Marco & Satvik Assignment 3 Report

Code Smell	Where it is found
Lazy Class	The commit for this change can be found at **e67de888**. This refactoring consists of the removal of a lazy class, the Tile.java class. This change also invoked changes which were made inside the TileManager.java class, as well as a small change in GamePanel, and changes in the test class for TileManager, TileManagerTest.java.
Data Clump	The commit for this change can be found at **79b40a75**. This refactoring consists of creating a Point class and replacing many x and y variables throughout the classes to Point object. Additionally, since this change was something that affected the majority of the code, the changes can be found in multiple classes.
Unused Variable/Imports	The commit for this change can be found at **63e28c5e**. Once we finished the Phase 3 of the project, we realized that while building several of our classes, we were sloppy with our code and left many unused variables and imports in the classes and tests. Since the unused variables and imports were scattered throughout the code, the changes were made in many places. But we focused on the classes made by our group, such as Entity, Bee and TileManager classes.
Duplicate Code	The commit for this refactoring can be found at **97fac99a**. In the UI.java class, we noticed a lot of duplicate code being used to draw the text buttons that are all similar and used in the game's Title screen, game over screen, and game won screen. The changes made to refactor this duplicate code took place solely within the UI.java class.
Long Methods	The commit for this refactoring can be found at **3faffc20**. Whilst looking through the relative code for long methods, a particular method that seemed unnecessarily long was the keyTyped method in the Keyhandler class. In this method, the switching of states was repeated for multiple game states.
Primitive Obsession	The commit for this refactoring can be found at **991f6fb3**. We found that throughout the project, we had made use of Strings to represent the 4 directions a player can move, and this is a case of primitive obsession. The changes for this refactoring introduced the Direction class, changes in the Entity, Bee, and Enemy classes, and changes in those classes' test classes.

Code Smell	How it was fixed and why
Lazy Class	The Tile.java class was a lazy class because it was not doing anything other than containing a single field, which was a BufferedImage (a subclass of the Image class) being used to hold the image of a tile which is set in the TileManager.java class. We fixed this by deleting the Tile class, and replacing the list of Tile's in TileManager with a list of BufferedImages. Each object in the list of BufferedImages has its Image set in the TileManager class to represent the different game tiles, instead of instantiating a Tile object and setting its image for each type of game tile.
Data Clump	As we looked through the code for things to refactor, one thing popped out more than others. The fact that we had lots of variables where we had, "worldX" and "worldY". This of course is a bad practice as we are creating unnecessary variables which could be handled by one object, a perfect example of Data Clump code smell. So to fix this issue, we decided to implement a Point class found in the util package. This class consisted of getters and setters, which were then used throughout the code.
Unused Variable/Imports	Leaving unused variables and code unattended can lead to issues in the future, as looking for bugs can take longer since you would be looking through more code to reach the issue and prevent developers from wondering what the variable is for in the later stages. We fixed this issue by going through the relevant classes, and removing any unused imports, and checking each class/function to see if there is any dead code.
Duplicate Code	In UI.java's draw___Screen() methods used to draw the different game screens, there was a code structure that we saw being used in each method multiple times to draw the text buttons that are part of the UI. We extracted this code and made the drawTextButton() method, which was then called in all places where this code structure previously existed. Unifying repeated code structures allows for better maintainability in the code.
Long Methods	This long method, keyTyped, found in the KeyHandler class had code that we repeated multiple times to switch the game play states. So rather than repeating code for each state, we decided to create another method, which took inputs from the keyTyped function and changed the states in general. This function is called updateCommandNum. This helped reduce the possibility of errors via copied code and also allowed for futuring refactoring in case the commandNums values changed.
Primitive Obsession	Throughout all of the classes in the entity package, the use of Strings to represent the 4 directions a player can move was leaving a great amount of room for errors caused by mistyping these strings, and affecting the overall understandability of the code. To refactor this, we made a Direction enumeration class, representing the directions up, down, left, and right. In both entity methods and entity test methods, we then replaced all instances of String directions to make use of the Direction class. We also refactored the Bee class' draw() method to use a switch statement on the new Direction enumeration.