Class Diagram Design Rational

1. Enemy:

Extends Actor.

* Enemy acts like a parent class of various enemy creatures, like grunts, goons, ninjas or the mini-boss. Enemy uses ActionFactory to make actions, such as attack, move or drop item. Enemy can deal damage to player and follow the player if they are ought to. When an enemy is defeated, a key may drop on the ground where the enemy was defeated.
* Moreover, if the mini-boss is defeated, player will receive a rocket plan as the prize of defeating the boss.
* Following behaviours such as following the movement of player might be a property of some type of enemy.

Advantages:

The use of enemy can make adding new enemy type trivial, since they may have common attributes and methods.

Disadvantage:

An enemy may not necessary need certain action (e.g. Doctor Maybe may not need to move), wasting memory space or impact time complexity.

1. Rocket:

Extends Item.

Rocket is a subclass of item which the player can build it using a rocket engine and a rocket body. Player can find the rocket plan in a locked room and ask Q for rocket body in return. A launch pad is a special ground where player can build rocket on it. The aim of the game is to build the rocket and exit the place.

Advantages:

uses item utilities, in order to execute operations like drop or pick up items. If new features added in item, all rocket class and its parts can utilize them.

Disadvantage:

Rocket must be built on launchpad therefore making it an item might be unnecessary. The rocket class does not have to exist in the first place, since the game ends once the player meet all the conditions of building rocket. But for the sake of further development, we decided to make it an object.

1. NPC:

Extends Actor.

None -Player-Actors are there to give out tasks, talk, and exchange items with players. The only NPC in this game is Q, but more NPCs may be added in the future development. NPCs have their own behaviour, they may be wondering around the map, or sit in a certain place, or they may follow the player.

Advantages:

NPC has its own behaviours, some need the method and attributes in actor to achieve them, making actions through ActionFactory. Wandering around the map can be achieved using the similar way as following behaviour.

Disadvantage:

The only NPC in this game is Q, which is OK to put it just under Actor, saving memory and work. However, in other to be more extendable in future development, we made the NPC class and made it an interface so that we can implement every new NPC easily.

1. Doors and keys:

* Doors are designed as a piece of ground that player cannot passes through without a corresponding key. There are other ways of implementing it, for example making it a subclass of Floor or Wall. The reason we designed in this way is that some of the methods in Ground class are more suitable for implementing doors than other two classes. Doors are only allowing access bidirectionally, either left and right or up and down.
* Key is a child class of item, because most of the item activities (drop, pick, etc.) are useful in key class. Keys are dependent on doors since there can be doors that do not need keys, but there is no point having a key while no door is present. Key may be consumed after the door is open, or it can be used multiple time and never expire.

Advantages:

Ground is providing a moderate base of building a door, it has methods that can be adapted to check if the player is eligible to cross. Doors are bidirectional to prevent violate usage of it.

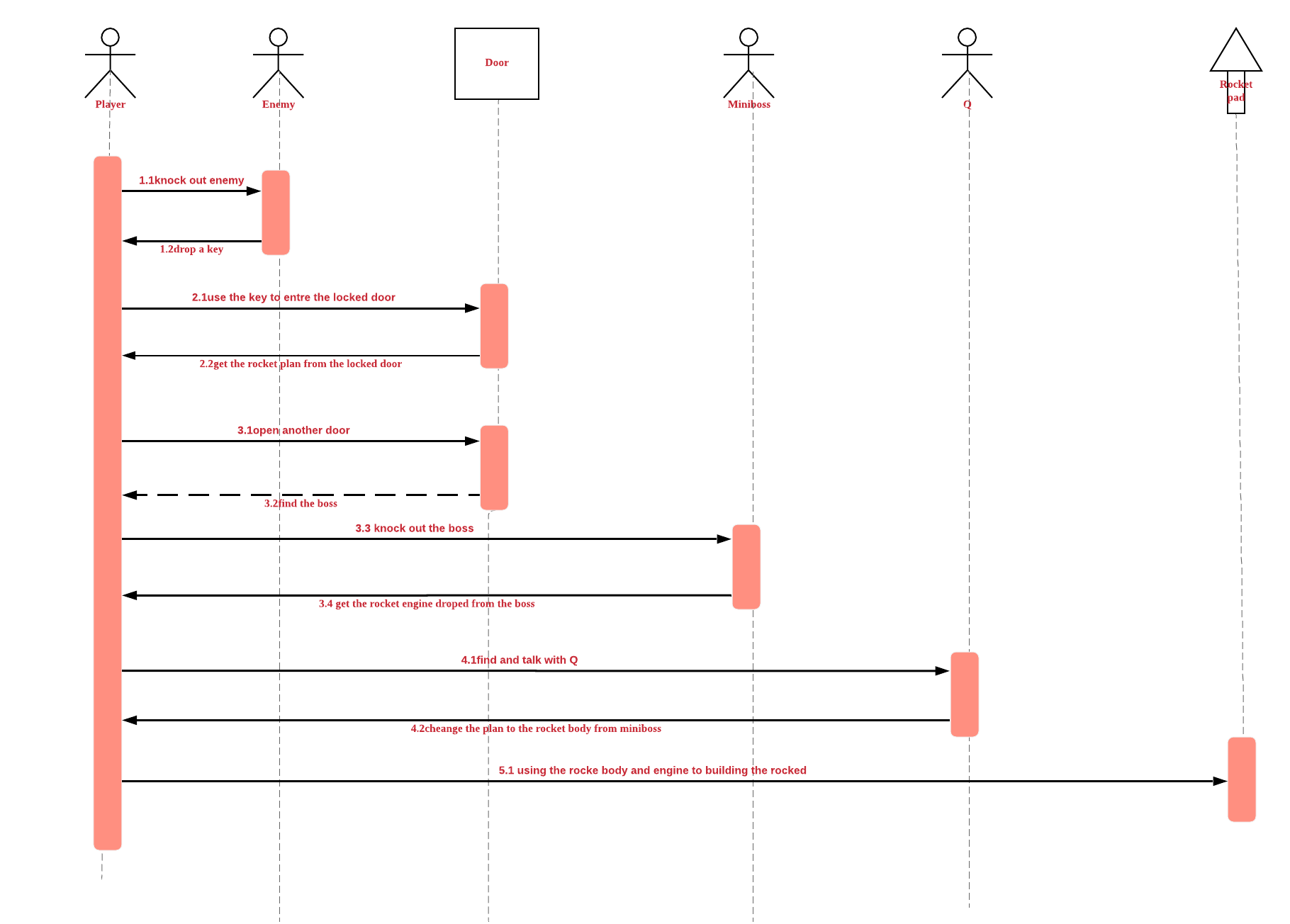
Disadvantage:

Doors are usually surrounded by walls, which is not likely to let players pass through. Making it bidirectional might be a waste of time. But there is a potential of bug which we do not want to take the risk.

In the development process we discovered some disadvantages of our previous design, we try to fix them by changing our design.

We decided to add some enemy behaviour such that for certain type of enemy, they have their own behaviour (e.g. ninja). For certain action such as talk and exchange item, we made it a subclass of action so we can directly take the advantage of action class, this made it easier for us to control the allowable actions. We changed the dependency on key and door as we decided a key can open every door in this dungeon.

A new version of player is used in this game for the propose of stun feature, using the old player is not enough to fulfil the requirement as there is no way to implement the stun function.



Rational of interaction diagram

The diagram above shows us the process of this game, at first, there are two locked doors on the ground, player can not access without holding the key, the way to get key is knocking down the enemy and the key will be dropped from enemy randomly. After getting the key, we can access one of the locked doors and get the rocket plan. Then access the other door to find the mini-boss(3.2 we have to open the door and verify the boss)we also need to get the rocket engine by knocking it out. After these, by carrying the rocket plan, go to find the NPC Q, then talk to him, we will change our rocket plan to body, now we get everything of building the rocket, so the final step is to put body and engine on the rocket pad to complete this game.