Lab 3 Team 4
Design Rationale
Yo Kogure

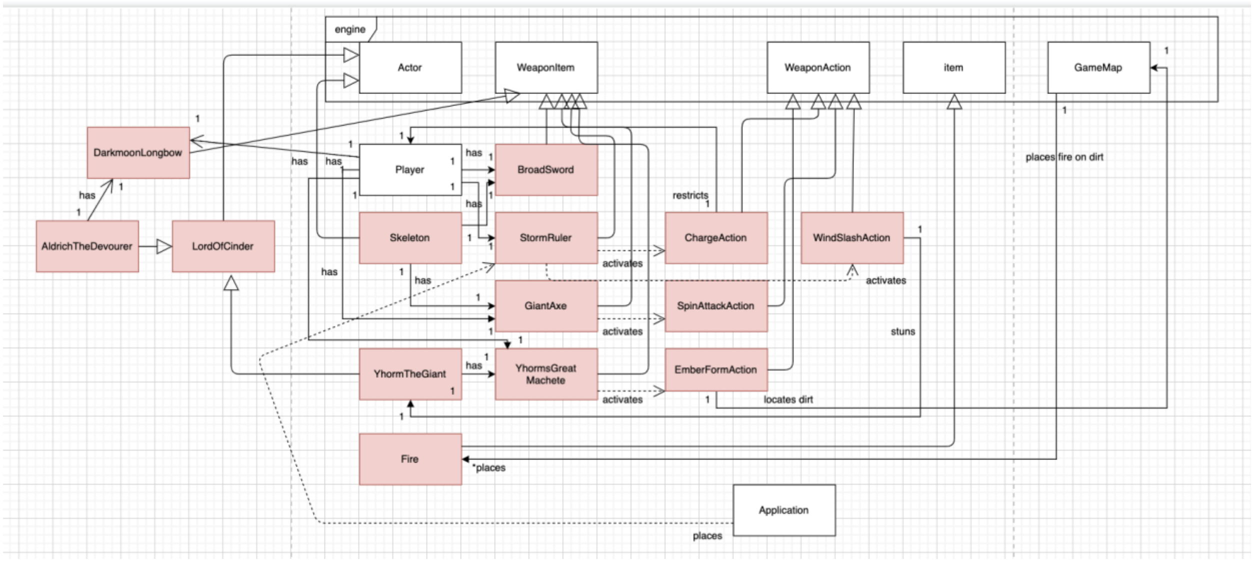
Requirement 7: Weapons

Assignment 3:
There are several changes made to this in Assignment 3. I have just updated the UML diagram for weapon since it is easier to see this way.
The new addition to this is AldrichTheDevourer here which takes control of DarkmoonLongbow, a WeaponItem. The reason Aldrich is inherited from LordOfCinder and not Actor nor Enemy class, is because they can be categorized as LordOfCinders, and in future implementation it is easier to expand by making a child of Lords, any time we want to add another boss.

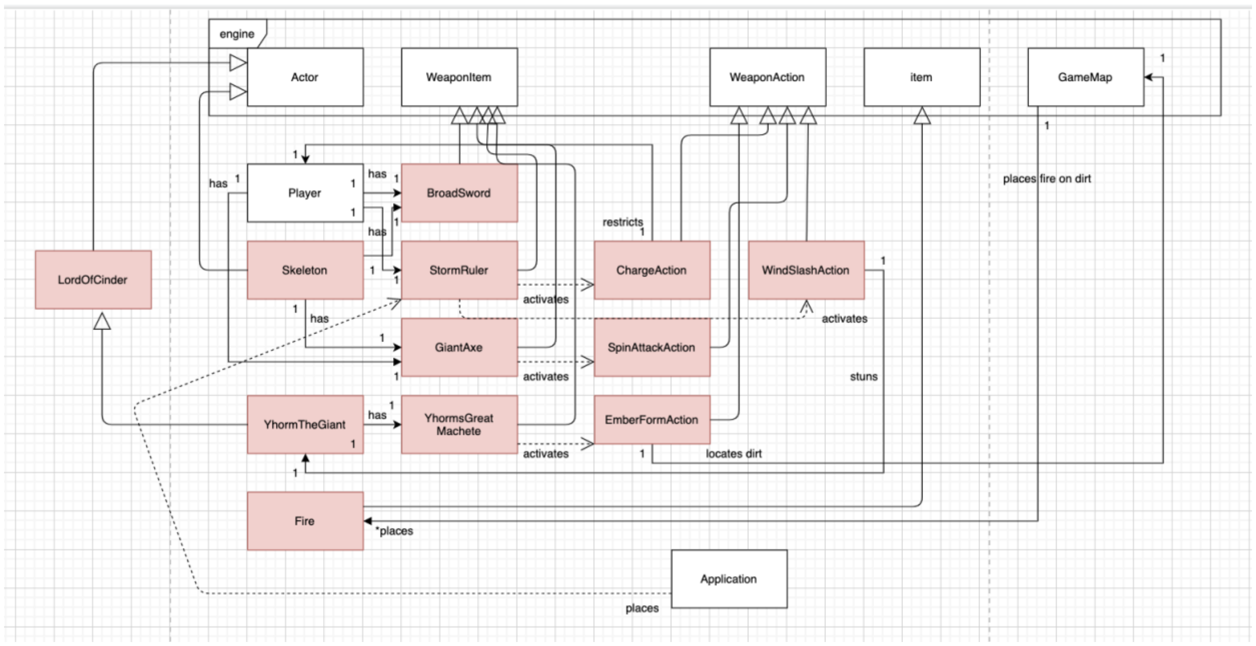
Another addition is an association from player to DarkmoonLongbow and YhormsGreatMachete. Although not visible here, I have made changes to the code to allow special abilities of Machete to be used by player without problem. The player can use the EmberFormAction any time to burn the ground, which recognizes the holder of the weapon and makes necessary adjustments.

Although DarkmoonLongbow is specified to have its own EmberForm in the future, it is optional so is out of my implementation here.

Since there is no changes in Interaction diagrams, this concludes the end of my Weapons.



Assignment 2:
There are few changes that I made in Assignment 2 for Weapons, and I would like to briefly explain why in this design rationale.

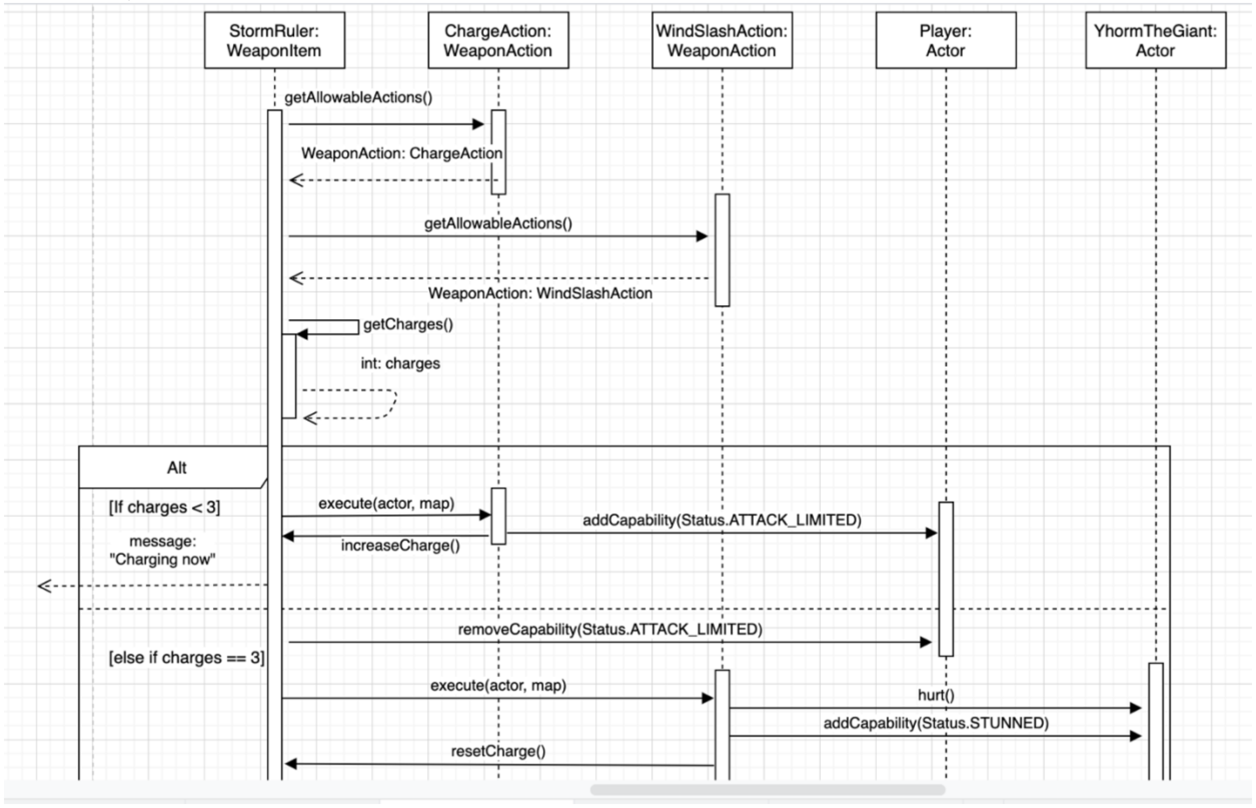


Change 1)
I made Fire inherit from Item instead of Ground. This is because, changing from dirt to fire is not permanent; rather temporary only lasting for 4 turns. Furthermore, Item have tick() and ways to detect Actors on top, so I figured that is a better approach.

Change 2)
I made YhormTheGiant inherit from LordOfCinder thanks to Ms. Bush's feedback. Now this is following the principles better and in the future we could add more LordOfCinders with different characteristics.

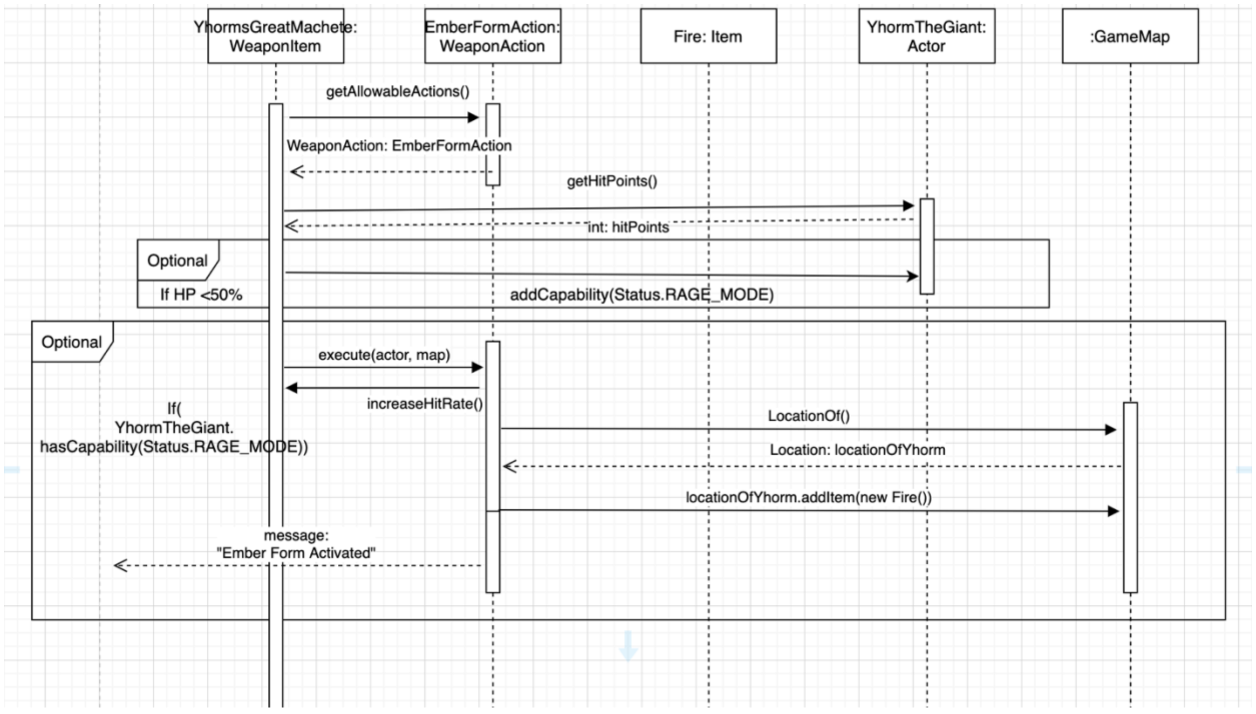
Change 3)
I deleted StormRulerOnGround, which act as an item for StormRuler. This is because, StormRuler inherits from WeaponItem in the first place which implements the functionality of Item as well. Therefore, I could simply use Application to place StormRuler next to Yhorm at the beginning of the game.

Other than that, no major change in the UML diagram as the initial approach was quite close to better implementation.

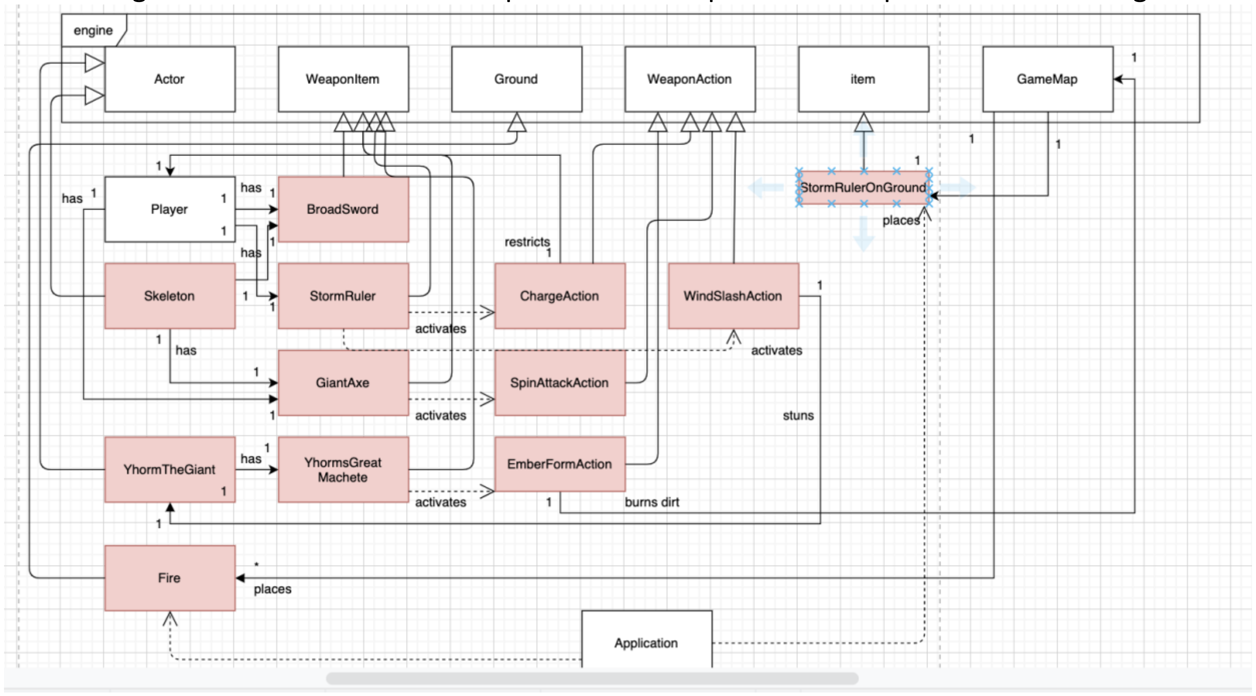


And then for StormRuler, most things stay the same. We use hurt() and resetCharge() methods for smoother implementation.

Lastly for Ember Action, I made better approach by adding status RAGE_MODE. This lets us track Yhorm better as the condition of Yhorm is now accessible from multiple classes, if only given the appropriate Yhorm object.
And we add new Fire() object in the location of the yhorm when needed.

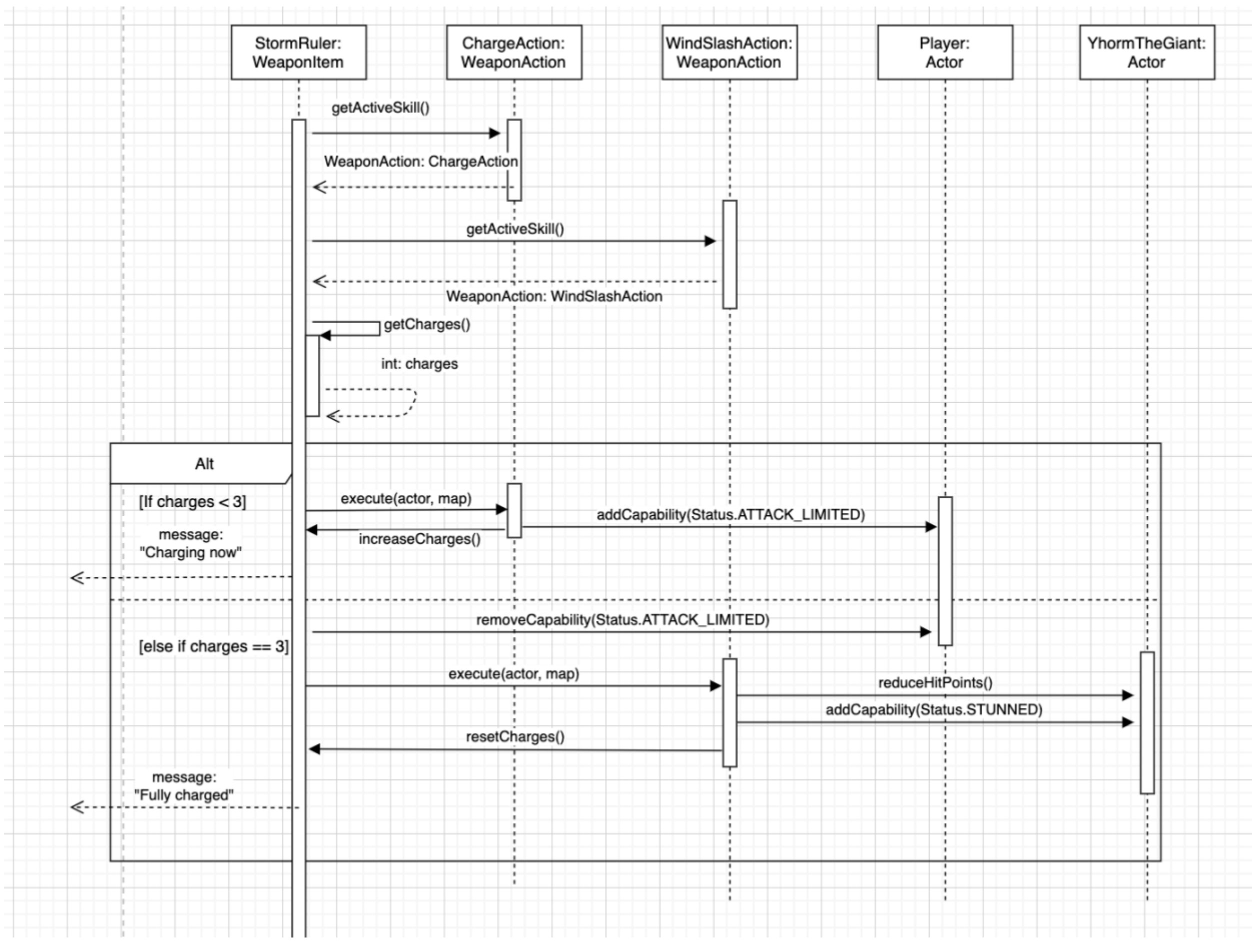

YhormsGreatMachete: WeaponItem
EmberFormAction: WeaponAction
Fire: Item
YhormTheGiant: Actor
:GameMap
getAllowableActions()
WeaponAction: EmberFormAction
getHitPoints()
int: hitPoints
Optional
If HP <50%
addCapability(Status.RAGE_MODE)
Optional
execute(actor, map)
increaseHitRate()
If(YhormTheGiant. hasCapability(Status.RAGE_MODE))
LocationOf()
Location: locationOfYhorm
locationOfYhorm.addItem(new Fire())
message: "Ember Form Activated"

Assignment 1:
In this requirement, I have a need to create multiple weapons with characteristics as shown in the specification sheet. And also, in my implementation I have decided to make ◯◯Action which inherits from WeaponAction. Weapons are inherited from WeaponItem, as I have decided it to be more suitable after looking at the codes in engine file in git. WeaponAction will be in charge of active actions and WeaponItem will implement Weapon while extending Item.



First, I have made BroadSword, StormRuler, GiantAxe, and draw associations between them to show the possession. Multiplicity is always 1 to 1 for them (except GameMap and Fire), because it shows one actor has one weapon each.

And then, StormRuler has options to trigger either ChargeAction or WindSlashAction. What determines is whether charge variable in StormRuler is less than 3 or not. When ChargeAction is called, it will restrict Player by not allowing attack actions.
Details are implemented in the interaction diagram below.



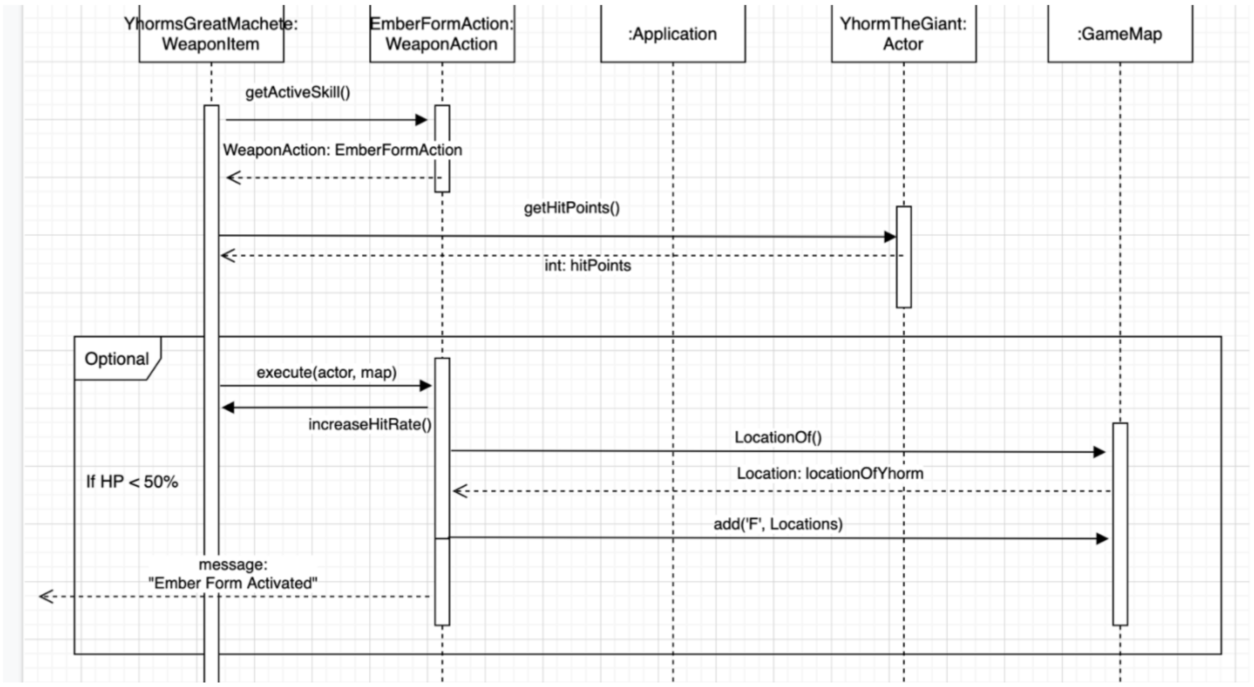
StormRuler will be responsible for calling each weapon skill in our implementation. It will use getActiveSkill() to get the information of the ChargeAction and WindSlashAction. After making sure of that, I will use getCharges() to access charges variable inside StormRuler. After getting an integer representing that, I will perform if-else statement. If charges is less than 3, we can execute ChargeAction using execute(actor, map).
While charging, it adds a capability(my original) to a Player, called Status.ATTACK_LIMITED. This will disable player from attacking, as specified. After that, it will increment the charge by one and return the message “Charging now”.

The alternative is when there are 3 charges. It will removeCapability(Status.ATTACK_LIMITED) as charging is complete. And then it will be able to execute WindSlashAction, which will damage YhormTheGiant, and give status called STUNNED which restricts its movement for 1 turn.
It will then reset its charge to 0 by using resetCharges() and return the message.

Also, please note that I have not shown the detail of passive skills, as that is something that I would write inside my code in assignment 2 and there is no show on the uml diagram about passive skills.

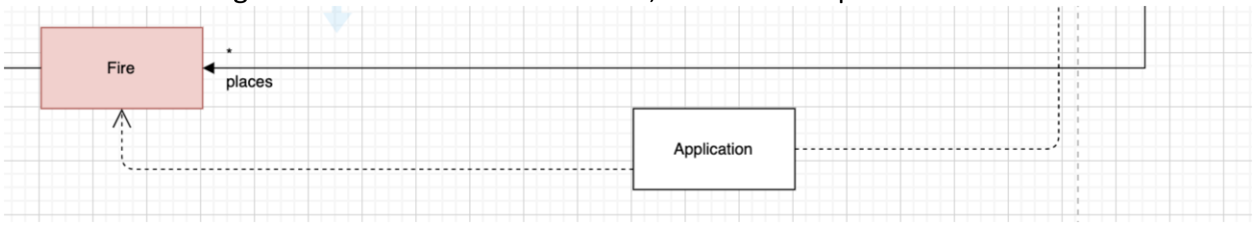
Now, GiantAxe does work very similar to this, except that its skill, SpinAttackAction is very simple implementation. There is no need for interaction diagram for non-complex connections, so there is none.

Lastly, I will talk about YhormsGreatMachete which will turn into EmberForm. Associations is from the Yhorm and its Weapon, and a dependency from Yhorm to EmberFormAction as this is an action to call. To be precise, in my class diagrams there might be an association from YhormsGreatMachete to YhormTheGiant to obtain Hit Points information. However since I consider this insignificant and unnecessary depending on how I Implement, I have decided to leave it out.



What it does in this, is first that it obtains detail about WeaponAction using getActiveSkill() by the YhormsGreatMachete. It then accesses YhormTheGiant to know its hit points detail. The reason for this is to find out whether or not it should trigger EmberFormAction. If HP is less than 50%, it will execute the action, which will increaseHitRate() of the weapon. This is not directed to Yhorm itself, as Weapon will hold the information such as Hit Rate.
It will also access GameMap class in engine to know the locationOf() the Yhorm, and will calculate which rectangles to burn. It will turn adjacent dirt to fire, so it will handle that

calculation, and does add(‘F’, Locations). Although details will be implemented in Assignment 2, this ‘F’ represent space for fire, and Location holds information about x y coordinates of the location nto burn.
It will send message that “Ember Form Activated”, and that completes.



The reason for having * multiplicity from GameMap to Fire, is because it can create many fires depending on the situation.

Finally there is StormRulerOnGround, which is an entity inherited from Item, which is placed by GameMap on the ground. Initially I was having it inherit from Ground, but since I have realized that Item also possess the characteristic to be shown on the ground, I have changed it to this way.
Also, there is Application class that has dependency on Fire and StormRulerOnGround. This is because, when I referred to game class diagram, application class has dependency on all entities such as Wall, Floor, Valley. To add on, line 22 in the Application file in interfaces in source code, has newFancyGroundFactory() that contains multiple arguments such as new Floor() and new Wall() etc.

Since Application will be in charge of bringing them to action, (depends on such objects) there is a dependency.

The reason there is seemingly 2 StormRulers, is because one has a characteristics of a Weapon which activates actions, and the other acts as an item that sits on the ground. It is specified that this sword is placed next to Yhorm The Giant on the map, so StormRulerOnGround will be just an entity placed on the ground.