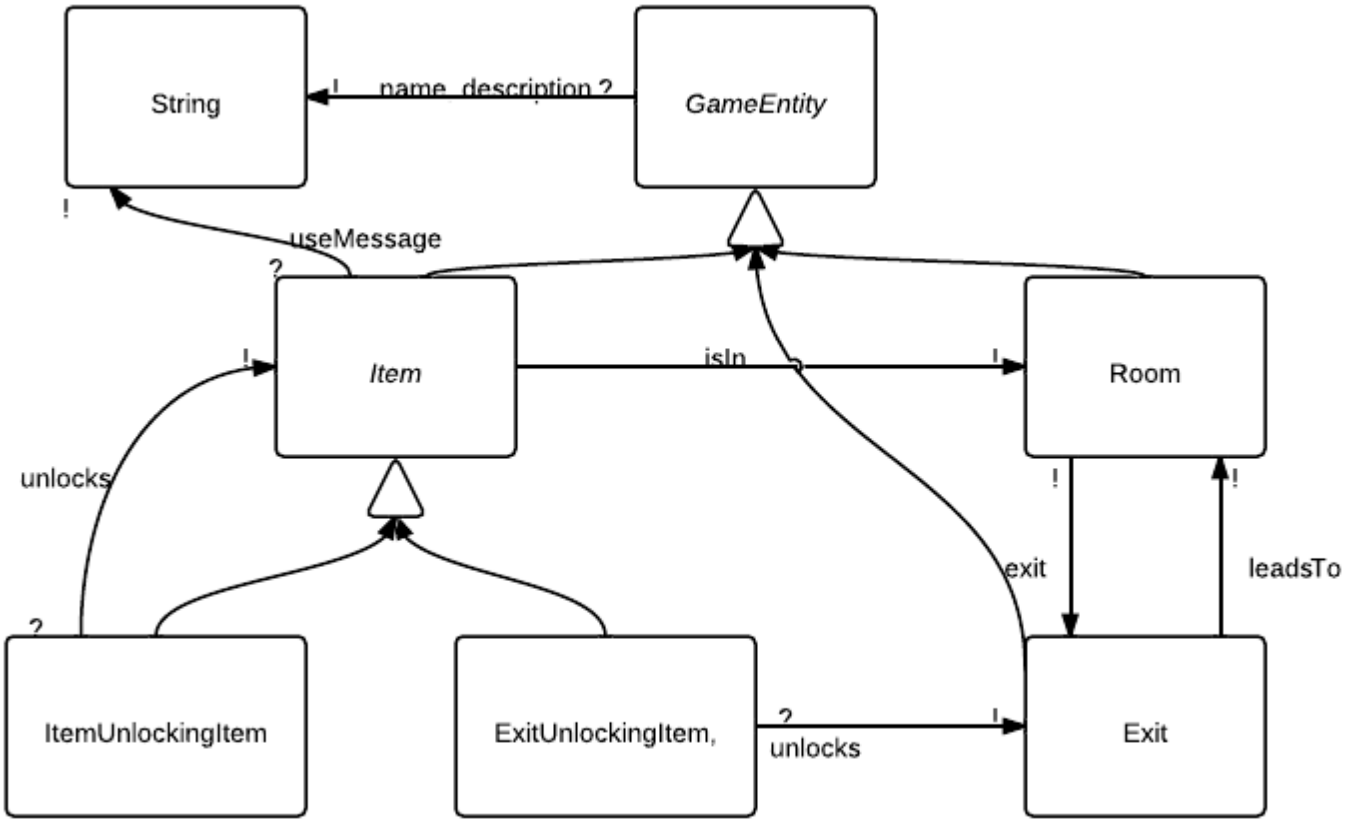
Rahul Rajagopalan
6.170

Object Models

Game Configuration Object Model



Ambiguities Found

It was not specified what function items should perform; only that they should allow the user to progress in some way. I decided to make it possible for them to unlock exits or other items, and categorized them in a different set based on that.

Complexities Ignored

My goal when designing this configuration model was to produce a model expressive enough to fulfill the requirements stated in the assignment, while remaining relatively simple and not "overfitting" to a particular game scenario. This model is expressive enough to handle all the given criteria for the game. There are however some advanced game features that are not included, including:

1. Stateful rooms (rooms that mutate in description depending on the scenario)
2. Stateful items
3. Exits that unlock without an item if you open them on the opposite side
4. Multiple players in a world

These features, while available in other games, are omitted in this model and the resulting application for simplicity.

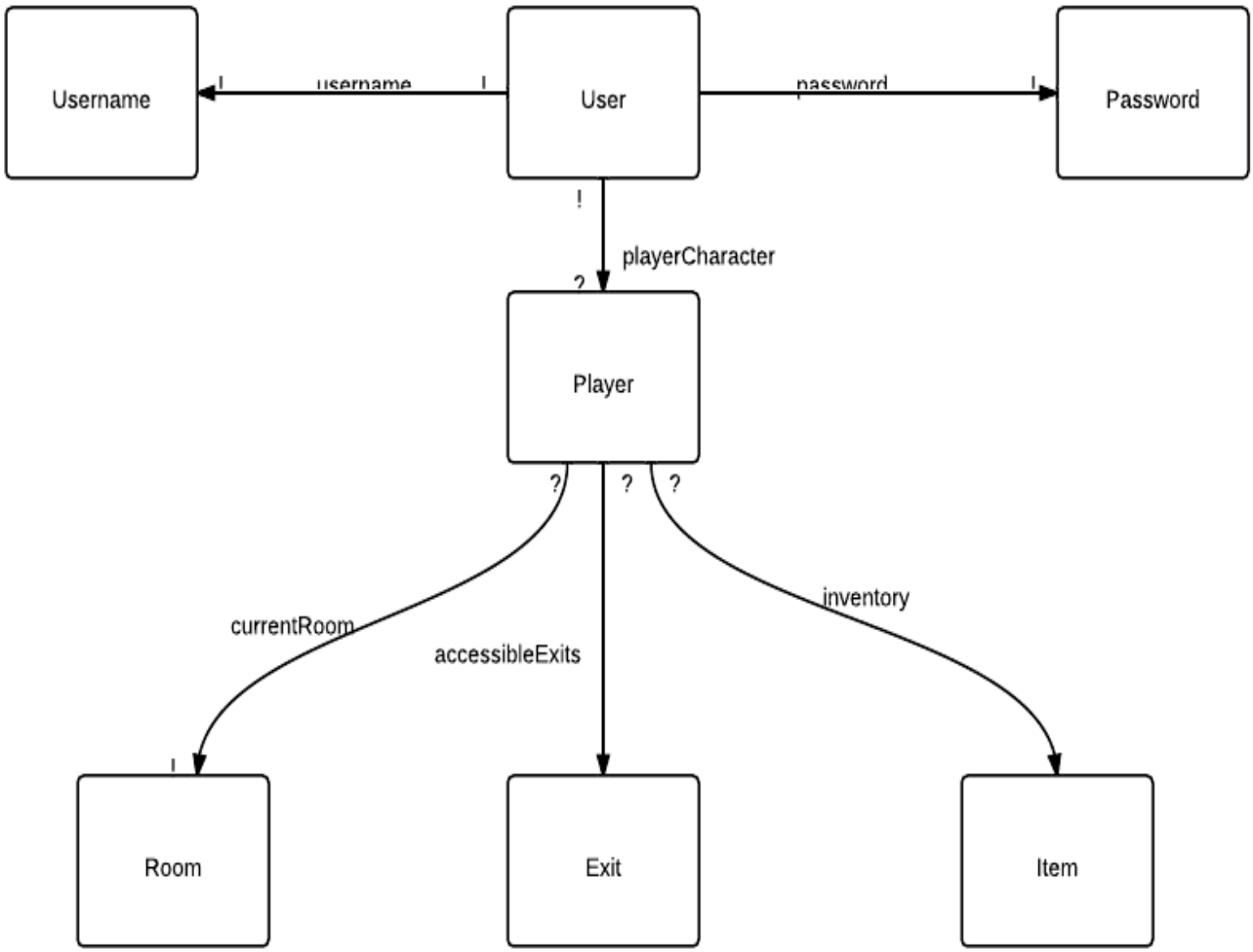
Designations

GameEntity is an abstract set representing the necessary attributes for anything in the game. At the very least, objects should have a name, a description, and an ID.

Items can be picked up by the player and used. They can either unlock other items or unlock exits, allowing the player to make progress through the game. The model does not show this, but in the application they are destroyed upon use.

Each location in the game is represented by a Room, which is connected to other Rooms by one or more Exits. There are actually two exit objects for any bidirectional connection between any Rooms A and B; the exit going from A to B and the one going from B to . This allows subtleties like making an exit locked on one side only.

Player State Object Model



Ambiguities Resolved

It was not specified how having multiple users playing concurrently should be implemented. I did this by creating a Player object for every game, and putting each game in its own “world” with a fresh set of game objects to interact with.

Complexities Ignored

Player state consists only of a room location, a set of exits that are accessible to the player, and their inventory. (In the application the set of exits was inferred from the room). The concept of any other player state (e.g. hit points, standing with NPCs, and so on) was omitted from the model and the game. It might be possible to add this in a future extension.

Designations

A User object represents a physical user who registers for the application and logs in to use it.

A Player represents a User’s character in a game. After a game is finished, the user can start a new game and another Player will be created. This model supports the concept of a User managing multiple Player characters, which could theoretically be implemented.

Rooms, Exits, and Items are described in more detail in the previous object model.