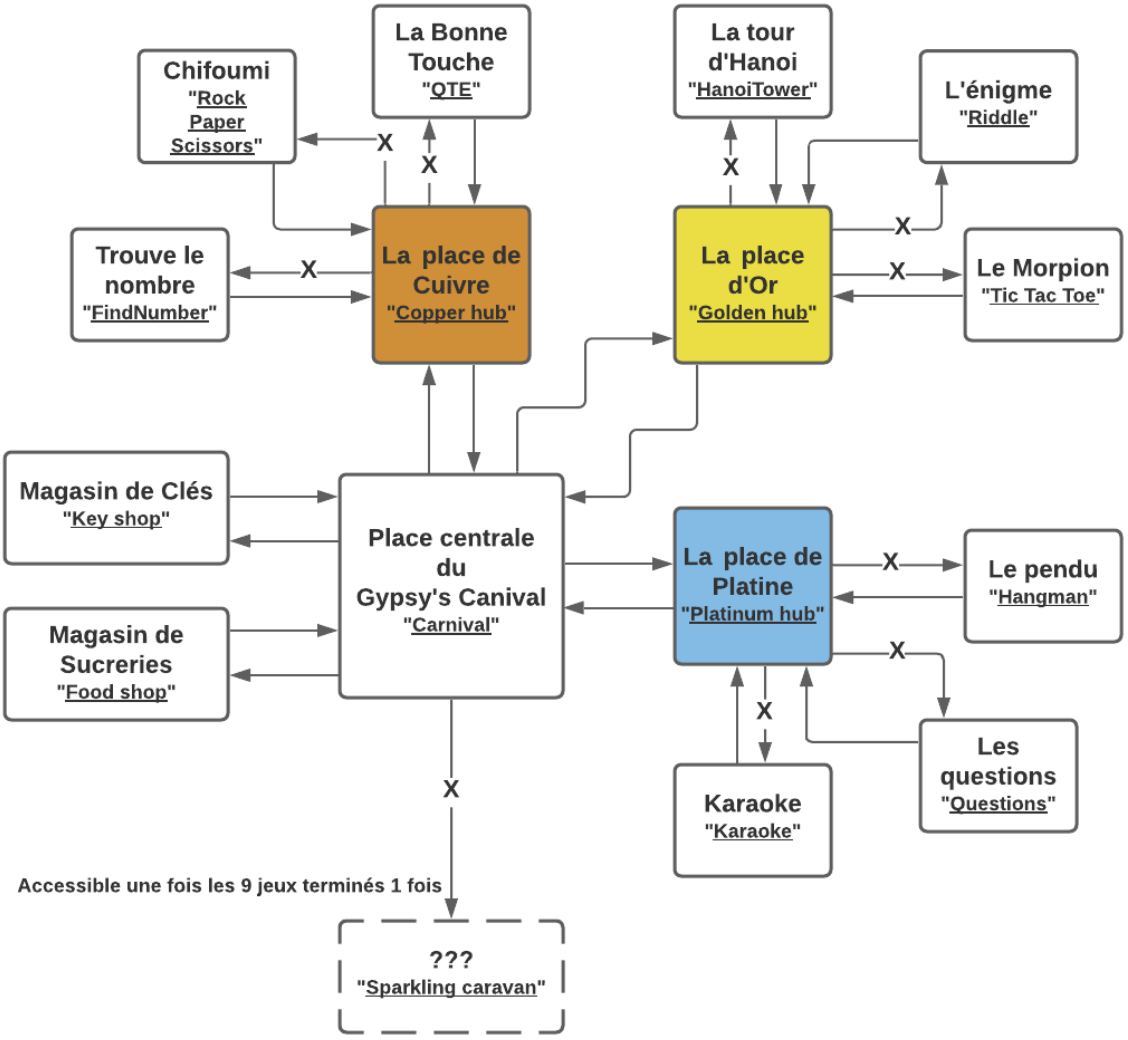
Pour tester votre culture générale, Samuel Outienne vous pose cinq questions qui ont chacune quatre réponses. Vous devez trouver la bonne réponse pour chaque question et avant de passer la question suivante, vous pouvez soit terminer en remportant l'argent que vous avez gagné jusque-là, soit continuer jusqu'à la fin pour gagner le jackpot.
Pour jouer, vous devez écrire le numéro correspondant à la réponse que vous jugerez correcte. Les réponses sont numérotées de 1 à 4, donc si vous pensez que c'est la troisième question qui est correcte, tapez simplement "3".

- Karaoké (niveau de platine)

Cette fois-ci, c'est votre culture musicale qui est testée par Kharra Okey qui vous laisse 3 essais pour deviner les paroles manquantes d'une phrase provenant d'une musique connue.
Pour jouer, vous devez simplement écrire le ou les mot(s) manquant.

La carte des lieux

Pour résumer les différents lieux accessibles pendant le jeu, un plan des lieux a été créé avec les sorties entre chaque lieu. Les flèches normales représentent les sorties déverrouillées, tandis que les flèches avec une croix représentent les sorties verrouillées.



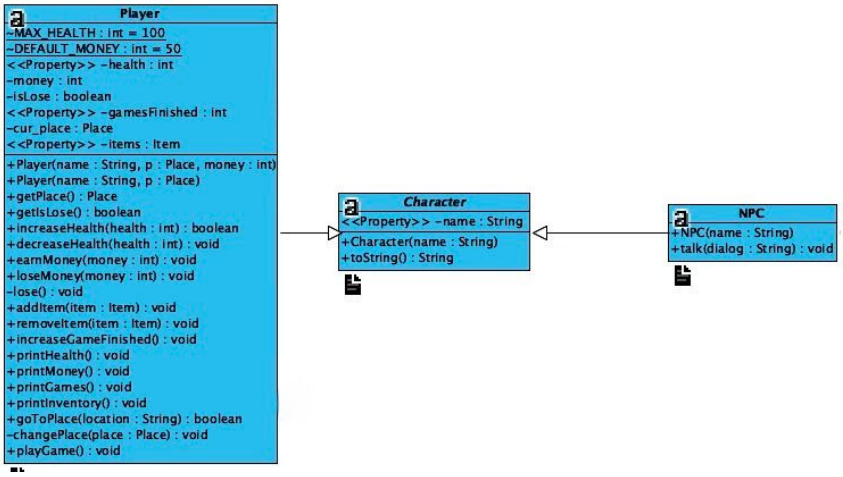
Documentation développeur

Les différents packages

- code.character

Ce package contient 3 classes, la classe abstraite *Character* qui contient la base de tous personnages (joueur ou non). Puis de la classe *NPC* qui contient la méthode *talk()* permettant à un npc d'afficher une phrase avec son nom. Et enfin la classe *Player* qui est l'une des classes principales du projet car c'est elle qui va, via ses méthodes, gérer l'intégralité des actions directes et indirectes du joueur.

Ces 2 dernières classes héritent de *Character*, mais ont un rôle différent. Une instance de *NPC* sera reliée à une seule et unique instance de *Place*. La seule instance de *Player* est, quant à elle, reliée à une instance de *Place* temporairement car le joueur peut se déplacer de lieu en lieu. Mais aussi interagir via les différentes commandes qui lui sont à disposition.

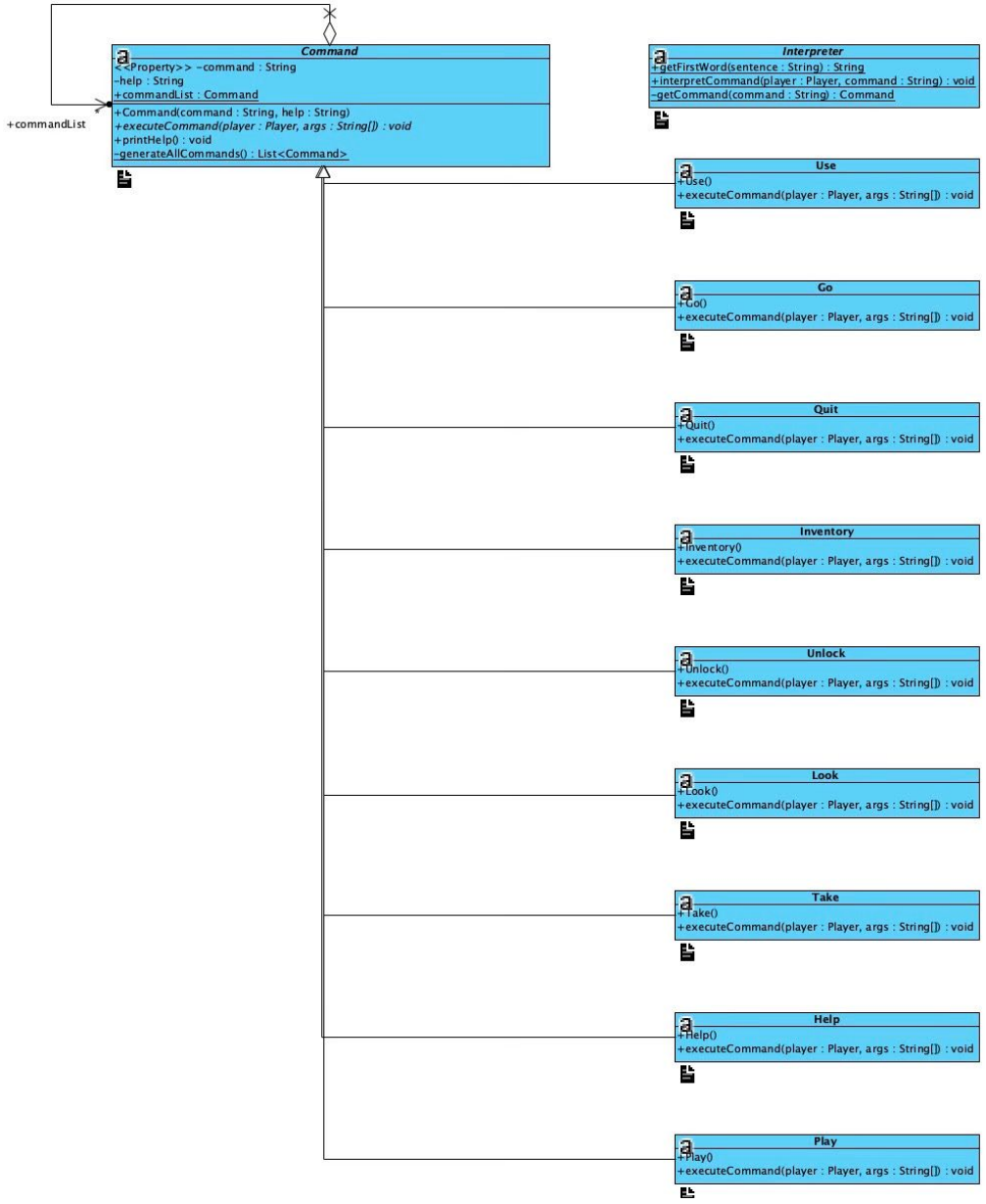


- code.command

Ce package contient une classe abstraite principale *Command* qui contient quelques méthodes, dont celle qui permet d'exécuter la commande. Cette méthode est abstraite et doit être définie par chaque sous-classe de *Command*. Cette classe a pour attribut de classe une liste de *Command* générées par une méthode privée où chaque sous-classe de *Command* est instanciée une fois.

Ces sous-classes de *Command* permettent de définir l'exécution des commandes listées dans Quelles sont les commandes ?. Si vous voulez rajouter une commande, alors il faut juste créer une sous-classe de *Command*, faire un constructeur avec le nom de la commande et son aide, puis définir la méthode *executeCommand()* pour finalement l'instancier dans *generateAllCommands()*.

Il y a aussi une classe abstraite *Interpreter* qui permet d'exécuter une chaîne de caractère rentrée par le joueur au clavier si elle existe, et possède une méthode *getFirstWord()* qui retourne le premier mot d'une chaîne de caractère.

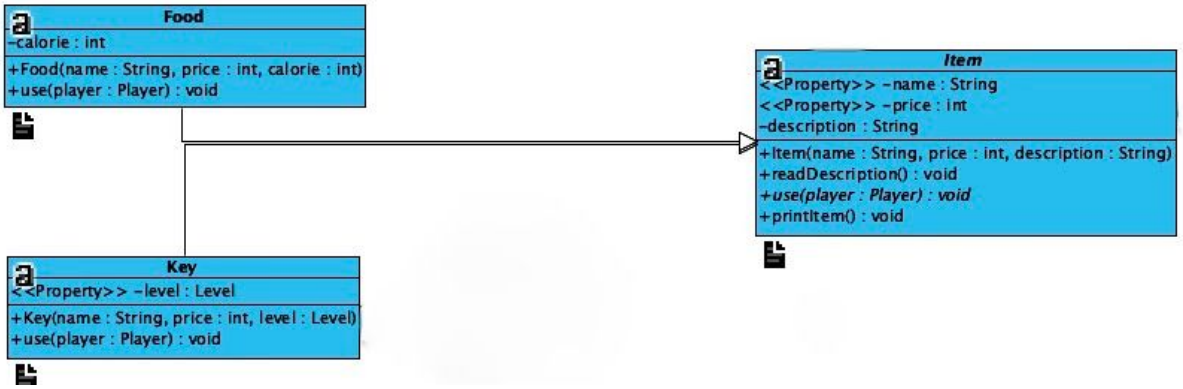


- code.item

Ce package a pour classe principale *Item* qui est abstraite et implémente l'interface *Describable*. Chaque item a un nom, une description et un prix et une méthode abstraite *use()* qui doit être définie par ses sous-classes.

Deux sous-classes sont déjà définies, la sous-classe *Key* et la sous-classe *Food*. Pour cette première sous-classe, un niveau doit être spécifié et la méthode *use()* sert à déverrouiller des mini-jeux du même niveau que la clef. Pour la deuxième sous-classe, un nombre de calories doit être défini pour l'ajouter au joueur lorsqu'il utilise une instance de cette classe.

Pour créer votre propre sous-classe d'*Item*, il vous faudra donc juste redéfinir la méthode *use()* et ajouter des attributs si nécessaire.



- code.place

Ce package a pour classe principale *Place* qui implémente l'interface *Describable*. Une instance de *Place* possède un nom, une description, un personnage et une liste de sorties. La méthode statique *generateAllPlaces()* permet de générer une liste contenant tous les lieux du jeu (La carte des lieux). Cette fonction gère l'affectation des sorties selon le type de l'instance du lieu, ainsi si vous voulez ajouter un lieu vous n'avez qu'à ajouter ce lieu dans la liste *placeList* avant l'appel à *Exit.generateAllExits(placeList)*.

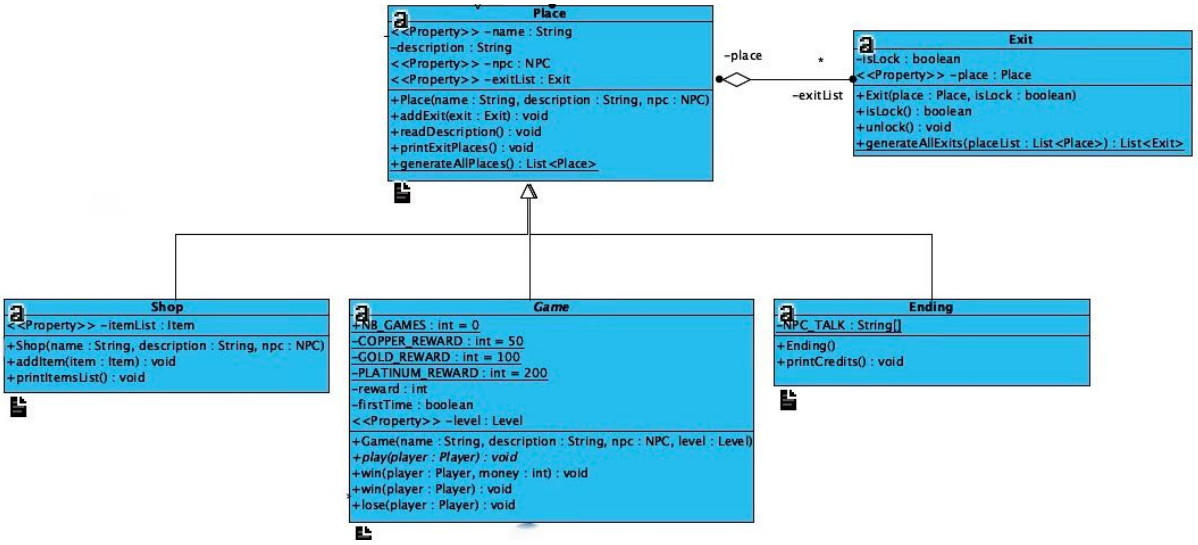
Trois classes héritent directement de *Place*, les sous-classes *Shop*, *Ending*, et *Game*. La classe *Shop* possède une liste d'items que le joueur peut acheter. La classe *Ending* n'est instancié qu'une seule fois et représente le lieu final du jeu.

La classe *Game* est une classe abstraite avec un niveau et une méthode *win()* qui adapte le gain d'argent en fonction du niveau du mini-jeu. Cette classe possède une méthode abstraite *play()* qui est appelée lorsque le joueur lance la commande "play" lorsqu'il se trouve dans un mini-jeu.

- code.place.exit

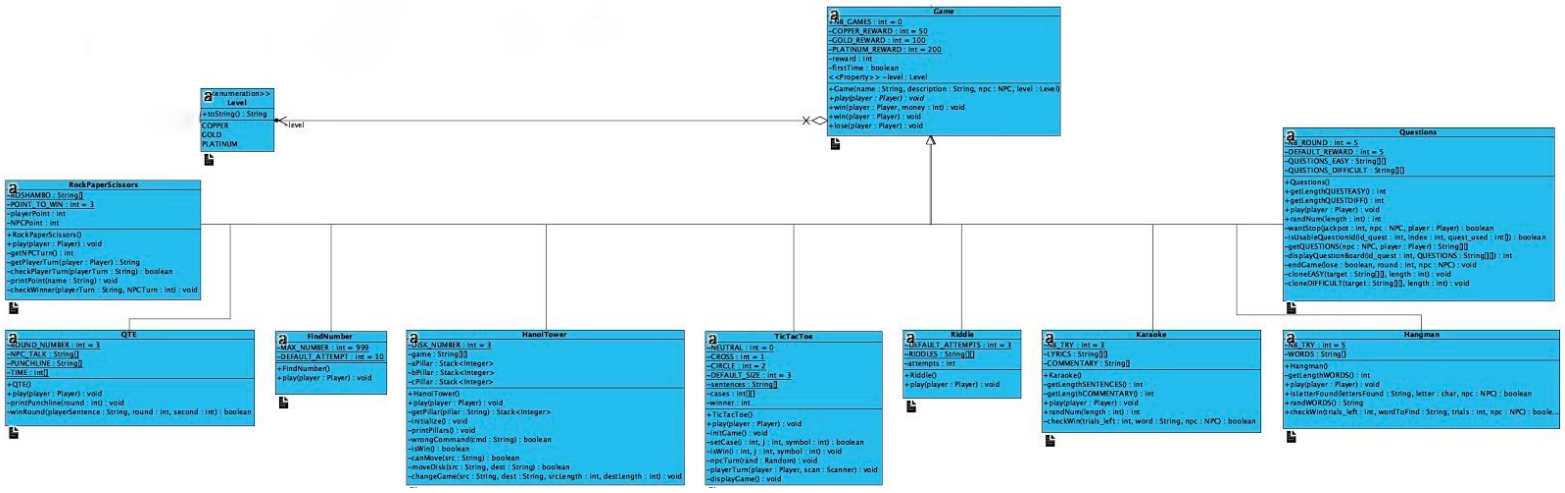
Le sous-package code.place.exit, contient une seule et unique classe *Exit*, qui contient le lieu vers lequel la sortie mène et un booléen pour savoir si la sortie est verrouillée ou non. La méthode statique *generateAllExits()* qui crée les sorties de chaque place de la liste qui lui est fournie en paramètre.

Vous pouvez créer des sous-classes de Exit si vous voulez faire d'autres types de sorties.



- code.place.game

C'est dans ce sous-package que se trouvent les neuf mini-jeux qui ont chacun une classe qui hérite de la classe *Game*. Chaque mini-jeu a donc sa propre implémentation de la méthode play. Ainsi, si vous voulez ajouter un autre mini-jeu, il faut créer une sous-classe de *Game* et définir la méthode play selon le principe de votre mini-jeu.



- code

Le package principal contient l'interface *Describable*, l'énumération *Level* et la classe principale *Gameplay*.

L'interface *Describable* est implémentée dans une classe pour lire la description de celle-ci, notamment avec la commande "look" (cf. Quelles sont les commandes ?). L'énumération *Level* est utile pour désigner les trois niveaux des mini-jeux et donc des clefs pour les déverrouiller.

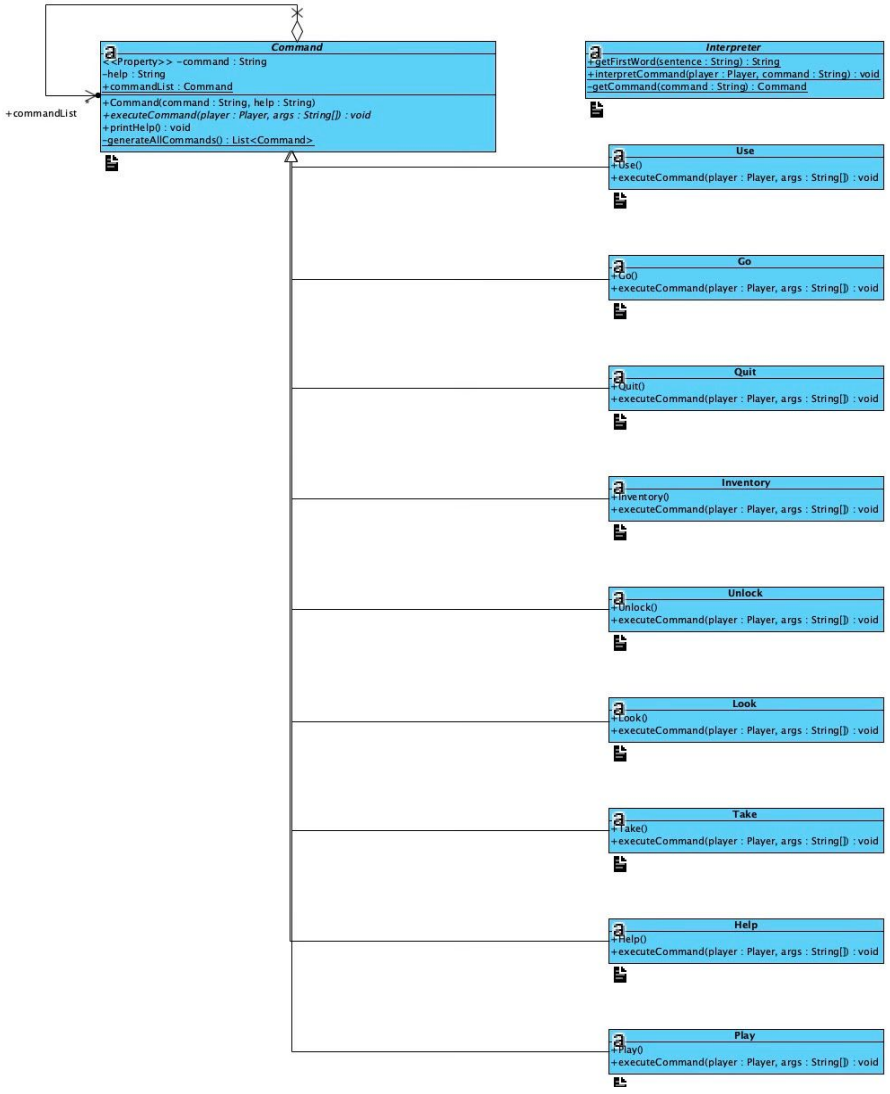
La classe *Gameplay* contient la méthode *main()* qui est appelée lorsque le programme est lancé. Cette classe a également un attribut statique qui est une instance de *Scanner* sur l'entrée standard puisque chaque jeu a des commandes différentes.

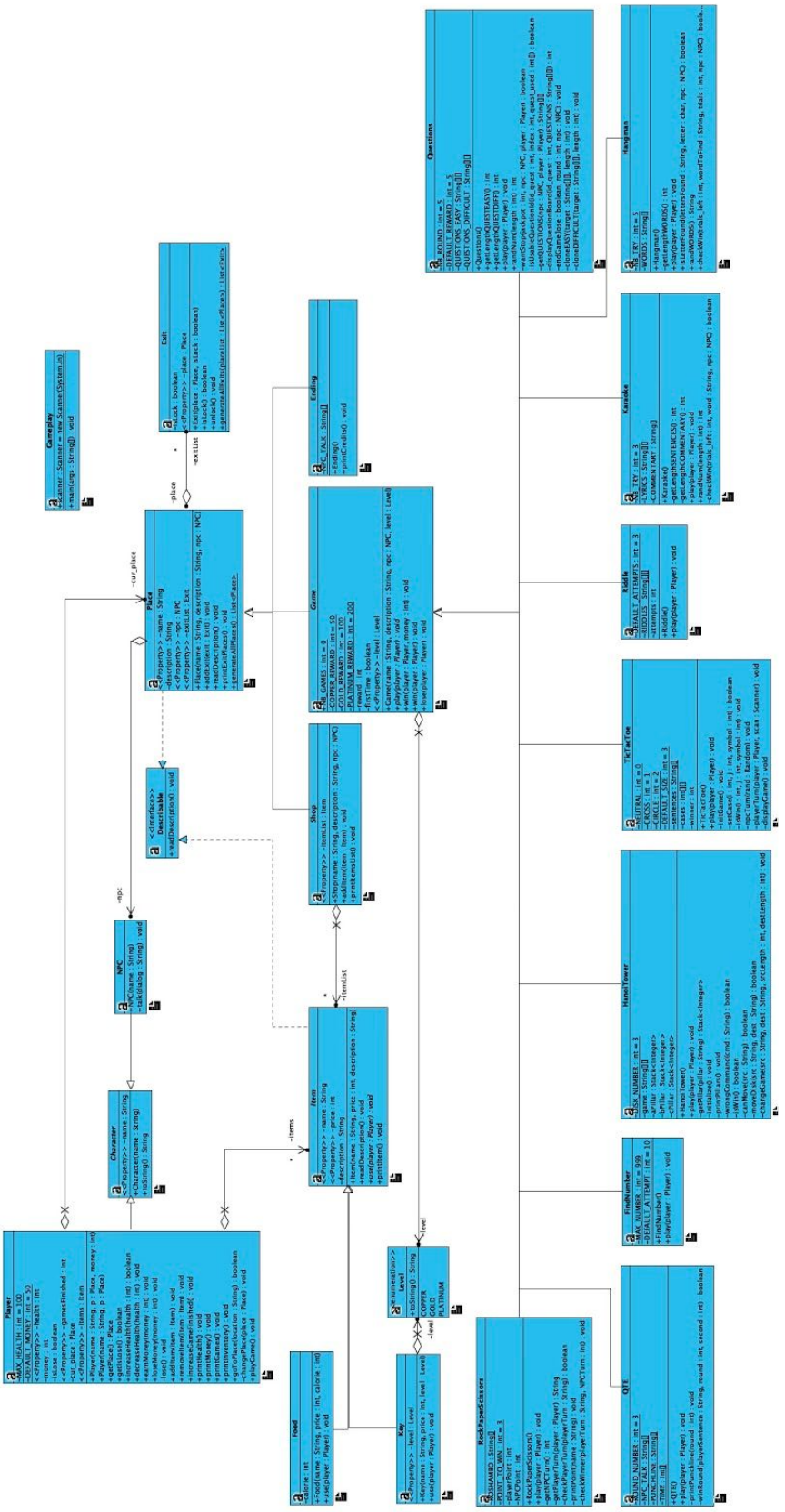
- test

Ce package implémente les différents tests unitaires qui ont été effectués sur certaines classes. Ainsi, vous pouvez rajouter des tests ou bien exécuter ceux déjà écrits grâce à la classe *TestAll* qui lance tous les tests unitaires. Ces tests ont été rangés dans des sous-packages correspondant aux sous-packages où la classe testée se trouve.

Le diagramme de classe

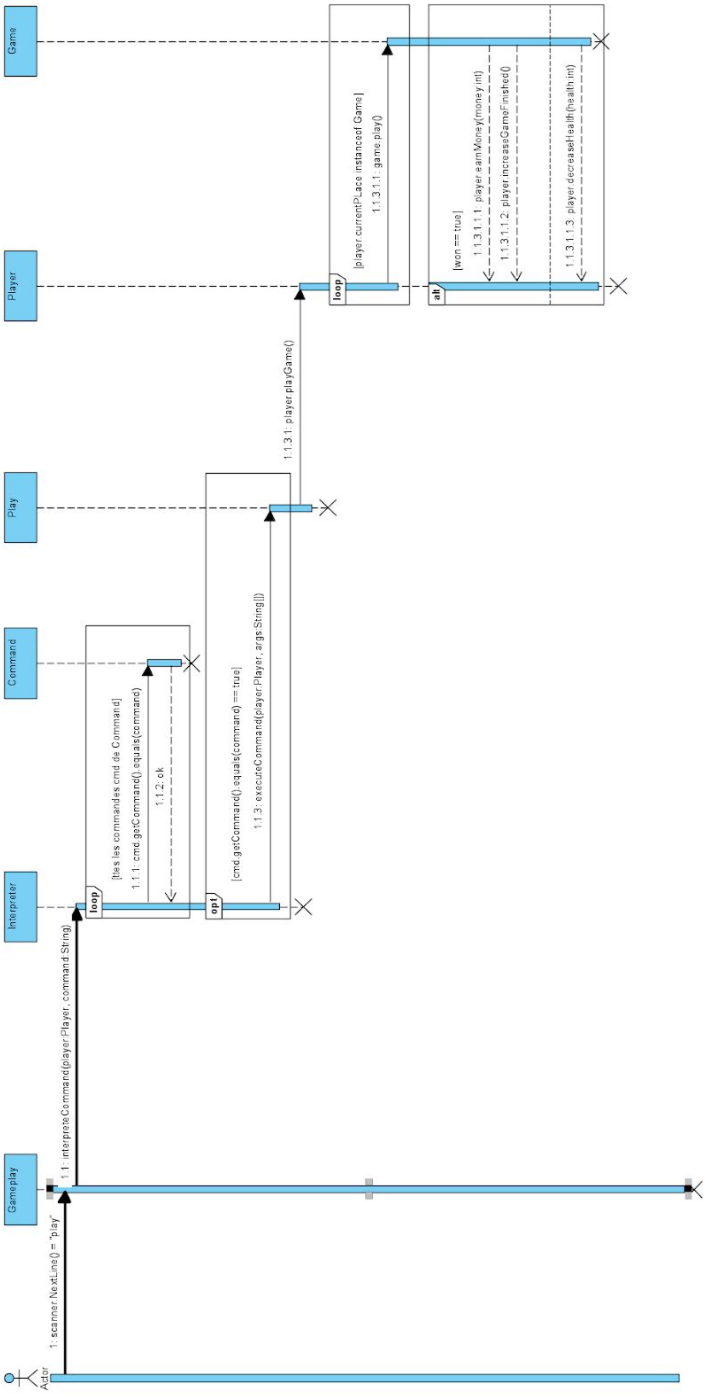
La préconception du projet a été faite dès le début, avec une idée de départ. Puis au fur et à mesure de l'avancement de celui-ci, nous l'avons modifiée pour qu'elle puisse respecter ce que nous voulions, ou pour régler certains problèmes rencontrés en cours de route.
Pour ainsi mieux voir la manière dont nous avons conçu ce jeu, voici l'ensemble du diagramme de classe correspondant, avec les tous les liens entre les différents paquets (en 2 images) : chacun des diagrammes présentés ci-dessous sont disponibles dans le dossier "uml_diagrams".





Le diagramme de séquence

Voici un exemple de diagramme de séquence simplifié (sans appel sur la même classe) lorsque l'utilisateur saisit une commande au clavier. La commande ici est *play*.



Organisation du groupe

La préconception :

Au départ, alors que n'avions que le cahier des charges avec toutes les consignes que nous nous devions de respecter, nous nous sommes tous les 3 attelés à la préconception et sur ce que nous voulions faire. C'est ainsi que le thème du jeu fut trouvé, ainsi qu'une conception de départ, mais surtout de base au projet.

Mise en commun et travail de groupe :

Pour ce projet, nous avons décidé de tout travailler sur le même environnement de développement : IntelliJ. Nous avons jugé celui-ci facile d'utilisation, pratique et bien fourni.
Nous avons organisé plusieurs réunions de travail, sur l'outil de communication Discord. Cela nous a permis d'établir les tâches de chacun pour pouvoir travailler ensemble et continuer de notre côté.
Afin de travailler sur les mêmes fichiers, nous avons décidé d'utiliser un dépôt Github. Le fait d'avoir choisi IntelliJ comme IDE nous a grandement facilité la mise en commun car celui-ci implante directement les fonctionnalités de Github.

La programmation Java, à l'aide d'IntelliJ

- Andrianarivony Henintsoa :

  Henintsoa s'est occupé des jeux *Hangman*, *Questions* et *Karaoke.* Il s'est chargé d'écrire les codes et tests correspondant aux classes *Shop*, *Exit*, *Items* et les classes qui héritent de cette dernière. Il s'est également occupé de réaliser le diagramme de séquence pour la commande *play*.

- Goubeau Samuel :

Samuel s'est occupé des jeux *Findnumber*, *TicTacToe*, et *Riddle*. Il s'est aussi chargé d'écrire les codes et tests correspondant aux classes *Character*, *NPC* et *Player*. Il s'est aussi chargé d'écrire le code de certaines classes du paquet *command*. Il s'est également occupé de réaliser le diagramme de classe final.

- Sauzeau Yannis :

Yannis s'est occupé des jeux *HanoiTower*, *RockPaperScissors* et *QTE*. Il s'est aussi chargé d'écrire les codes et tests correspondant aux classes *Place, Game, Ending* et le reste des classes du package *code.command*.

Mais dans la majorité, nous avons regardé le code de chacun afin de régler certains défauts. Ou encore nous nous sommes aidés les uns les autres pour régler les problèmes que nous n'arrivons pas à gérer.
Nous avons plusieurs fois revu la conception de certaines parties du code, comme par exemple avec la partie qui gère les commandes. Nous étions en premier partie sur un 'switch case'. Mais nous nous sommes ravisés pour partir sur une classe *Command* avec plusieurs héritages sur celle-ci (cf. code.command).