Doors and keys

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.

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 inherits from it because all subclasses share a common set of instance variables and methods. The methods in this class include a method to create a key after an enemy is knocked out, a method to add the key into the inventory, an override playTurn method and a method which adds the behaviour of the enemy in the subclass. Subclasses which inherit from the Enemy class will have to include the behaviour of its newly instantiated object into the constructor to initialise the 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 calls 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 contains a method which checks if the player is stunned. If the player is not stunned, a method is called which will instantiate a new WeaponItem stunPowderBag which throws a bag of stun powder with a 50% chance of hitting the player. The player will be stunned for 2 turns and the player cannot perform any actions other than waiting. If the player is stunned, the stun powder has no effect on the player.

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.

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.

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.

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.

Explain why new classes are added

1. Why is enemy class created? And why is it abstract ?
   Based on the project requirement, we have a total of 3 different enemies. The fact that they're all ENEMY objects suggests that they will be implementing similar method, therefore we have designed a new abstract enemy class that implements DRY to reduce duplicated codes as much as possible.

   The purpose of the abstract ENEMY class is to provide a base template that mulitiple The methods within the abstract ENEMY class can be overidden by the subclasses

Explain how the new classes will interact with the existing system and deliver their functionalities

Doors and keys

The new classes added to implement a locked door are the classes LockedDoor and UnlockDoorAction. The new classes interact with the existing system by inheriting and overriding their superclass (which are the classes Ground and Action respectively)'s methods. The overridden LockedDoor method canActorEnter() calls the Actor's class's getInventory() to check if the actor has an Item with the GameSkills.UNLOCKDOOR. If yes, then return true else, return false. The overridden allowableActions method in LockedDoor calls the canActorEnter method in its own class and calls the UnlockDoorAction class if the actor can enter. The UnlockDoorAction class overrides its superclass's methods including execute, menuDescription and hotkey to its suitable value. The Item key is instantiated in the Enemy class in its getInventory() method when the Enemy.isConcious() is false.

New types of enemies

The new classes added to implement instantiate new enemies are an abstract Enemy class, Goon class and Ninja class. The Enemy class creates a new Item key when

Refer to the interaction diagram

Explain how the proposed system will work and why

The proposed system is a text based game that consists of multiple game maps which requires the player to knock out enemies and obtain items to

complete the game. The player will first be instantiated in a new game map whereby they have to knock out an enemy to obtain a key to unlock a door. The unlocked door will lead the player to a new destination whereby they will obtain a rocket plan. The player has to knock out another enemy to obtain a key to gain access into the locked room which contains the miniboss. After knocking out the miniboss, the player would have rocket plans and a rocket engine in their inventory. In order to build a rocket, the player has to trade their rocket plans with Q to obtain a rocket body. The player then has to search for a rocket pad so that they player can build a rocket with the rocket body and rocket engine on the rocket pad.

Interaction diagrams
http://www.cs.unc.edu/~stotts/145/CRC/Interactions.html
http://www.agilemodeling.com/artifacts/sequenceDiagram.htm
https://creately.com/blog/diagrams/sequence-diagram-tutorial/#WhatIs

Proposed system means explaining what you are going to do this project. What is your project and what is new in your project other than existing things. And how you are going to do this. In short proposed system is explaining your project

Explain how the new classes will interact with the existing system
Explain how (existing & new) classes will deliver the functionality

https://greenbay.usc.edu/IICMSw/practice.tech.sys_and_sw_arch_development.base/tasks/analyze_the_proposed_system_7CEFD45C.html

Work Breakdown Agreement of Team Kimchi

Subtasks	Due date	Description	Partner(s) responsible for producing the content	Partner(s) responsible for reviewing the content
Class Diagrams	16/4/2019	To produce UML class diagrams using UMLET showing the interaction between classes in the engine and game package	Ye Zea	Megan Ooi
Sequence Diagrams	17/4/2019	To produce diagrams showing how objects interact with each other in time sequence	Megan Ooi	Ye Zea
Design Rationale	18/4/2019	Document the overall responsibilities of classes	Megan Ooi	Ye Zea
	18/4/2019	Explain how the proposed system will work and why	Ye Zea	Megan Ooi
	17/4/2019	Explain how the new classes will interact with the existing system	Ye Zea	Megan Ooi
	18/4/2019	Explain how (existing & new) classes will deliver the functionality	Megan Ooi	Ye Zea

I, Hew Ye Zea, hereby accept this work breakdown agreement.

I, Megan Ooi Jie Yi, hereby accept this work breakdown agreement.

Notes
public abstract class Enemy extends Actor
@override
public Action playTurn(Actions actions, GameMap map, Display display) {

```
    return actions.get(rand.nextInt(actions.size()));
}
```

public class Ninja extends Enemy
- Has attribute WeaponItem stunPowder = new Weapon(String name, char displayChar, int damage, String verb);

public class Goon extends Enemy
- Has attribute IntrinsicWeapon shoutInsults = new IntrinsicWeapon(int damage, String verb);
- Has attribute Speech //this is to shout insults after attacking with intrinsic weapon

public class Grunt extends Enemy
- Has attribute IntrinsicWeapon slap = new IntrinsicWeapon(int damage, String verb);

public class Q extends Actor
-

public enum gameSkills (refer to demoSkills)
- UNLOCKDOOR
- BUILDROCKET

Notes: to print a: kick feature, b: attacks feature etc in player.showMenu()
player.playTurn() calls show menu, playTurn() in Actor
World class world.processActorTurn() calls player.playTurn()
World.run() calls world.processActorTurn() calls player.playTurn()

I think spitbehaviour extends action and implements actionfactory because actionfactory creates an action object but spitbehaviour also wants to implement action abstract methods

actionFactory creates an Action object

In world when processing actor, each item in inventory will get allowable actions so key actions constantly called even when i dont want to

12/4/19- added UnlockDoorAction

Design Rationale
https://docs.oracle.com/javase/9/docs/api/java/lang/String.html
https://docs.oracle.com/javase/9/docs/api/allclasses-noframe.html
https://www.academia.edu/12695666/Snake_game_documentation

The classes added into the system are:

Enemy /
Goon /
Ninja /

Q /
DrMaybe /
RocketPad /
WanderBehaviour /
TalkAction /
GivePlansAction /
UnlockDoorAction /
StunAttackAction
LockedDoor /
NinjaBehaviour /
StunAttackAction /
GivePlansAction /

An enum added into the system is:
GameSkills

Checklist

WBA (done)
UML class diagram (done)
Interaction diagram
Design rationale

- Why add the new class
- Functionality (roles & responsibilities) (done)
- How new classes relate & interact with existing system
- How (existing & new) classes will deliver the functionality