Assignment 3: Code Review Report

Introduction:
Initially in this assignment, we had discussions where we broke down our game code into chunks that were implemented by us on the majority scale. Each of us looked over the sections that were developed by the other to recognize any bad smells within our code. We used the guidelines provided in the Assignment Manual to help us recognize the bad smells. When overviewing the entire code, we had found the code to follow high cohesion, and low coupling for the majority of our classes and methods. On the contrary, we found the code to include several sections of poorly structured code, and significant lack of documentation. The details of several other bad smells including the ones mentioned above are discussed below.

Detected Bad Smells and their Refactoring:
A very common bad smell within our code was poor documentation and code structure. This included methods and classes that were significantly long with minimal documentation on the process. Specifically the checkCollision class included one large method to perform all the collision detection. Upon further observation, it was notable that the method was unnecessary long due to duplicate/similar code in several conditions of the if statements. With a slight change in some variable placements, this could be refactored by adding a helper function to handle duplicate sections of the code. This helper function was then called inside each if statement. The detailed refactored code can be seen in commit `b295f2b5` which was applied over the checkCollision class within the checkTile method. The helper function added was named setTileCollisions.

Similar bad smell was detected within the inputKey class inside the keyPressed() method which handled all the cases and scenarios for player inputs. To account for all input cases in different screens (Main Screen, Tutorial Screen, Settings Screen) all solutions were coupled into one large method. This led to multiple sections of code duplication, and in some cases dead code. As per commit `00edd632`, we implemented additional performActionWithEnter() methods with different input parameters to handle player inputs in each screen of the game. This was then called within the if statements which made the code more readable and less clustered. This commit also solved the lack of documentation issues inside the inputKey class. We added additional comments regarding each scenario and their functionalities.

We found the GameFrame class to have very low coupling, as this class was used as an interface to integrate all other major classes. This resulted in the class being significantly long and not legible. After further observation we found the class could use helper classes to handle some functionalities implemented in GameFrame. Inside the same commit, `00edd632` along with `9541a6df`, we added a Command class which worked with returning and setting the game settings attributes such as the chosen level and its relative Game objects. Additionally, we added the EndScreenCommand and ChangeLevelCommand classes which extend the Command class. The EndSceenCommand class handles functionalities of the screen the player sees once they finish the level. The ChangeLevelCommand class implements functionalities within the screen which the player sees when changing the game level. After adding these classes, we

were able to integrate them inside the GameFrame class and eliminate clustered code. This made the GameFrame class manageable in any cases of debugging in the future.

The GameFrame class also included a constructor which was significantly long due to many declarations of image import. We found this to be poorly structured and an use of unsafe constructs. We were able to easily solve this by implementing methods which handled the image imports. This was then called within the constructor to ensure observance of the same features and GUI within the game. The detailed refactored code can be seen inside commit `f9b71883`.