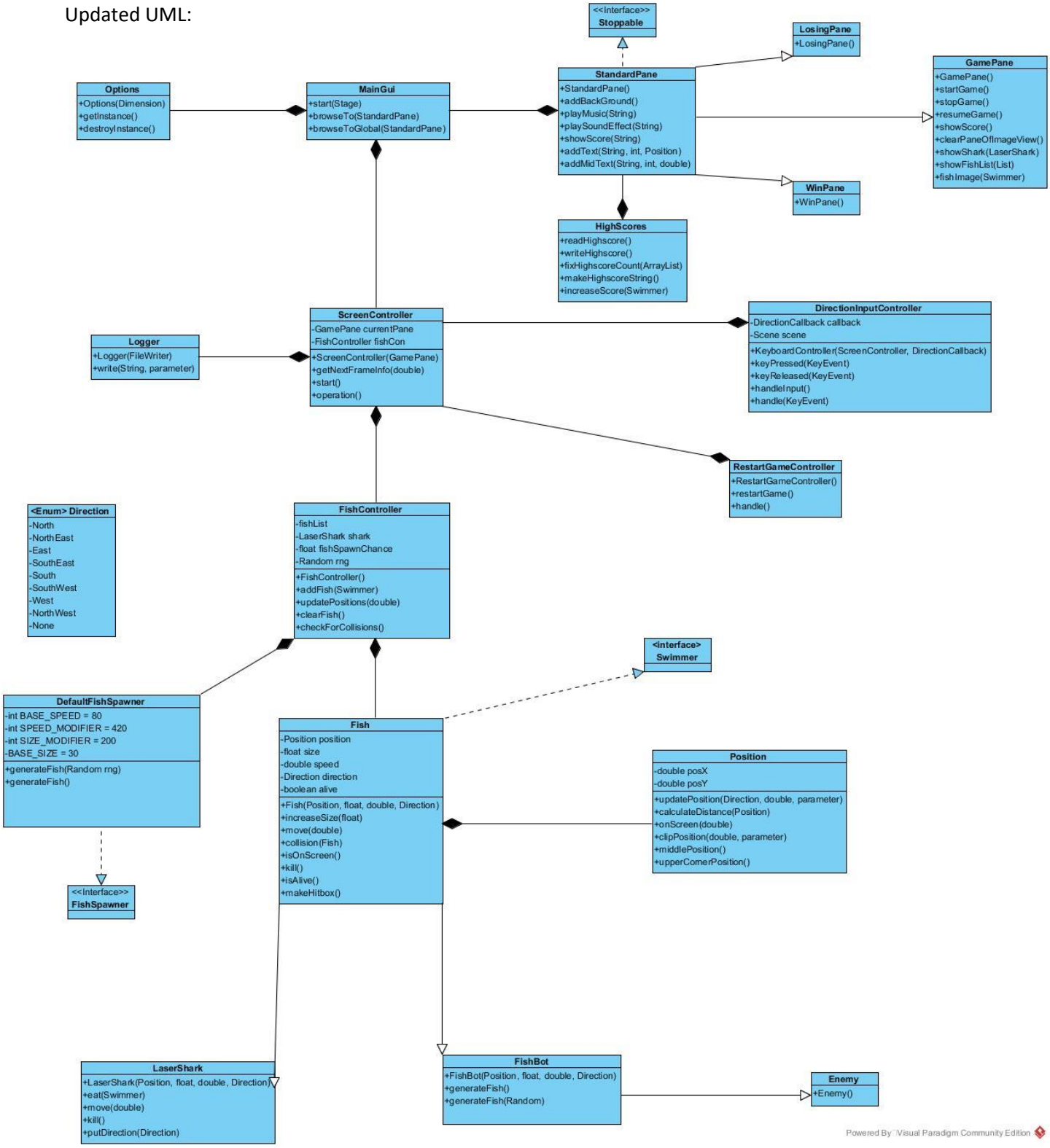
Assignment 3 Group 27

Exercise 1

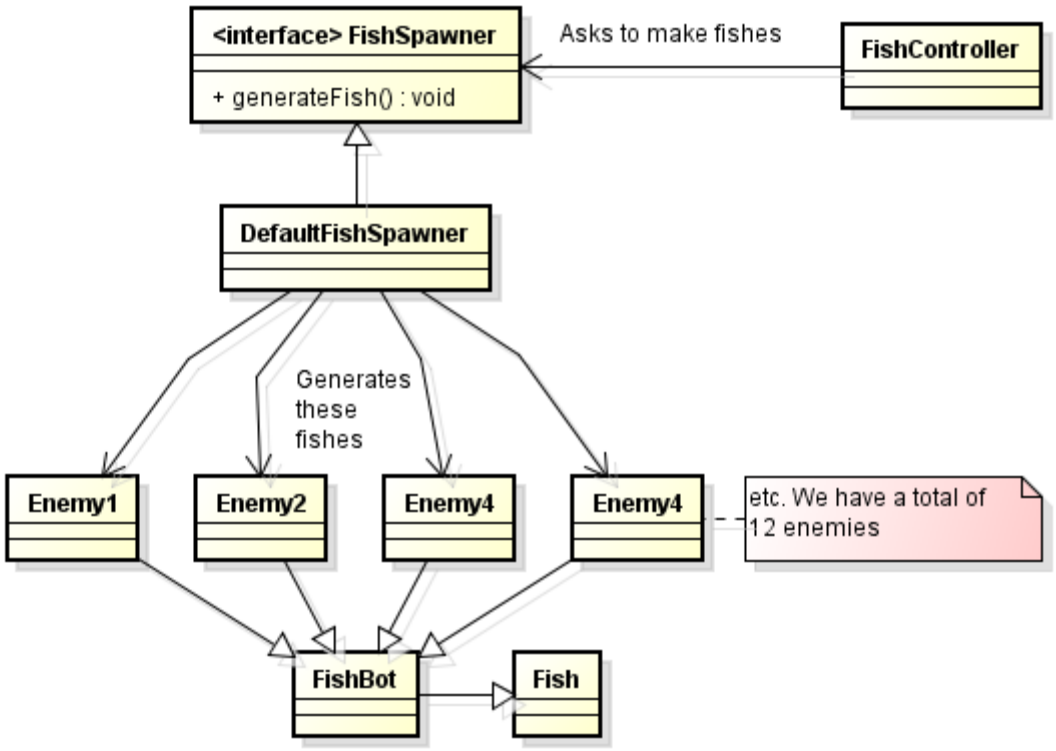
Updated UML:



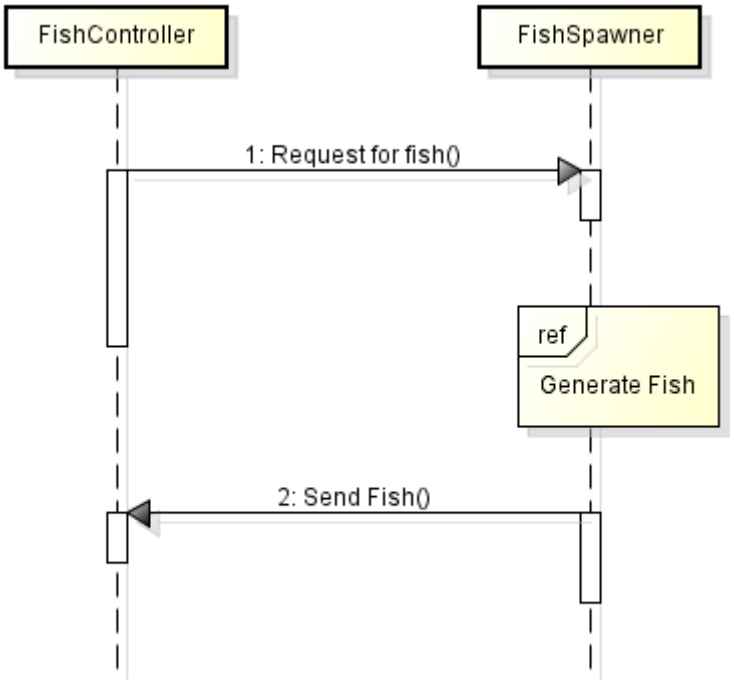
Exercise 2

Design Pattern 1:
The factory design pattern for creating fishes.

1. Before, spawning fishes was implemented statically inside of the FishBot class. Following the responsibility driven design, we thought this should actually be moved to a separate class. The factory design pattern fits great with this. We created a FishSpawner interface and a DefaultFishSpawner class that implements the FishSpawner. All the methods for creating Fishes were moved to this class. If we'd like to have an alternate way of how fishes are spawned, we could do this by making a new factory class. This might be useful if we'd like the player to change difficulty in the future.
2.



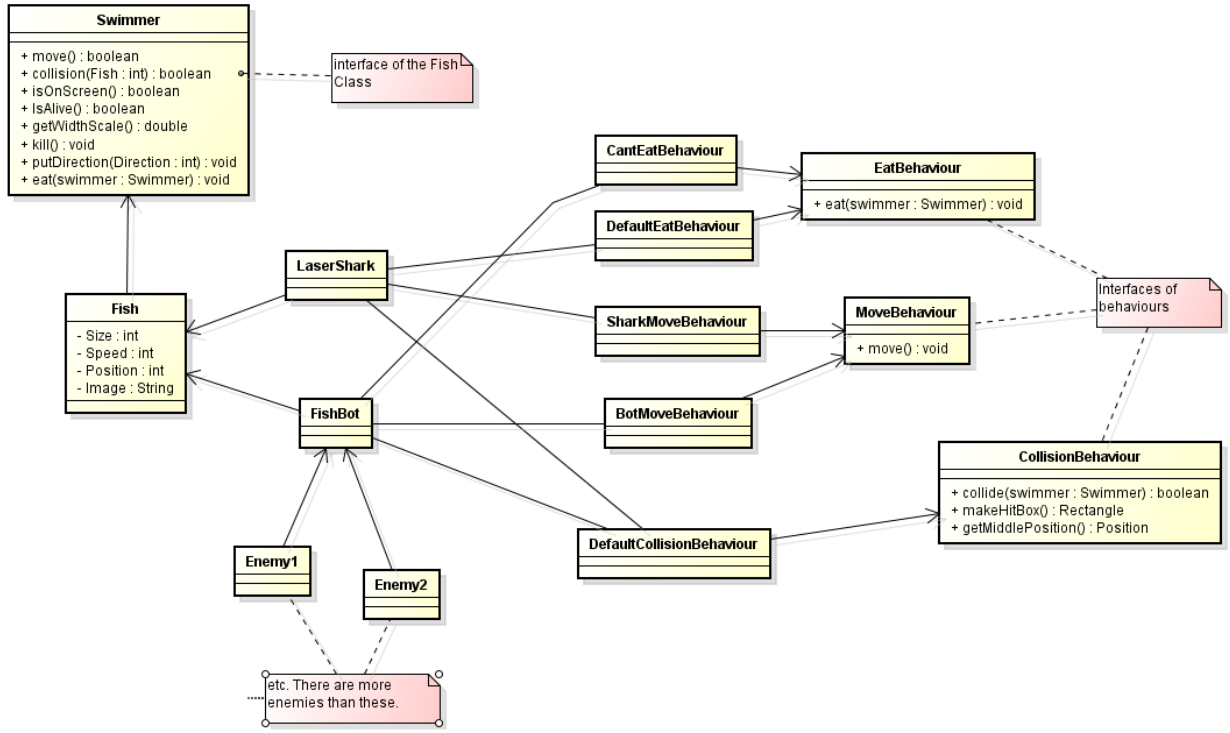
3.



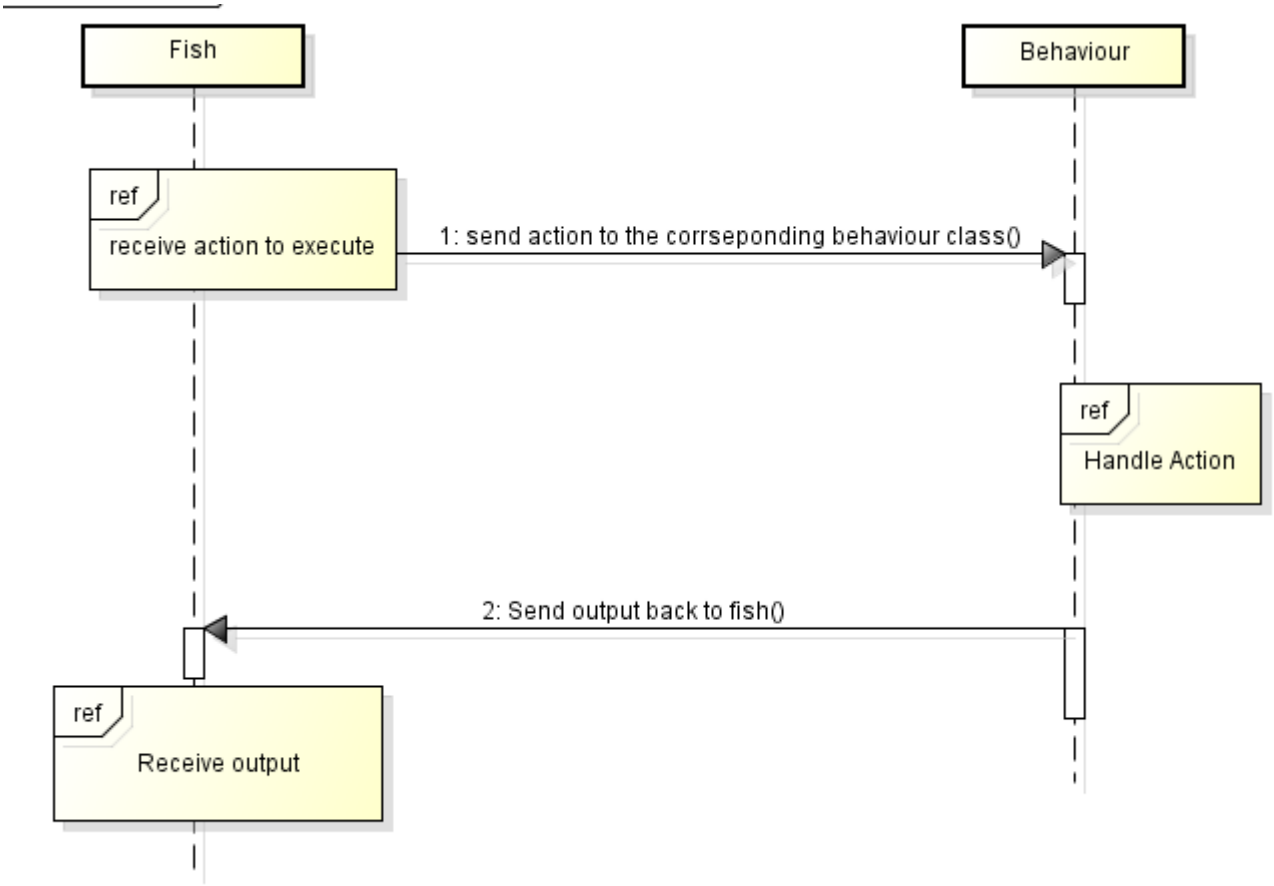
Design Pattern 2:
Strategy pattern for our fishes.

1. The fishbots and the shark have quite some similarities. They both move around the board, have a position, size etc. A strategy pattern could be an useful design pattern to implement this.We have a Fish class which implements the interface Swimmer. It contains some behaviour objects, these are implemented as an interface. These behaviour interfaces have some classes who implement this interface. Using combinations of behaviours make different types of fish. This way we could implement a fishbot or a shark, without having a lot of code inside the LaserShark or FishBot. We think this keeps the code organized and more responsibility driven, which makes it easier to implement further extensions. I decided it wasn't useful to implement a strategy pattern for enemy1 to enemy12. This is because these classes have a lot in common and should function almost the same way, These only have different images.

2. A class diagram.



A sequence diagram.



Exercise 3

1. To recognise good and bad practice, we need performance on both cost and duration of a project. When the performance on both cost and duration is better than the average cost and duration corrected for the applicable project size, it is considered as good practice. When the performance on both cost and duration is worse than the average cost and duration again correct for the project size, it is considered as bad practice

2. One might assume that in the three given companies, according to the paper, the Visual Basic skills were stronger than other skills. It is more likely that Visual Basic project environments on average are less complex than others, and due to that, end up in the Good Practice quadrant more often.

3. 1. Testing: testing belongs to good practice, because testing is really required to point out the defects and errors that were made during the development phases. Some of these defects and errors are expensive or dangerous. We need to check everything we produce, to make clear there are no defects or errors in de code.

   2. Reviews: reviews belong to good practice, it is important to review other people's work. Reviews can identify issues earlier and more cheaply than they would if they are identified by testing. Early and frequent reviews of small work samples can identify systematic errors in the author's work processes, which can be corrected before further faulty work is done.

   3. Requirement management: requirement management belongs to good practice, because gathering and agreeing on requirements is fundamental to a successful project. A requirement is a feature, specification, capability or constraint your product must have on delivery to the customer. Requirements management is a continuous process throughout a project.

4. Many team changes: many team changes belong to bad practice. When people join or leave the team during a project, they are not part of the whole development process. When a new team member joins a team, he has to understand what is already made and what has to be done. This takes a lot of time, especially when there are a lot of team changes.

   Bad relation with external supplier: a bad relation with an external supplier belongs to bad practice, because you have to work with the external supplier. It is self-evident that a bad relationship with the external supplier is bad for the project development. A bad relationship can lead to irritations, which are not beneficial for the development process.

   Inexperienced team: an inexperienced team belongs to bad practice, because working with inexperienced team members destroys productivity. Mistakes and failures are discovered late and have to be fixed late in the development process which is time consuming.