PROGRAMMATION ORIENTEE OBJET ET INTERFACE HOMME-MACHINE

PROJET

30 NOVEMBRE 2023

Julien Lefevre
Antoine Promis
Téo Itsweire-Krebs

Table des matières

Introduction

Au cours de notre 3ème année de Licence Informatique à l'Université de Poitiers et dans le cadre de l'Unité d'Enseignement Programmation Orientée Objet et Interface Homme-Machine, nous avons eu pour projet la programmation d'un jeu d'aventure textuel. L'inspiration de ce projet vient du jeu *Colossal Cave Adventure*, un jeu des années 70 qui est considéré comme le premier jeu d'aventure interactif. Dans ce jeu, une situation était décrite, le joueur pouvait alors saisir des commandes décidant de l'action qu'il allait faire, ce qui entrainait une nouvelle situation et ce jusqu'à atteindre la fin du jeu.

Nous avons donc créer notre propre histoire. Cette dernière prendra place dans une école où l'utilisateur se rend pour son premier jour. Son objectif : voler les sujets des futures examens afin de les étudier à l'avance et s'assurer de valider son année.

Dans un premier temps, nous vous présenterons la partie documentation utilisateur, puis dans une seconde partie nous expliciterons la conception du projet. Pour finir, en dernière partie nous parlerons de l'organisation du groupe.

1. Documentation utilisateur

Durant cette partie nous présenterons rapidement comment installer et jouer au jeu, puis dans un second temps, nous vous présenterons les commandes utilisables dans le jeu.

1.1. Installation et exécution du jeu

Concernant l'installation, il faudra dans un premier temps lancer un terminal (Windows : Invite de commandes / Linux : Terminal). Il faudra ensuite se placer dans le répertoire nommé : « Lefevre_Promis_Itsweire-Krebs »

- Windows :
  Saisissez : « javac CLASS*.java CLASS\ACTIONS*.java CLASS\Character*.java CLASS\Door*.java CLASS\Item*.java CLASS\Game*.java CLASS\Room*.java »
  Puis pour exécuter :
  « java -cp . CLASS.Main »

- Linux :
  Saisissez : « javac CLASS/*.java CLASS/*/*.java»
  Puis pour exécuter :
  « java -cp . CLASS.Main »

1.2. Commandes du jeu

Ce jeu vous propose l'utilisation de nombreuses commandes vous permettant alors de vous déplacer dans l'environnement et d'interagir avec ce dernier. Nous allons vous les présenter en détails ci-dessous :
(PNJ = Personnage Non Joué)

- HELP
  Cette commande ne nécessite aucun argument et renverra dans le terminal la liste des commandes utilisables avec une courte description de ces dernières.

- QUIT
  Cette commande ne nécessite aucun argument, elle vous permettra de quitter le jeu instantanément, cependant la partie qui était en cours ne sera pas sauvegardée.

- ATTACK
  Cette commande se devra d'être accompagnée d'un argument qui sera le nom d'un PNJ, par exemple :
  ATTACK Heisenberg
  La commande fonctionnera uniquement si le PNJ cité se trouve dans la même salle que vous. Le perdant sera le moins puissant.

- LOOK
  Cette commande pourra être utilisée avec un argument, ou aucun. Si vous ne saisissez aucun argument, vous obtiendrez la description de la salle dans laquelle vous vous trouvez, avec notamment les objets et personnes qui s'y trouvent. Enfin, si vous saisissez cette commande avec argument, ce dernier pourra être soit un PNJ soit un objet présent évidemment dans la salle ou dans votre inventaire, par exemple :
  LOOK Heisenberg
  Vous obtiendrez alors la description de l'objet ou du PNJ saisi en argument.

- USE
  Cette commande pourra être utilisée avec soit un argument, soit deux arguments. Commençons par l'utilisation avec un argument. Ce dernier devra être un objet présent dans votre inventaire. Si il s'agit d'une arme, vous équiperez alors cette arme (si vous réutiliser la commande avec l'arme équipée, elle sera alors déséquipée et reviendra dans votre inventaire). Cela peut aussi être une clé pour ouvrir une porte (nous en reparlerons), ou encore un journal dans lequel vous pourrez alors écrire ce que vous pensez avoir besoin de retenir, ou pour finir les sujets d'examens qui permettront alors de terminer le jeu. Par exemple :
  USE Diary
  Comme mentionné précédemment, cette commande pourra être utilisée avec deux arguments, qui devront également être des objets présents dans votre inventaire. Ainsi, ces deux objets fusionneront pour n'en donner qu'un seul qui se retrouvera dans votre inventaire, comme par exemple :
  USE Glue Hair

- GO
  Cette commande se devra d'être accompagnée d'un argument qui sera le nom d'une

salle, par exemple :
GO LivingRoom
La commande fonctionnera uniquement si la salle existe et qu'une porte relie cette dernière avec la salle où vous êtes.
Le cas échéant, vous pourriez être confrontés à certaines portes fermées et qui nécessiteront alors un code ou une clé. Si il s'agit d'une clé. Il faudra utiliser la commande :
USE NomDeLaClé

- INVENTORY
  Cette commande ne nécessite aucun argument, elle affichera dans le terminal la liste des objets présents dans votre sac.

- TAKE
  Cette commande devra s'accompagnée d'un argument, ce dernier devra être un objet présent dans la salle où vous êtes, comme par exemple :
  TAKE Hair
  L'objet saisi en argument se retrouvera alors dans votre inventaire.

- TALK
  Cette commande devra être accompagnée d'un argument, ce dernier devra être un PNJ présent dans la même salle que vous, comme par exemple :
  TALK Heisenberg
  S'affichera alors dans le terminal une discussion avec ce personnage.

Il est important de noté que pour chacune de ces commandes, si lorsque vous la saisissez le nombre d'arguments qui l'accompagne n'est pas correct, il ne se passera rien en dehors de l'affichage dans le terminal d'un message indiquant que le nombre d'arguments n'est pas bon. Aussi, si le nom de la commande n'est pas reconnu, s'affichera dans le terminal un message indiquant que la commande n'existe pas.

2. Conception du projet

Concernant la conception du projet, nous avons dans un premier temps déterminé l'histoire du jeu, puis nous avons pu procéder à la création du Diagramme de Classes.

2.1. Diagramme de Classes

Après avoir construit l'histoire de notre jeu avec notamment les dialogues et quêtes, nous avons pu construire notre UML. Vous retrouverez ce dernier en **Annexe 1, Annexe 2, Annexe 3 et Annexe 4**. Le fonctionnement de notre jeu implique notamment la création de nombreuses classes héritant de « Adult » et « Student » (ces dernières héritant elles-mêmes de « Human »). En effet, les nombreux dialogues qui diffèrent selon les PNJ, tout comme les quêtes liées à ces derniers, auraient été ingérables si nous avions choisi de les créer à partir d'une seule et même classe. On notera aussi que la classe « Hero » héritant de « Student » implémente toutes les interfaces, qui représentent les actions que l'utilisateur peut exécuter.

Une spécificité à noter est notamment la classe « TheGame », c'est via cette classe que tout les objets (PNJ, salles, etc) sont instanciés. Nous avons fait ce choix afin que le main ne soit pas trop chargé. Cela nous permet de créer une et une seule instance du jeu, mais aussi d'avoir accès aux futurs objets depuis les classes.

2.2. Diagramme d'états

Dans notre jeu, nous avons un objet d'instance « DoorWithCode », son fonctionnement est assez simple. Si l'utilisateur essaie d'allez dans une salle et que cet objet sépare cette dernière de la salle où il se trouve, un message s'affichera : « veuillez saisir le code ». L'utilisateur saisi alors un code, si il est correct la porte s'ouvre et il peut entrer, sinon la porte reste fermée et il ne change pas de salle. Vous retrouverez le diagramme d'état de « DoorWithCode » en **Annexe 5**.

2.3. Diagramme de séquences

Pour le diagramme de séquences, nous avons estimé pertinent de vous présenter la fonctionnalité « Take » de notre jeu. Celle-ci permet à l'utilisateur de récupérer un objet présent dans la pièce où il se trouve. Cet objet se retrouvera alors dans son inventaire, il sera également supprimé de la liste des objets présents dans la salle. Ce diagramme vous montre donc les communications entre différentes classes rendant possible l'exécution correcte de la fonctionnalité. Vous pourrez retrouver ce diagramme de séquences en **Annexe 6**.

3. Organisation du groupe

Concernant l'organisation de notre groupe, nous avons tout simplement travaillé ensemble à chaque fois, que ce soit dans des bâtiments de l'Université de Poitiers, ou dans nos logements respectifs en communicant via Discord. Une fois que nous avons fini de déterminer l'histoire de notre jeu, nous avons fait ensemble le diagramme de classes en débattant notamment sur les méthodes et attributs que nous devions utiliser. Enfin, pour la répartition d'un point de vue de la programmation, nous nous sommes réparti chacun une ou plusieurs classes sur lesquelles travailler pendant une session donnée et ce jusqu'à ce que l'on termine. Si l'un d'entre nous avons besoin d'aide ou une question, les autres l'aidait.

Annexes

Annexe 1

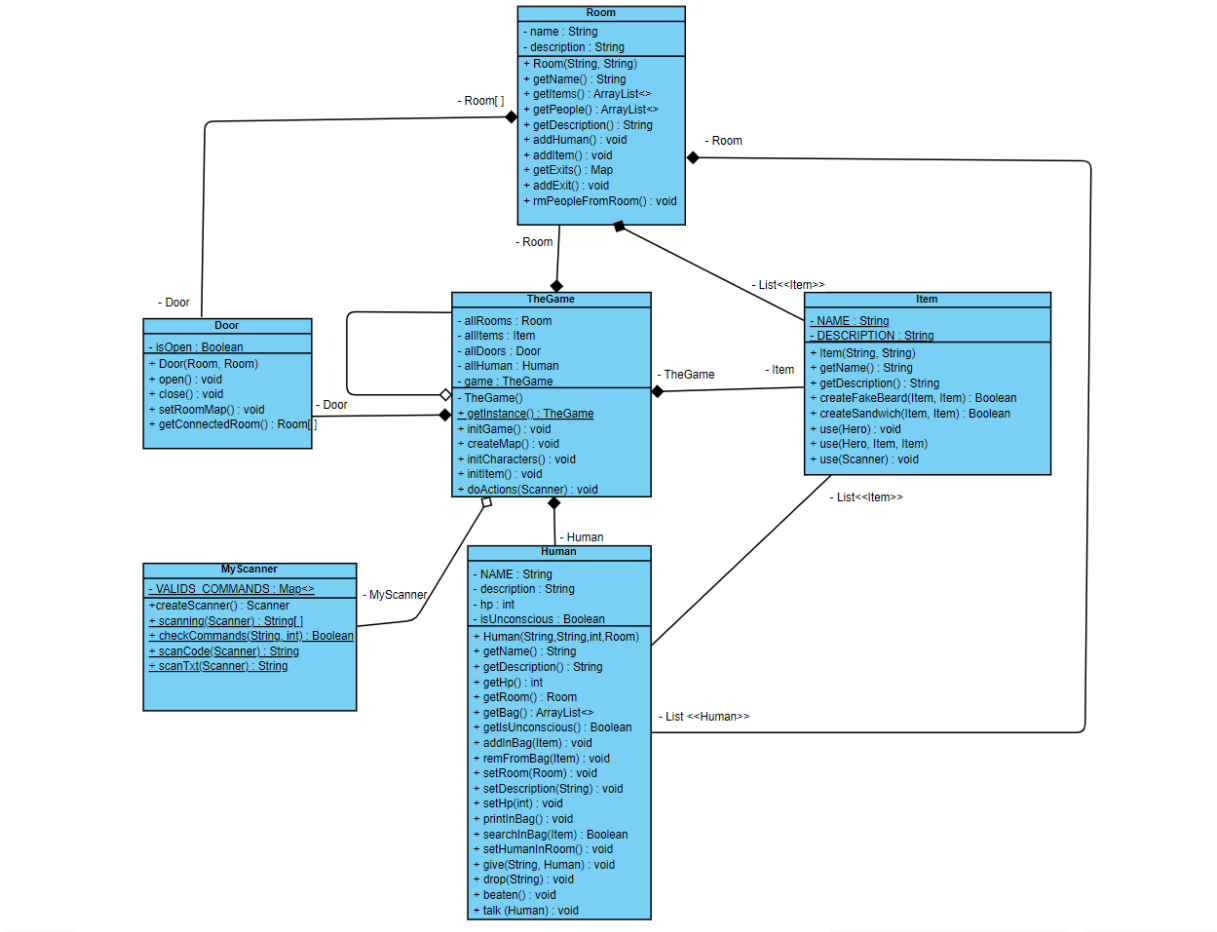

Diagramme de classes (UML) général, il représente les différents liens entre nos classes majeures.

Annexe 2

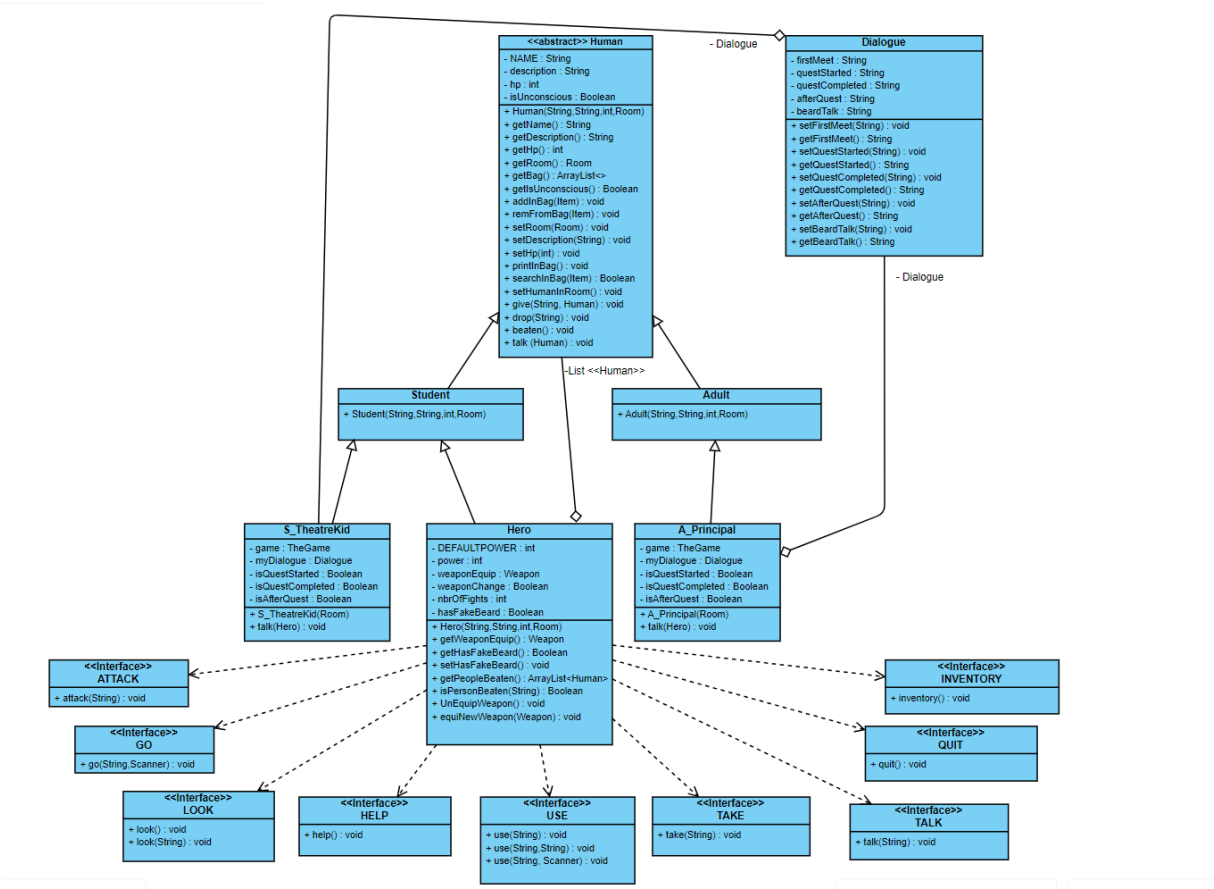

Diagramme plus spécifique représentant la classe « Human » et ses classes filles.

Pour ce diagramme, nous avons choisi de représenter une seule des classes héritant de « Adult » avec « A_Principal », en réalité il en existe 8 autres qui sont soit identiques, soit ont moins d'attributs (nécessaires aux quêtes).

Annexe 3

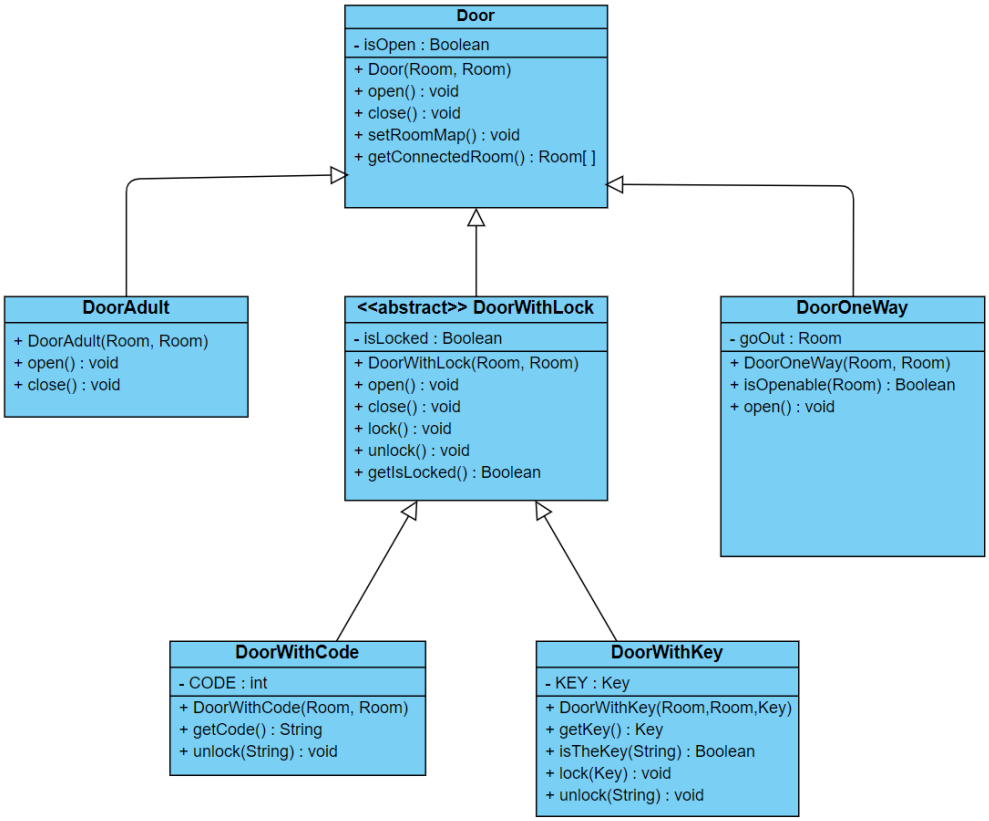

Diagramme de classes représentant la classe « Door » ainsi que toutes ces classes filles.

Annexe 4

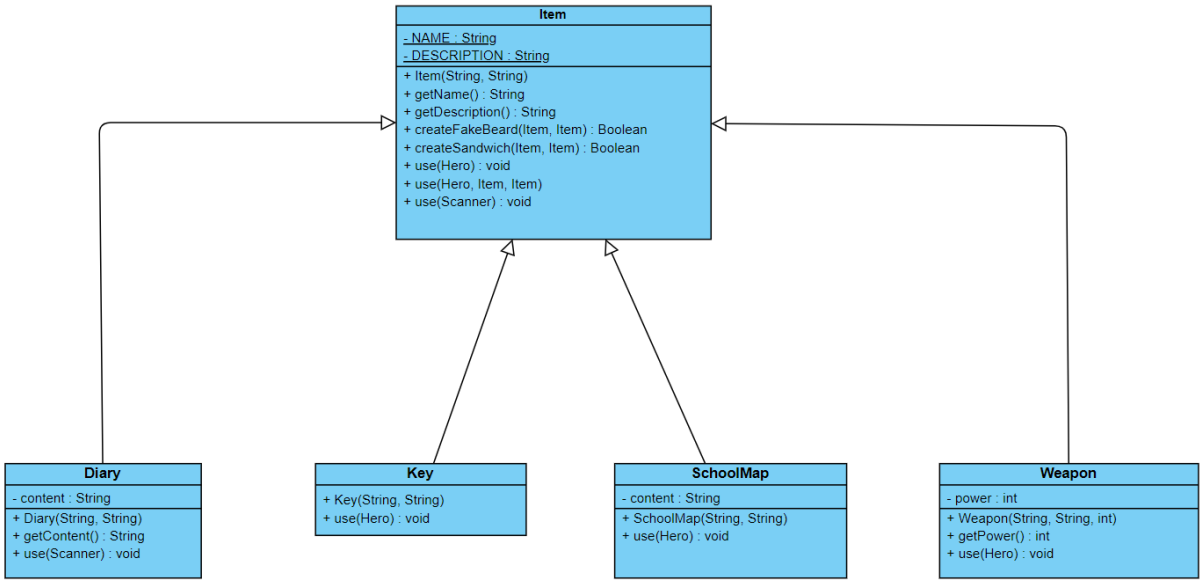

Diagramme de classes représentant la classe « Item » et ses classes filles.

Annexe 5

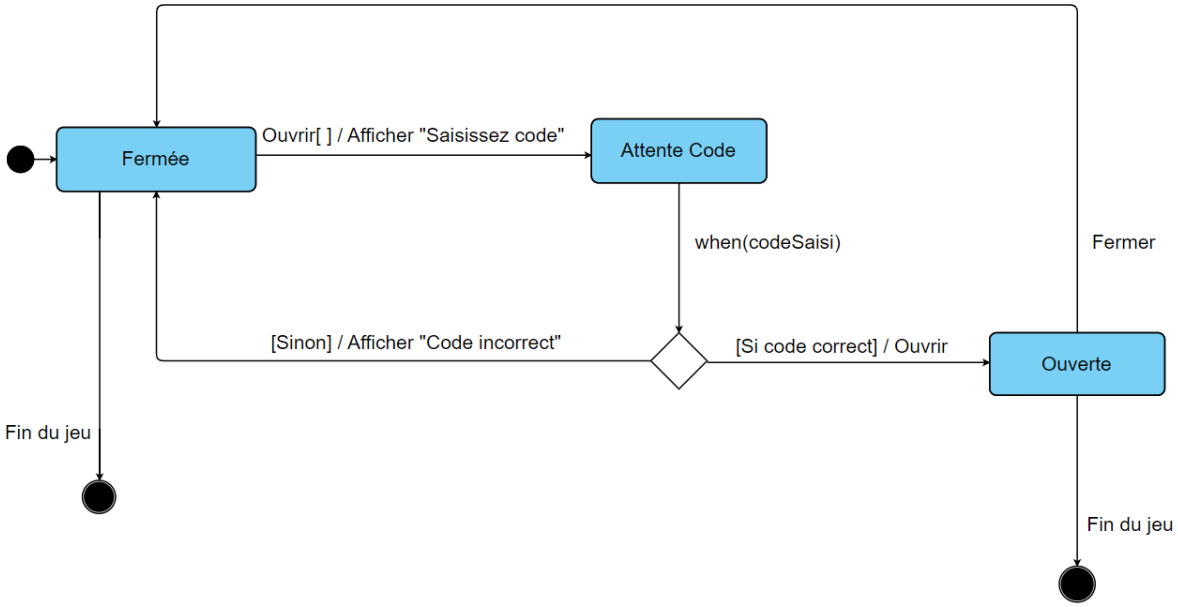

Diagramme d'états de l'objet « DoorWithCode ».

Annexe 6

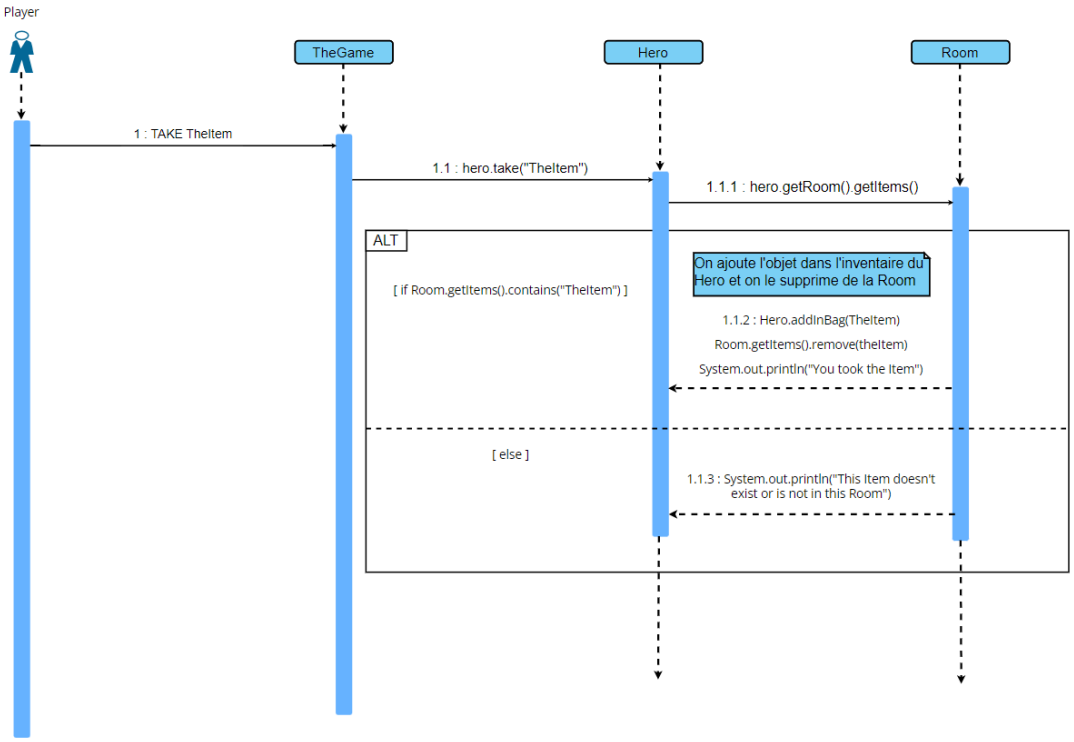

Diagramme de séquences de la fonctionnalité « Take ».