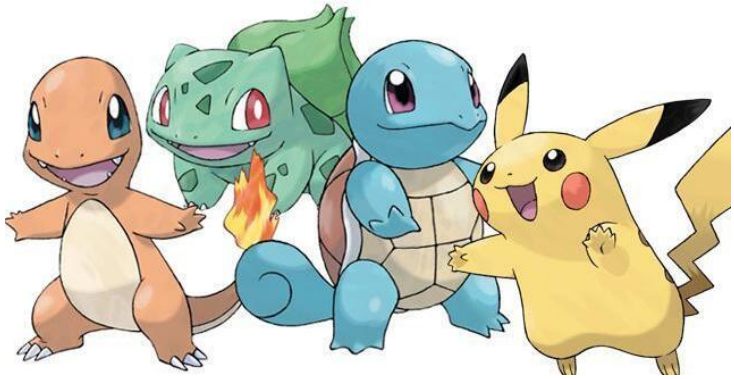
JUNIA ISEN

SMA Simulation

Concept Object / M2



Nicolas DEBEURNE

Mathieu RANC

Swann DE ROQUEMAUREL

Andrea FILIPOVIC

Maja RADOICIC

Marta LJUBOJEVIC

20/11/2022

1. Introduction

This report describes the architecture of the project. The purpose of this project is to develop a simulation of a multi-agent system in Java.

In this project we have implemented basic concepts in Object Oriented Programming such as inheritance, polymorphism and encapsulation. We also used design patterns such as singleton. We can see the use of encapsulation for example in class Trainer, where we have TakeMessages function and the attribute messages are private. The use of inheritance is seen for example in Trainer which inherits CommonTrainer, where we can also see the use of polymorphism. The use of singleton is seen in Master being classes, to provide making only one object.

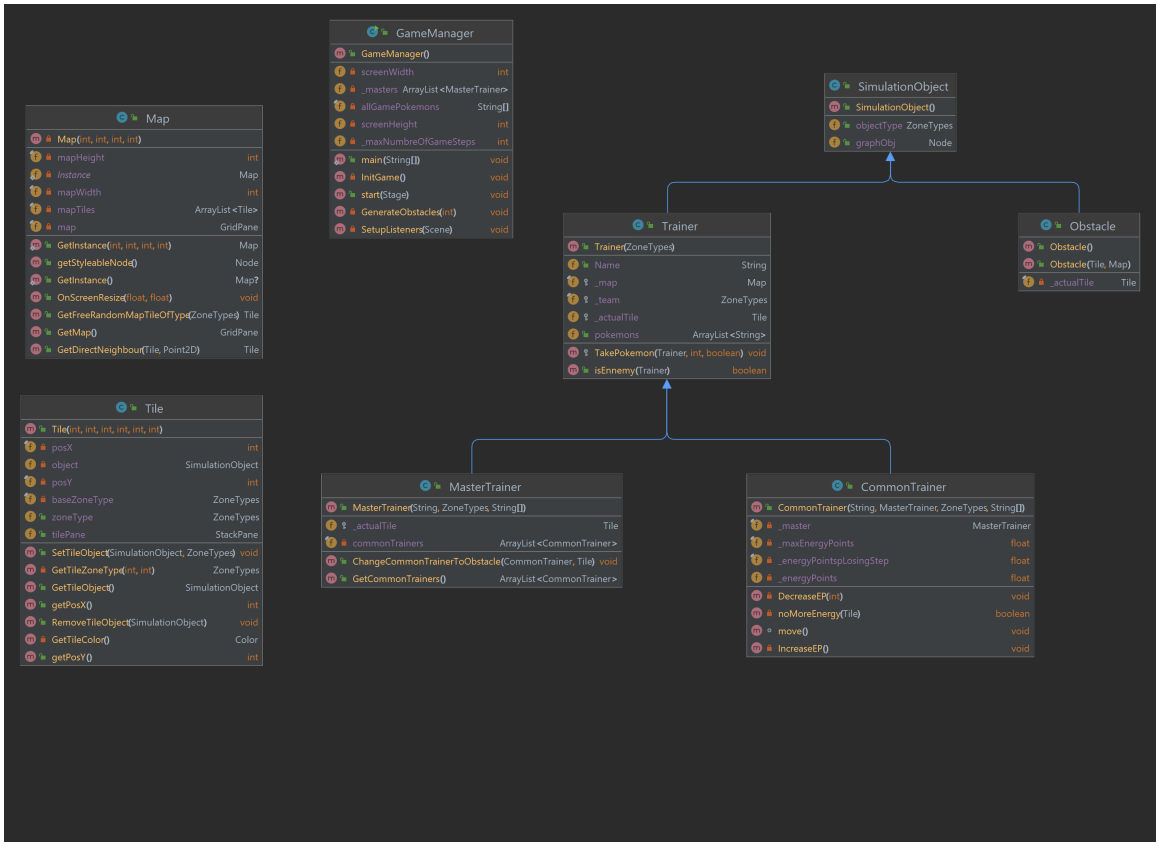
2. About the project

It's a simulation of a multi-agent system, more likely to board a game with four groups of beings (Pokémons) : Fire Team , Water Team, Plant Team, Thunder Team. These 4 groups are separated into two alliances : Fire/Water and Thunder/Plant.

When the game begins, the Common beings (Pokémon trainers) start to move in random directions. Each Common beings consume energy points when they move so when they have less than 20% of energy they go back to the safe zone of their team to recharge. They can collect Pokémons when they meet enemies and win the fight (random boolean) or exchange Pokémons when they meet their allies. The main goal is to collect as many different messages as possible. Every group has its own master who collects all these messages. The master from a specific group who has collected all the messages from the game or has the most messages at the end of the simulation will win.

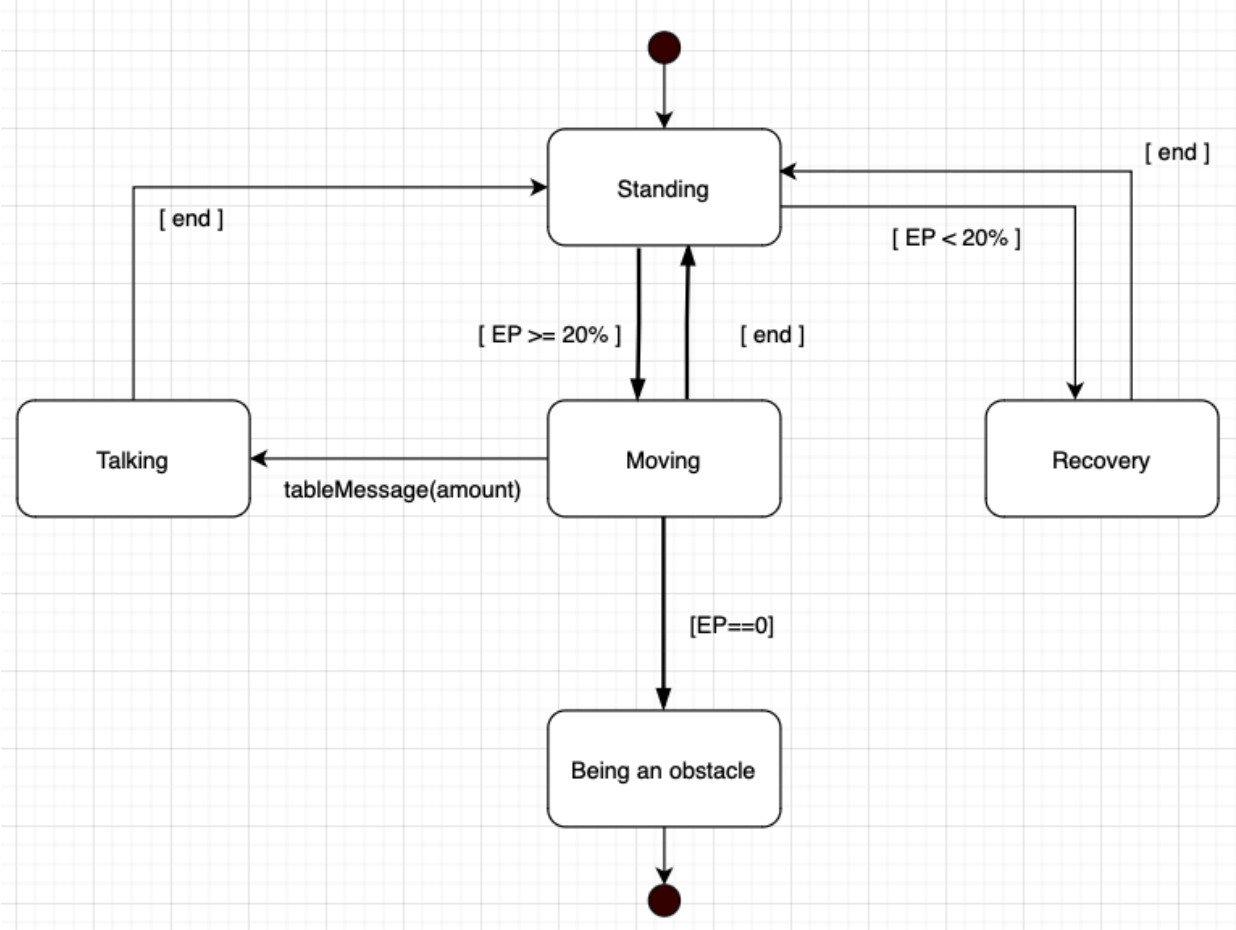
3. UML Diagrams

3.1. UML CLASS DIAGRAM



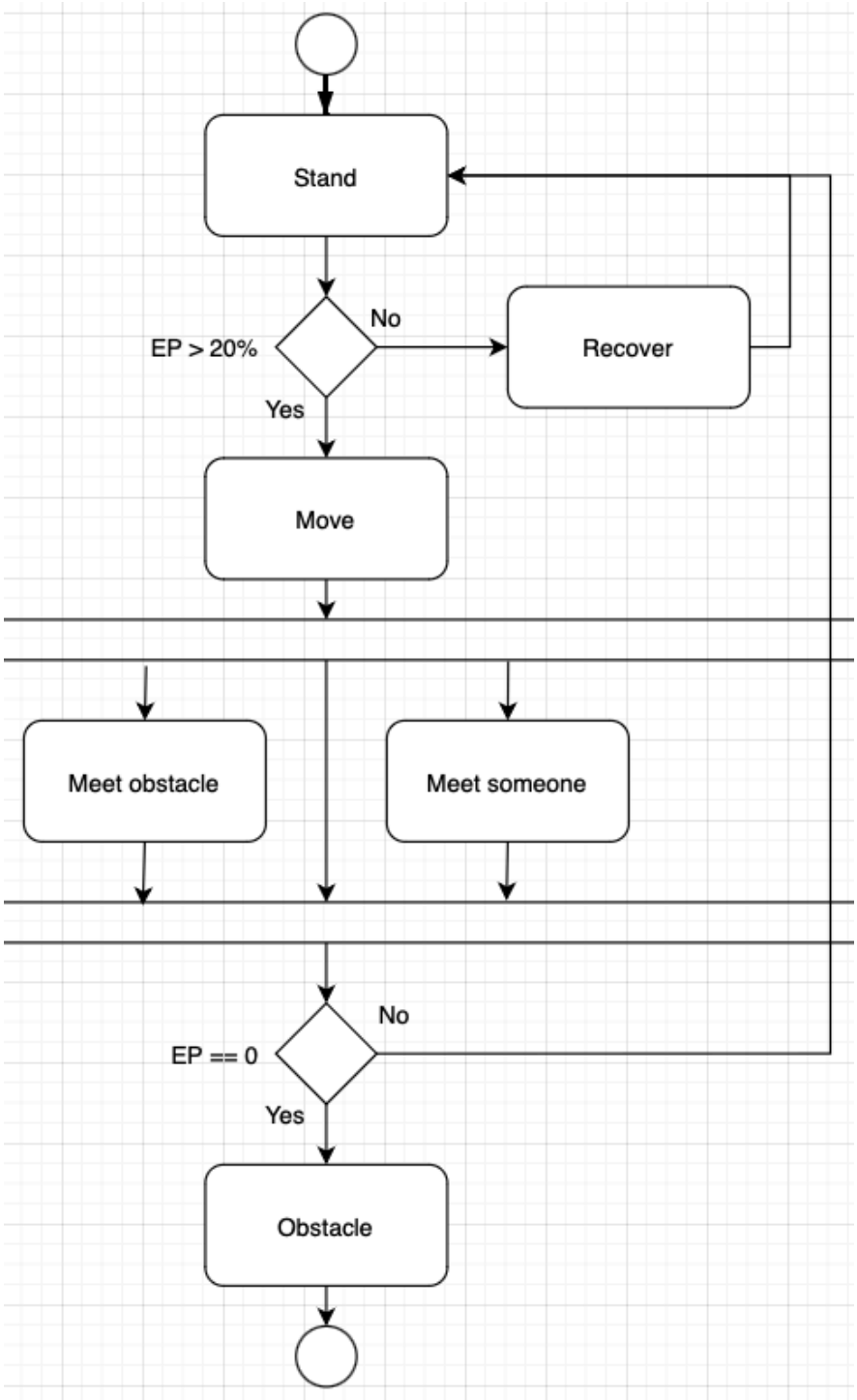
Picture 1 - Uml diagramme

3.2. STATE-MACHINE DIAGRAM



Picture 2 - State-Machine Diagram

3.3. ACTIVITY DIAGRAM



Picture 3 - Activity Diagram

4. Analysis model of the game project

The Map class will contain a constructor which will generate the size of the map, the Tile class define if the tile is neutral or safe. The start method of GameManager launch the simulation and call some functions to set up the map and the beings (Pokémon trainers), such as GenerateObstacles and InitGame. The Map has a singleton pattern to ensure that multiple instances of this class cannot be created.

Trainer class contains the Pokémons in an array of Strings, energy points which will be set on 100 at the beginning of the game, the maximum number of messages which can be 20, and also their total number.

CommonTrainers take random Pokémons from enemies and exchange Pokémon with allies using TakeMessage function.

If a CommonTrainer reaches the MasterTrainer of his team, the Master will pick up all the messages from him using TakeMessages function. CommonTrainer class contains several functions such as Move, decreaseEP, increaseEP (to manage energy points). If a CommonTrainer has no more energy the noMoreEnergy function will be called to transform the trainer to an obstacle.

5. Roles

Nicolas DEBEURME - Writing the code

Mathieu RANC - Writing the code

Swann DE ROQUEMAUREL - Writing the code

Andrea FILIPOVIC - UML diagrams / report

Maja RADOICIC - UML diagrams / report

Marta LJUBOJEVIC - UML diagrams / report