Assignment 3: Further Design and Implementation

FIT2099: Object-Oriented Design and Implementation

Team: CL_Lab2Group6
Zhijun Chen
Minh Tuan Le
Ishrat Kaur

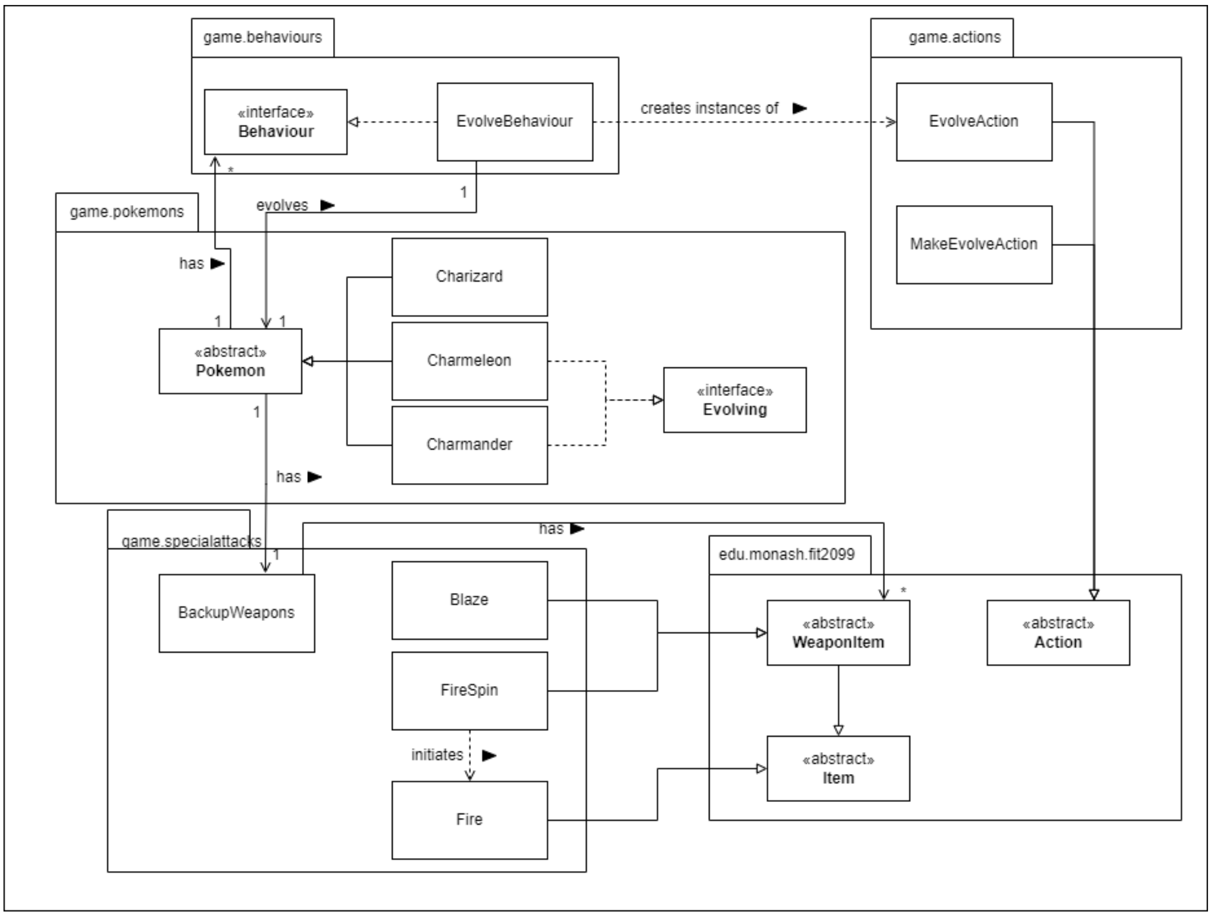
Table of contents

Design documentation

This document contains four UML (unified modelling language) diagrams for four requirements specified in the project. UML diagrams are followed by corresponding design rationales explaining major design decisions and the underlying principles.

REQ1:

Evolution

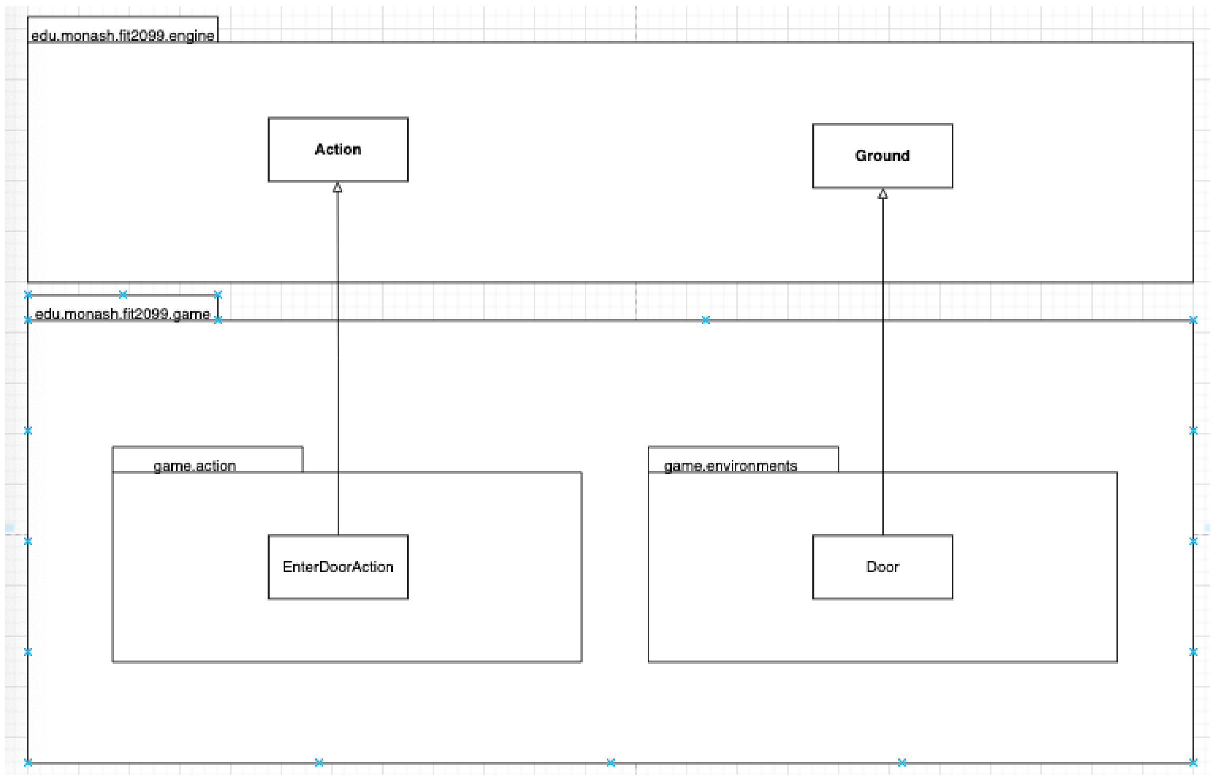


Design rationale

The evolution of pokemons (Charizard and Charmeleon) are implemented as part of the Pokemon abstract class and share all its attributes. In order for a pokemon to evolve, it has to implement the Evolving interface, which has the evolve() method to return the next evolution in order. These are in line with the interface segregation and Liskov substitution principles. The evolve behaviour will return an evolve action and will be added to the behaviour list in Pokemon with highest priorities when conditions are met. The backup weapons or special weapons are refactored so that BackupWeapon classes can contain more than one WeaponItem. The Fire Item is a new Item class used to implement the special effect of the WeaponItem FireSpin. The MakeEvolveActon is an action to allow the player to evolve pokemon manually from the console game options when all conditions are met.

REQ2:

Pokemon Centre

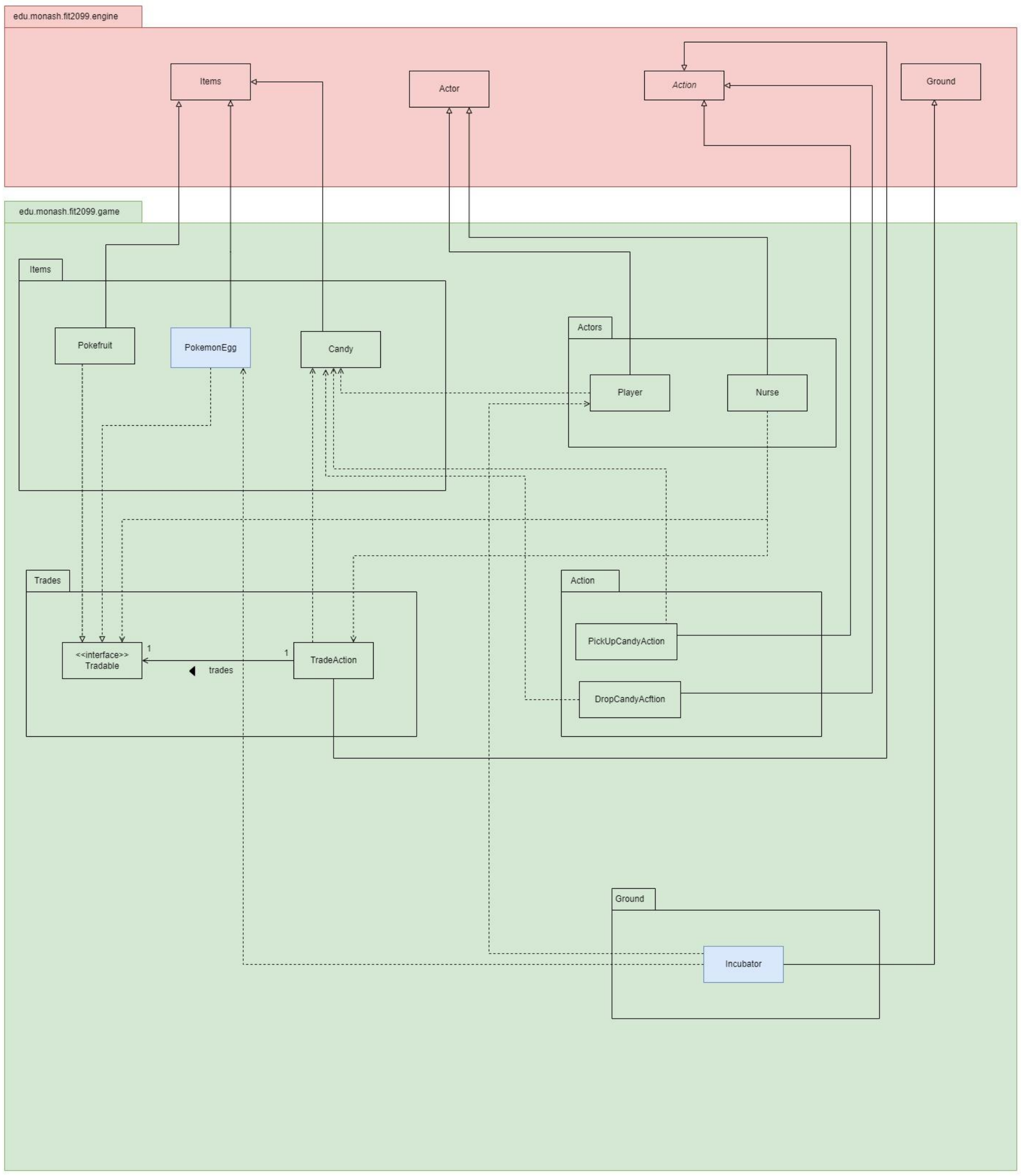


Design rationale

A new map of Pokecenter has been added to the map list of the game world, in order to teleport between two maps through a reusable door, a new Door class extending from the superclass ground has been created. For the teleport action between different maps, the EnterDoorAction has been created and generalised from the superclass Action. The map will be manually put into the game, and the sample action of teleport of the reusable door will be done in the application based on the EnterDoorAction as well

REQ3:

Pokemon Egg and Incubator



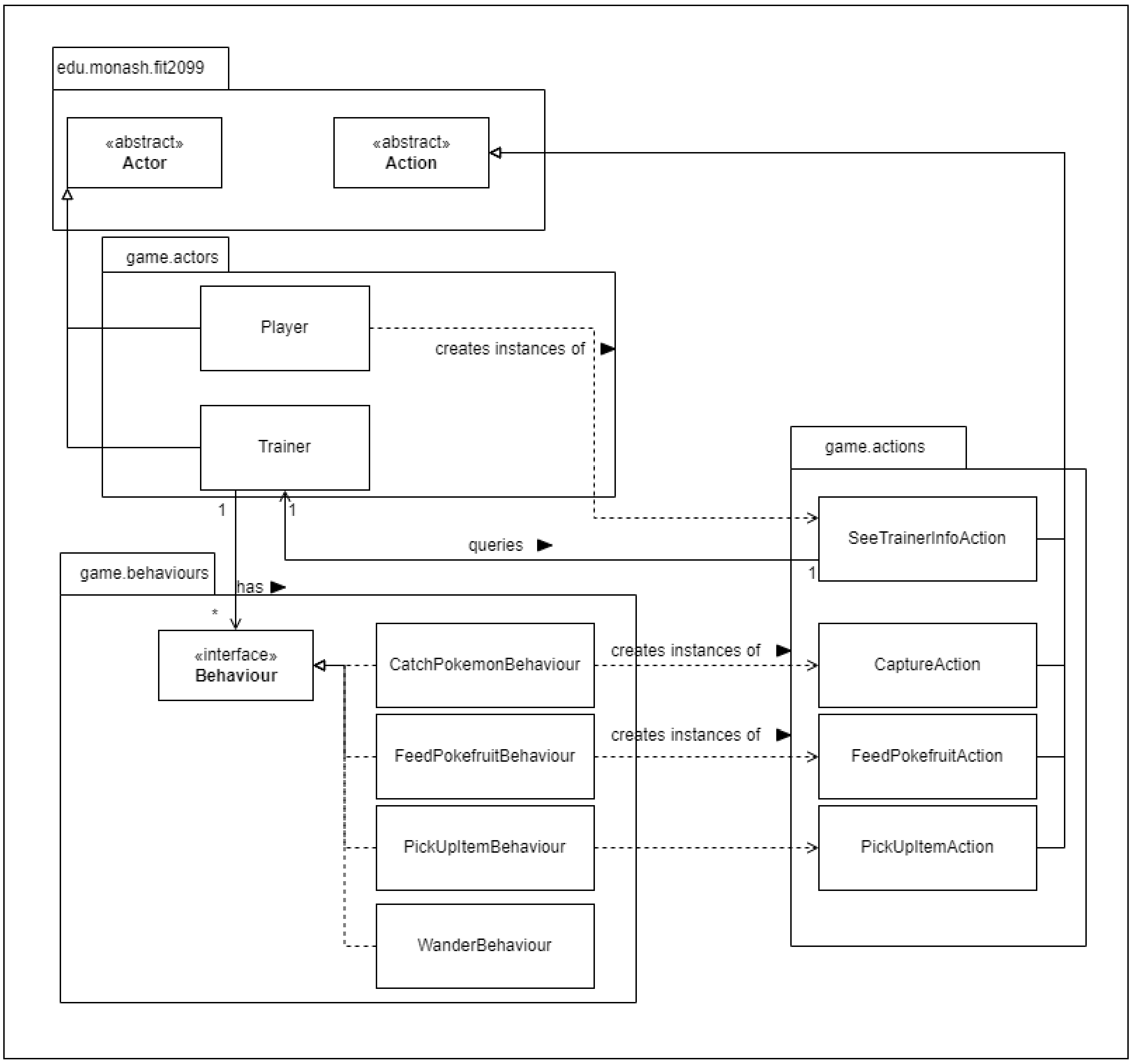
Design rationale

A new *Tradable* item, PokemonEgg is introduced which is obtained by the player by trading Candies with Nurse Joy. The generalisation of items from Item class in the game engine, allowed PokemonEgg class to inherit all characteristics of an item, thereby, allowing better reusability of the code. The realisation among PokemonEgg class and Tradable interface adheres with Dependency Inversion Principle.

The Incubator class extends from Ground class in the game engine. Again, generalisation here allows Incubator to inherit all characteristics defining a ground, allowing better reusability of code. Incubator also has a dependency relationship with PokemonEgg and Player classes.

REQ 4:

New Trainer



Design Rationale

The non-player trainers are implemented through the Trainer class, which contains a list of behaviours in order to automatically and orderly conduct the actions available to the character. This design is in consideration of the single-responsibility principle. In the behaviour lists, newly added classes implements Behaviour: CatchPokemonBehaviour, FeedPokemonBehaviour, PickUpItemBehaviour will be used to prioritise the character behaviours and to get the actions needed for the new trainer playTurn(). The SeeTrainerInfoAction will contain the queried trainer as an attribute in its class to conduct the query. This action has to be added to the action list of the Player through the Player.playTurn() method because World.run() only adds allowableActions of character surrounding the player.