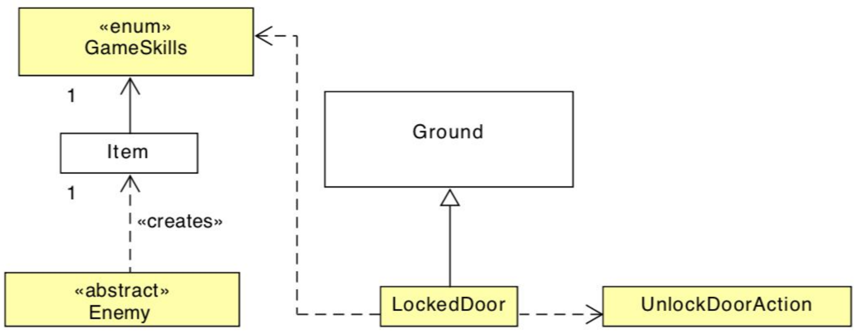
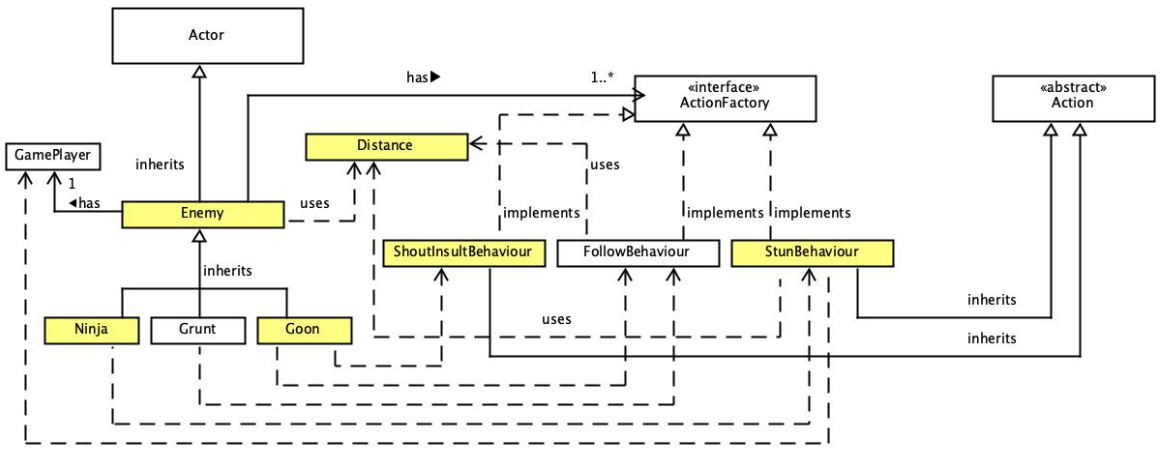
Updated class diagrams

Doors and keys



Based on the **class diagram** above, the Enemy class creates an Item, key in a method. The key has the skill GameSkills.UNLOCKDOOR. The locked door can be unlocked if the player has a key with GameSkills.UNLOCKDOOR. So, the LockedDoor class depends on GameSkills. If the door can be unlocked, the LockedDoor class has an overridden method allowableActions that calls the class UnlockDoorAction.

New types of enemies

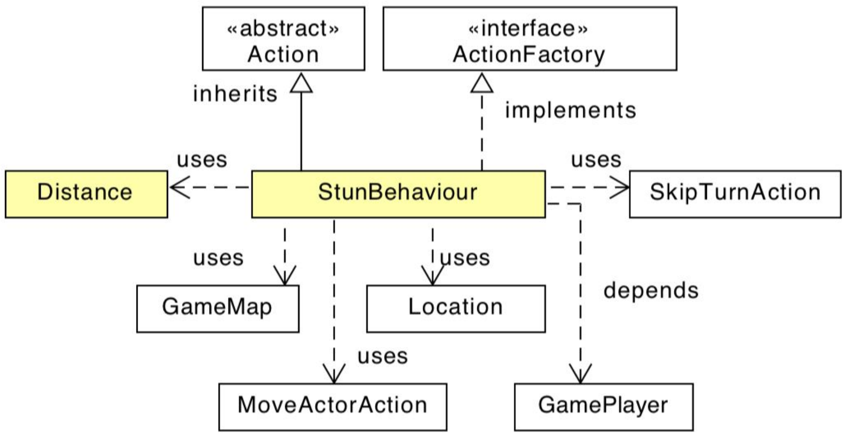


Based on the **class diagram** above, the abstract Enemy class inherits from the Actor class. The Enemy class has an attribute of GamePlayer type. The Enemy class uses the Distance class's method to check if it is adjacent to another Actor.

The Ninja, Grunt and Goon class inherit from the Enemy class. The Enemy class has a List attribute of type ActionFactory that stores the behaviours of the enemies. Grunt class adds its behaviour FollowBehaviour through its superclass's addBehaviour method. Goon class adds its behaviour FollowBehaviour and ShoutInsultBehaviour through its superclass's addBehaviour method. Ninja class adds its behaviour StunBehaviour through its superclass's addBehaviour method.

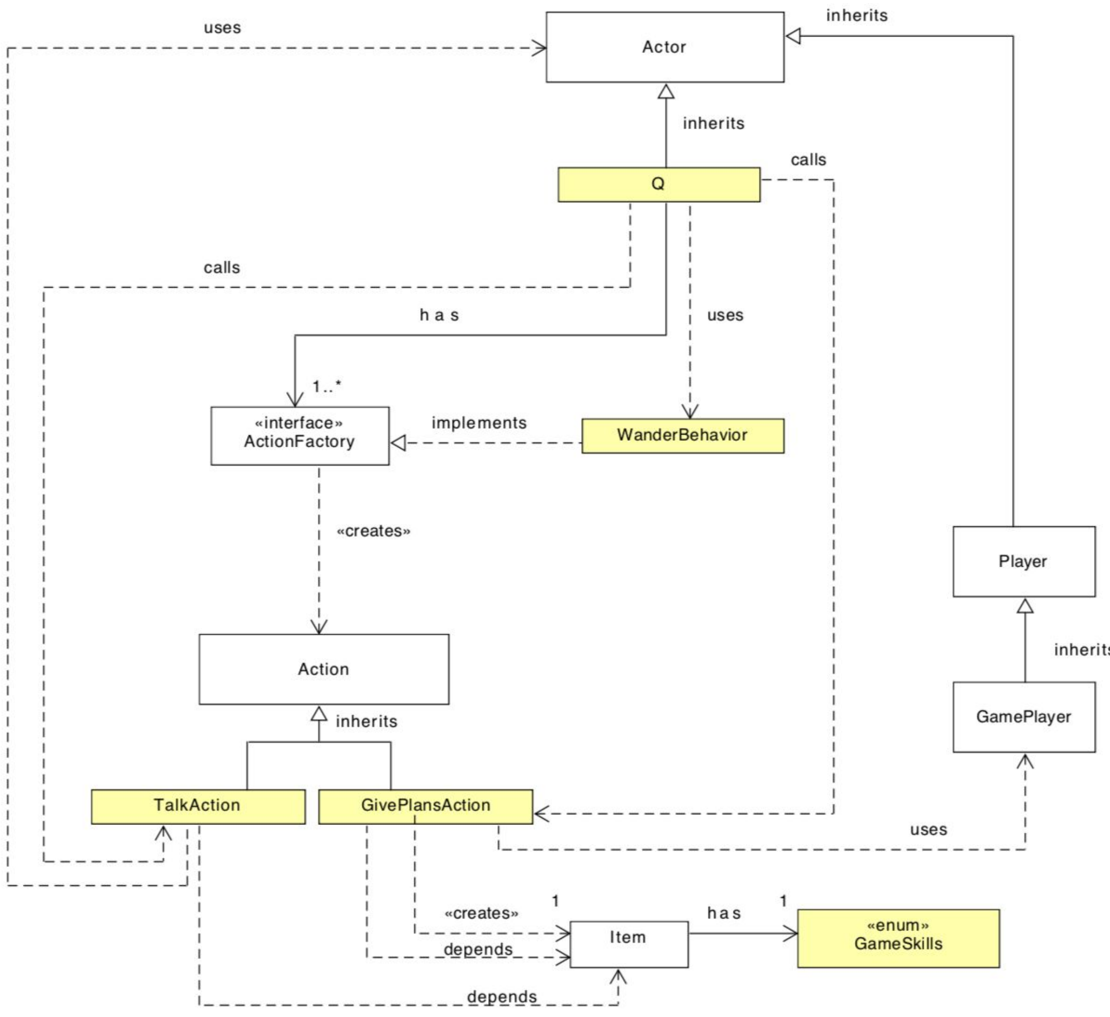
The classes ShoutInsultBehaviour, FollowBehaviour, StunBehaviour implements ActionFactory as these classes represent behaviours of the actors that use it. The ShoutInsultBehaviour class and StunBehaviour class inherit from Action. The StunBehaviour and FollowBehaviour class use the Distance class to get the distance between the Actor and the Player. The StunBehaviour class has a dependency with the GamePlayer class.

Class StunBehaviour



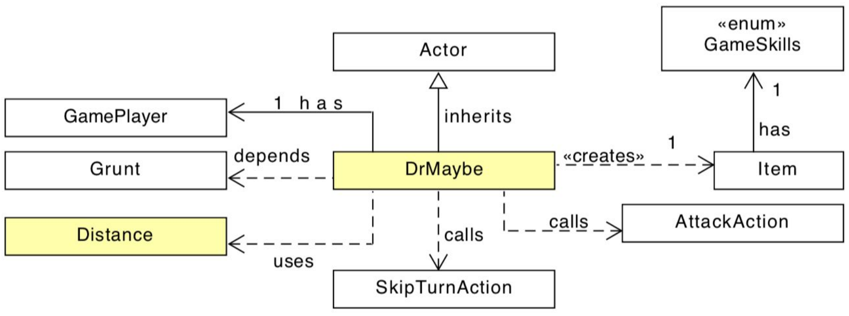
Based on the **class diagram** above, the StunBehaviour class inherits from Action and implements ActionFactory. It uses the Distance class to get the distance between the actors. It uses the Location class to get the current location of the Actor object and the GamePlayer object to check if there are terrains that block thrown objects between them. If the distance is less than or equals to 5 squares apart and there are no terrains that block thrown objects, the StunBehaviour class calls its own execute method and returns a MoveActorAction to move away. If the distance is more than 5 squares apart or there are terrains that block thrown objects between them if the distance is less than or equals to 5, the StunBehaviour class calls the SkipTurnAction class to stay in one place and do nothing.

Q



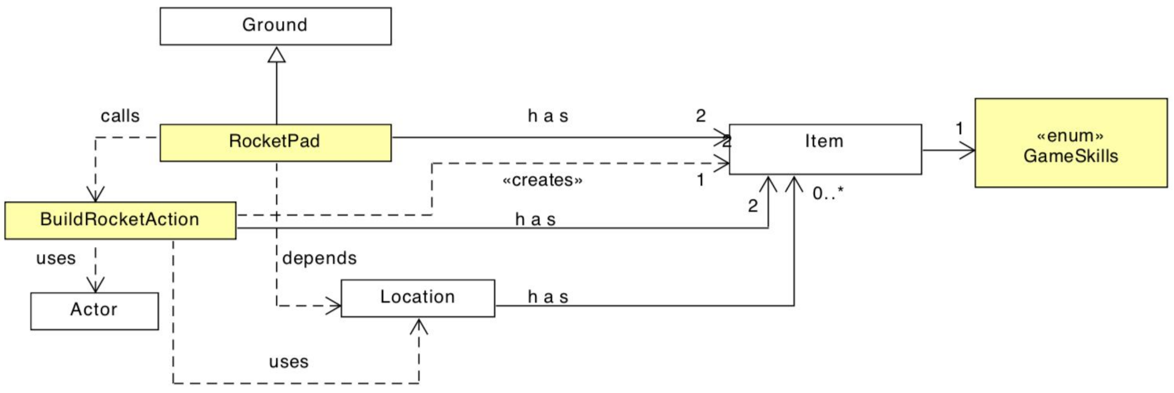
Based on the **class diagram** above, the Q class inherits from Actor. It has a List of type ActionFactory and calls the WanderBehaviour class through the addBehaviour method. This allows Q to wander around the map. Q has an overridden method getAllowableActions which calls the TalkAction class to enable Q to talk and calls the GivePlansAction class to enable the player to give the Item rocket plans.
The TalkAction and GivePlansAction classes check if the player has rocket plans which has GameSkills.GETROCKETBODY. The GivePlansAction class uses the GamePlayer class as it will remove the rocket plans item and creates an Item rocket body that is added to the player’s inventory. If the player successfully gives their rocket plans, Q will disappear from the GameMap with a cheery wave.

Miniboss: Doctor Maybe



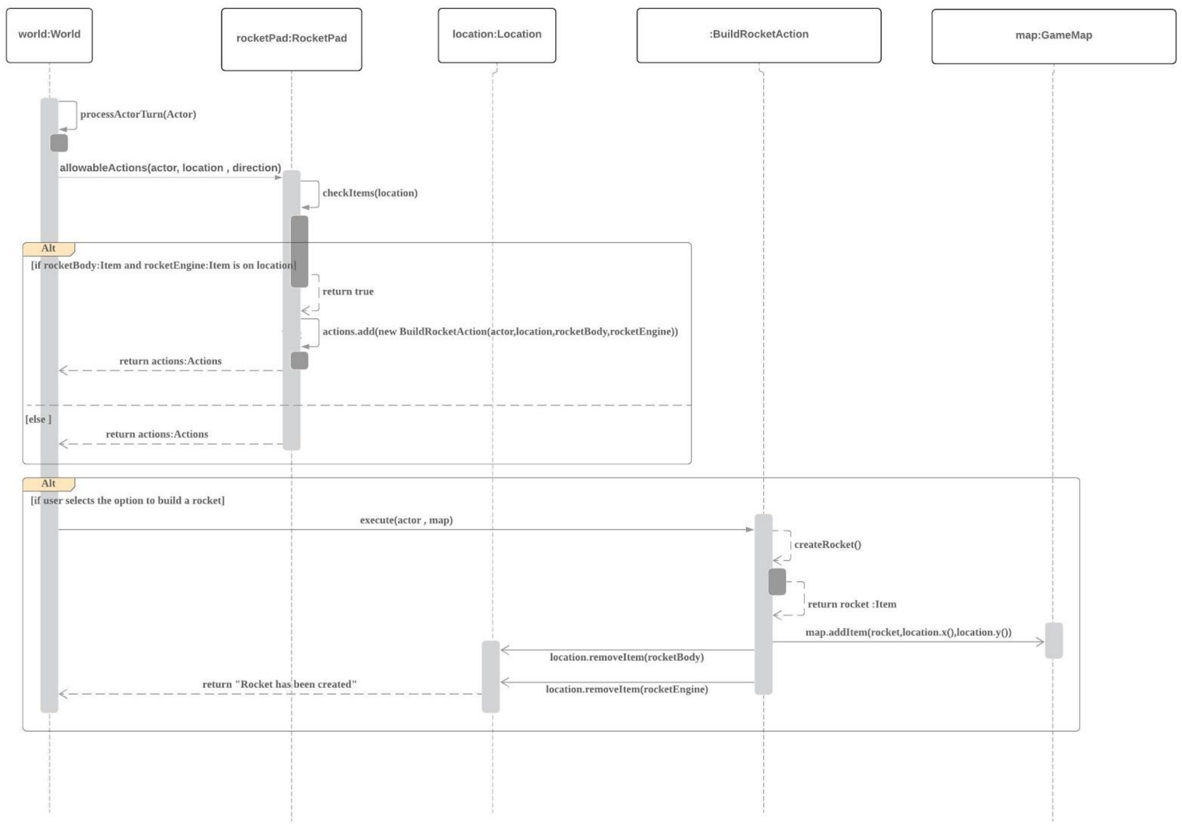
Based on the **class diagram** above, DrMaybe inherits from Actor. The DrMaybe class has a method that creates an Item, rocket engine. The rocket engine has GameSkills.BUILDROCKETBASE. DrMaybe has a GamePlayer attribute. It depends on Grunt's static variables for its own hitPoints and damage as Doctor Maybe's hitPoints and damage are half of Grunt's. It uses the Distance class to check if the player is adjacent to it. If it is, it will call the AttackAction class to attack the player. If the player and Doctor Maybe are not adjacent, it calls the SkipTurnAction class and does nothing.

Building a rocket



Based on the **class diagram** above, RocketPad inherits from Ground. The RocketPad class has 2 attributes of type Item. It depends on the Location class and checks if there are items with GameSkills.BUILDROCKETBASE and GameSkills.BUILDROCKETTOP. If yes, the RocketPad class calls the BuildRocketAction class. The BuildRocketAction class has a method which creates a new Item object called rocket. It uses Actor to return the correct menu description.

Interaction diagram (for building a rocket item):



world:World
rocketPad:RocketPad
location:Location
:BuildRocketAction
map:GameMap
processActorTurn(Actor)
allowableActions(actor, location , direction)
checkItems(location)
Alt
[if rocketBody:Item and rocketEngine:Item is on location]
return true
actions.add(new BuildRocketAction(actor,location,rocketBody,rocketEngine))
return actions:Actions
[else]
return actions:Actions
Alt
[if user selects the option to build a rocket]
execute(actor , map)
createRocket()
return rocket :Item
map.addItem(rocket,location.x(),location.y())
location.removeItem(rocketBody)
location.removeItem(rocketEngine)
return "Rocket has been created"

The interaction diagram above illustrates how an Item object, rocket is built.

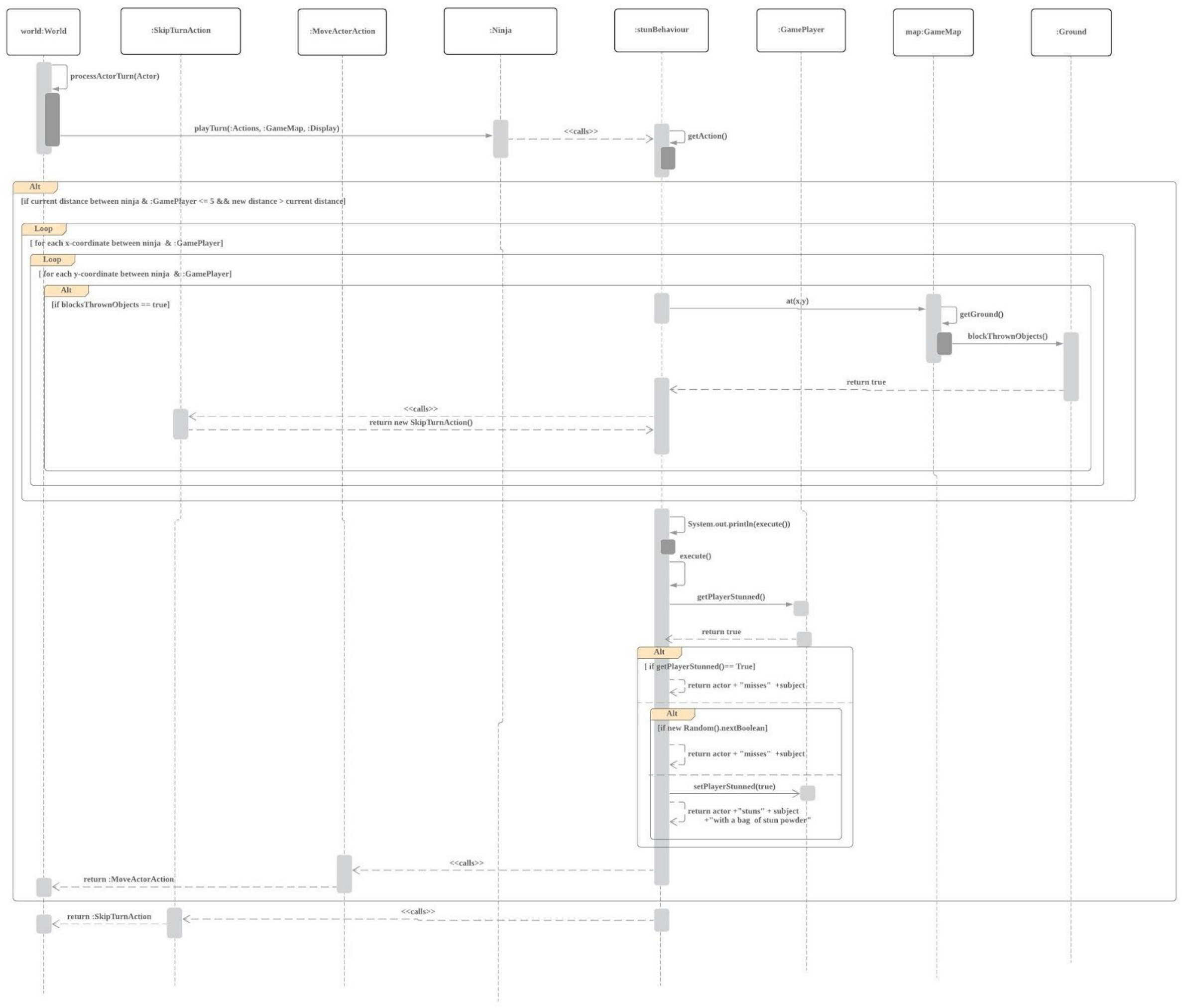
If the player is adjacent to the rocket pad, it calls allowableActions(actor,location,direction) in the rocket pad class, the program first checks if the following items are placed on the rocket pad :

1. Item that has the GameSkills.BUILDROCKETBASE (namely Item rocketEngine)
2. Item that has the GameSkills.BUILDROCKETTOP (namely Item rocketBody)

If the location of rocket pad contains both a rocket body and a rocket engine, a newly instantiated BuildRocketAction is added into a list of actions and the list of actions is returned.
However, if either or both of the items are not present on the location of rocket pad, an empty list of actions will be returned.

If the condition to build a rocket is fulfilled, a build rocket option will be printed on the menu. A rocket will be built and added on the location of rocket pad as a furniture item when user selects the build rocket option. Once the rocket is created, both rocket body item and rocket engine item will be removed from the location of rocket pad. The process ends with a message, “Rocket has been created”.

Interaction diagram (for ninja to stun the player) :


world:World
:SkipTurnAction
:MoveActorAction
:Ninja
:stunBehaviour
:GamePlayer
map:GameMap
:Ground
processActorTurn(Actor)
playTurn(:Actions, :GameMap, :Display)
<<calls>>
getAction()
Alt
[if current distance between ninja & :GamePlayer <= 5 && new distance > current distance]
Loop
[for each x-coordinate between ninja & :GamePlayer]
Loop
[for each y-coordinate between ninja & :GamePlayer]
Alt
[if blocksThrownObjects == true]
at(x,y)
getGround()
blockThrownObjects()
return true
<<calls>>
return new SkipTurnAction()
System.out.println(execute())
execute()
getPlayerStunned()
return true
Alt
[if getPlayerStunned()== True]
return actor + "misses" +subject
Alt
[if new Random().nextBoolean]
return actor + "misses" +subject
setPlayerStunned(true)
return actor +"stuns" + subject +"with a bag of stun powder"
<<calls>>
return :MoveActorAction
<<calls>>
return :SkipTurnAction

The interaction diagram above illustrates how ninja:Ninja performs stun attack action on player:GamePlayer

In the first phase, it uses the Distance class' distance method and checks for the following conditions :
1. The current distance between ninja and player is less than or equals to 5
2. The new distance between ninja and player is greater than the current distance

If the distance between ninja and the player is less than or equals to 5, it checks if there's a terrain that blocks thrown objects between ninja and player. If yes, it will return SkipTurnAction and ninja does not attack.

If there's no terrain that blocks thrown objects, it checks if the player is stunned. If player is already stunned, the ninja misses the player this round. If player is not stunned, ninja will attempt to stun attack the player at a 50% success rate. Then, it calls the MoveActorAction class to move one space away from the player.

In the event where the current distance between the ninja and player object is more than 5 squares apart, Ninja calls the SkipTurnAction class and does nothing.