MONASH University

Design Rationale FIT2099 | Assignment 1

Group members : Chen Xi Diong, Gde Putu Guido Parsanda, Karen Zicheng Xia

We created a Waterfall class, a Crater class and a Hay class, which extends the Ground class to cut down redundant implementations of methods and attributes. This adheres to the Do not Repeat Yourself (**DRY**) principle.

The Pokemon interface is implemented by Charmander, Squirtle and Bulbasaur classes. The interface can create the AttackAction to be returned to the Player instead of implementing the same method in all three classes. This adheres to the Do not Repeat Yourself (**DRY**) principle.

We declared an AffectionManager class, which only manages the affection level between the pokemons and the main player and nothing else. This adheres to the Single Responsibility Principle (**SRP**).

We declared a Utils class to generate random possibilities for the spawning of a Pokemon, spreading and destruction of a ground, and a Pokemon using its special attack. This adheres to the Single Responsibility Principle (**SRP**).

As some grounds can only spawn Pokemon and not spread to other grounds, whereas other grounds can only spread and not spawn Pokemon, we created the Spawnable and Spreadable interfaces for the different Ground classes to implement. This adheres to the Interface Segregation Principle (**ISP**).

We created a Pokemon interface and a Spawnable interface to reduce the dependencies between a Ground class and a certain pokemon (e.g. Waterfall spawns Squirtle, Crater spawns Charmander etc.). This adheres to the Reduce Dependency (**ReD**) principle.

Squirtle and Bulbasaur classes will implement the Catchable interface. When the player stands adjacent to a catchable Pokemon, the Pokemon object (no matter whether it is a Squirtle or Bulbasaur) will check its affection points by implementing the method from Pokemon Interface, and if it meets the requirement, the Catchable interface will create CatchAction and return to Player to interact with. Therefore, we don't need to check the class type of certain pokemon as using an if-else statement is not recommended in the object-oriented as it involves extra dependency. Our design can thus break such coupling between Player, Pokemon and CatchAction. This adheres to the Reduce Dependency (**ReD**) principle.

We created a Tradable interface for the items that can be traded with Nurse Joy to reduce the dependencies between Nurse Joy and the current tradable items (i.e. Charmander and Pokefruit). This adheres to the Reduce Dependency (**ReD**) principle.

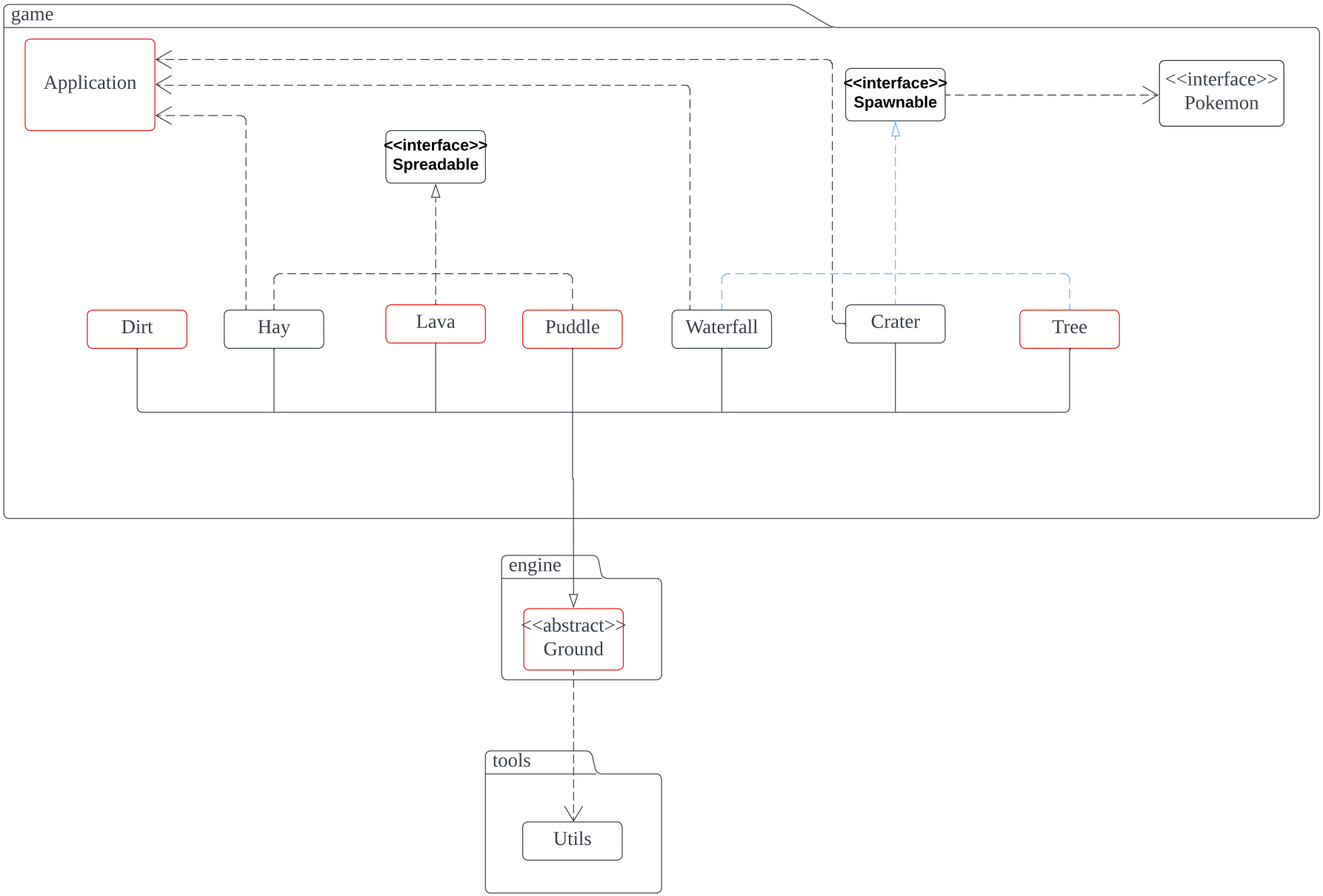
We made factory methods for the AffectionManager, Player and NurseJoy classes, since there will be only one instance of each throughout the whole game. Our application will maintain a single class instance for these three classes.

We added a Pokemon attribute to the Pokeball class for each Pokeball instance to keep track of what kind of Pokemon the Pokeball contains.

We made Charmander, Squirtle, Bulbasaur, Puddle, Lava and Tree classes implement the TimePerception interface so that they can perceive Day and Night turns.

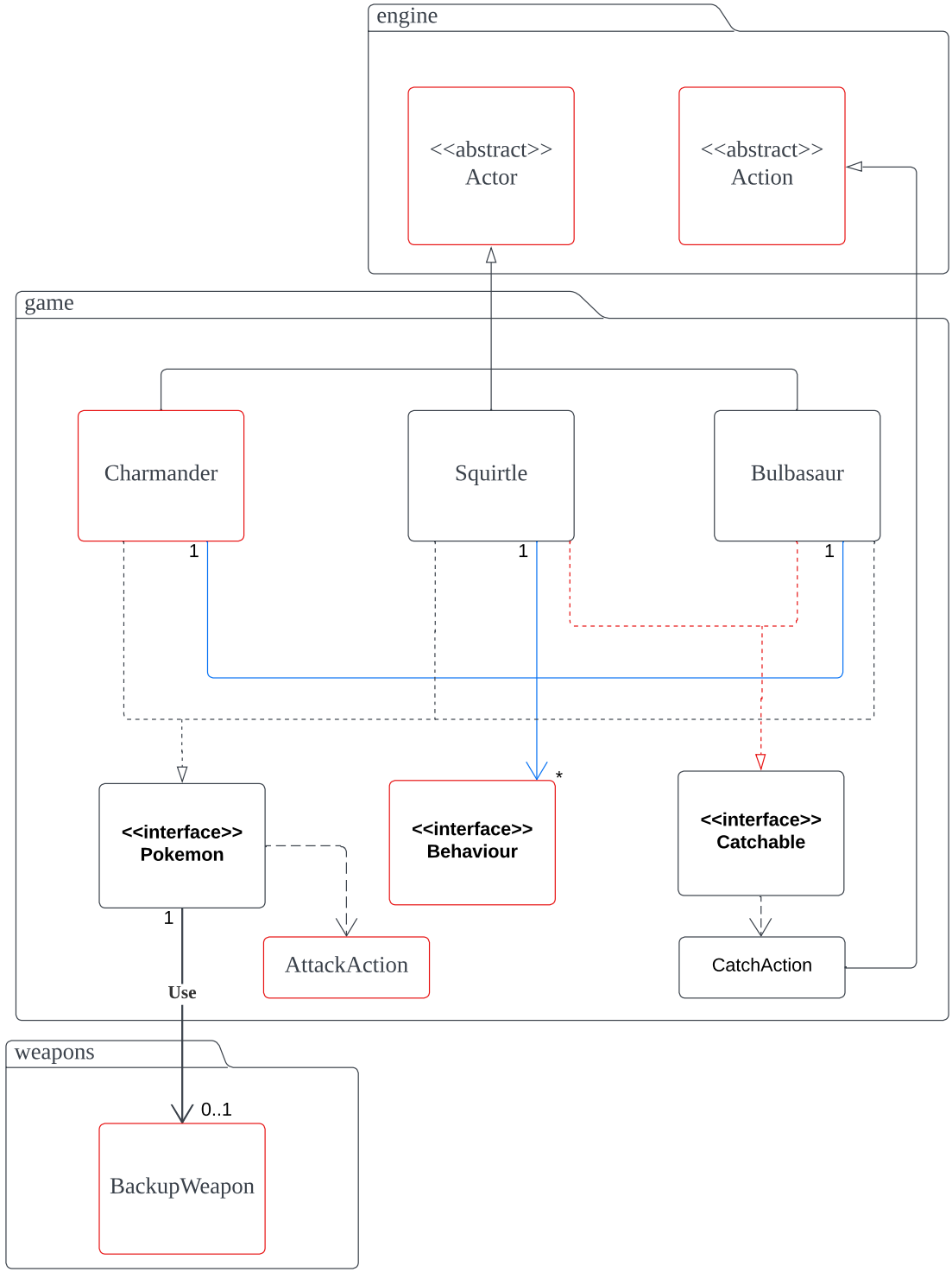
FIT2099 Assignment 1 Lab04 Group05

REQ1: Environment



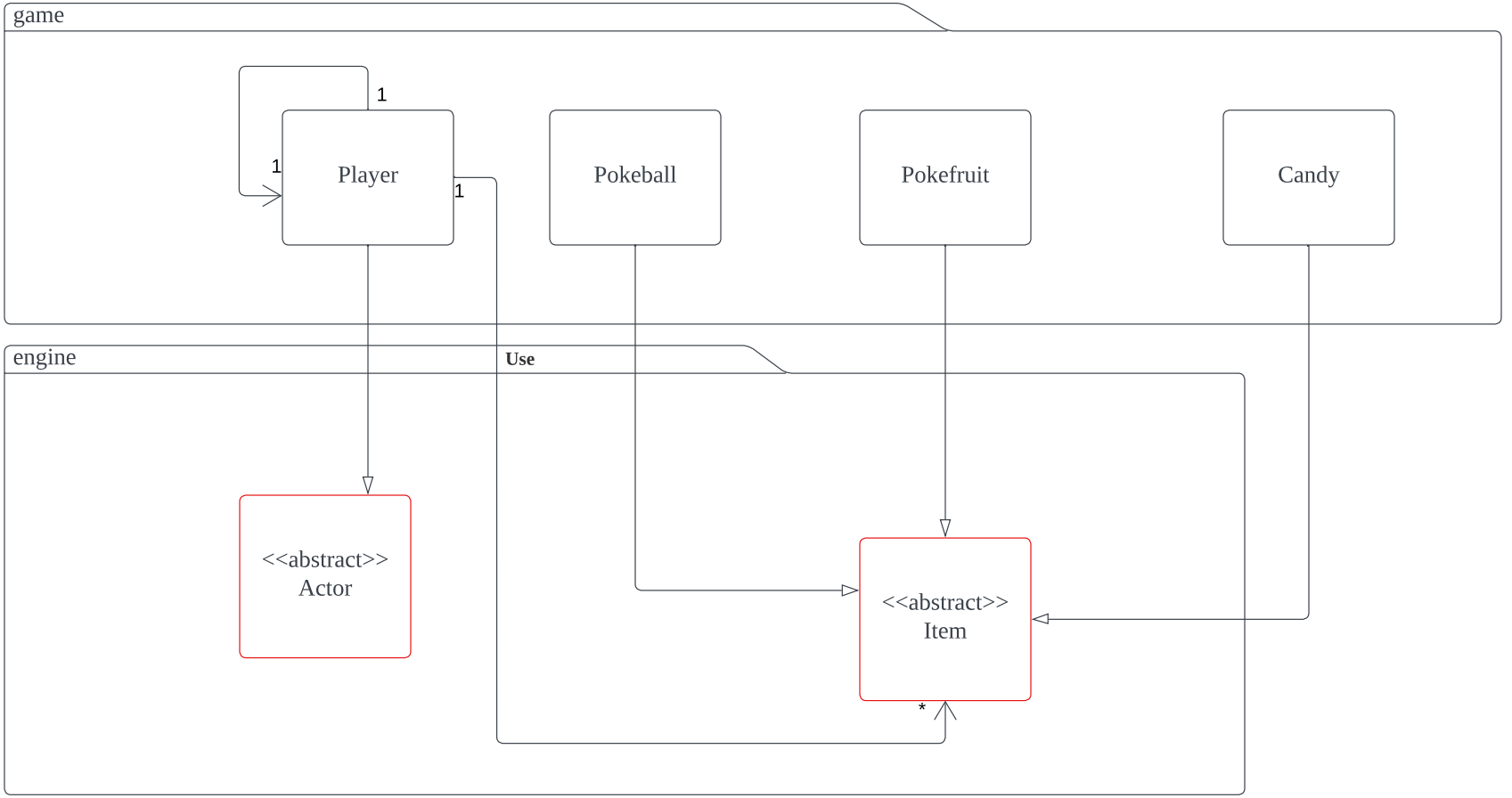
FIT2099 Assignment 1 Lab04 Group05

REQ2: Pokemons



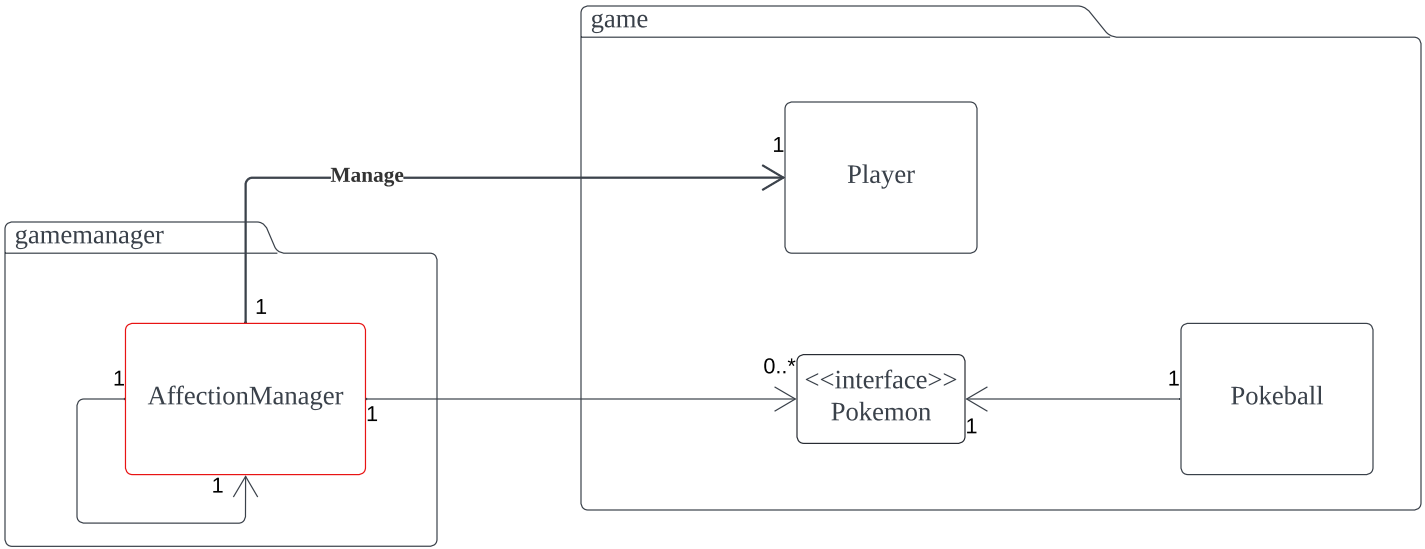
FIT2099 Assignment 1 Lab04 Group05

REQ3: Items



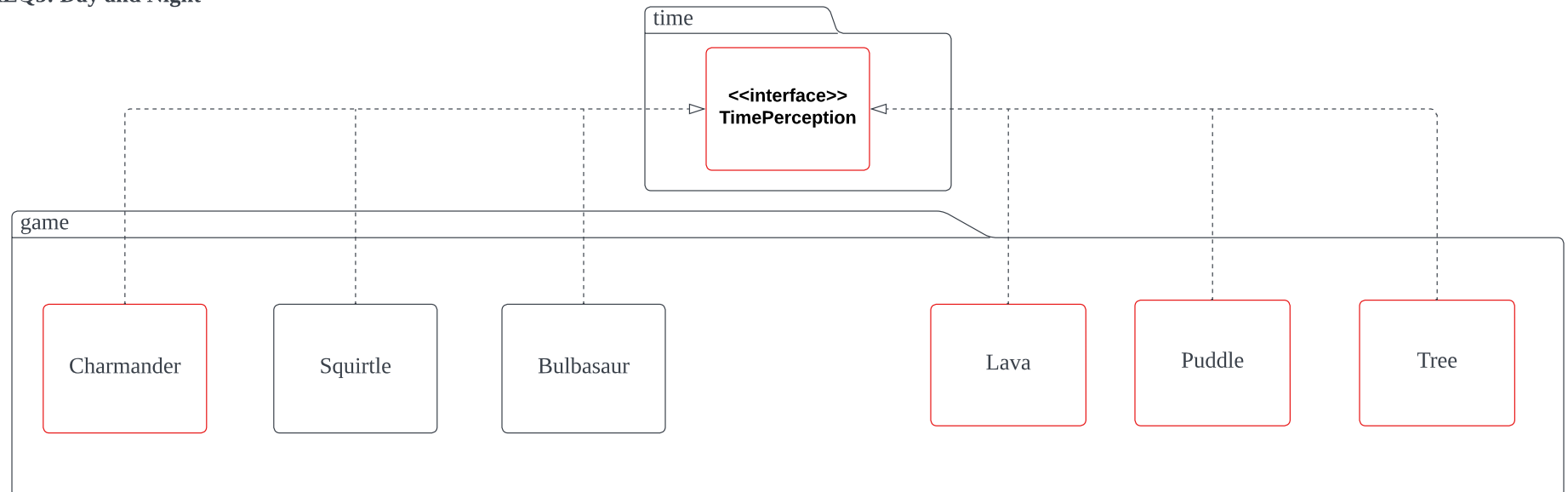
FIT2099 Assignment 1 Lab04 Group05

REQ4: Interactions



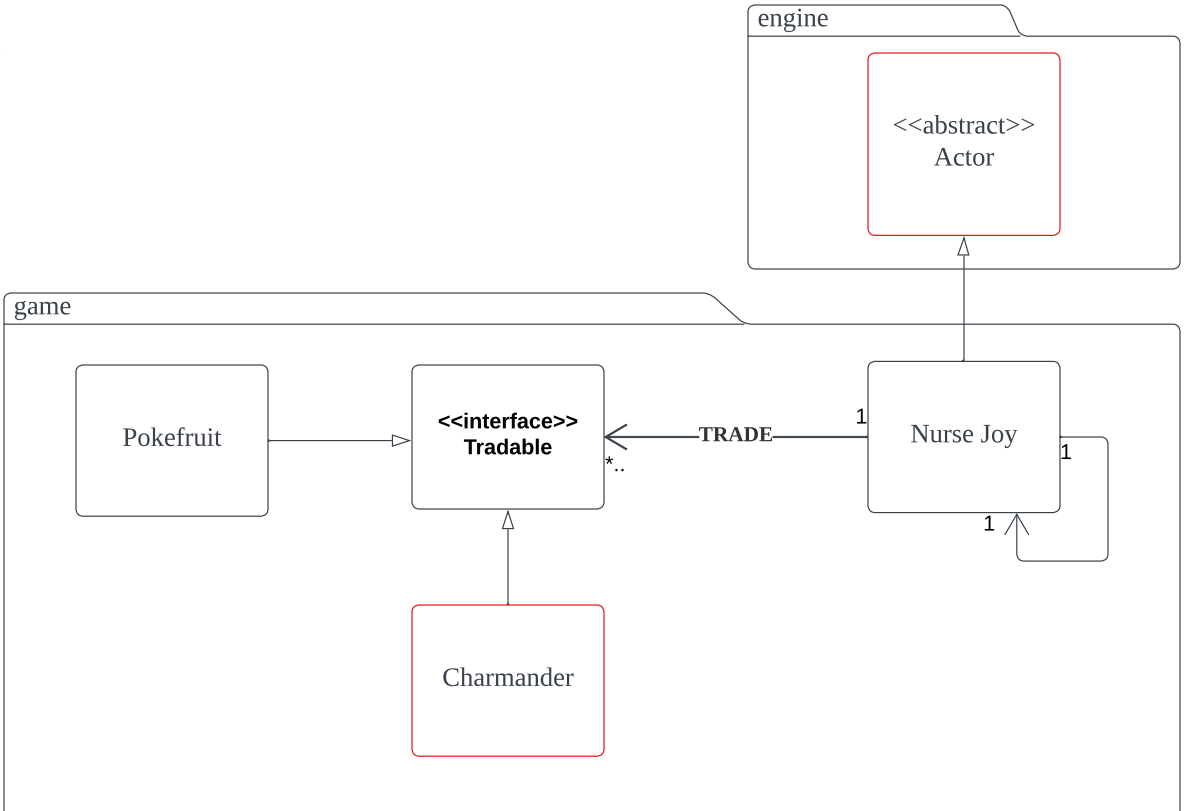
FIT2099 Assignment 1 Lab04 Group05

REQ5: Day and Night



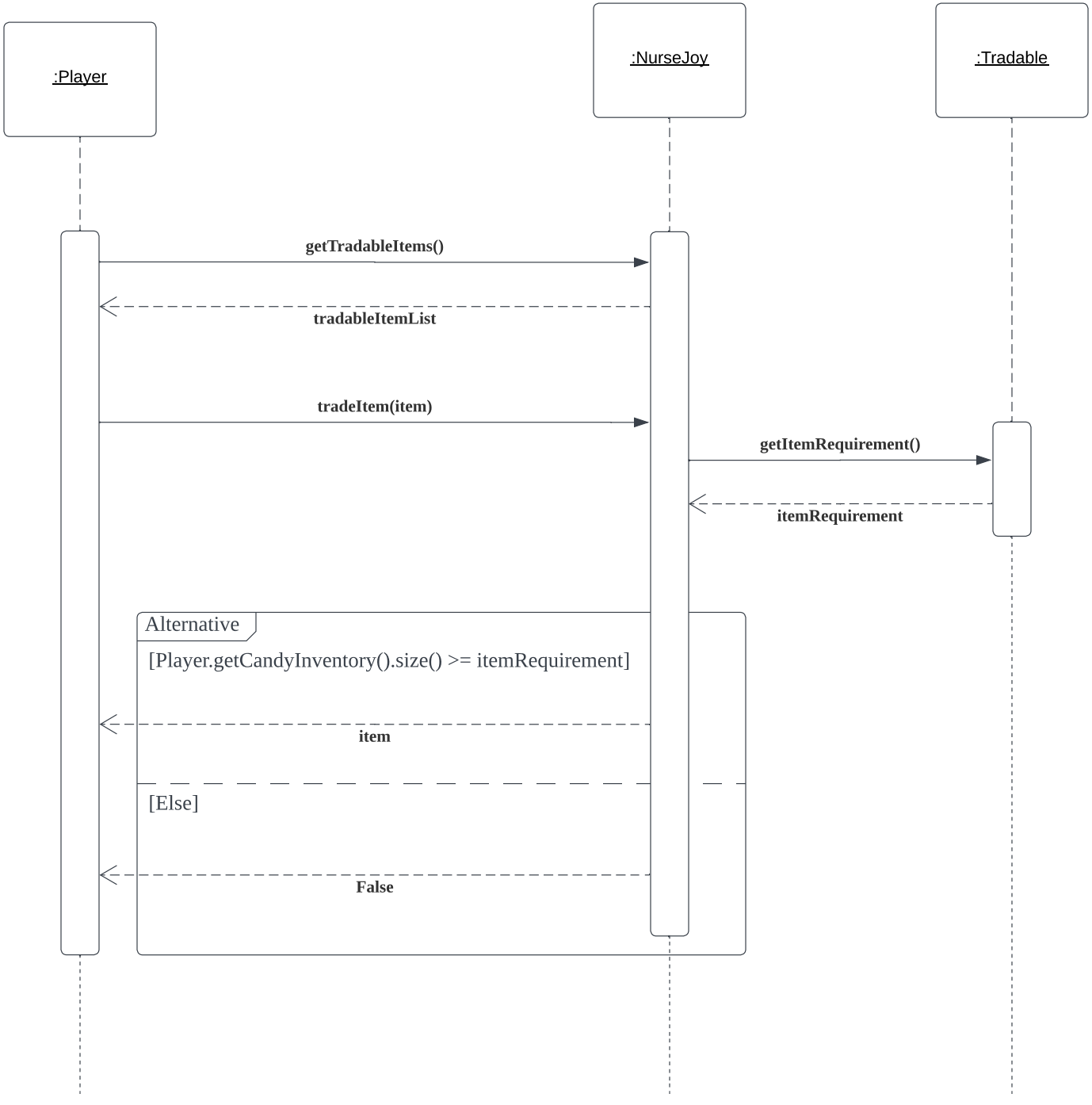
FIT2099 Assignment 1 Lab04 Group05

REQ 6 NURSE JOY



FIT2099 Assignment 1 Lab04 Group05

TradeItem Sequence Diagram



FIT2099 Assignment 1 Lab04 Group05

givePokeFruit Sequence Diagram