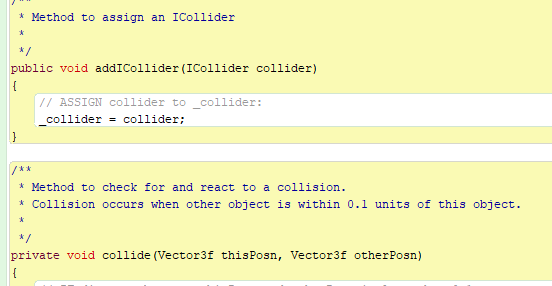
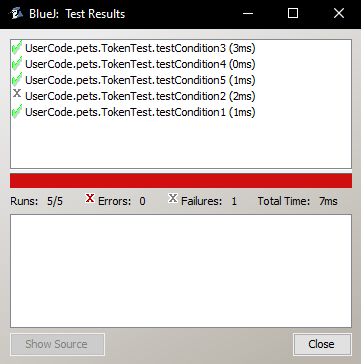
1.Make an initial assessment of the project so far: does it compile and execute without error, what does it do, does it appear to function correctly based on the description above, is it well-commented, does the Javadoc give sufficient details, does the code look efficient, has it been tested fully?

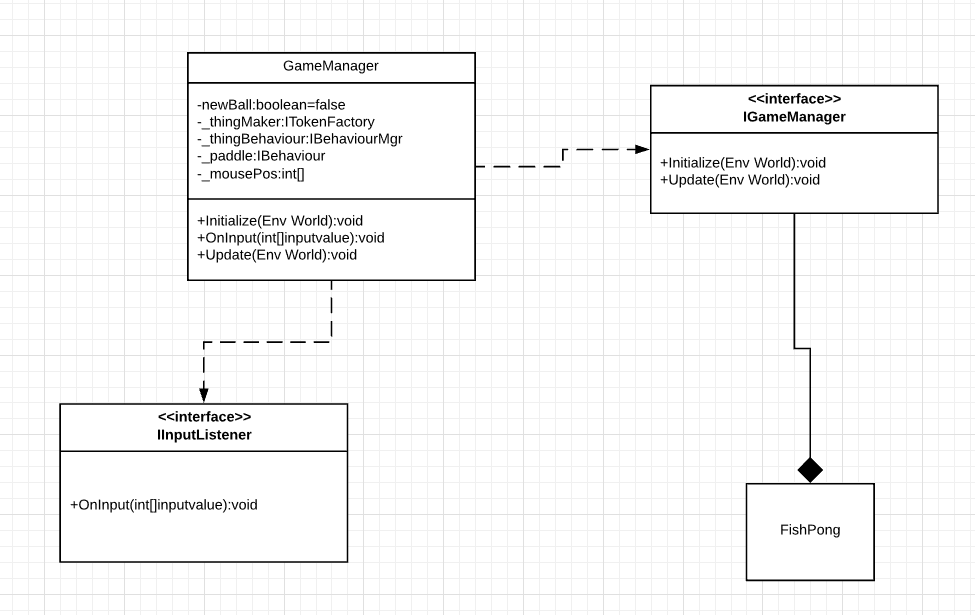
On initial assessment of the projects, when searching for the Javadoc in the directories, it does not exist, meaning that it has not been regenerated. However, the project does compile and execute without error. The description of the project is a two-player bat and ball game, where one player throws the ball and the other player hits it. After running the code, the game has some of its base functionalities missing or not behaving as they should. For example the piranha can be controlled with the up and down arrow keys and the ball can be thrown with left mouse click, but after the ball collides with the player once, every ball that is instantiated afterwards travels right, instead of towards the piranha. This makes the game unplayable after one throw and is therefore a bug. After regenerating the Javadoc, I can see that there are class descriptors for each class, sufficient use of key words when commenting, for example ‘DECLARE’ for each declaration. Method descriptors describe what the method does, but for some classes like BallBehaviour for example, there is no description for the parameters(As seen in screenshot below).

The project includes two separate unit tests, one for the mouse handler class and one for the token class. The mouse handler unit test is done with a test harness, meaning that there are no parameters that are being tested, only that the methods are being called correctly, which this test does just fine. The second test is fully testing the token class with each parameter correctly being tested. When the test is run however, there is one failed test. Test condition 2 has failed its test. This tests to see if a tokens coords are out of range of the aquarium. This means that even when the coords are out of range, the OutOfBoundsException is not executed.



2. The GameMgr class violates at least one of the five SOLID principles. Name one of them and explain how/why the class violates the principle. Modify the design in a way that you think will rectify this problem.

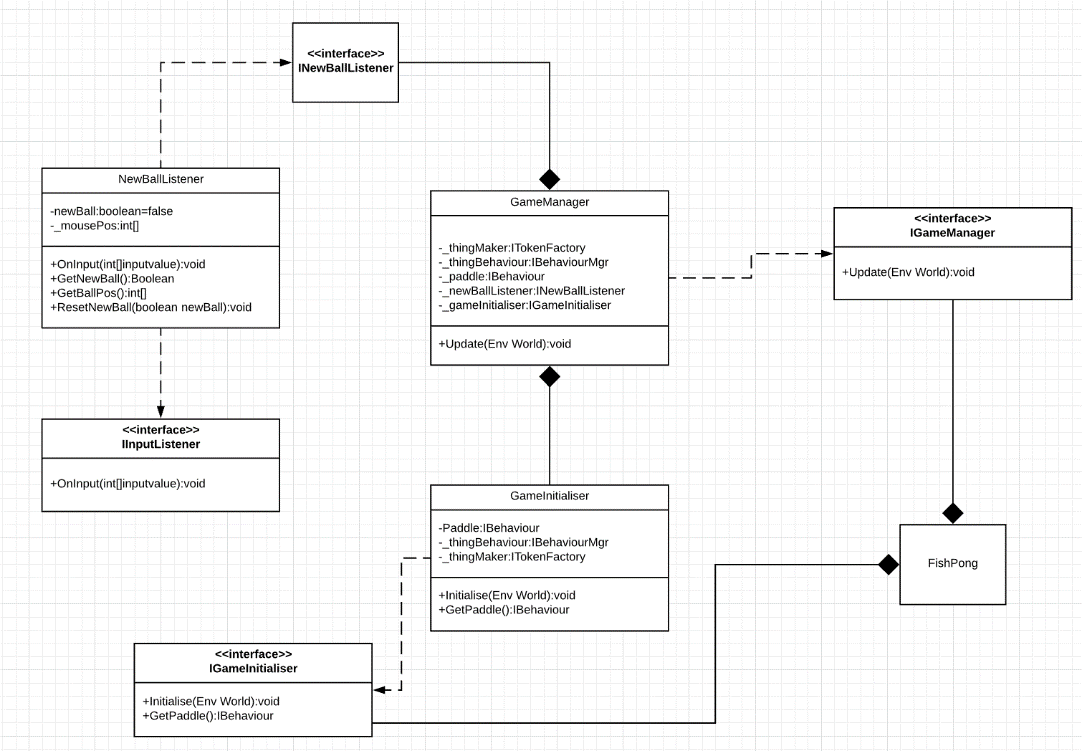
Previous design



The single responsibility principle states that any class that the developer makes cannot have more than one responsibility Janssen, 2018.). For example, a class that creates an entity and enacts their behaviour would not be following this principle because it has two completely separate jobs Janssen, 2018.). A benefit of using this principle is that helps prevent unexpected side effects of future implementations. Because if you need to alter a responsibility within a class or the class itself, then having one responsibility per class means that other responsibilities won’t be affected Janssen, 2018.).The GameManager class violates the single responsibility principle because it has 3 responsibilities- Updating the behaviour of the ball and paddle, Initialising the Paddle and listening for new input from the input publisher.

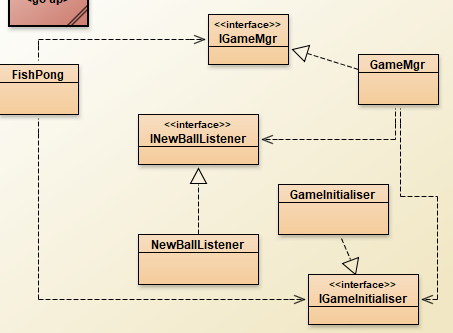
So, to rectify this issue, I must refactor each job into a separate class. I decided to keep the update responsibility inside the GameManager class and take out the listening for input and initialising the paddle responsibilities. The new design is below.

New Design



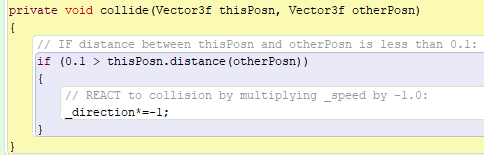
The idea of this new design is to have one class that listens for input and returns a Boolean to see if a new ball should be created and returns the position at which it should be created at. The GameInitialiser class takes the initialising responsibility away from the from GameManager. This initialiser class with create a paddle behaviour and return the paddle behaviour back to the GameManager.

3. Implement your modification to the design in 2 above.



4. When the program is running, there is an error in its behaviour: specifically, when a ball (fishfood) collides with the paddle (piranha), the response is not as intended. This is due to a bug in the BallBehaviour class. Explain the erroneous behaviour, diagnose the bug, and implement the fix.

When the ball collides with the paddle the speed gets reversed, but because in the update of ball behaviour, the position is being added by speed x the direction of the ball, the direction never changes. This results in the ball moving backwards each time it is spawned because the speed is always reversed. To get around this I change the speed from being revered in the collision code block to the direction being reversed. This allows the ball to always moving in the desired direction.



References

Janssen, T., 2018. *SOLID Design Principles Explained: Dependency Inversion Principle With Code Examples*. [online] Stackify. Available at: <https://stackify.com/dependency-inversion-principle/> [Accessed 16 March 2020].