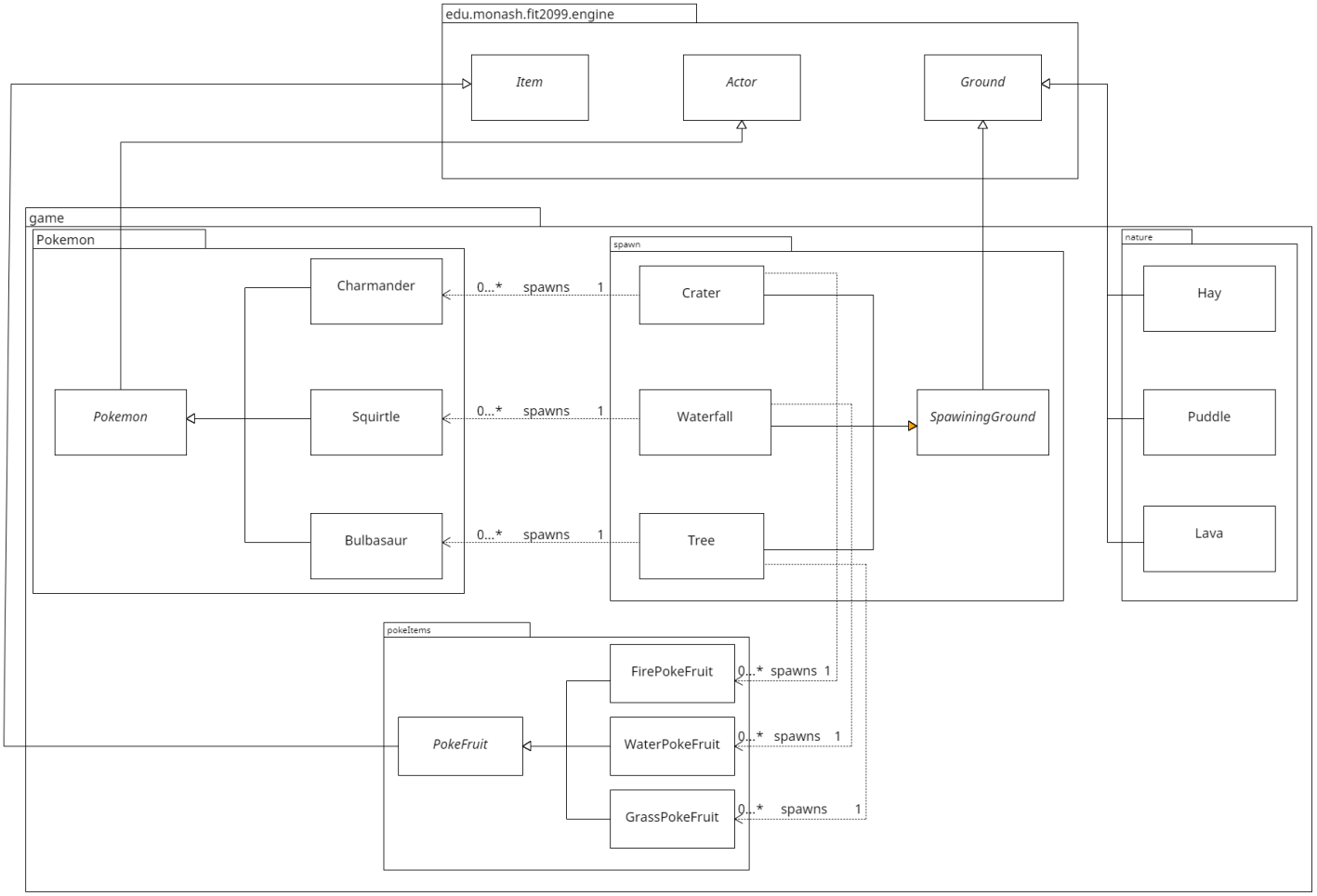
RQ1 - Environment

Class Diagram



Design Rationale

In this requirement, SpawningGround extends Ground.

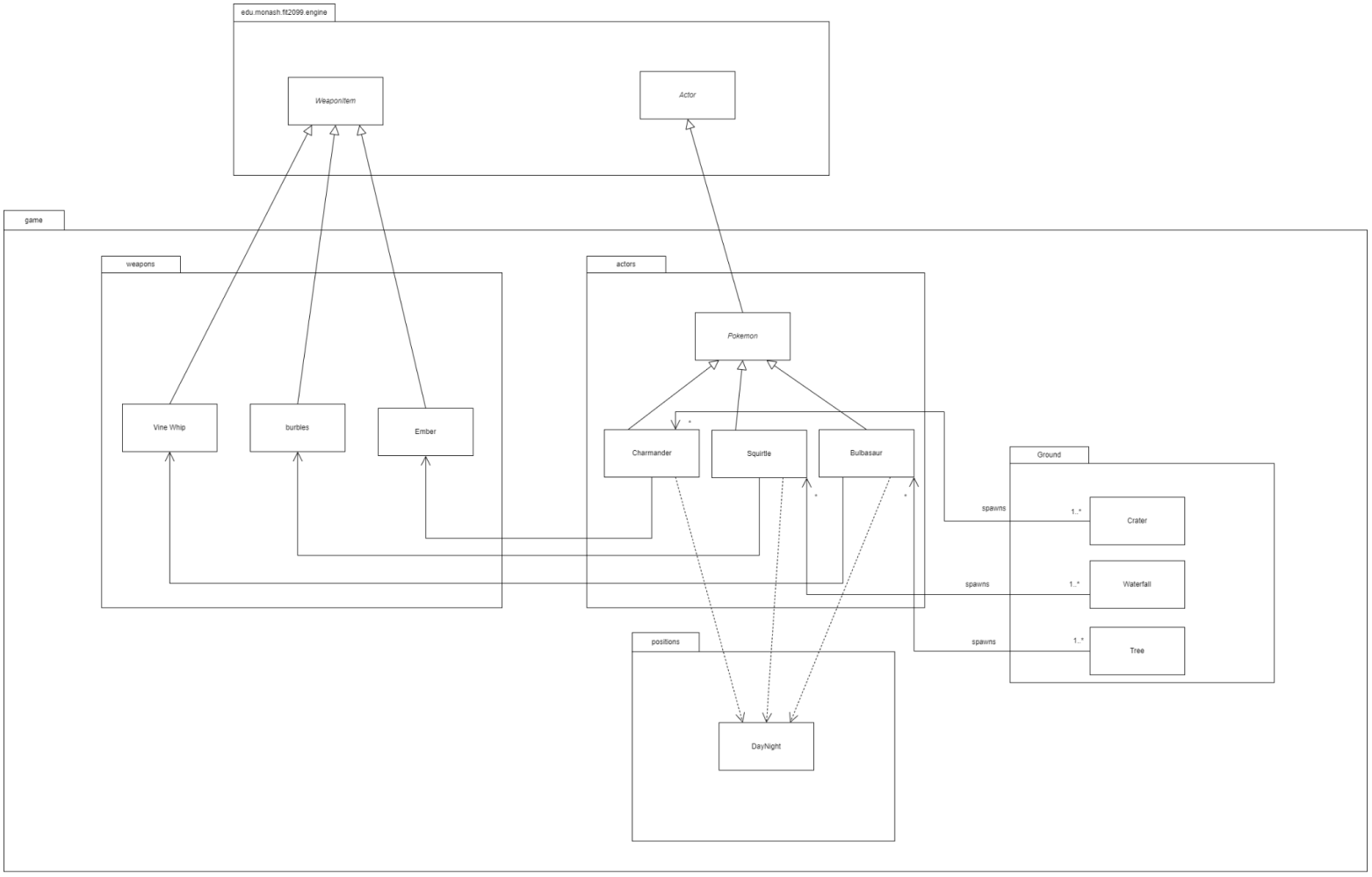
In the SpawningGround class, it has a method that helps check whether the chance of something spawning occurs or not.

Crater, Waterfall, Tree ---<<extends>>---> Spawning Ground. Crater, Waterfall and tree are the object that causes SpawningGround to randomise and checks whether the probability of a Pokemon or PokeFruit spawning.

Alternatively, we can create a method for probability inside the Ground class. However,doing this requires that all ground types can use the SpawningGround, which is not true in this requirement. This results in tight coupling as well as doesn't stick with DRY principle. Therefore the team sticked on the idea of another abstract class to be made called SpawningGround.

RQ2 - Pokemon

Class Diagram



Design Rationale

The diagram represents a part of a game that includes 4 packages which have 9 concrete classes and 3 classes inheriting an abstract class.

All three types of pokemons extended the abstract Pokemon class. As all pokemons will share some common methods and attributes, it makes sense to place those into an abstract class to avoid repetition.

Each pokemon has an association with the environment that they spawn in, as one environment can spawn one or more of the pokemons.

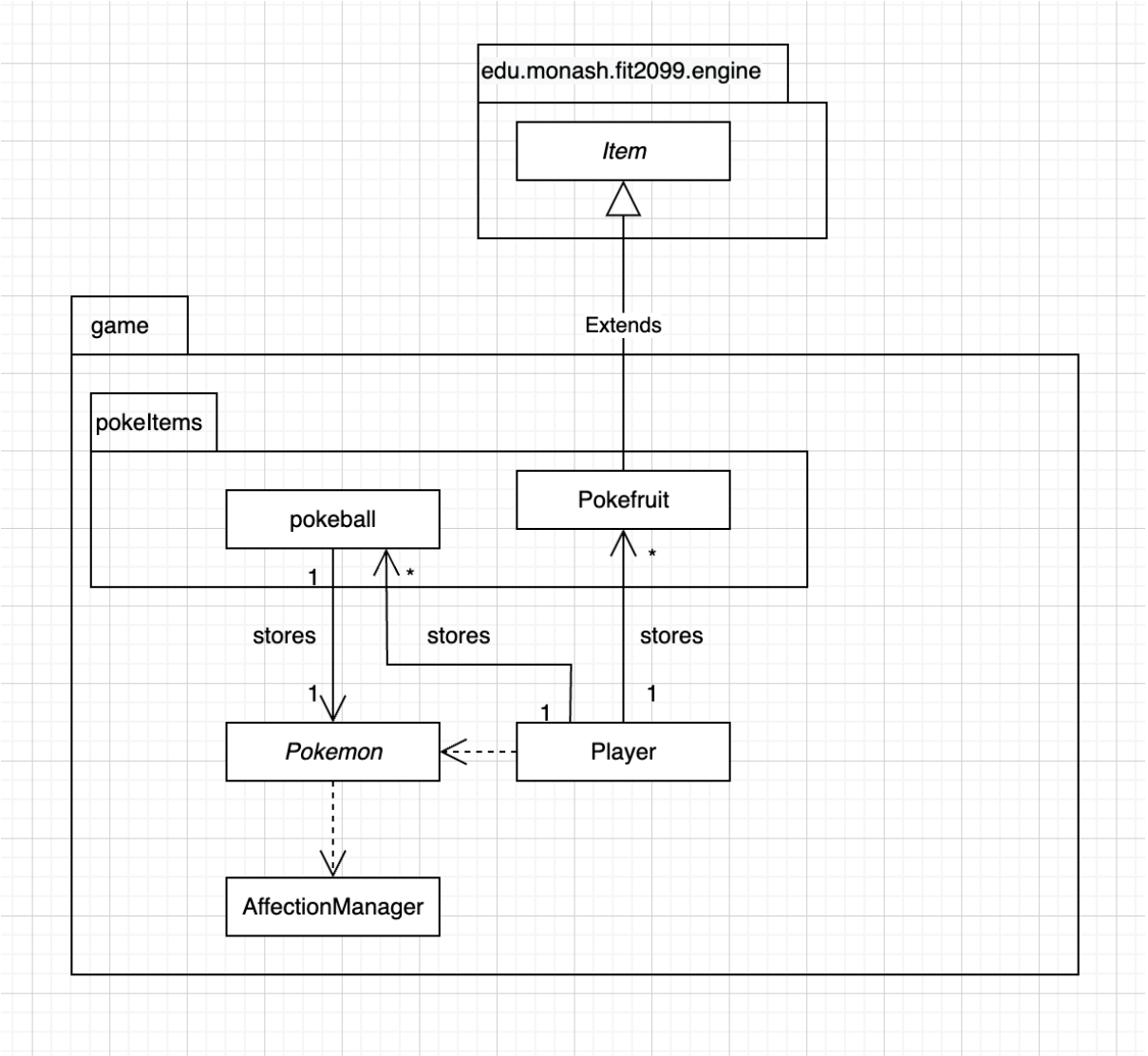
Each pokemon also has a dependency on DayNight class. Everytime the game switches between night and day, each pokemon will either get hurt or be healed. By adding methods from the DayNight class that hurts or heals a pokemon, we do not have to repeatedly add those methods to each pokemon class.

The abstract Pokemon class extends the Actor class. As all types of pokemon are also included as an Actor and has attributes common to an Actor class, which means it will inherit all Actor attributes.

Each Pokemon also has a dependency on their special attack, which extends the WeaponItem class as all special attacks are a weapon item. So we only need to call the methods from the abstract WeaponItem class to modify each special attack.

RQ3 - Items

Class Diagram



Design Rationale

Candy, Pokefruit, and Pokeball extends *Item* . All items that inherit the abstract item class have the methods required to make the items behave properly.

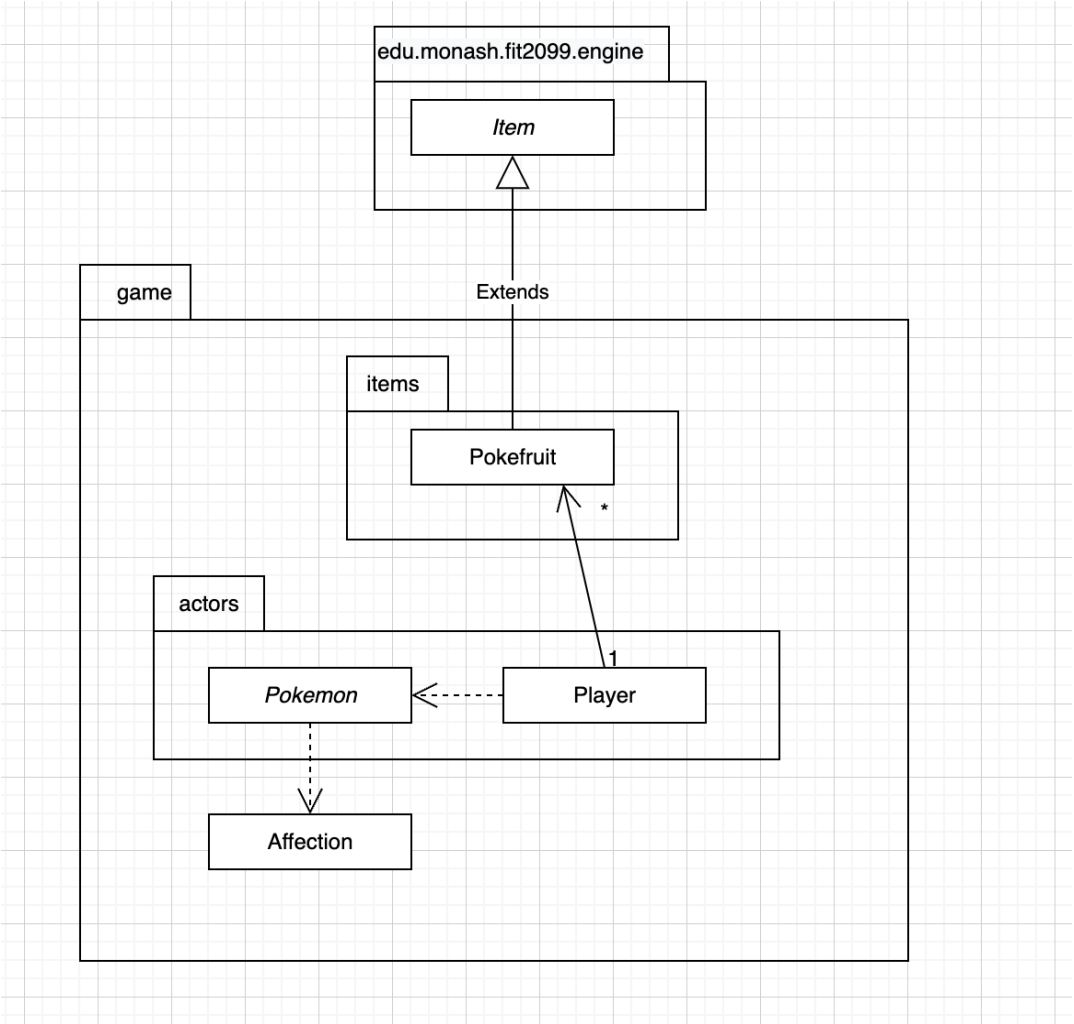
All items such as Candy, Pokefruit, and Pokeball are inherited from the *item* class which comes from the engine. All candies are given to the player by captured pokemons which is why the player has a dependency on the pokemon for the candies.

Pokefruits are picked from the ground by the player. Our design choice is the player will have an arraylist of Pokefruits in the global attribute. The arraylist will act as an inventory of pokefruits.

Everytime a pokemon is captured it will be stored into a pokeball which after that the pokeball will be stored in the player class. It's the same as pokefruit where the player will have an arraylist of attributes filled with pokeballs with pokemons inside the pokeballs.

RQ4 - Interactions

Class Diagram

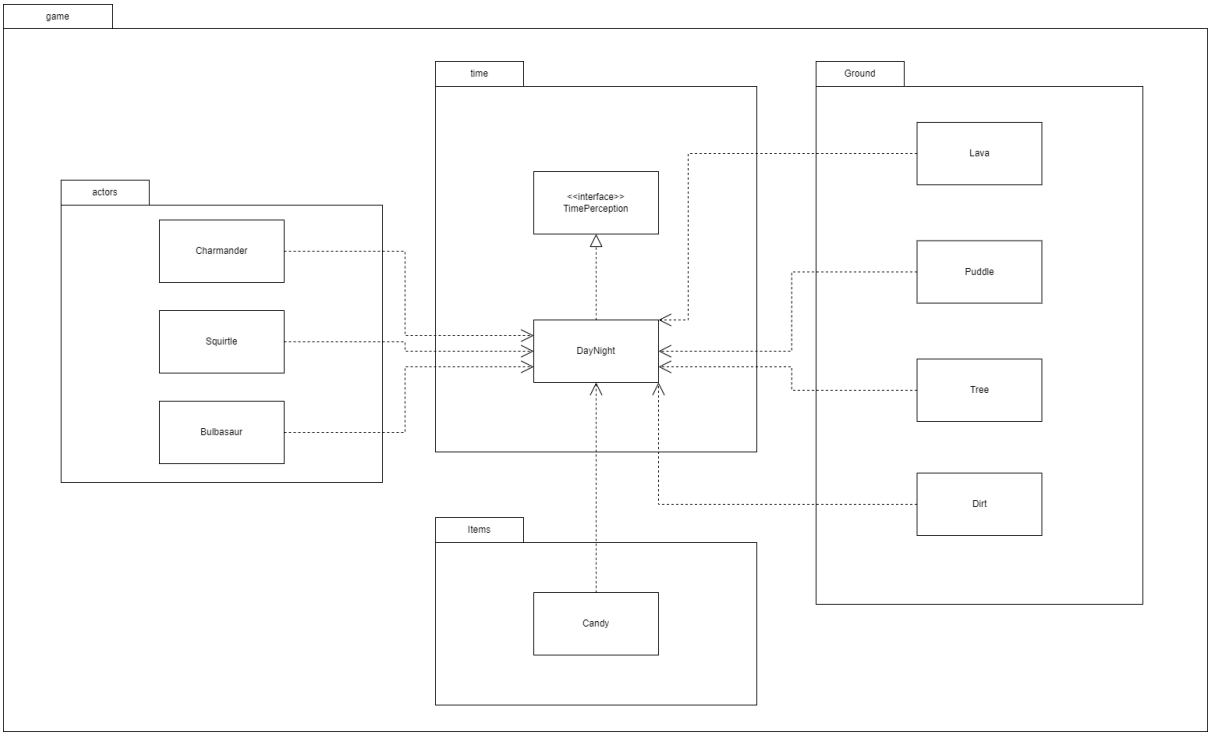


Design Rationale

The same as REQ3 pokefruits are inherited from the item class in the engine. The Player class will have dependency on the Pokemon class to feed the pokemons. In our design the pokemon would create an object called Affection where the Pokemon can get and edit their affection stats.

RQ5 - Day and Night

Class Diagram



Design Rationale

The diagram represents a part of a game that includes 2 packages which have 6 concrete classes.

Charmander, Squirtle and Bulbasaur class all have a dependency on DayNight class as whenever the game switches between day and night, a pokemon will either lose health or gain health. By adding methods from the DayNight class that hurt or heal a pokemon, we do not have to repeatedly add those methods to each pokemon class, we could just call those methods from the DayNight class.

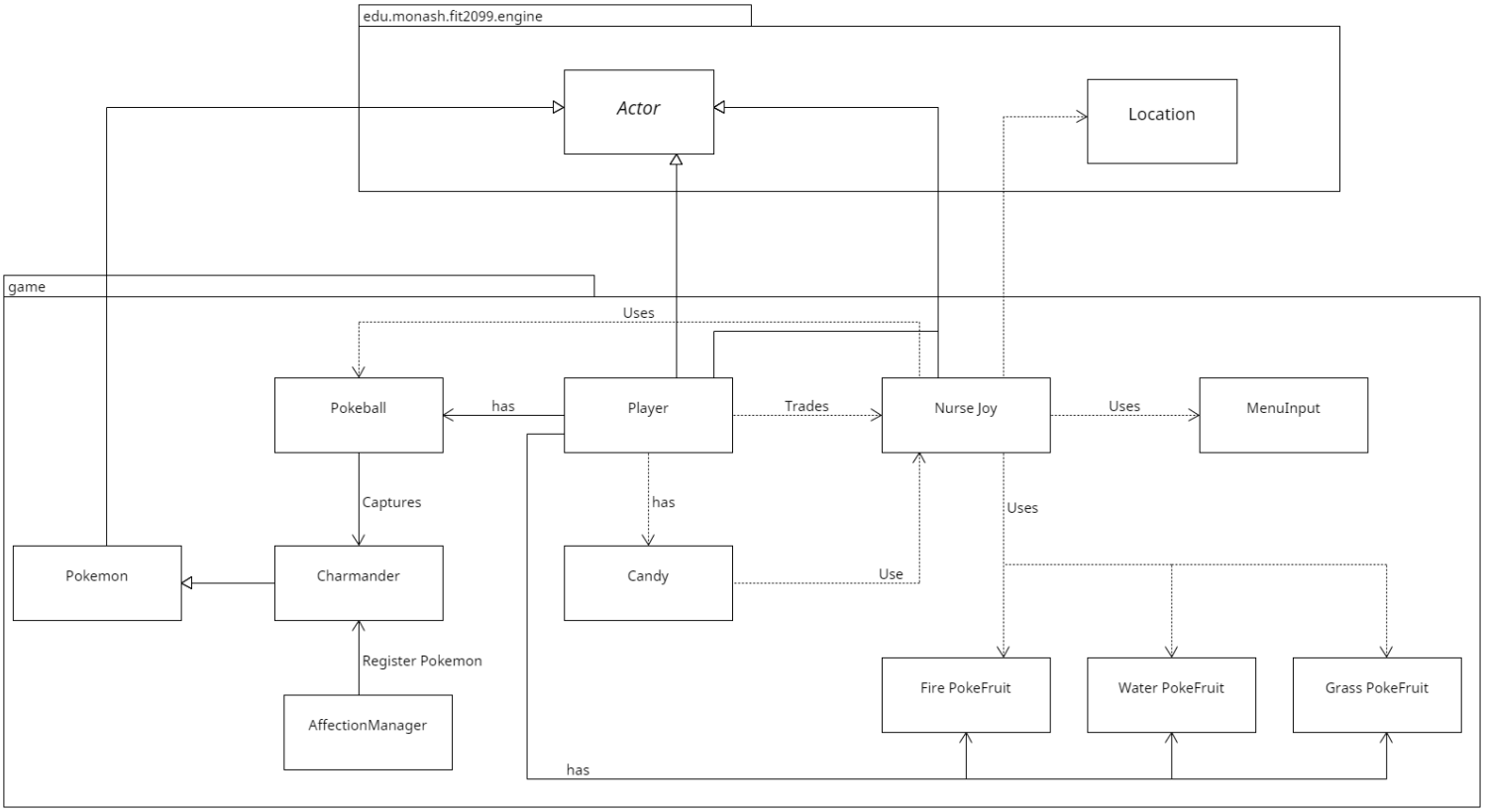
Lava, Puddle, Dirt and Tree class have a dependency on DayNight. Whenever the game switches between day and night, each environment in the ground class will either expand or get destroyed. By adding methods from the DayNight class that allows a certain environment to expand or get destroyed, we do not have to repeatedly add those methods to each class, we could just call those methods from the DayNight class.

Candy has a dependency on DayNight class, as when it is day time, Tree will have a 5 percent of dropping Candy. DayNight class will have to call on the dropCandy method on Tree whenever it is day time.

Location class has a dependency on DayNight class. DayNight class calls on methods like setGround, getGround and equals to expand and destroy a certain ground whenever the day night switch happens.

REQ 6 - Nurse Joy

Class Diagram



Design Rationale

In this requirement, my approach was first to know what classes it would be connected to from Nurse Joy. The team knows that NurseJoy trades different element Pokefruits as well as a Charmander inside a Pokeball with an affection level of 0 for certain amounts of candy. Therefore it means we have to create a class called MenuInput to help navigate the player to trade candy for their product. As Nurse Joy is an NPC, therefore it inherits the Actor class.

In this requirement, it is also known that Nurse Joy will spawn at the middle of the map, therefore it is known that using

Interactive Diagram (Sequence Diagram)