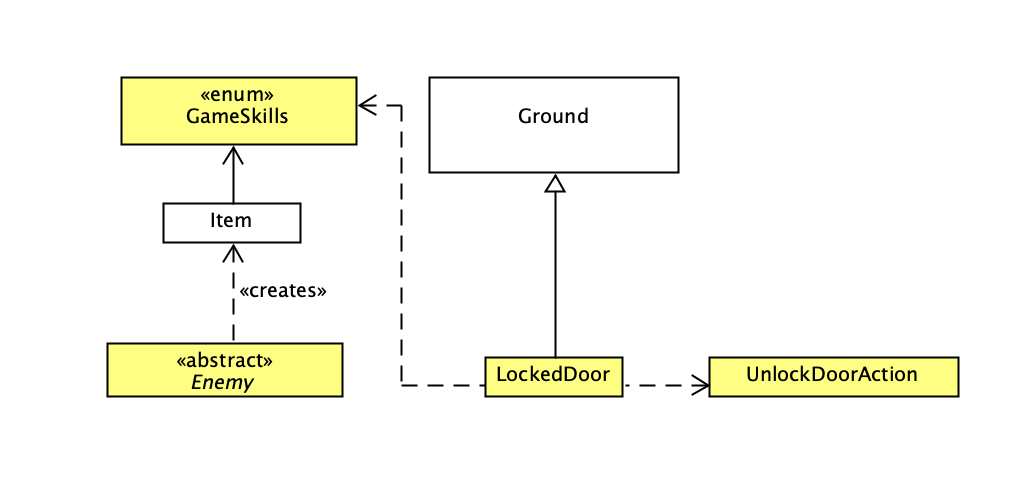
**Doors and keys**

Class Diagram:

Class LockedDoor

public class **LockedDoor** extends Ground

The LockedDoor class inherits from Ground and instantiates a new locked door.

How it works:

The locked door can be unlocked when the player has an Item key which contains the skill UNLOCKDOOR. The LockedDoor class also has an overridden method allowableActions which calls the class UnlockDoorAction.

Class UnlockDoorAction

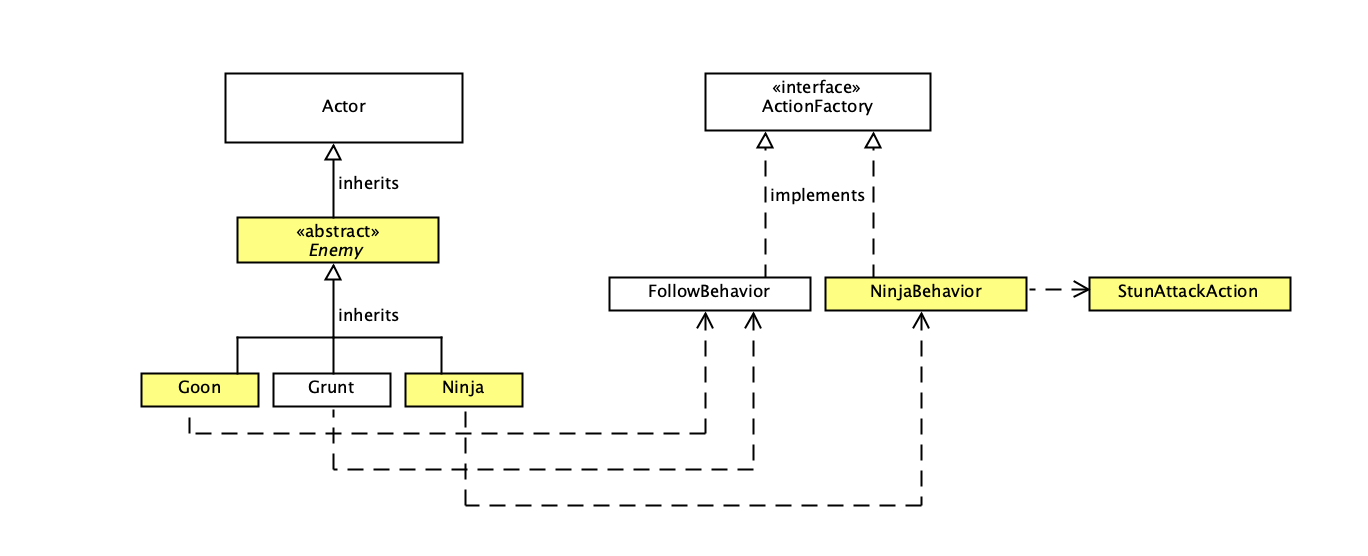
public class **UnlockDoorAction** extends Action

The UnlockDoorAction inherits from Action and has overridden methods execute and menuDescription.

Design Rationale:

The LockedDoor class inherits from Ground because it can override the existing methods in the Ground class.

**New types of enemies**

****

Class Enemy

public abstract class **Enemy** extends Actor

The Enemy class inherits from Actor and serves as a template for subclasses that inherit from it because all subclasses share a common set of instance variables and methods. The methods in this class include a method to create an Item key after an enemy is knocked out, a method to add the Item key into the inventory, an overridden playTurn method and a method which adds the behaviour of the enemy in the subclass.

Constructor

Enemy(String name, char displayChar, int priority, int hitPoints, ActionFactory behaviour)

Class Grunt

public class **Grunt** extends Enemy

The Grunt class inherits from Enemy and instantiates a new Grunt object. FollowBehaviour is added into the superclass’s constructor when instantiating a new object so that it follows the player. The Grunt class has an overridden method, getIntrinsicWeapon so that Grunt can slap the player.

Class Goon

public class **Goon** extends Enemy

The Goon class inherits from Enemy and instantiates a new Goon object. FollowBehaviour is added into the superclass’s constructor when instantiating a new object so that it follows the player. The Goon class has an overridden method, getIntrinsicWeapon so that Goon can shout insults at the player. The Goon class has a shoutInsult method which calls the getIntrinsicWeapon method at a 10% chance on each turn and shouts insults to the player by printing the insult onto the console.

Class Ninja

public class **Ninja** extends Enemy

The Ninja class inherits from Enemy and instantiates a new Ninja object. NinjaBehaviour is added into the superclass’s constructor when instantiating a new object.

Class NinjaBehaviour

public class **NinjaBehaviour** implements ActionFactory

The NinjaBehaviour class implements ActionFactory and creates a new Action in which the actor stays in one place unless the player is within 5 squares away from them which they then call the StunAttackAction class and the actor moves 1 space away from the player.

Class StunAttackAction

public class **StunAttackAction** extends AttackAction

The StunAttackAction class inherits from AttackAction. The class allows Ninja to stun the player. This class contains an int attribute counter and a method which checks if the player if stunned. The class also includes another method which instantiates a new WeaponItem stunPowderBag.

How it works:

If the player is not stunned, counter = 0.

If the player is stunned, counter = 1 or 2.

If the player is not stunned (counter = 0), a method which instantiates a new WeaponItem stunPowderBag is called which throws a bag of stun powder with a 50% chance of hitting the player. If the player is hit, the player cannot perform any actions other than waiting. Hence, the SkipTurnAction class is called and the counter is incremented by 1 (counter = 1). For the next round, since the counter = 1, this indicates that the player is stunned, the stun powder has no effect on the player and the SkipTurnAction class is called again. Then, the counter is incremented by 1 (counter = 2). Since the player can only be stunned for 2 rounds, in the next round, the counter will reset to 0 (counter = 0).

Design rationale:  
Firstly, the Enemy class has been declared abstract because this class serves as a template for subclasses that inherit from it. Furthermore, the Enemy class has been declared abstract as it should not be instantiated to create an Enemy object since it does not represent any specific type of enemy in the game.

Next, the Goon class, Grunt class and Ninja class inherit from the Enemy class and each class passes their specific behaviour (ie, FollowBehaviour, NinjaBehaviour) by calling the superclass’s constructor. The superclass, Enemy will then call a method which adds the behaviour passed through the constructor to the subclass’s objects. This reduces code and implements the DRY concept so that each subclass of Enemy do not have an addBehaviour method.

**Q**

Class Q

public class **Q** extends Actor

The Q class inherits from Actor and instantiates a new Q object.

How it works:

Q overrides the playTurn method to call the WanderBehaviour class so that Q wanders around the map. Q has a method which calls the TalkAction class to enable Q to talk. Q also has a method which calls the GivePlansAction class to enable the player to give the Item rocket plans and calls a method to create an Item rocket body which has the BUILDROCKETBASE skills when the player gives Q their rocket plans.

Class WanderBehaviour

public class **WanderBehaviour** implements ActionFactory

The WanderBehaviour class implements ActionFactory and creates a new Action in which the actor wanders around the map.

Class TalkAction

public class **TalkAction** extends Action

The TalkAction class inherits from Action. The TalkAction class overrides the menuDescription method to return the correct String based on whether the player has an Item with the skill GETROCKETBODY.

Class GivePlansAction

public class **GivePlans** extends Action

The GivePlans class inherits from Action and contains a method which allows the player to give Item rocket plans which contains the skill GETROCKETBODY and subsequently have the rocket plans Item be removed from the player’s inventory.

Design rationale:

The Interface ActionFactory has a method which is used by classes that implement it to give an Actor in the game a behaviour. Since Q wandering around the map is a behaviour, the WanderBehaviour class implements the Interface ActionFactory. Furthermore, WanderBehaviour implements ActionFactory for abstraction as WanderBehaviour has to implement the method in the Interface ActionFactory based on the requirements of its own class. WanderBehaviour implementing the Interface ActionFactory also introduces loose coupling between classes that implement ActionFactory which reduces dependencies and uses the ReD concept.

**Miniboss: Doctor Maybe**

Class DrMaybe

public class **DrMaybe** extends Actor

The DrMaybe class inherits from Actor and instantiates a new DrMaybe object. DrMaybe contains a method which creates an Item rocket engine after being knocked out and a method to add the rocket engine into the inventory.

Design rationale:

**Building a rocket**

Class RocketPad

public class **RocketPad** extends Ground

The RocketPad class inherits from Ground and instantiates a new rocket pad. The class overrides the allowableActions method which checks for each item in the player’s inventory if it has the skills BUILDROCKETBASE, BUILDROCKETTOP. Then it calls the BuildRocketAction class.

Class BuildRocketAction

public class **BuildRocketAction** extends Action

The BuildRocketAction class inherits from Action and instantiates a new rocket pad. The class has a method which allows the player to build a rocket when the player has items in their inventory which have the skills BUILDROCKETBASE, BUILDROCKETTOP.

Design rationale:

**Others**

Enum GameSkills

public enum **GameSkills**

GameSkills is an enum consisting of skills that can be added to Item objects. The enum of skills include UNLOCKDOOR, BUILDROCKETBASE, BUILDROCKETTOP, GETROCKETBODY.

Class Application

public class **Application**

The Application class contains the driver and instantiates objects for the game. The Application class has 1 GameMap object, namely, startMap which contains the map that the player starts on. The Application class has walls with a locked door which creates a room that contains a DrMaybe object and another room with rocket plans. The Application class instantiates 1 Player object, two Grunt objects, 2 Ninja objects, 2 Goon objects, 1 Q object, 1 DrMaybe object, 1 Item rocketPlans that has the skills GETROCKETBODY in a locked room.