Lab 3 Team 4
Design Rationale
Yo Kogure

Requirement 8: Vendor

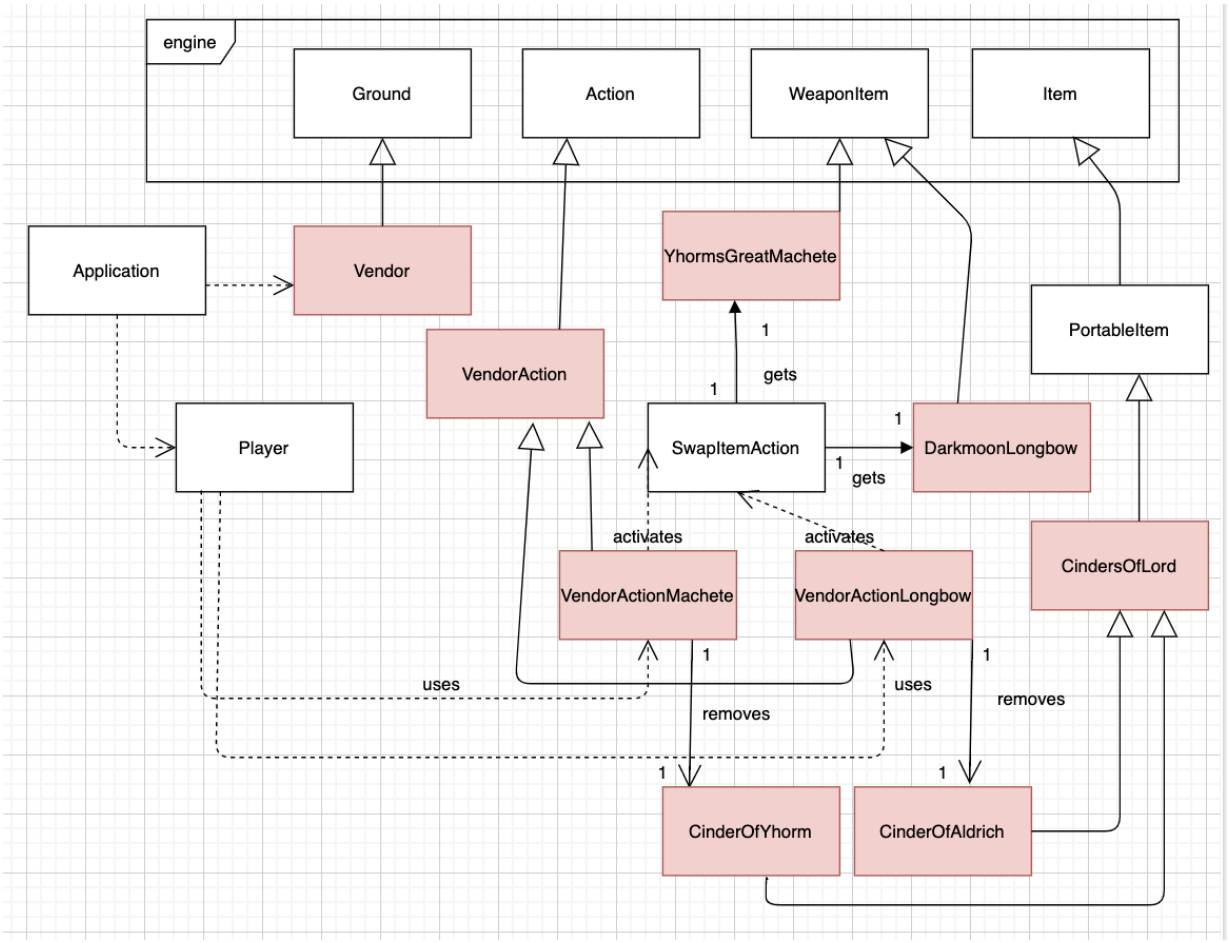
Assignment 3 Update:

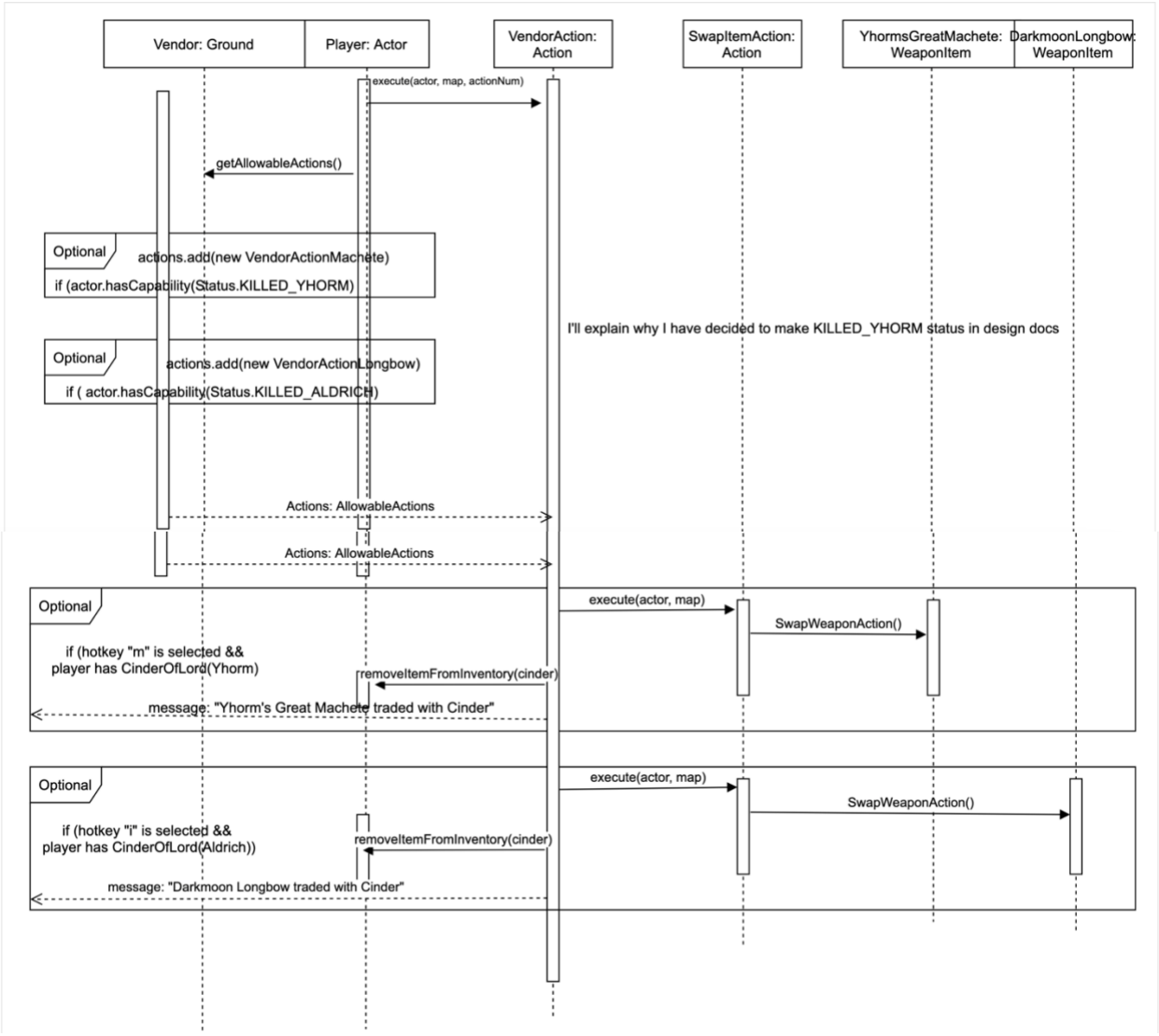
I have added to git the newest part for my Vendor. The new parts introduced here are in Red, and those are VendorActionMachete and VendorActionLongbow. They work similarly with previous VendorActions, and very few modifications were needed as my codes were easy to re-work with. One thing to note in my design is that, Vendor will show VendorActionMachete as an allowable action, when player has a capability called KILLED_YHORM. The other option was to iterate through inventory to find CinderOfYhorm. However, considering that this is a game and I am a hard RPG game fan, I would like to add it as an Achievement – that I have beaten a boss. When I want to go wild in the future, I could simply refer to KILLED_YHORM to see the player is onto a next stage. Of course, considering cases that the player has forgotten to pick up the Cinder, or has already paid Cinder, I have added special codes in VendorActionMachete itself to make sure the player has CinderOfYhorm in the inventory. The same application goes for CinderOfAldrich.

Speaking of Cinders, CinderOfYhorm and CinderOfAldrich both are items which are CindersOfLord, which inherit from PortableItem which derive from Item.

I was wondering between whether to add a variable in CindersOfLord to differenciate between Cinders that came from Yhorm and Aldrich. But since I would like to add many bosses in the future and I want to keep Cinders consistent, I have decided to make multiple childs of it to act as a portable item which proves that the boss has beaten.

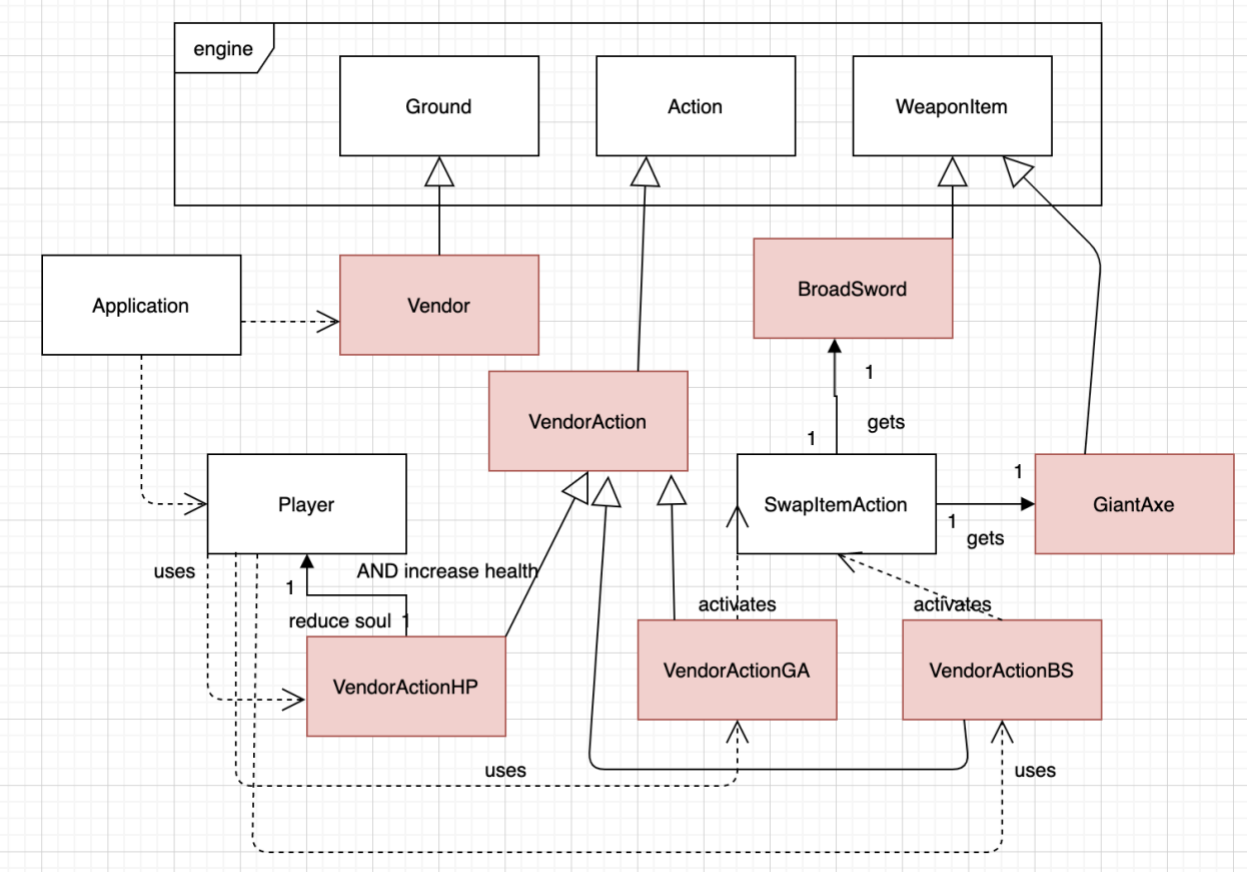
And then, as we can see, VendorAction works pretty similar to the previous ones in Interaction diagram. One thing to note is that the item checks works inside the Action to give out necessary message like “You have killed the boss but do not have Cinder with you!”
After that, it will remove from inventory the necessary Cinder and concludes the programme.
Thank you for marking!


engine
Ground
Action
WeaponItem
Item
Application
Vendor
YhormsGreatMachete
PortableItem
VendorAction
1
gets
1
Player
SwapItemAction
1
DarkmoonLongbow
1
gets
activates
activates
CindersOfLord
VendorActionMachete
VendorActionLongbow
1
1
uses
uses
removes
removes
1
1
CinderOfYhorm
CinderOfAldrich



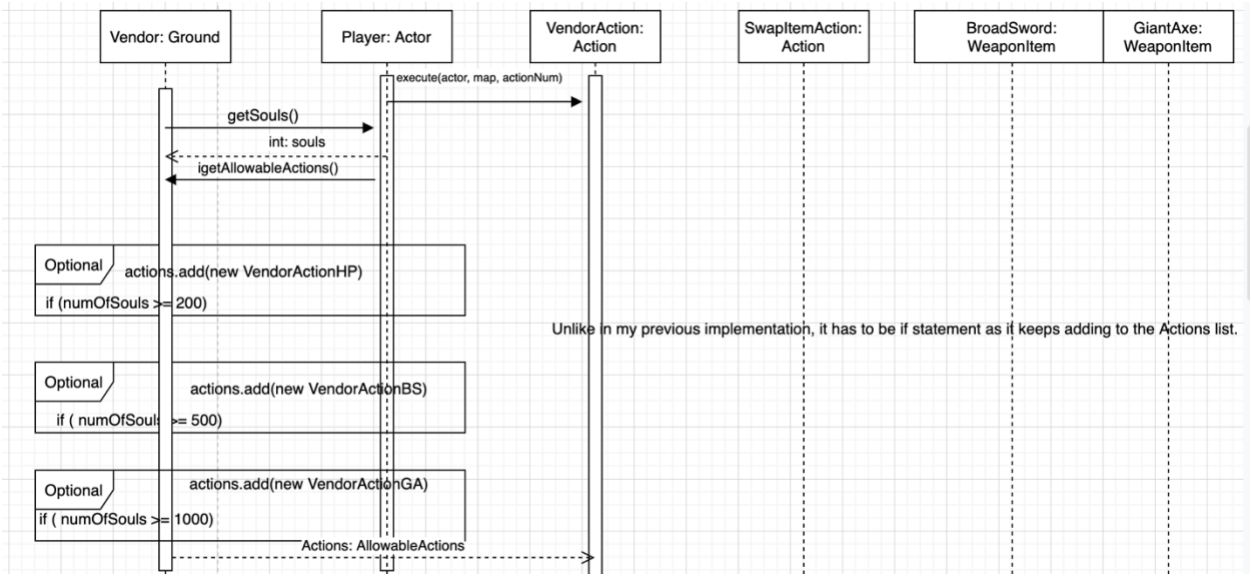
Assignment 2 Update:

There are a couple of changes that I made after considering assignment 2, and I would like to explain why I made those changes.



First change is that I made 3 different VendorActions◯◯ which inherit from VendorAction class which extends Action. This is for better design, and I could use super() in my codes to store default values specialized in the VendorActions. VendorActionBS for BroadSword, VendorActionGA for Giant Axe, VendorActionHP for Hit Points.

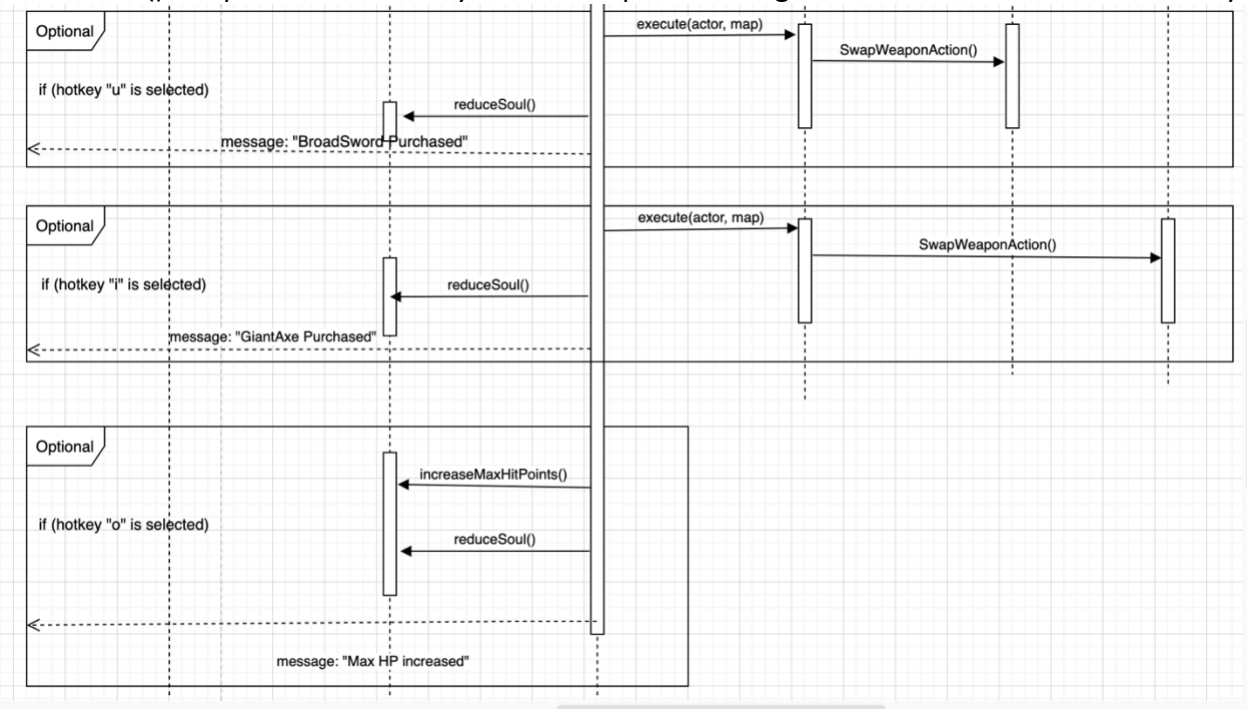
And then, Vendor inherits from Ground, and we use the getAllowableActions() in Vendor class to obtain list of Actions which is possible by the Player. This way, only the options that are buyable is shown to the menu console, and we can easily append new VendorAction into the Actions thanks to the engine codes built on it.



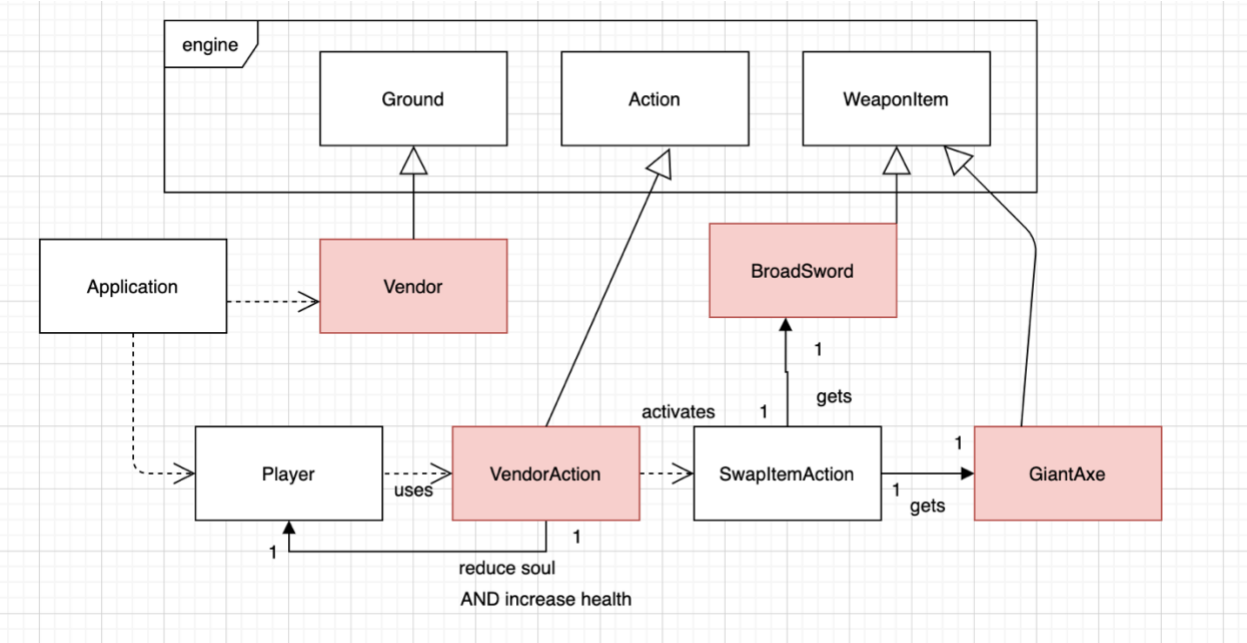
This way, I should use if statement instead of if else alternatives, since we might add one options or 2 or 3 options, depending on souls that the player has.

Another thing to note here that I used to have actionNum, 0-2 in my earlier diagram, however, we will actually detect which VendorAction has been selected by the hotkey that the user put. The classes that detect and deal with the input are deep under engine code, but depending on what is selected out of the Actions returned by getAllowableActions(), the player could execute() VendorAction accordingly. As expected, we use SwapWeaponAction() as it is responsible for adding and removing Weapons. As for WeaponActionHP, we can directly alter the player's health easily. And finally, after any one of those 3 actions selected, we must

reduceSoul() the price from the Player. This is a perfect design that is efficient and user-friendly.



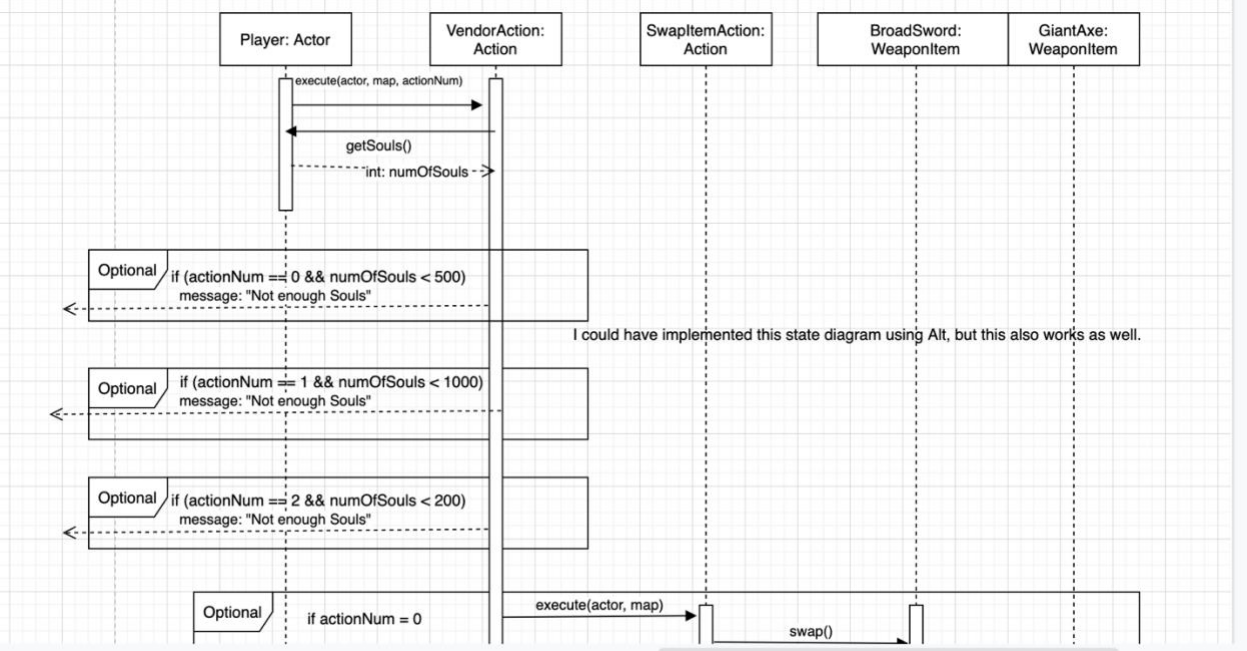
Assignment 1:



The souls can be traded to buy a new weapon and to upgrade the Player's attributes (stats) through a vendor. When the Player buys a new weapon, the weapon in the current inventory will be automatically replaced with it. Replacing the weapon will cause the old weapon to be removed from the game(drawn from the specification sheet).

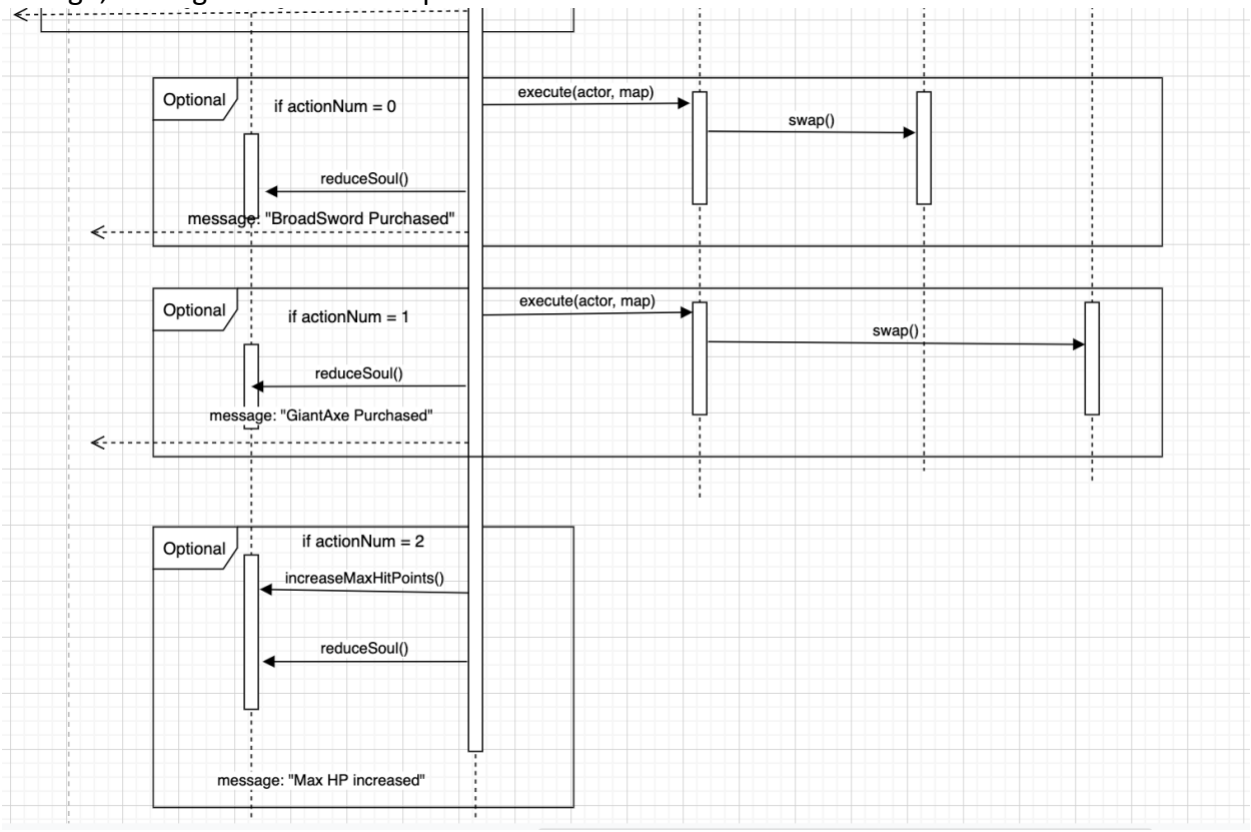
The class diagram of Vendor is pretty simple. I have made Vendor to be inherited from Ground, so it will sit as a type of ground in the Firelink Shrine.
Then, player calls something called VendorAction which inherits from Action. Well, it could have been something called VendorManager just like ResetManager, but I wanted it to use the current template that Action class has, so selected this method.

Vendor is just an entity on the ground.
Application will depend on Player to trigger VendorAction. Detailed coding later on, but I will override execute() method in VendorAction, to accept another argument called integer actionNum.
actionNum 0 will represent purchasing BroadSword.
actionNum 1 will represent purchasing GiantAxe.
actionNum 2 will represent purchasing maximum Hit Points of a player. There is no option to buy StormRuler and Yhorm'sWeapon, as they are not inside the Vendor options.



So, Actor will first execute VendorAction which inherits from Action.it will call getSouls() to obtain the number of souls that player has. Right after that process, we will have to perform checks to see if player has sufficient souls. All of these are written as if statements, depending on which actionNum the player chose. Also note that actionNum is always either 0, 1 or 2 and it is made sure in the earlier part of the process when executing.

If player has insufficient souls, it will return corresponding error message, and if they have enough, it will go on to the next part.

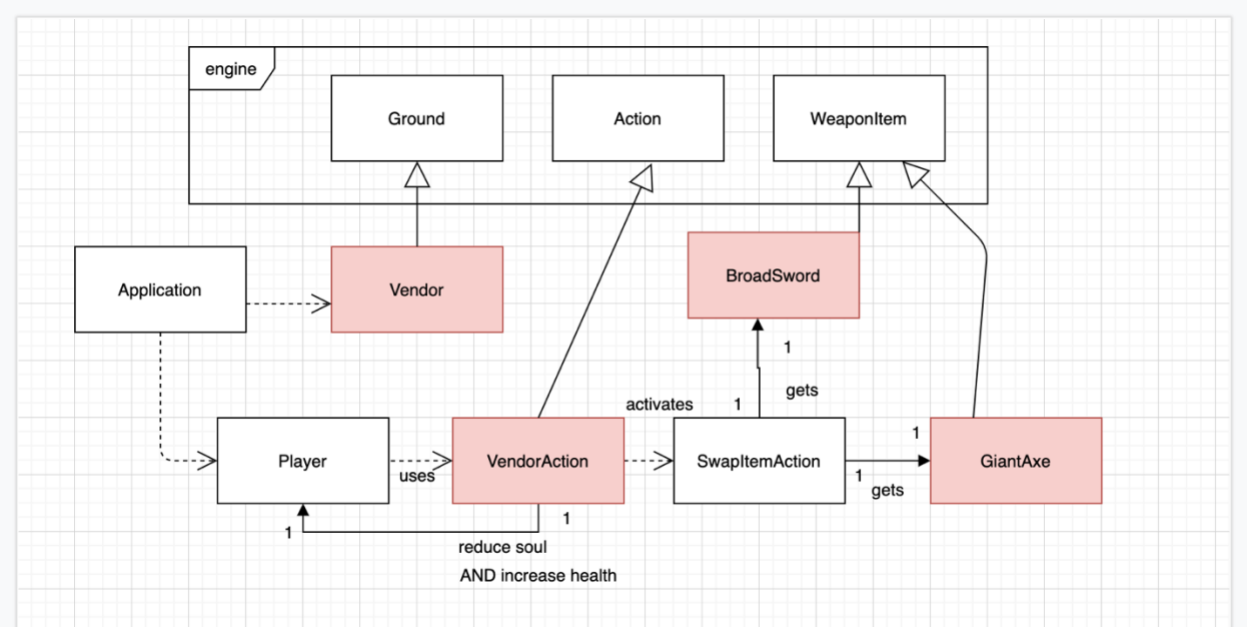


If actionNum – 0, purchase BroadSword is selected, since I have made sure they have enough souls, it will execute() another action called SwapItemAction. This is an action defined already in game file, and is used to obtain new weapon/replace old weapon with new one. Therefore, it will have an association 1 to 1 from SwapItemAction to BroadSword and GiantAxe. After performing weapon swap by swap()(detailed method will be implemented later), it will call reduceSoul() to reduce corresponding soul price as a payment(500). It will also give out message that it has been purchased successfully.

Very similar interactions for the case if actionNum – 1, except it is for buying GiantAxe of price 1000. Since if-else statements and if statements have no difference in this specific interaction diagram, I have decided to apply Optional, to help us visualize better for 3 conditions.

Now if actionNum = 2 is selected, it will increase the maximum hitPoints of the player, and also reduces 200 souls as a payment. Lastly, it will message completion message and concludes the

end of Vendor interaction diagram.



All associations in this part is 1 to 1 as each one of their object only associates with one of the other. VendorAction also associates with Player, because it will have to call its method to reduce soul and also to increase health.