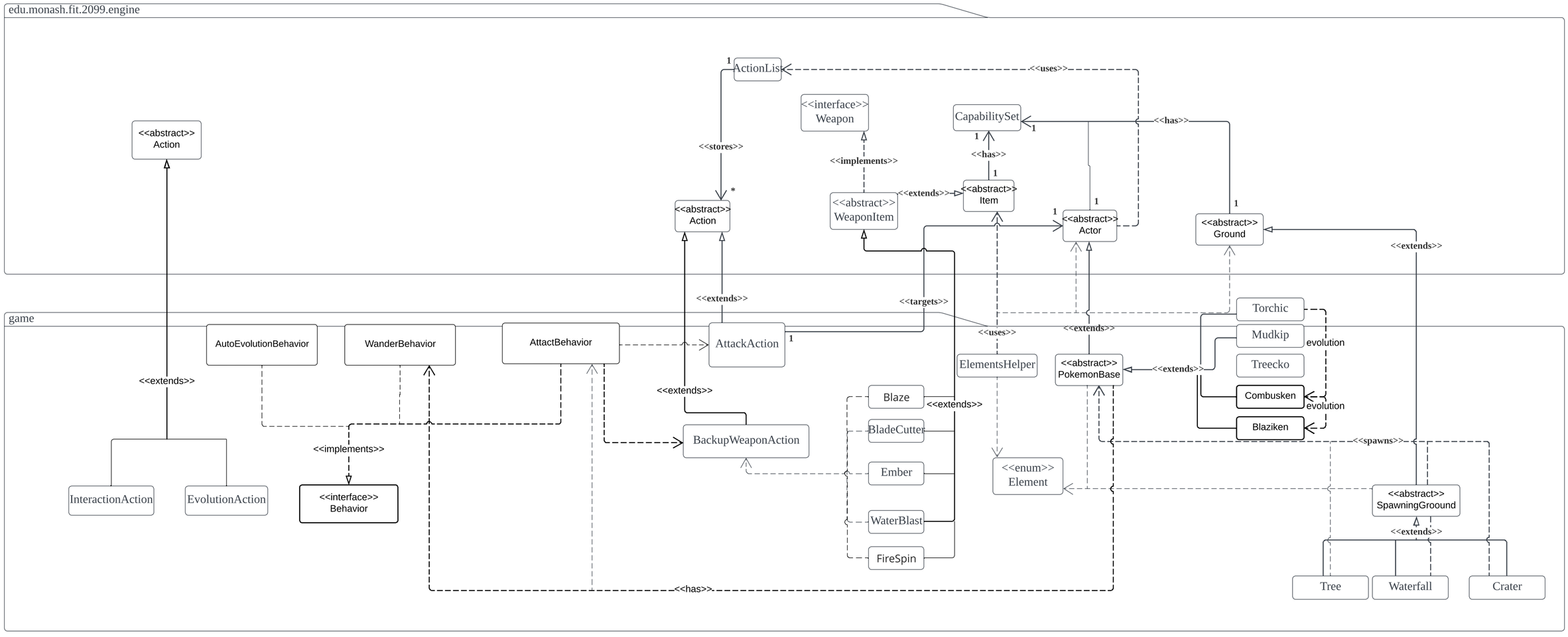
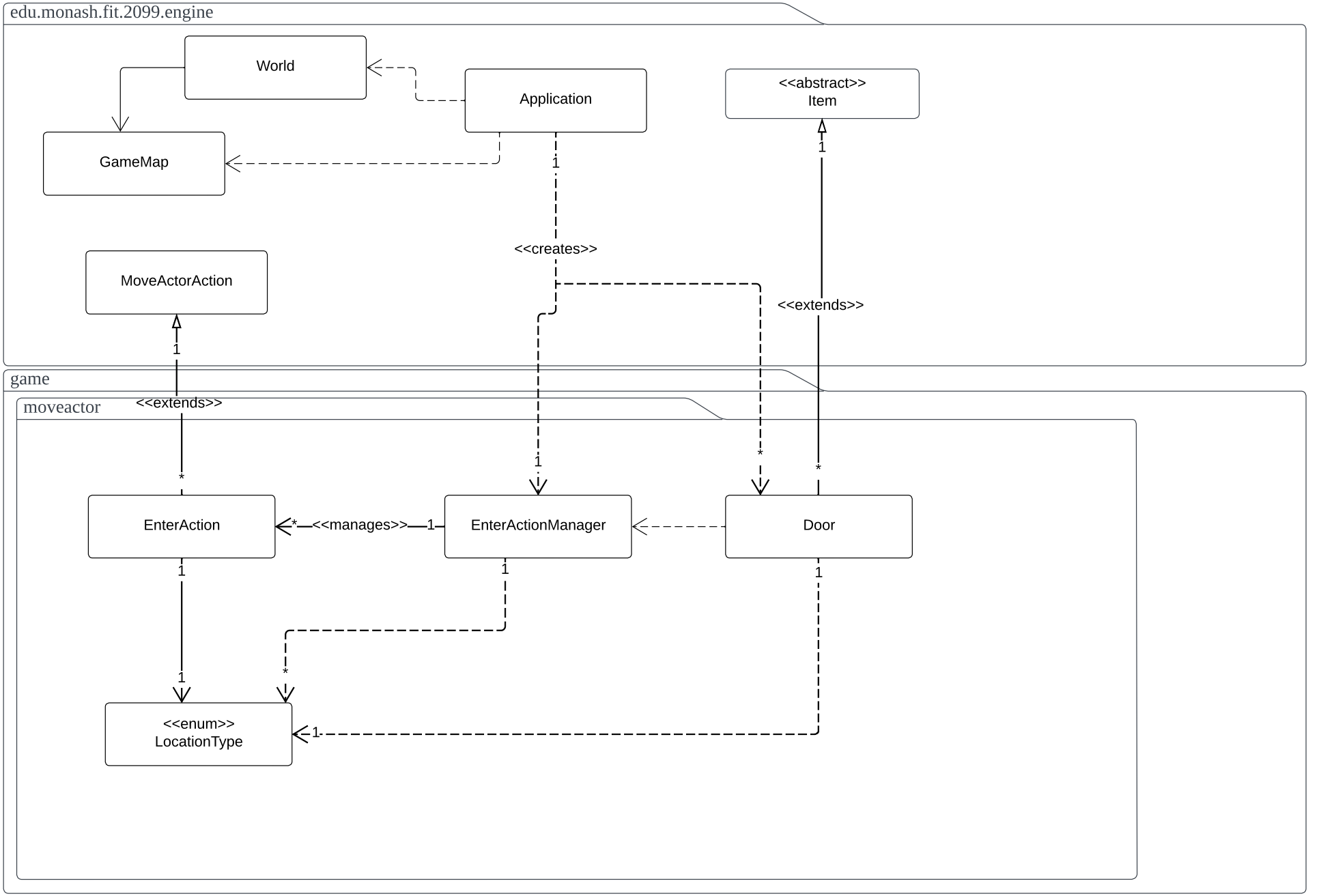
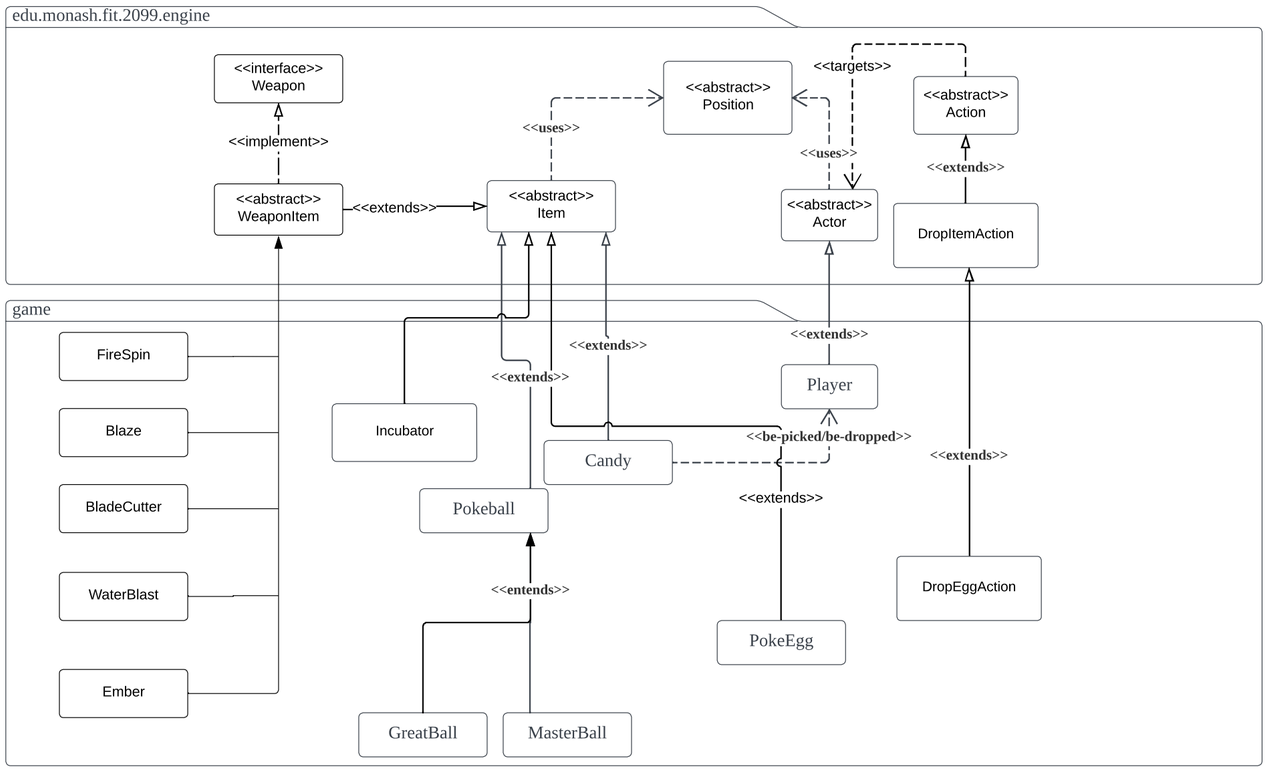
REQ1: Evolution



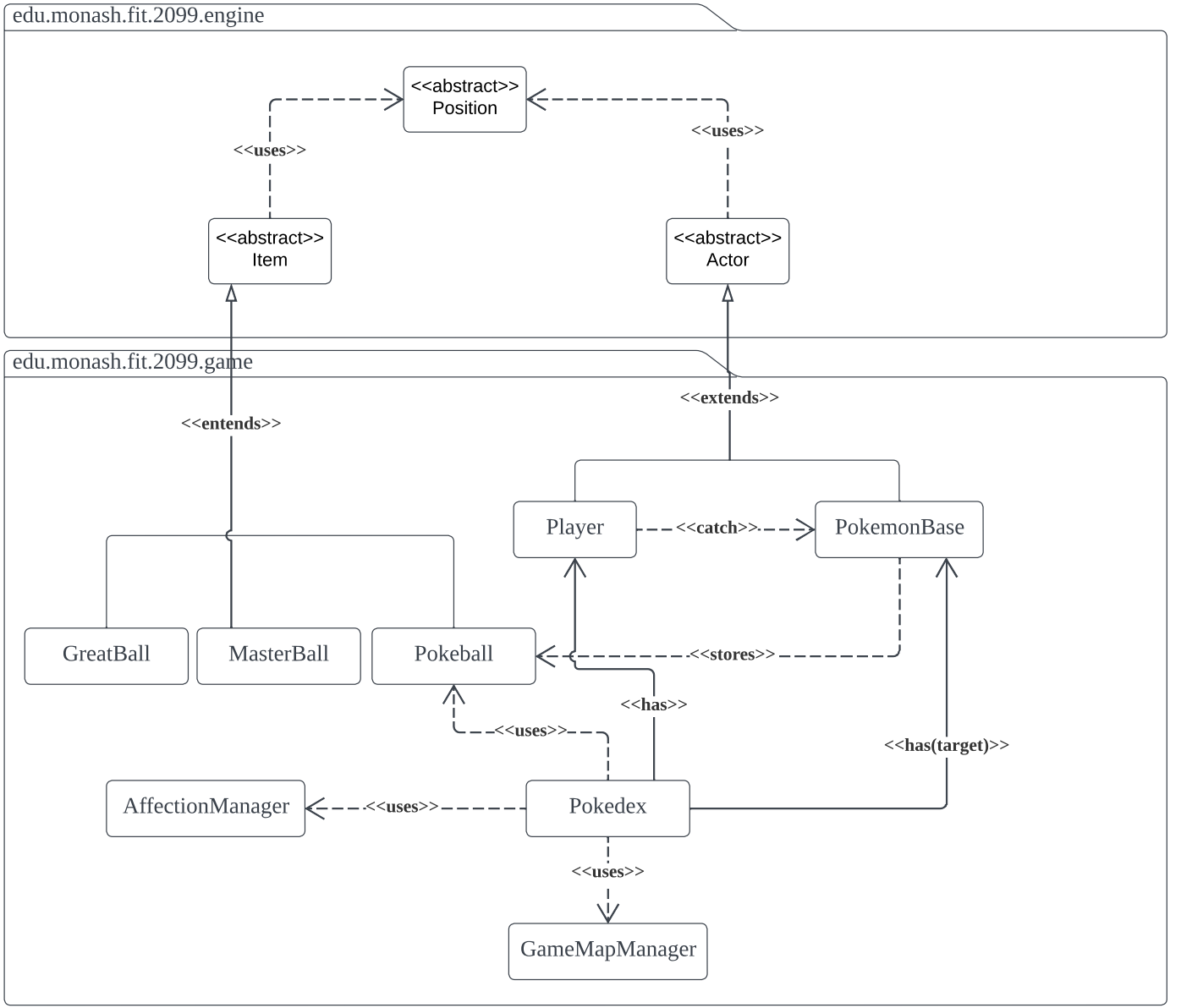
REQ2: Pokemon Center (New Map)



REQ3: Pokemon Egg and Incubator [Structured Mode]



REQ4: Pokedex



Group CL_Lab02Team03
Assignment 3

Design Rationale

Design Goals

The project is a “rogue-like” game that has a Pokemon theme. In this game, the user can control the player character to catch and battle pokemons. The design goals for the this project are:

- Be able to extend the number of Pokemons
- Have Single Responsibility that delegates the responsibilities of the game to the respective class
- Have the base game function be open for extension but closed for modification

The design rationale for the project game will be split into 5 parts, covering all requirements for the game.

Req1: Evolution

In this task, My idea is to build Combusken and Blaziken class which extends from Torchic class. The Comusken and Blaziken have different name, displaychar, hitpoints. And I will add a random method for specialattack. I will add dayEffect and nighteEffect with nothing to make them not affected by the Day or Night. And I will add new status Evolvable and Unevolvable. We add Evolutable to Torchic, and Combusken. Unevolvable to the Blaziken, Mudkip, and Treecko. In EvolutionAction class, I will use name check the next evolution. I want to write a method that will check the AP >= 100 and the surrounding has an empty location to check the evolute. In pokebase, I want to add method allowableActions which will use to interact with pokemon to improve ap and method playTurn to record pokemon alived time. In Torchic class, the Torchic has a new behaviour AutoEvolutionBehavior when it is alive for more than 10 turns.

Req2: Pokemon Center

The following diagram represents the features for the game’s new game map, the Pokemon Center. The main goal for this new feature was to create a new Door class that can be reused as doors for buildings and even act as a teleporter for example when players go into battle in a battle stadium.

All Door Class instances are items and they extend the Item class from the game engine as this allows it to have actions when the player interacts with the doors. The EnterAction class was created to extend the feature of the MoveActorAction class to override the menu description to explain that the actors enter a location. To make the assignment of EnterActions for Doors more reusable, an EnterActionManager class that has a hashmap that uses the enums LocationType to check for the EnterAction instances was created. This allows all distinct EnterActions instances to be set only in the EnterLocationManager reducing the chance of creating errors for where the actor is teleported to by reducing total instances. In addition, EnterActions instances can be set with the enum location type reducing the possibility for errors when repeatedly assigning new locations to move the actors as the enum names clarify where we want to send the actor instead of specific coordinates. In the future, if there is a need for doors that allow multiple types of locations

that it can teleport actors to, the door class can be modified through parametric polymorphism to allow for such doors.

Req3: Pokemon Egg and Incubator

In this task, I exchange the BuyAction in assignmnet2. And I add a new item which is PokeEgg. I want to add the judgment of hatching time and surrounding empty position in the egg class. And the spawning of pokemon will depend on the egg name. In this way, it will help us to reduce the number of classes. In BuyAction, the player will buy TorchicEgg,TreeckoEgg, and MudkipEgg to please buy Torchic. I will create a new ground named Incubator near ShopKeeper. I will create a DropEggAction that the player can put the PokeEgg in the Incubator. Then PokeEgg will start to judgment method until reaches to a certain time. The PokeEgg will spawn a Pokemon and remove the PokeEgg.

Req4: Pokedex

The main point for the requirement 4 is to support an button(action) in the display screen so the player can check all the pokemons information include the HP and AP that are caught in inventory.
As the UML diagram shown below, I made the Pokedex class extends from the action so that can be execute via pressing “z”. Due to the caught pokemons are stored in the pokeballs from player’s inventory. Therefore the main idea for Pokedex is buid by some loops like for loop. Firstly, I made a for loop to find unique Element type of the caught pokemons and stored them in an arraylist called pokeTypes so that make sure it ready to print out. Then I print each pokemons’ informations that in the player’s inventory by another for loop. And print their HP, AP and location by using AffectionManager and GameMapManager. In this way, the structure of execute the Pokedex is done. Finaly, only need to execute this action in player class so the player can see the button “z” and check the pokemons’ informations.