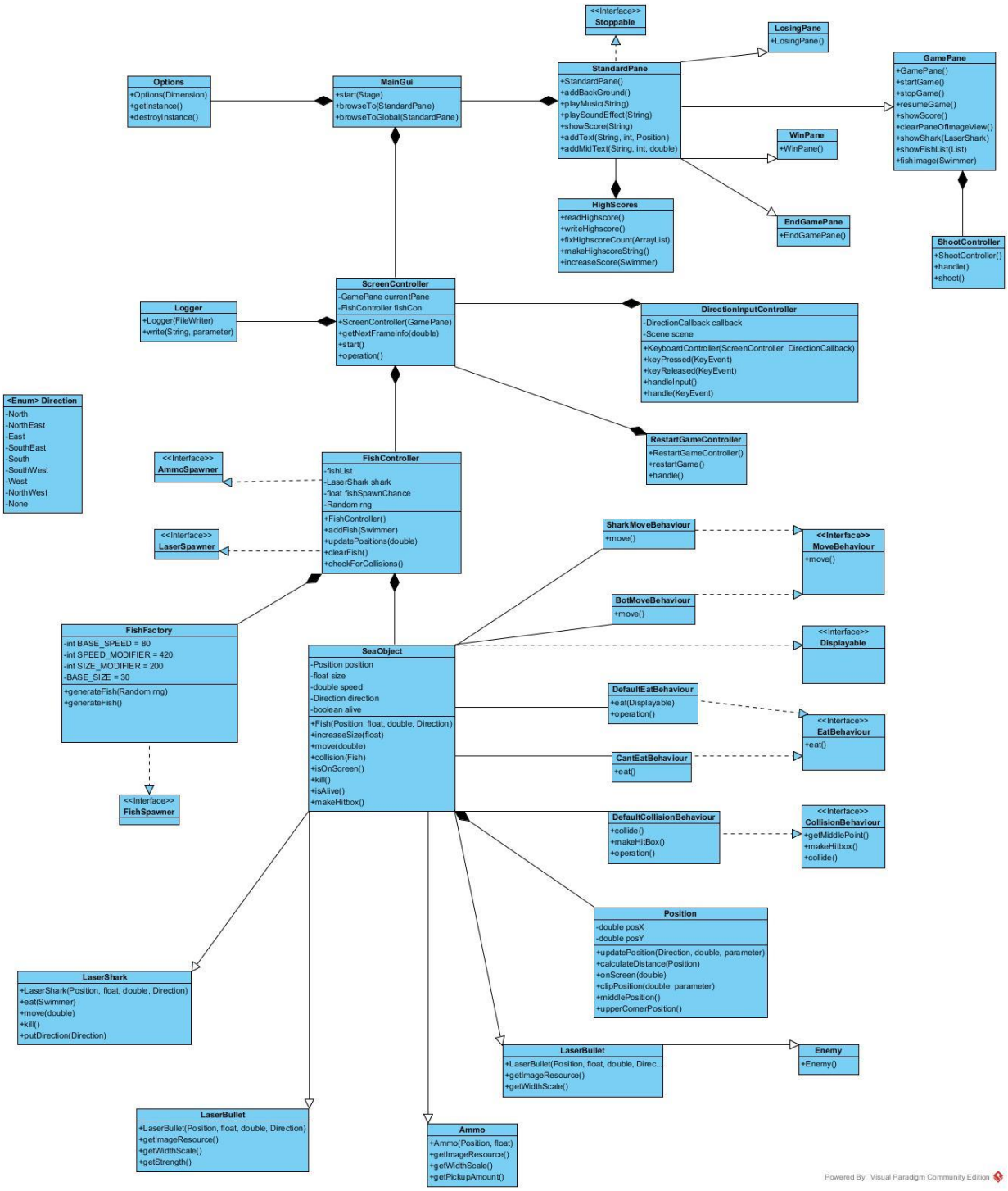
Group 27 Assignment 4

Exercise 1

Updated UML:



Exercise 2 - Software Metrics

1. The resulting analysis file is located at LaserSharks/Documents/20151013LaserSharks_initial.result

   The analysis gave only two design flaws.

2. Design flaw one, :
   a. The first design flaw that showed up was duplication between two classes namely the LosingPane and the WinPane classes. After taking a look we saw that almost everything in the two classes was the same apart from a single String value. We decided that since these Panes are both only shown at the end of the game we decided that we would have them inherit all the duplicate code from a class that we called EndGamePane. The way we then changed the string was by passing it through in the constructor. We probably missed that these classes were near identical because they were probably made by two different people. then someone went ahead and changed the way the graphical user interface was structured and missed that those two things were nearly identical.
   b. Instead of two classes with near identical code we now have two classes that extend a standard EndGamePane.

   Design flaw two, internal duplication in DirectionInputController class:

   a. This internal duplication is shown up in the analysis because in the DirectionInputController are two methods, keyPressed and keyReleased, which are almost identical. These two methods could be reduced to one method which is called from the keyPressed and keyReleased methods.
   b. This design flaw is solved in the DirectionInputController class.

   Design flaw three, factory design pattern:

   a. Our last design flaw did not show up in the analysis of the inCode program. This is one of the reasons that we think it is the least severe. Another reason is that it did not have any negative impact on the code. The only thing that was wrong with it is that it was not created according to a design pattern and using a certain design pattern it can be implemented in a much more clean way. This design flaw was the fact that we had implemented the different enemies as 12 different classes where a factory design pattern as described in lecture 07 - Design Patterns I would have been perfect. The reason for not using this design pattern was the fact that we had not seen lecture 07 yet. Because of this, the factory design never came to mind and we chose for a simple way of implementing the possibility of spawning multiple different fishes. This simple way was making 12 different classes, with every class containing one of the multiple enemies.
   b. The 12 enemy classes have been replaced by a FishFactory class.