Part 1- Removing hard-coded numbers/magic numbers and using symbolic constants to add meaning

1. Replaced the number 0.9 with the private variable initialHunger
2. Replaced the number 1.0 with the private variable initialSize
3. Replaced magic number 0.2 with private variable starvingBorder
4. Replaced magic number 0.7 with private variable hungryBorder
5. Replaced magic number 7.0 with private variable plantFishBorder
6. Replaced magic number 5.0 with private variable bigFishBorder
7. Ran tests, all passed

Part 2 – Replace Conditional Calculations with Strategy

1. Created abstract class FishStrategy with multiple subclasses SHungryFishStrategy, LHungryFishStrategy, LStarveFishStrategy, SStarveFishStrategy, and fullFishStrategy based on their conditionals in move.
2. Delegated each strategy class their calculations according to the calculations within the conditionals.
3. Removed all conditions in move() method and placed it in updateMoveStrategy()
4. Updated conditionals so that it is easier on the eyes (removed doubly nested if statements)
5. Created private FishStrategy object and instantiated it with its respective strategy object within the conditionals.
6. Simplified move so that the FishStrategy object only has to call the move method (fishStrategy.move(this, pond, x, y))
7. Placed updateMoveStrategy in constructor and at the end of the age method so that we know which algorithm to perform every time a Fish is instantiated or its age called.
8. Ran tests, all passed.

Part 3 – Replace Hard-coded Notifications with Observer (Goal is to decouple FishReport and Fish since new classes may be added)

1. Fish is the Subject in this scenario and FishReport the Observer since FishReport is getting the attributes (hunger, size, etc.) from Fish.
2. Created Observer interface with a method defined – update(double[] params)
3. Created Subject class

- Created protected list of observers where observers will be stored
- Created addObserver method to add new observer types (like FishReport)
- notifyObs method for updating all the observers

4. Had Fish class extend from Subject class
5. addObserver(report) in constructor
6. Added updateVars method to get values for params[], which consists of hunger, size, x, y. The method also will call notifyObs with the params
7. Every time one of the values stated above is modified (constructor, age method, and all the swim methods), updateVars will be called so that the observers are notified with the new parameters
8. Had FishReport implement the Observer interface
9. Created update method in FishReport to call all the other update methods in FishReport with the new parameters