DESIGN RATIONALE: FIT2099 SSB ASSIGNMENT 1

GROUP NAME JAVAPLUSPLUS

GROUP MEMBERS MOHAMED SHAKEEL MOHAMED RAFI 28021452

MATTI HADDAD 29708966

KERRY YUE SONG ZHENG 28794346

This document relays the design rationale and thought process for the tasks that were assigned.

This document includes:

1. Leave Affordance
2. Give Affordance
3. Wand and Spell Class Implementation
4. Dementor (SubClass of HPActor) Implementation
5. Potions
6. Leave Affordance

After an item is 'taken', it needs to have the option to be 'left' as well.

Hence, leave needs to be a type of affordance, which in the code, will eventually give the actor the action of leaving it.

Once an item has been left by an actor, the Take affordance needs to be reassigned to the item so that an actor could take it again.

1. Give Affordance

If an actor is holding an item (ring, wand, dagger), the actor has the option of ‘giving’ the item to another actor if and only if both actors are in the same location and both are from the same team. In addition, the item can be rejected by the reciever, if so, it will stay with the actor that tried to ‘give’ it, this action consumes one turn.

The Give class is a subclass of HPAffordance that depends on:

* MessageRender to display a message to the user, a form of player feedback.
* HPEntetityInterface as it deals with entities and therefore it needs to be managed by EntityManager.
* HPActor as actors are the ones who initiate the action.
* HPAction as to check if the action can be performed.

In addition, it implements *HPActionInterface* as it needs to implement some methods from it, such as execute and getDuration.

1. Wand and Spell Implementation

Wand is an entity that can only

For a Spell to be Casted, there must be a Wand class and a Cast Action / Affordance.

Furthermore, Cast may target both entities and actors, which will require further distinction.

* 1. Wand and Cast Class

A Wand, like a dagger or any other item in the game, can be picked up by any Actor. The existence of a Wand allows an Actor to Cast Spells onto other Actors, or itself.

* 1. Spell Abstraction

Every spell needs to have a common target attribute, and a common method which denotes the spell's function.

Hence, abstraction of the Spell class allows subclasses to share that, and it makes it easier to Cast any subclass of Spell.

* 1. Cast and Actor’s Known Spells

Casting a spell requires a new action - Cast. This is different from Attack as it requires a Spell to be an input as well.

It will then execute the Spell's effect if the Actorknows the Spell.

Every actor will have a set of its known Spells

* 1. Targeting Items

In order to allow Spells to be Casted on items, the interface HPEntityInterface is used in Cast and Spell to target both subclasses.

Additional checks have to be implemented for individual spells to ensure that the target is the intended class type.

1. Potions

Potions are used to allow an actor to replenish a random or a certain number of hit points if and only if the actor’s health is not full. Potions cannot increase the actor’s health beyond its default health. furthermore, potions are immediately consumed when an actor moves to the potion’s locations and do not consume the actor’s turn.

Potions are a subclass of HPentity