RAPPORT DE PROJET

Unité d'enseignement Programmation Orientée Objet

1 DECEMBRE 2023

FLORENTIN DEUMIE-LACOUT
THEO CARTIER
ARNAUD BONNET

Table des matières

1. Documentation Utilisateur :

1.1 Installation :

Pour jouer au jeu, vous avez besoin d'être sur un ordinateur faisant tourner un système d'exploitation Linux.

Téléchargez l'archive contenant les fichiers du jeu puis faites un clic droit sur le fichier ZIP. Cliquez sur l'option extraire et choisissez l'endroit où vous voulez installer le jeu. Un dossier va être créé, ce dossier s'appelle DEUMIE-LACOUT_CARTIER_BONNET. Faites un clic droit sur le dossier et cliquez sur ouvrir dans un terminal.

A ce moment-là une page du terminal s'ouvre, tapez la commande :

cd Projet_POO/src/

Puis tapez la commande:

java Main

Votre jeu vient de se lancer.

1.2 Le Jeu :

Notre jeu est un Maze like, nous sommes partis sur la base du jeu The Binding of Isaac. L'histoire du jeu est simple : vous êtes un héros qui a perdu son village qui a été détruit par des monstres. En quête de vengeance, vous entrez dans cette grotte pour y affronter le boss et en finir une bonne fois pour toutes.

Tuez le boss pour gagner et évitez de mourir.

1.3 Comment jouer ?

Le jeu est une aventure textuelle ouverte dans un terminal. Il n'est donc pas possible d'interagir avec les éléments du jeu en utilisant une souris. Tout se passe par des commandes écrites en majuscules qui peuvent utiliser un argument pour préciser le but ou l'élément du jeu avec lequel on veut interagir.

1.4 Commandes :

GO [nom Destination]: la commande GO suivie du nom de la destination permet de se déplacer d'une pièce à une autre.

- Attention : s'il y a un monstre ou un boss dans la salle où vous venez d'entrer, il est impossible de sortir de la salle tant que vous ne l'avez pas battu/tué. Vous avez la possibilité de voir les sorties possibles grâce à la commande LOOK décrite plus loin. La Map est à la fin de ce rapport, attention, elle dévoile les différents

types de portes ainsi que leur emplacement.

HELP : la commande HELP affiche toutes les commandes possibles dans la salle où vous êtes.

- La commande s'adapte à l'environnement dans lequel vous êtes (s'il y a un personnage dedans vous pouvez utiliser TALK par exemple, mais si la salle est vide vous ne verrez pas l'option TALK en utilisant la commande HELP).

LOOK : la commande LOOK permet d'avoir un aperçu des personnes dans la pièce.

- Vous avez la possibilité de regarder votre inventaire avec la commande LOOK INVENTORY.
- Vous avez la possibilité de regarder les informations liées au héro (arme et armure équipée s'il y en a, points de dégâts du héros et points de vie) avec la commande LOOK HERO.
- Vous avez la possibilité d'afficher la map avec la commande LOOK MAP.

TAKE : la commande TAKE permet de récupérer un objet donné par un pnj ou un coffre. Elle place automatiquement l'objet dans votre inventaire.

USE [nom Objet] : la commande USE suivie de l'objet à utiliser permet d'utiliser un objet qui se trouve dans notre inventaire ou de l'équiper s'il s'agit d'une épée ou d'une armure.

ATTACK : la commande ATTACK permet d'attaquer le personnage présent dans la même pièce que le héros.

- Attention : lorsque vous entrez dans la pièce d'un monstre ou d'un boss, vous serez obligé de le battre pour sortir de la pièce. En revanche si vous attaquez un autre personnage vous aurez la possibilité de sortir de la pièce, de parler… Vous n'êtes pas obligé de les vaincre pour faire autre chose. Si vous attaquez le personnage vous attaquera en retour s'il n'est pas mort sur le coup et si vous voulez attaquer un coffre, cela est possible, mais bon courage pour le vaincre.

TALK : la commande TALK permet de parler à un personnage du jeu, peut être qu'il décidera de vous donner quelque chose, les gens sont parfois imprévisibles.

- Attention : il est possible de parler à un monstre, mais vous ne comprendrez pas ce qu'il dit, la commande HELP n'affiche donc pas l'option TALK si vous êtes dans une salle monstre bien que ce soit possible. Il s'agit d'un choix de notre part dans la conception du jeu.

DELETE [nom Objet] : la commande DELETE suivie du nom de l'objet permet de le supprimer de l'inventaire.

QUIT : la commande QUIT nous fait quitter le jeu.

Attention : il n'y a pas de système de sauvegarde, vous risquez donc de perdre toute la partie.

2. Documentation Développeur :

2.1 Conception du Projet :

I Les idées :

Voulant faire un Maze like, nous avons décidé de faire 3 grands axes dans notre UML : les Locations, les Objets et les Personnages.

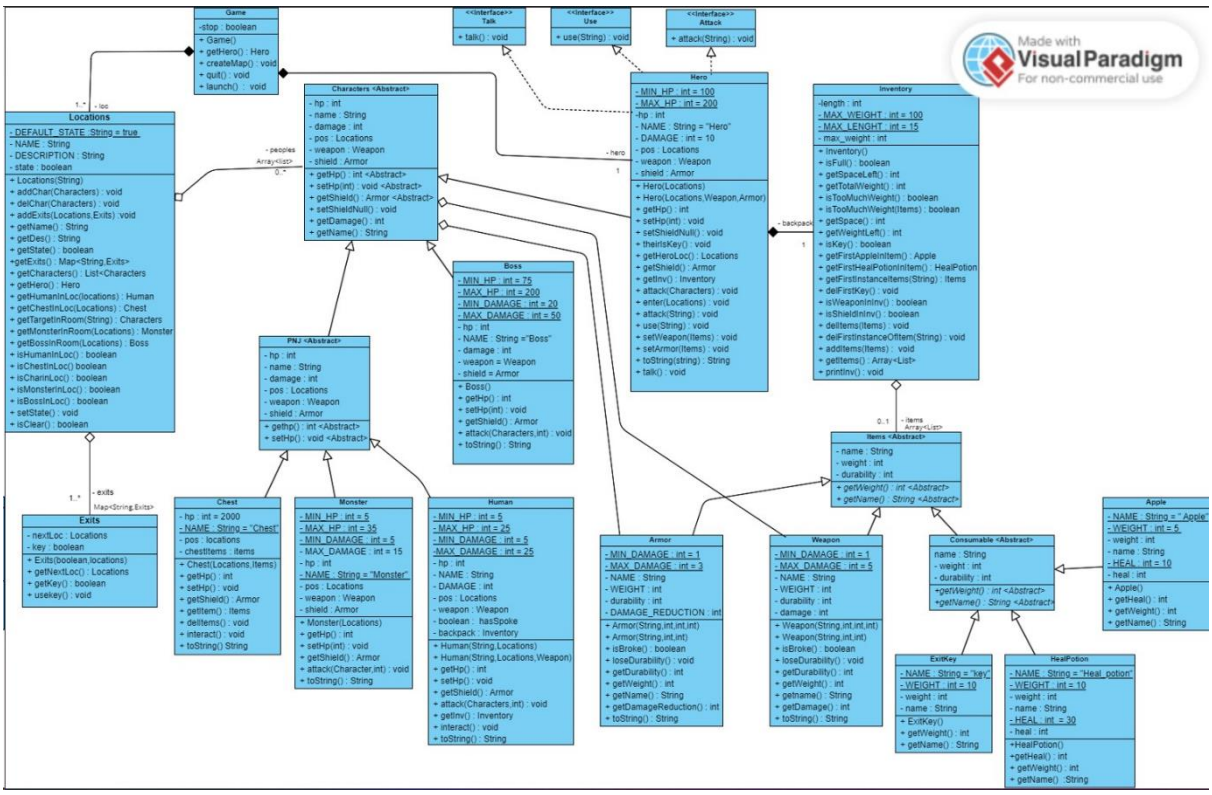
Ensuite, nous avons détaillé ces 3 axes pour en faire des classes principales.

- La classe Items, qui regroupe tous nos types d'items : Arme , Armure et Consommable qui lui-même regroupe : Clef, Potion de vie et pomme
- La classe Characters qui regroupe tous nos type de personnages : Héros, Boss, PNJ qui regroupe : Monstre, Humain, Coffre
- La classe Locations qui regroupe toutes les Sorties

Enfin, nous avons aussi une autre classe principale Jeu, qui nous permet de faire tout le design du jeu.

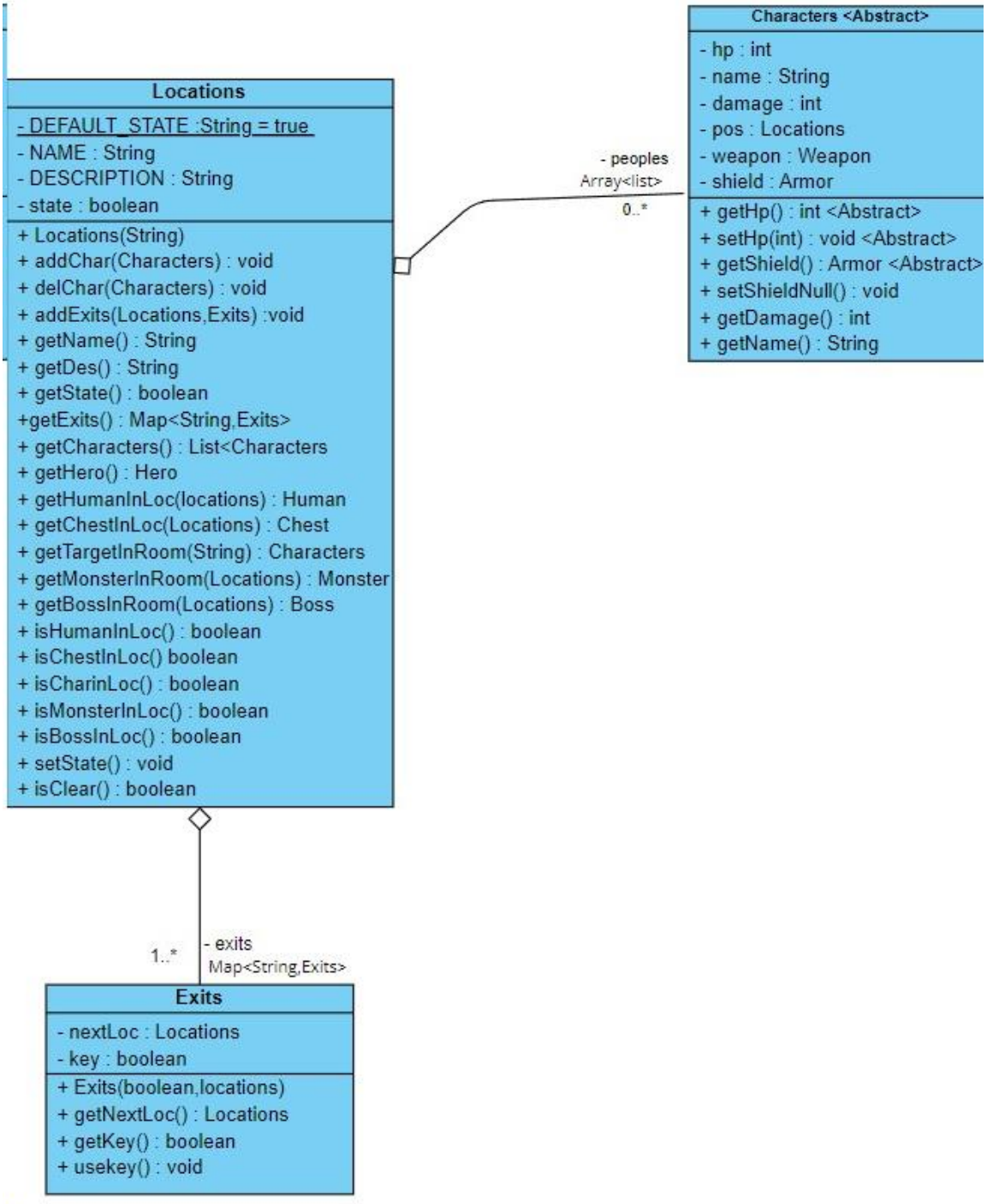
II L'UML

Nous avons fait une ébauche d'un schéma UML (voir annexe 1 dans organisation des tâches de groupe) qui nous a donné une vision schématique de nos 3 grandes classes. Ce schéma a été modifié au fur et à mesure que nous codions pour répondre aux problèmes trouvés lors du code du jeu. Finalement nous sommes arrivés à cet UML final :



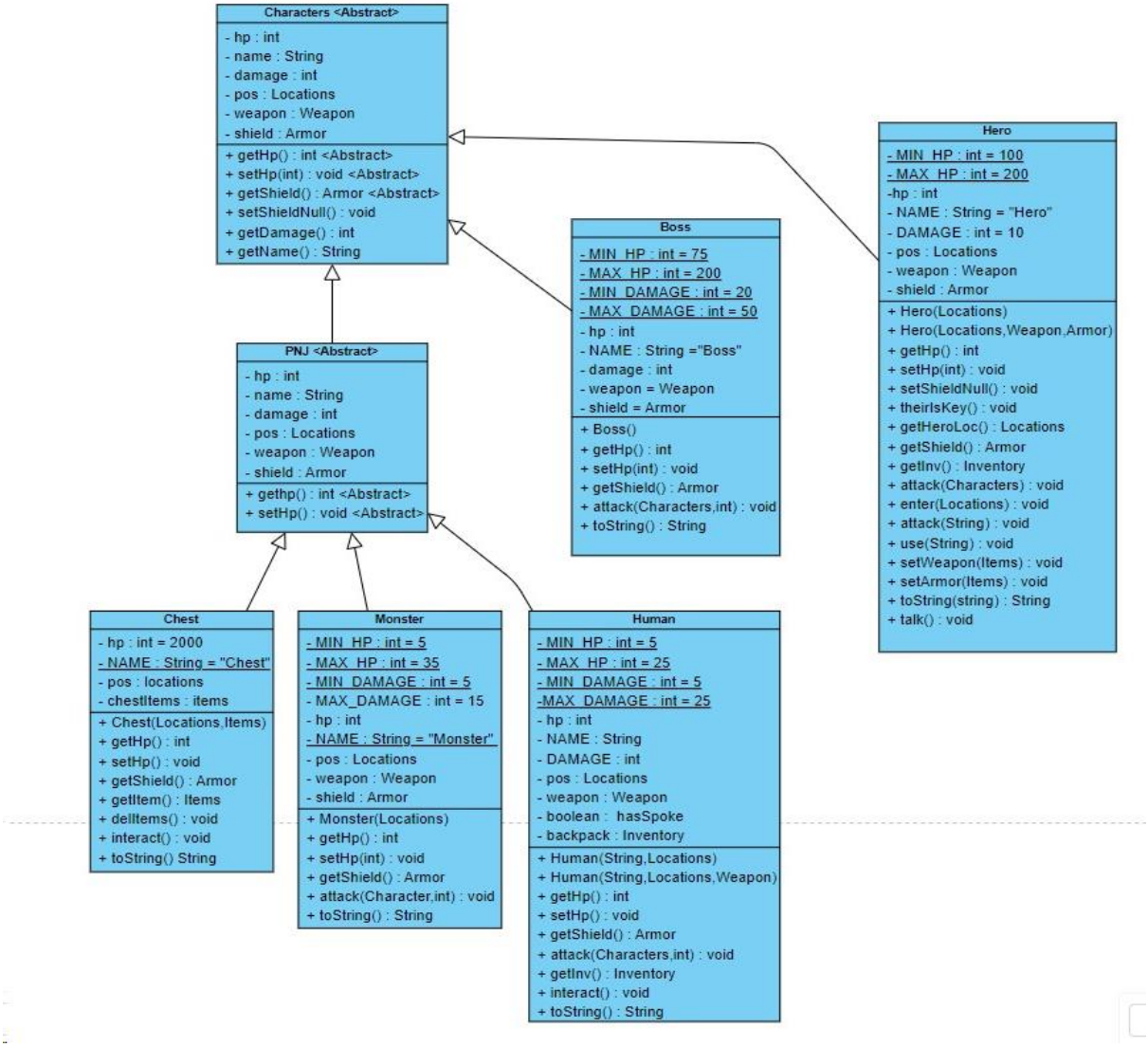
(il y a dans le fichier du projet une image SVG et une PNG (en fonction de ce que vous préférez) pour une meilleure vision de l'ensemble de notre UML)

1) La classe Locations



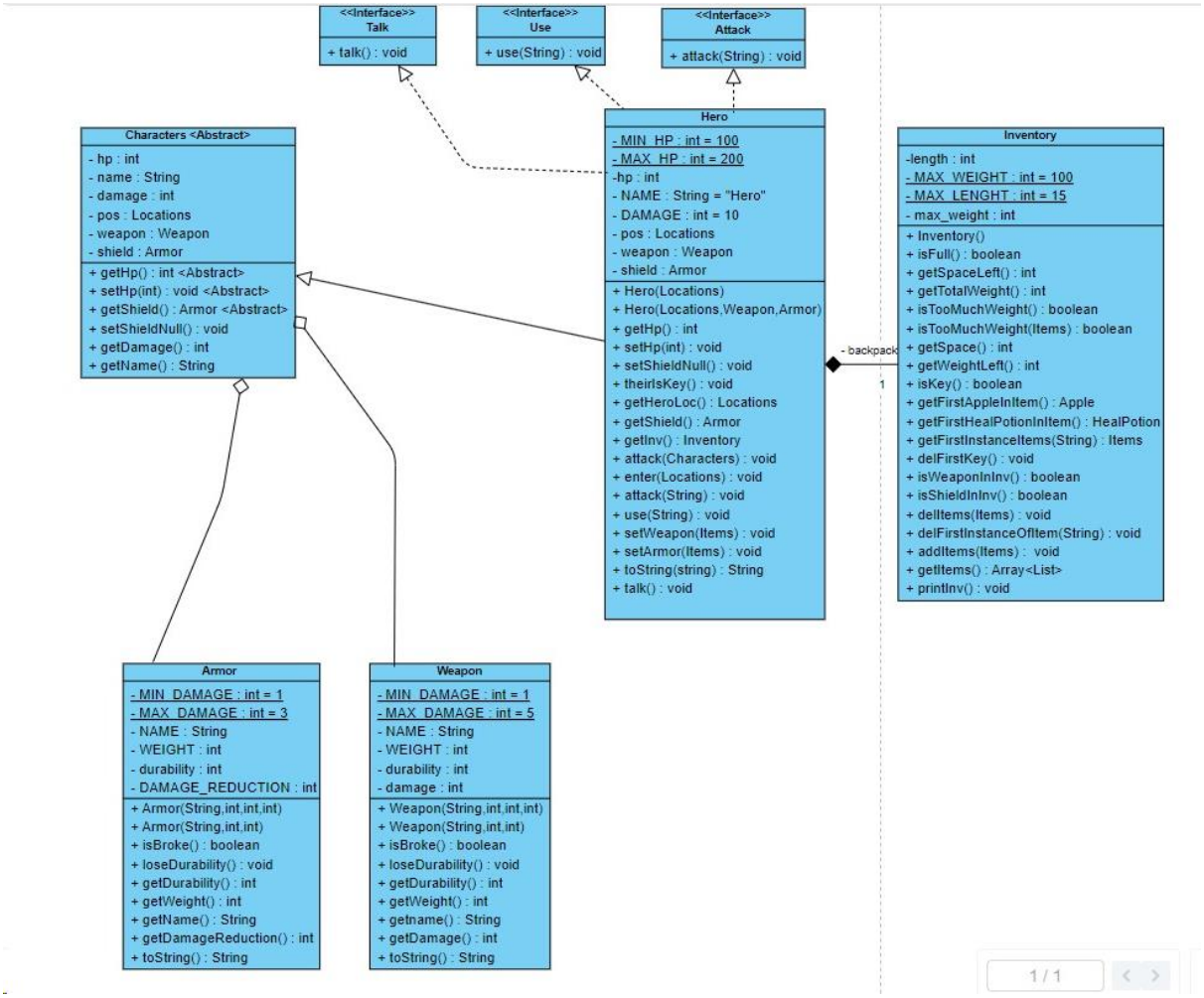
Notre Classe Location est essentielle dans notre jeu. C'est elle qui va regrouper toutes nos sorties ainsi que les personnages que chaque pièce contient. Elle relie par deux agrégations la classe Exits et Characters

2) La classe Characters



Notre Classe Characters regroupe tous nos types de personnages, en particulier la plus intéressante, le héros. Nous avons décidé de ne pas mettre Boss dans PNJ car c' est l'autre personnage le plus important et si nous voulons faire des modifications pour améliorer notre jeu, ce sera plus simple.

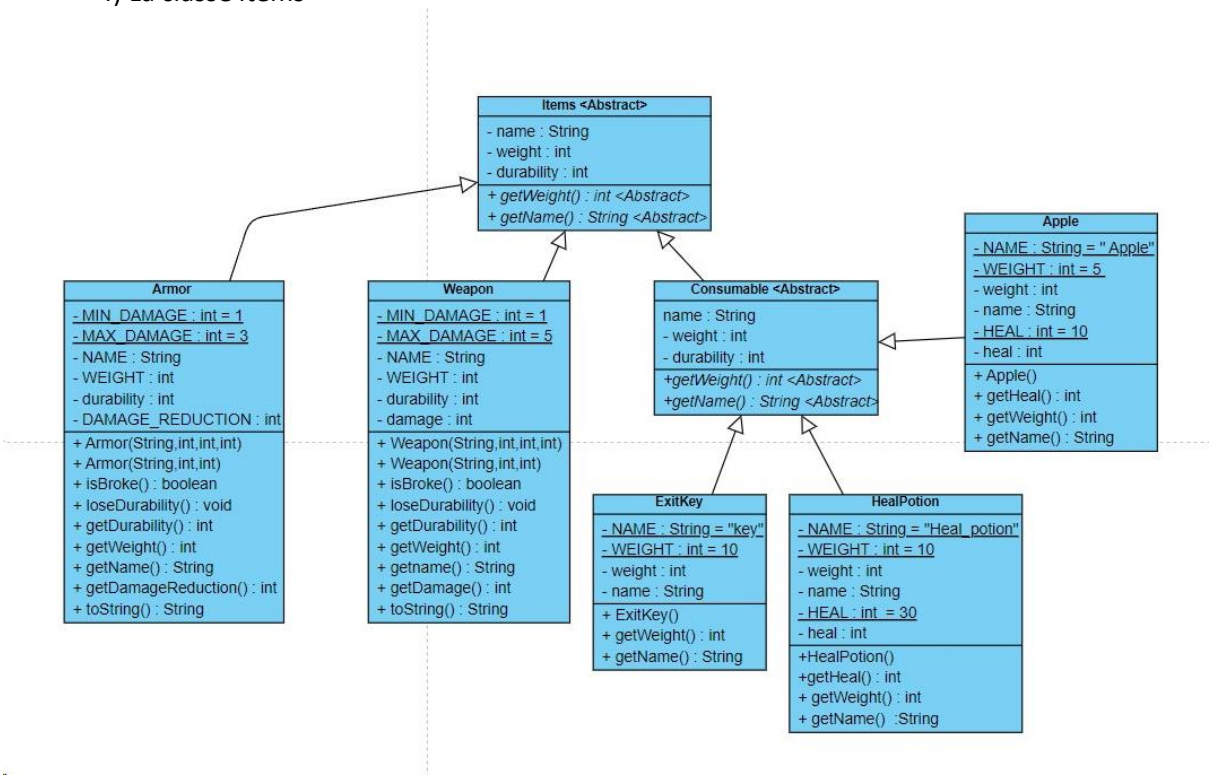
3) Précision sur la classe Hero



La classe Hero est sans doute la classe la plus importante de notre jeu. C'est avec elle que sont reliées nos 3 interfaces. Elles ne sont pas forcément utiles, mais si nous voulons faire des modifications sur d'autres classes, elles sont déjà présentes et prêtes à l'emploie. C'est la seule classe qui a un inventaire. Elle a aussi une arme et une armure. Toutes les autres classes de Charactères en ont aussi.

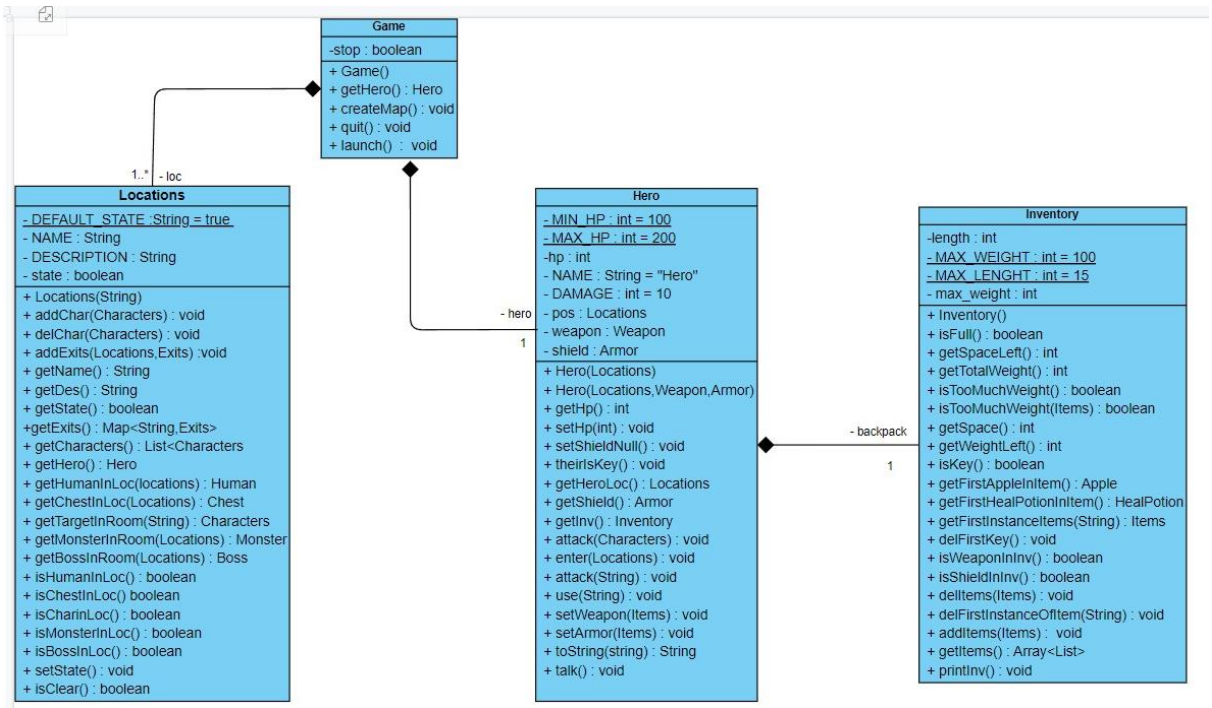
Surtout, c'est la classe qui regroupe une majorité des sous fonctions de nos différentes commandes.

4) La classe Items



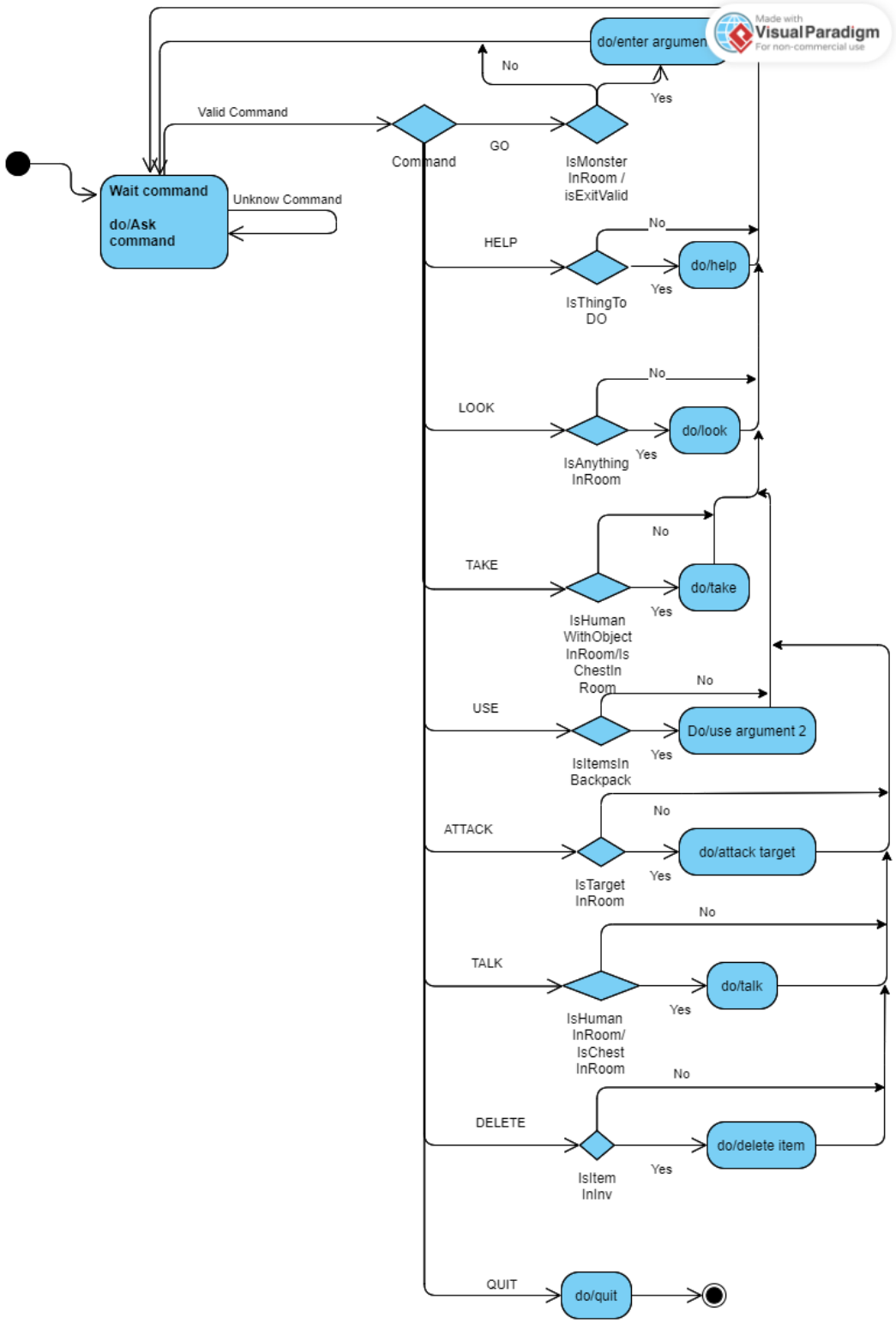
La classe Items regroupe tous nos objets, les classes Armor et Weapon sont indépendantes. La classe Consumable quant à elle, regroupe tous les objets utilisables à effet unique du jeu.

5) La classe Game



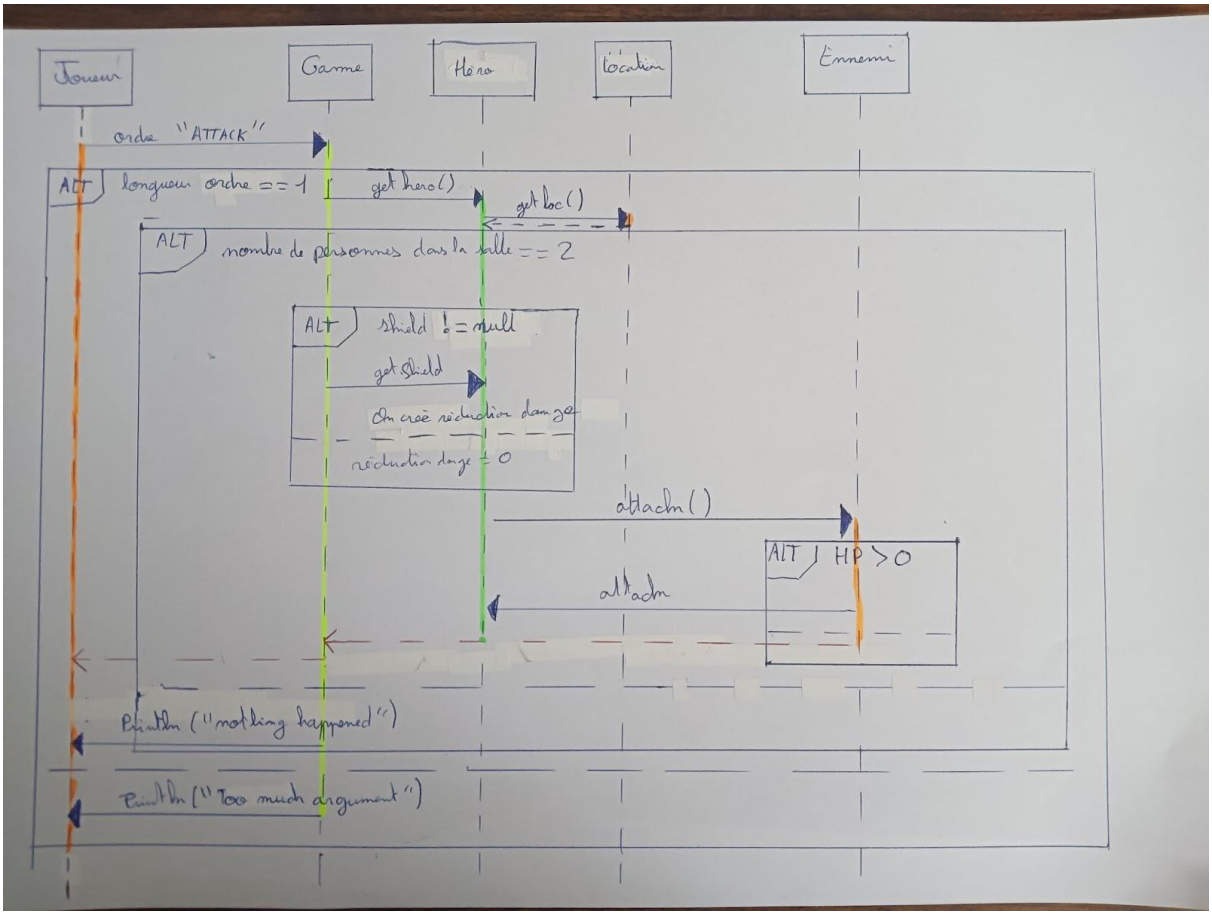
Enfin, notre classe game qui crée le jeu. C'est elle qui va créer la carte du jeu, donc nos différentes salles et ce qu'elles contiennent. Aussi, elle est chargée de créer toutes nos commandes utiles au jeu. Ce sont les fonctions de Game qui vont être appelées dans le main.

III Diagramme d'État et diagramme Séquentiel



Ce diagramme d'état montre les différentes actions faisables en tant qu'utilisateur de ce jeu.

(Il y a dans le fichier du projet une image SVG et une PNG (en fonction de ce que vous préférez) pour une meilleure vision de l'ensemble de notre diagramme d'état).



Voici le diagramme de séquence correspondant à l'ordre ATTACK nous n'avons fait qu'un seul diagramme de séquence par manque de temps.

IV Potentielles améliorations

Nous avons réfléchi à plusieurs optimisations et améliorations du jeu si nous étions amenés à continuer le projet.

Il serait intéressant de faire un système d'étage pour que le jeu soit plus long et pour rajouter de la difficulté dans la partie, actuellement le jeu est trop court pour exploiter pleinement le système de consommables avec les épées, les armures, les pommes et les potions de vie.

Une deuxième amélioration serait de prendre en charge la sensibilité à la casse pour les commandes du jeu (actuellement le jeu ne comprend pas les commandes telles que go, Go, gO…), cela permettrait de rendre le jeu plus facile, car la moindre faute de frappe nous oblige à retaper la commande.

Une troisième amélioration serait, comme précisé dans « 2.)La classe Characters » d'ajouter des fonctions particulières au Boss ou lui ajouter des classes liées par agrégation.

Une quatrième amélioration serait d'augmenter la diversité de nos types de Characters(monstre, Boss,..) et d'items(armure,arme,...) en transformant ces classes en sous-classe mères avec chacune plusieurs filles.

V Problèmes

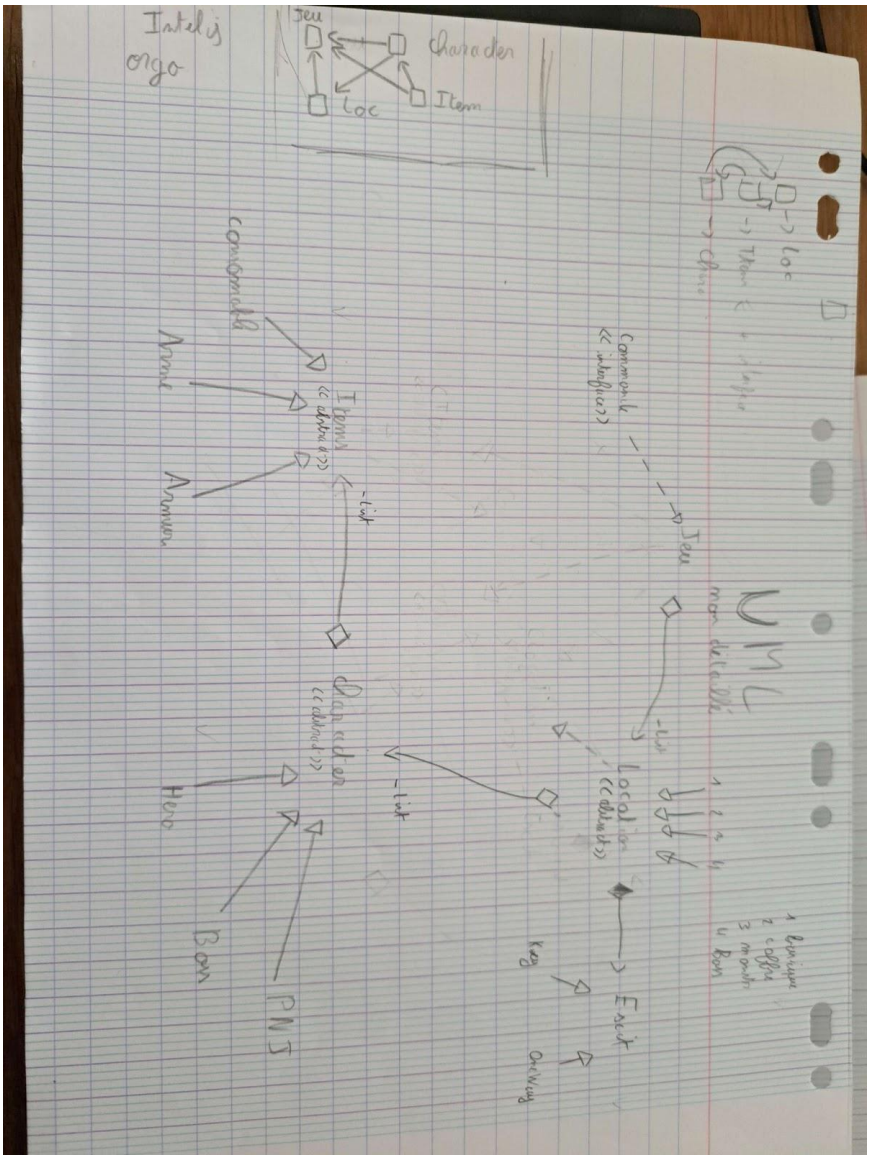
Lors de la conception de notre jeu nous voulions faire beaucoup plus de créations procédurales comme notre source d'inspiration The Binding of Isaac, Nous voulions le faire que ce soit pour les salles , les items ou les monstres, mais nous nous sommes vite rendus compte que cela allait nous poser des problèmes vis-à-vis de la date de rendu et surtout de nos connaissances en java.

Un deuxième problèmes rencontrés fut le délai pour rendre le projet qui a été assez court si on compte le reste du travail que nous avions à faire à côté. A cause de cela, nous avons revu à la baisse nos idées.

3. Organisations des tâches dans le groupe :

Pour ce projet, nous avons peu réparti les tâches. Au début du projet, nous avons réfléchi ensemble aux bases de notre projet. Savoir quel type de jeu faire, l'histoire que nous voulions faire. Puis pour commencer nous avons réfléchi à 3 pour faire une ébauche de notre UML. Ainsi, nous avons une base que nous pouvons modifier si nécessaire.

Avec cette ébauche, nous avons essayé de détailler un peu plus chaque classe pour avoir une idée plus concrète de chaque classe. Nous avons réparti les classes entre nous. Ensuite, nous avons confronté nos classes et nous les avons essayées de finaliser leur ébauche à 3.



L'étape de conception étant finie, nous avons commencé à coder chacun de notre côté. Nous nous sommes vite rendu compte que cela n'était pas très productif. Finalement, nous avons décidé de travailler au maximum sur un git à 2 ou 3 quand nous le pouvions. Les créations de classes n'ont pas été réellement réparties. Chacun a codé ses classes. Chaque classe est un mélange de nos idées. Dès que nous trouvions un problème, une personne s'en occupait indépendamment de qui a créé la classe.

Quand notre jeu était enfin un minimum jouable, nous avons essayé continuellement de le déboguer. Lorsque notre jeu fut en grande majorité codée, Florentin a fait un premier gros débogage en testant les commandes du jeu, Théo et Arnaud quant à eux ont corrigé les éventuels bugs trouvés.

Le code du jeu étant fini,Théo et Arnaud se sont occupés de faire tous les tests pour vérifier une ultime fois que toutes les fonctions et classes fonctionnent parfaitement.

Florentin, lui, s'est occupé de mettre au propre l'UML.

Finalement, nous avons fini ensemble sur l'écriture du rapport.

Annexe 1
Map du jeu