Assignment4Report

Kiran Khubbar & Amar Mondair

Kiran Khubbar (Player.c, NPC.c)

The first smell identified in the Player class was that the setItems method was a dead method. Since this function was never used anywhere in the code, it can lead to unnecessary confusion and extra complexity. To fix this, the function was completely removed, allowing for someone to better understand the class without having to look for the dead method.

The next smell was that the update method had multiple responsibilities and was way too long. Having the update method handle movement, collision detection, input handling, as well as interactions resulted in harder readability, as well as requiring more time to comprehend and maintain the code. The solution was to then break this update method down into several, smaller methods that each handle one responsibility. This results in the code being easier to read and update for future use.

Another smell identified was that variables ended up being hard coded in multiple places. For example, playerSpeed was hard coded numerous times throughout the class. This results in a difficult time having to modify the code later on. To fix this, these variables were instead changed to constants, resulting in improved readability and more safety when it comes to adjusting values.

In addition to the previous smells, another smell found was that the Player class contained several unused import statements. These import statements result in more clutter and result in a more difficult time understanding the code. All of these unused statements were removed, allowing for cleaner code and only necessary imports.

Throughout this entire class, multiple spelling errors were found in variable and method names. This can lead to some confusion as well as possibly distracting someone when reading the code. To fix this smell, spelling was corrected to improve code readability.

Another smell found was that comments left on the code were just repeating information that the code already expresses. This results in a lot of redundancy, which can reduce someone's efficiency when understanding the class. To counteract this, the comments were rewritten to provide more context instead of repeating information. This allows someone to gain a better understanding of how the code functions and its purpose.

A smell found in the NPC class relates to how the NPC dialogue was set up. The NPC dialogue was stored as one long string inside the class. This results in a method that is hard to edit as well as the method mixing code and text content used for the dialogue. To fix this, the text used was moved to a separate text file (npcText.txt), while the method is used just to read from the text file. This improves flexibility, as well as allowing easier changes to be made to dialogue.

Additionally, the NPC class contained a lot of commented out code which only results in harder to read code as well as more confusion. These comments were then removed from the file, allowing for easier readability.

Amar Mondair (EventHandler.c, Entity.c, TileManager.c)

The first smell that was identified was that EventHandler imported com.group.entity.Player at the beginning. However, this import is never actually used once in the file. The result is an unused import that can sometimes lead to confusion and clutters the file for no reason.

The next smell identified was the vague method naming used for checking if the player collided with an event tile. The method was previously named hit() which used a variable with the same name. This was very vague and could cause a lot of confusion as to what hit meant exactly. Therefore, the names were changed to isEventTriggered() and triggerEvent to help document the code usage.

Another naming related smell was the variable name getPreviousEventY. To some developers, this can often be confused with a function name as some functions are named like this. To reduce confusion, the variable was named to previousEventY, allowing for more clarity and following naming conventions.

The biggest smell that was identified for EventHandler was the checkEvent() method. This method had multiple responsibilities, having to check events related to both fire and bomb punishments as well as handling the player's ability to switch levels. This results in the violation of the single responsibility principle, resulting in the method being more difficult to maintain. The solution was to break the checkEvent() function into multiple smaller methods which include canTriggerEvent(), handleFireEvents(), handleBombEvents(), and handleTeleportEvents(). These extra methods broke the multiple responsibilities into their own separate methods, allowing for someone to easily update the functionality of these methods.

In addition to splitting up the checkEvent() previously mentioned, the actual code involved a lot of nested if statements as well as hard coded numbers for the map number, map row, and map column. This smell was fixed by introducing a loop as well as using a 2D array to store the event tiles X and Y position.

Another smell identified was that the variables tempScreenX and tempScreenY were not used in the final version of the code. This can cause confusion for someone who does not know the purpose of these variables. Hence, these variables were removed, allowing for easier understanding.

TileManager contained unused code related to updating tiles that correspond with the enemy pathfinding algorithm. This can result in a lot of confusion as there is already code that handles drawing the enemy pathfinding tiles. The code was removed as it did not contribute to the overall class.

TileManager uses a method referred to getTileImage() that stores the sprite image and collision of each tile used. However, this function relied on calling the function setUp() for each tile that was used. This resulted in modifying tiles tedious. To change this, a loop was used to iterate and initialize a 2D array containing all the information needed for the tiles including name, sprite name, and collision.