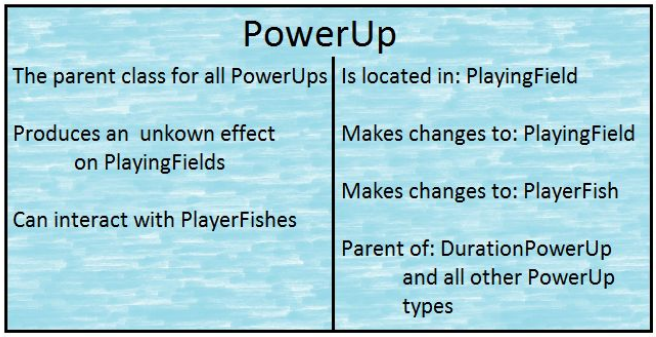
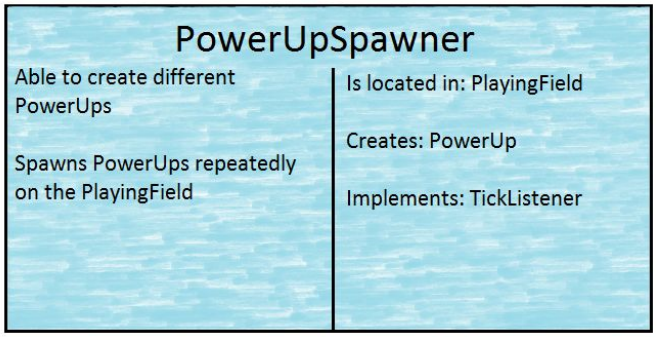
Assignment 2 - Group 26

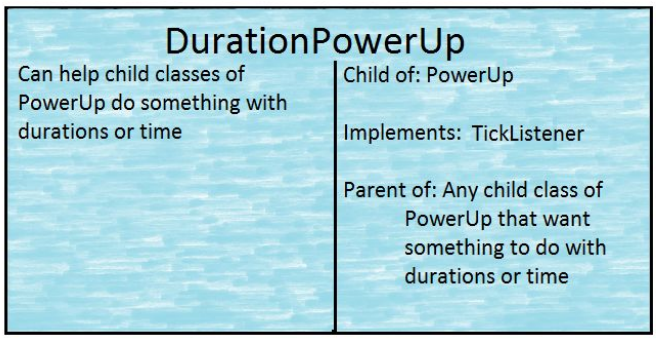
Exercise 1 - 20-time

As the extra feature, we've decided to implement PowerUps. PowerUps are entities that execute a certain effect on the PlayingField/PlayingFish once a PlayerFish has eaten it.

These following are the CRC cards are for the main classes of the PowerUp feature.



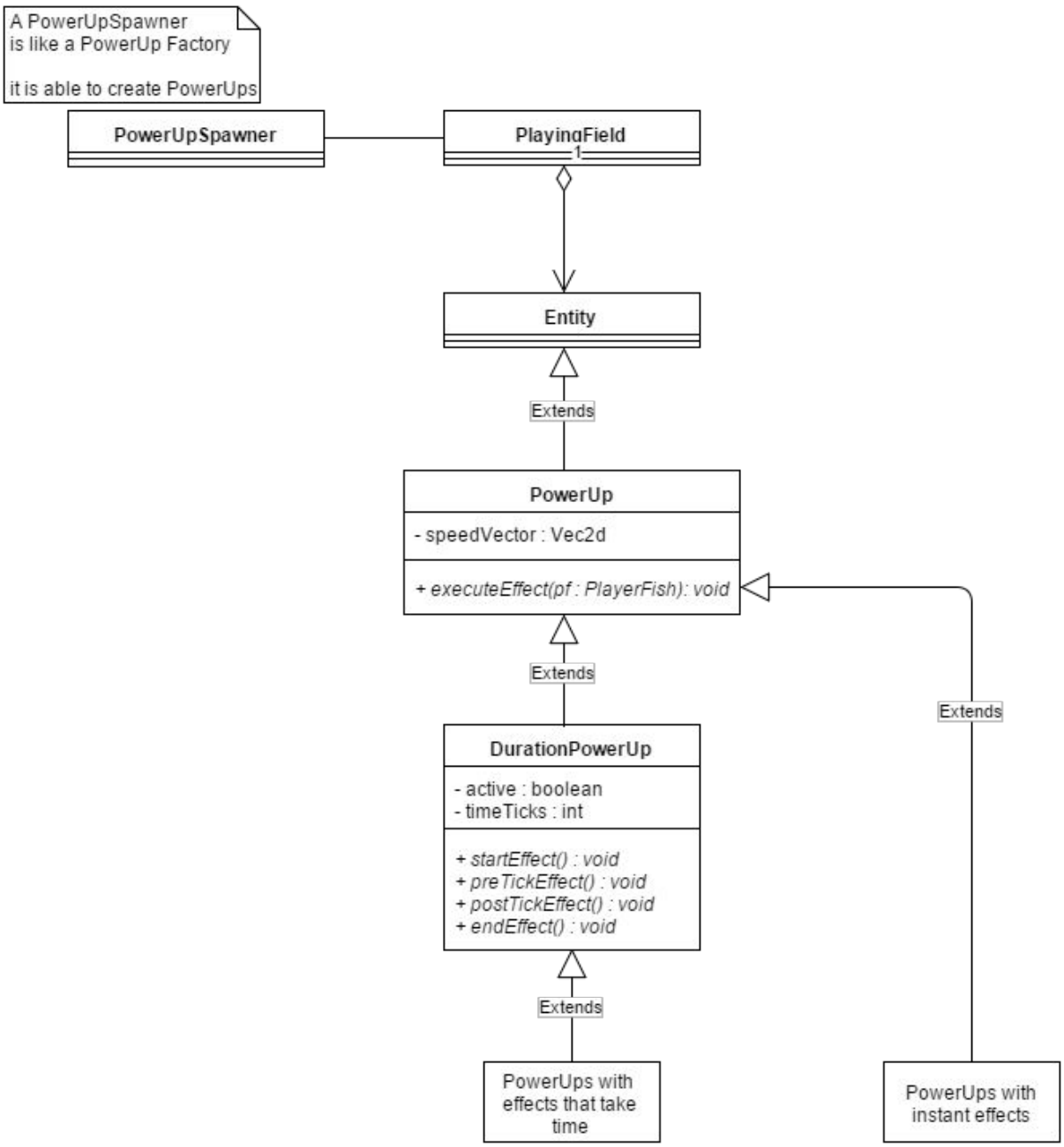




There are several other classes for the PowerUp feature, but these are only different children of the PowerUp class that implement the unkown effect.

An example child of the PowerUp class is PuExtraLife. This class has the simple effect that it gives a PlayerFish an extra life. A different Powerup child is PuFreeze, which freezes all EnemyFishes on the PlayingField for ten seconds. Because there are more PowerUp children that last for a certain duration, it's simpler to make one class that does this for them. That is the responsibility of the DurationPowerUp class. It can play a different effect at the start, each tick and once the duration is over.

The (simplified) UML can be found on the next page.

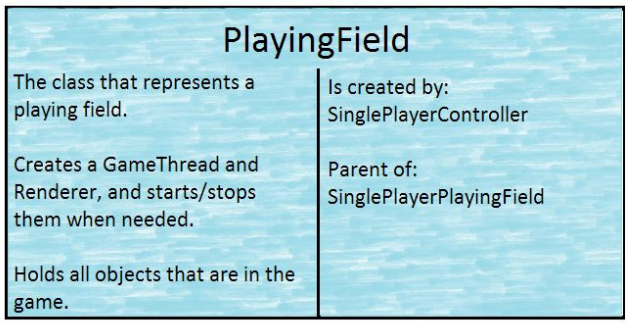


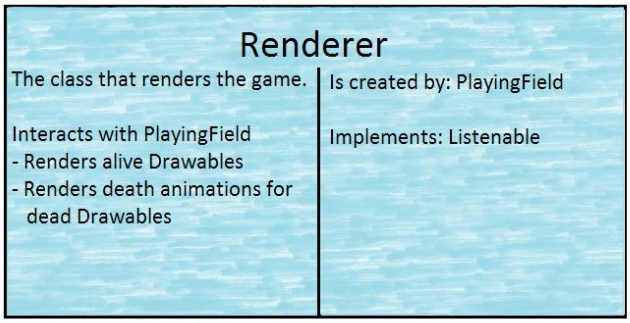
Note: the classes PowerUp, DurationPowerUp and all their child classes are located in a seperate package com.github.fishio.power_ups.

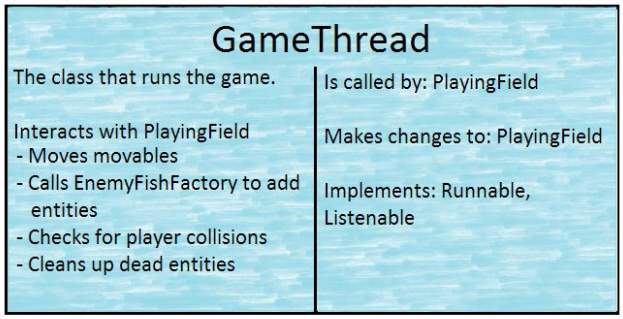
Exercise 2 - Your wish is my command

As the TA assignment, we had to split up the PlayingField class into 3 parts: GameThread, Renderer and PlayingField. The reason for this is that it has too many responsibilities.

We first created CRC cards for these 3 classes, in order to decide what responsibilities would be assigned to each class.

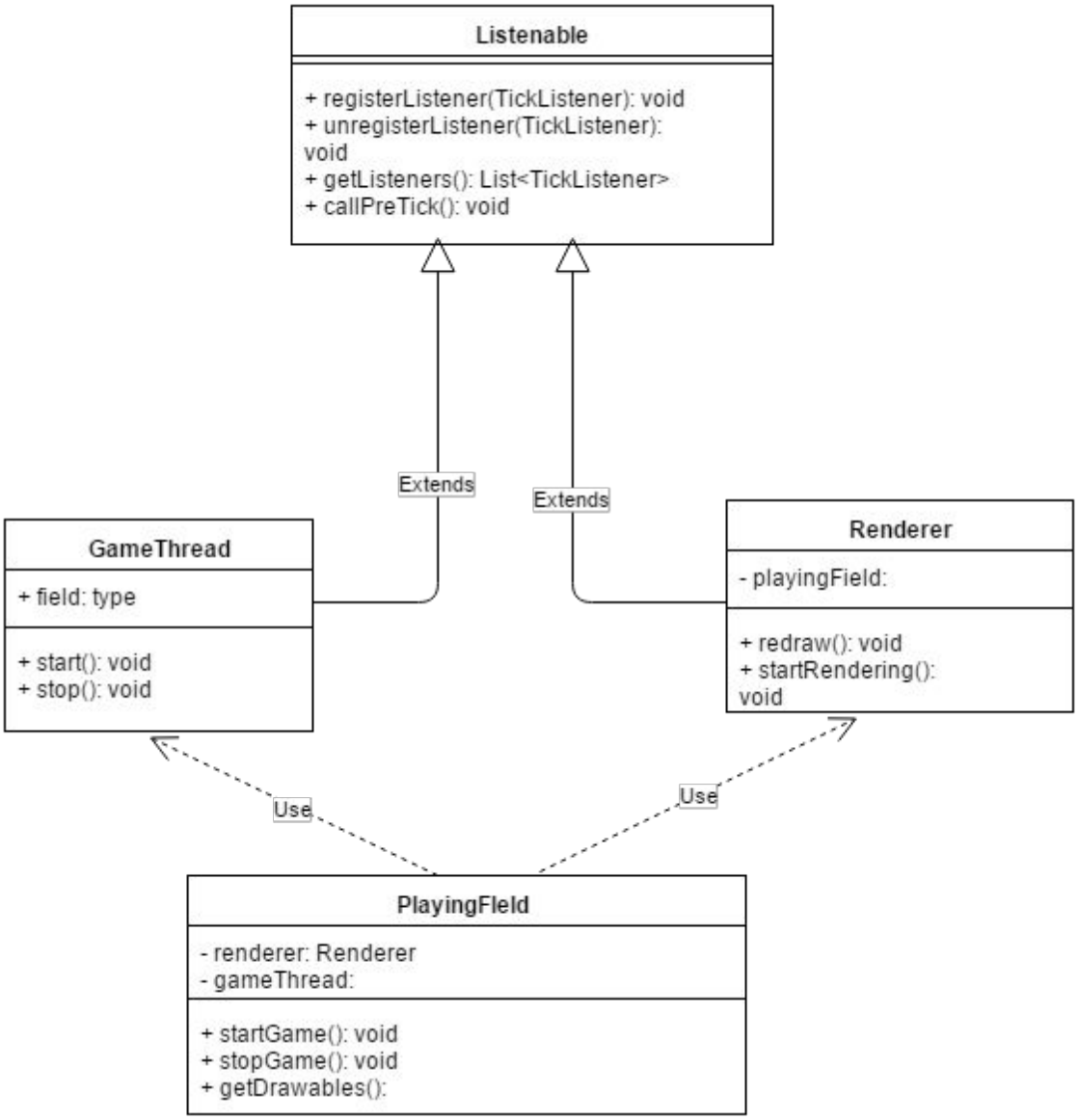






We decided that the PlayingField would still create the GameThread and the Renderer, and act as an interaction point to them (start/stop them), but that the PlayingField would no longer be tasked with the actual game or rendering itself.

The (simplified) UML can be found on the next page.


Listenable
+ registerListener(TickListener): void
+ unregisterListener(TickListener):
void
+ getListeners(): List<TickListener>
+ callPreTick(): void
Extends
Extends
GameThread
+ field: type
+ start(): void
+ stop(): void
Renderer
- playingField:
+ redraw(): void
+ startRendering():
void
Use
Use
PlayingFleld
- renderer: Renderer
- gameThread:
+ startGame(): void
+ stopGame(): void
+ getDrawables():