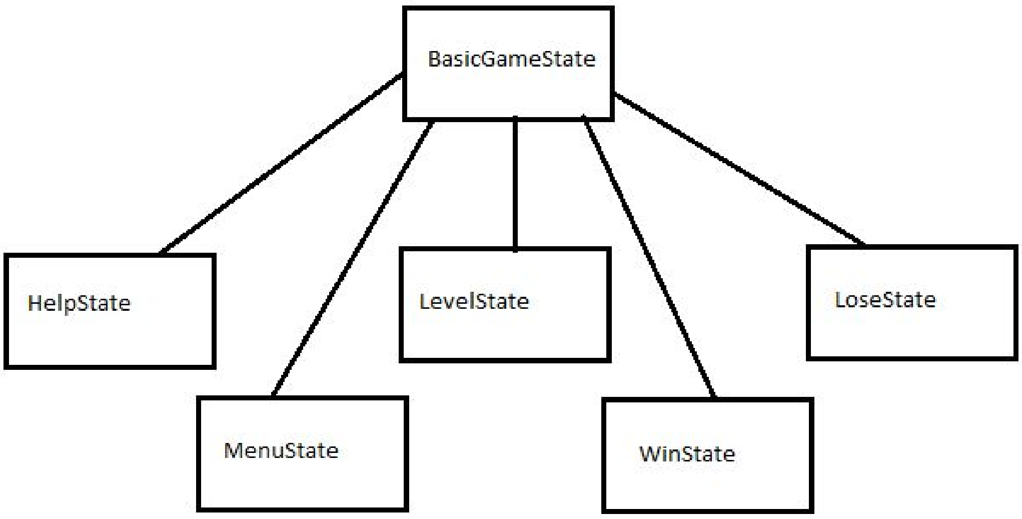
Exercise 1:



Exercise 2:

1. The analysis file of inCode can be found in the folder:
   fishy > docs > assignments > metrics > fishy_1444726078371.result
2. There was only 1 type of design flaw in our project: Data Clumps. We decided that the other two design flaws we would consider would be: God Classes and Feature Envy.

Design flaw: Data Clumps
The design flaw that was detected by InCode in our project was ‘Data Clumps’. This means the attributes of an external class are used a lot by other classes. In the Fishy project, the *Entity* class has four attributes concerning movement of the entity. These are dimensions, position, velocity and acceleration. All of these vectors were passed from constructor to constructor, all the way down until the *Entity* class was invoked. This is not a good practice, because it requires your constructor to always have at least four arguments. Which decreases the readability.

The solution to this problem was to create a separate class *Moveable* to store these four vectors. Also the generic logic for movement, like updating the position and calculating drag forces, could be moved to this new class. Right now every class that inherits from *Entity* should call the parent constructor with a new instance of the *Movement* class to store information about these four vectors mentioned before. Rewriting solved all warnings that were previously given by InCode.

Design flaw: God Class:
We don’t really have god classes at the moment. But we do have some potentials.

Take Player.java for example. This class is getting really crowded. It was already said in one of the pull requests, but we intend to address the situation after merger with development. Right now it computes a lot of problems, although for itself, that could easily be taken over by other or even new classes. We have avoided a god class thus far by splitting the classes when they get to big. So instead of a controller also controlling the spawns, we split them into a controller and a factory as is done with power up but is yet to be done with the opponentController.
The classes with most chance of turning into a god class are player, and the controller classes. Although controller classes have tendencies to do everything, they are called controller classes for a reason. And I think we managed to keep them in check as good as possible. Opponentcontroller needs some splitting but it has reached it peak in level of “godness”.
Player needs some work, it is too big right now but we will address that problem soon.

Design flaw: Feature Envy:
We currently do not have any methods with feature envy. Though it is very easy to get these in our project. Since a lot of methods and classes rely on eachother a lot. Both the controller classes (PowerupController and OpponentController) use a lot of methods from other classes, namely the opponents and powerups and thus are very susceptible for this design flaw.
InCode could also probably say that some of our bigger testclasses have Feature Envy, since these also use a lot of methods from other classes. This is something we do not mind and will not change because they are test classes/methods and their purpose is to use these methods to check if they work correctly.