Assignment 3: Code Review Report
Group 19 - Cole Thacker, Ellie Neufeld

The first code smell we encountered in our code review was that our GameLoop class was doing far too much and suffered from the Big Class code smell. We identified collision checking as a potential candidate for decomposing into a separate class. Throughout our inspection of GameLoop we also noticed that several of the switch cases the function nextGameScreen could be merged, to reduce duplicate switch cases, which was a very easy fix to implement via Intellij IDE tooling.

When we created and extracted the collision logic into the Collision Check class, we also found that there were several switch cases that had duplicates, and consolidated those cases again, in a similar manner to the nextGameScreen function in GameLoop.Though we only intended to decompose collision checking into a separate class to handle the Big Class code smell, we found that the Duplicate Code code smell also reared its head on closer inspection, and was easily resolved. Therefore, we found and dealt with both the Duplicate Code, and Big Class code smell. Changes in Commit f3047ba1, message is "refactored collision functions from GameLoop into separate class. Refactoring code smell big class".

Again GameLoop suffered from the Big Class code smell, where too many variables were contained in the class. Moving those variables to the GameState class made sense, as those variables don't need to change but still have to be accessible to GameLoop class through getters in GameState class to prevent the accidental setting of values. Commit 48d89726, message is "moved data fields from game loop to game state, refactoring, big class code smell".

The GameLoop class also suffered from the Dead Code code smell, as the Fry Cook always tracks the player. Removing the check to see if fry cooks tracking players made sense since the bounds of the condition were the entire map, forcing the fry cooks to always track the player. Since the condition is always true, moving the frycooks in random directions never gets executed, resulting in dead code. Commit is db747903, message is "eliminated dead code of moving individual frycooks randomly, frycooks only have tracking movement, refactor code smell dead code".

The next smell we identified was high coupling between the MapReader and Board classes. MapReader's method getMap was highly dependent on the implementation Board, since it took in a Board as its parameter and changed one of Board's data members (a BoardTile array). Our solution to decouple these classes was to change the method getMap into a static method that only took in the game's difficulty level as a parameter and that returned a BoardTile array. We also had to change the way getMap was called in Board's constructor. These changes can be seen in commit 7119c4c9 (refactored MapReader by changing it to a static method).

Another bad code smell we noticed was the high coupling between the GameLoop class and the creation of Entities. To fix this, we created a class called EntityFactory with static

methods for creating all Entities. All the code used to create the DeepFryers and FryCooks was moved into this new class and out of GameLoop, where it didn't make sense for it to be. These changes are in commit 8e0c1410 (refactored by creating EntityCreator).

We identified the bad smell of dead code again in the FryCook class. Since FryCooks are always tracking the player, we removed the data member trackingPlayer and method setTrackingPlayer from the FryCook class. This can be seen in commit b8c76e9f (refactored by removing dead code).