Code Smell 1: Unnecessary If Statement
Commit ID: 96374e4

Problem: The program checks the same condition twice. Once in the outer if statement and once in the nested if statement, the second if statement is redundant.

Fix: Removed the second if statement to stop the program from doing extra work

Code Smell 2: Poorly Structured Code
Commit ID: 7a4eb1f

Problem: Unnecessary use of a for loop and a list to get the Door object. This was considered unnecessary as there will only be 1 door in the game, so no point in using a list to store a single door. This also means we don't need a for loop to get the door.

Fix: Replaced the previous method "getAllDoors()" with getDoor(). The new getDoor() method traverses the original StaticEntities list to find and store the door in a variable so we can directly return the door object. This change also prompted modifications to our collision checker and playing class as those initially used the list and traversed the list with a for loop to find the door.

Code Smell 3: Long List of Parameters
Commit ID: 80bda99

Problem: The checkPlayerDoorCollision method used a door object as a parameter. This was unnecessary since there is only 1 door, and we can already get the door object using the static getter method.

Fix: Removed the extra parameter and used the static getter method to get the door from the StaticEntity class. This also resulted in us having to update all the other classes and test cases that used the original method.

Code Smell 4: Code Duplication and Dead Code
Commit ID: 5404a4d and 5083fa4

Problem: We realized that the method getBoundingBox() from static entities was never used, so it was dead code. We also realized we were manually creating the rectangles used for collision detection instead of using the getBoundingBox() method.

Fix: Initially, we intended to make all the subclasses of static entity implement the method so we can use it in the collision checker class. However, we soon noticed that both rewards and traps share the same hitboxes, so we left that as the default implementation in the static entity class to avoid code duplication. We then implemented the method in the door class. Since the door has its own unique hitbox, we have to override the default implementation. To ensure these methods were used, we replaced the original code that manually created the hitboxes in the collision checker class with the new methods. This makes things less repetitive and more understandable while addressing dead code.

Code Smell 5: long list of method parameters
Commit ID: bb32d0e

Problem: The method playmusic() and playsound() had been using parameters: filepath and volume; which was unnecessary and made calling the function complicated as you needed to know the location of different sound files you wanted to use.

Fix: For MusicManager and SoundManager, I got rid of parameters and made them into attributes in the class. Since soundManager has multiple sounds to use, I made a separate method for each type of sound to be used; examples are playWinSound(), playTrapSound(), playCollectSound().

Code Smell 6: Unused variable/file
Commit ID: 754a964

Problem: We were planning to have a java file with enum's for selecting the difficulty of our game. We later never implemented a difficulty system so it had been used/useless.

Fix: Deleted Difficulty.java and informed the group about it.

Code Smell 7: lack of documentation
Commit ID: 0a9eac8

Problem: The camera class of our game had been undocumented, so understanding what each method was used for was time wasting. There are now comments.

Fix: Added JavaDoc comments to all methods in the camera.java file.

Code Smell 8: Hardcoded values/Magic numbers
Commit ID: 72bc9fd

Problem: We had been hardcoding number literals, meaning not using variables, to assign values to different things, such as the number of spawn points for the player, max rewards able to be displayed, and the number of static entities present in the game. This made changing these values unnecessarily hard.

Fix: Made three private static final variables called: SPAWN_POINT_COUNT, MAX_REWARDS_DISPLAYED, STATIC_ENTITY_COUNT. Assigned them a value that made sense for our game.