Design Rationale : FIT 2099

**Door Class:**

Class Responsibility:

This class holds the ability to allow a player to pass through it only if the player has a key(an instance of the Key class) in their inventory. We decided to have unique keys for each door, therefore to enter a particular room we need its equivalent key(1 Door has 1 Key relationship)

We observed that the Door class is very much similar to the Ground class(which is a generalized(abstract) class of terrain types- eg: Floor class & Wall class) as it has all of the methods(Need to Override) we require for the Door class. (Most significant ones mentioned below)

The door class will therefore extend(inherit methods and attributes) Ground class present in the game engine. This will eliminate a lot of code that would otherwise have to be repeated(DRY principle followed)

Inherited Methods/Attributes:

+allowableActions(Actor actor, Location location, String direction)

+getMoveAction(Actor actor, Location location, String direction, String hotKey)

+canActorEnter(Actor actor)

+blocksThrownObjects()

Overridden Methods:

+canActorEnter(Actor actor) : Boolean

Check if the key in inventory matches the lock in the door. Returns True

Else: Return False

+blocksThrownObjects() : Boolean (Made True)

Addon Methods/Attributes:

-doorID:

+isDoorOpen(): bool

True if the door has been opened by the player before: So it doesnt need to be opened again.

Else: Return False (Indicating the Door has **not** been unlocked yet)

+keyMatches(): bool

True if the keyID attribute of the key in the player’s inventory matches the doorID.

Else: Player is not allowed to pass through the door (println: “Seems like you have obtained the wrong key, come back when you have the right one”)

**Key Class:**

In order to obtain a key, the player must defeat an enemy. After the enemy has been knocked out, it will drop a key. This key is used to open a door(Door Class gets involved here). Once the key is used, it should be removed from the players inventory.

We observed that we will need to use the Item Class as well as the Door Class, as it has the necessary methods we require to create the Key class(Few mentioned below). This way we will follow the DRY principle again.

The key class will generate a unique key(has attribute keyID which matches a doorID which is an attribute of Door Class) that will be given to an enemy

The key class will extend(inherit methods and attributes) Item class present in the game engine

Inherited Methods/Attributes:

+ int doorID (Door Class)

+

Overridden Methods:

+

Addon Methods/Attributes:

+ int keyID

**Goon Class:**

The Goon class will extend(inherit methods and attributes) Actor as it has similar behaviors as the Grunt class mentioned below:

Inherited Methods/Attributes:

+FollowActor() \_CHECK ON THIS

+AttackPlayer() \_CHECK ON THIS AS WELL

Overridden Methods:

+

Addon Methods/Attributes:

+ ArrayLists<Insults>

+ShoutInsult(): String

**Ninja Class:**

Description

Inherited Methods/Attributes:

+

Overridden Methods:

+

Addon Methods/Attributes:

+

**Q(NPC) Class:**

Description

Inherited Methods/Attributes:

+

Overridden Methods:

+

Addon Methods/Attributes:

+

**Doctor(Doctor Maybe) Class:**

Description

Inherited Methods/Attributes:

+

Overridden Methods:

+

Addon Methods/Attributes:

+

**RocketPad Class:**

Description

Inherited Methods/Attributes:

+

Overridden Methods:

+

Addon Methods/Attributes:

+

**StunBehavior Class: (Implements ActionFactory interface)**

Class Responsibility:

Inherited Methods/Attributes:

+

Overridden Methods:

+

Addon Methods/Attributes:

+

**insultBehaviour Class: (Implements ActionFactory interface)**

Class Responsibility:

Inherited Methods/Attributes:

+

Overridden Methods:

+

Addon Methods/Attributes:

+