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)

Relationships With  Other Classes:

We observed that the Door class would require attributes and methods from the Ground class(which is a generalized 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.

Methods/Attributes:

- Inherited attributes from Ground Class

+ Inherited Methods from Ground Class

Overridden Methods:

+canActorEnter(Actor actor) : Boolean

         If key is present in inventory, and the keyID of key in  inventory matches the doorID:

                   Returns True

         Else:

                   Return False

+blocksThrownObjects() : Boolean (Made True)

Addon Methods/Attributes:

-doorID: String

+isDoorOpen(): bool

If the door has been opened by the player before,  Return True. So it doesn’t need to be opened again

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

+keyMatches(): bool

If the keyID attribute of the key in the player’s inventory matches the doorID: True

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”),  Return False.

**Key Class:**

Class Responsibility:

In order to obtain a key, the player must defeat an enemy. After the enemy has been knocked out, the enemy will drop a key. This key is used to open a door. Each Key, which is an object of the Key class, has a attribute called the keyID which is used to uniquely identify each key. The keyID matches with a specific doorId so that, that key can be used only to open that Door. .Once the key is used, it should be removed from the players inventory.

Relationships With  Other Classes:

The key class inherits from the item class because when we thought about the implementation we realised that we needed all the attributes and methods used in item class, mainly, the displayChar attribute, the allowable actions attribute, the getAllowableActions method and the newInventoryItem method. This is because when we went through the code given for the Item class, the getAllowableActions and the newInventoryItem  method already implements a way for the player or enemy to pick up an item and drop an item . This will be useful to us when player defeats an enemy (Grunt,Goon,Ninja) and drops the key into the location where the enemy was.

The Key class has a one to one relationship with the door class and it is a dependency relationship because the door (a locked one) cannot exist without a key according to our understanding and our plans of implementation of the game.

Methods/Attributes:

- Inherited Attributes from the Item class.

+ Inherited Methods.

Overridden Methods:

None

Addon Methods/Attributes:

+ keyID: String

**Goon Class:**

Class Responsibility:

The Goon Class is used to implement a enemy character. The Goons can either attack the player with twice as much damage as Grunt or the Goons have a 10% chance of shouting insults at the player. The Goons also follow the player like Grunt.

Relationships With  Other Classes:

The Goon class will extend(inherit methods and attributes) Actor as it is an enemy with all required attributes of the Actor class. The Goon class also requires all the methods (like isConscious, hurt, getIntrinsicWeapon) in Actor class to be implemented properly in the game.

The Goon has two behaviors insultBehaviour and FollowBehaviour. The way FollowBehaviour is implemented for Goon is the same way that it is implemented for Grunt so it is not shown in our class diagram. However, the insultBehaviour is a new Behaviour so we showed it in our class diagram. The Goon class uses ActionFactory interface to generate actions to perform, we show this using the simple association relationship. The Goon Class also can hold a key which is shown by a simple association relationship. The Goon Class is also dependant on Action class because the Goon Class cannot exist without an Action to perform.

Methods/Attributes:

-Inherited attributes from the Actor Class

+ Inherited methods from the Actor Class

Overridden Methods:

+playTurn() - currently this method randomly chooses an Action to be performed by the user. This method should be overridden in a way that an appropriate action is performed at an appropriate time. Eg. Either follow the player,attack the player or insult the player depending on the players location.

+getIntrinsicWeapon() -  this method should be overridden so that the Intrinsic Weapon’s damage points and the name of it can be changed so that the damage of the Goon is twice of Grunt.

Addon Methods/Attributes:

None

**insultBehaviour Class:**

Class Responsibility:

The insultBehaviour class is used to print an insult only with a 10% chance at each turn. The way we decided to implement insultBehaviour is that it has an attribute called insults which is an arraylist of strings with different insults. Every time the insultBehaviour class is called, we iterate through the arraylist of insults and print it.

Relationships With  Other Classes:

The insultBehaviour class inherits from Action class as it is an Action that can be performed by the Player and it has all the required methods for us to implement the class. The class insultBehaviour, implements ActionFactory so

Methods/Attributes:

- Inherited Attributes from the Action Class.

+ Inherited Methods from the Action Class.

Overridden Methods:

**Ninja Class:**

Class Responsibility:

The Ninjas have the ability to stun the player but there is only a 50% chance of that happening. When the player is stunned he cannot perform any actions for two turns. After throwing stun powder at the player, the ninja moves back one step.

Relationships With  Other Classes:

The relationships of Ninja Class are the same relationships of the Goon class as they are both enemies the only difference is that Ninjas don’t have FollowBehaviour so we omit that relationship for Ninja. We plan on overridding the method of the Actor class for Ninja so that it is able to spot a player if player is within 5 squares of Ninja and Stun the player.

Methods/Attributes:

- Inherited Attributes from the Actor Class.

+ Inherited Methods from the Actor Class.

Overridden Methods:

+playTurn() - currently the playTurn method chooses a action randomly from a list of possible actions for the actor. We need to override it for Ninja in a way that there is only one action which is only performed if the player is within 5 squares of Ninja. There are two actions, throw stun powder and move one step back but it should be implemented as one for the sake of simplicity.

**Q(NPC) Class:**

Note:

Q inherits from actor class and the playturn method should be overridden so that the actions list for Q is a list of MoveActorActions chosen randomly. And the talk action and give plans action should be in the actions list accessed by the getAllowableActions  method which should be overriden as well.

When ever position of player changes we need to get allowable actions of player from depending on the surroundings. When actor moves to location of item the actor picks it up. When an actor is defeated the actor drops the item.

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:

+