CMPT 276 Assignment 3
Patrick Shaw | 301537454
Chris Pinca | 301396939

Scoreboard & FoodTruck:

Long Constructor
The Scoreboard class had a long constructor method that was doing too many things and contained some hard-coded content, making it difficult to maintain. To improve this, we added helper methods for each panel, such as createLeftPanel(), createCenterPanel(), and createRightPanel(), which simplify the maintenance of the scoreboard in the future. These methods allow for changes to be made to the scoreboard content in the corresponding createPanel() method, ensuring that the constructor only creates the scoreboard's structure while the content is supplied by the helper methods. We also added helper methods for updating each component of the scoreboard individually. These changes will make further modifications to the scoreboard's design easier.

Duplicate Code
Meanwhile, the FoodTruck class was too long and contained duplicate code. To simplify it, we removed the attributes score, ingredientsDiscovered, recipesDiscovered, damage, and fines, which were already in the Scoreboard class. This change also eliminated all the getters and setters for these attributes in FoodTruck, greatly reducing the amount of code. This makes the FoodTruck class shorter, simpler, and easier to read.

Tight Coupling
We noticed a tight coupling between the FoodTruck and Scoreboard classes, making them overly dependent on each other. To reduce this coupling and make the classes more independent, we updated the FoodTruck class to take a Scoreboard object as a parameter, rather than the other way around. We also removed the update() method from Scoreboard, which previously called getters on the foodTruck object to determine data for the scoreboard. Instead, the foodTruck now passes updates to the scoreboard data as they arise during gameplay. This greatly reduced the coupling between the two classes, making them easier to read and simpler to deal with less code. This, along with the reduction of duplicate code in both classes, helped to improve the maintainability and flexibility of the project.

Variable Names & Types
Finally, we made improvements in naming and type declaration. We made the fonts and colors used in the scoreboard constants with more descriptive names, making the code easier to read and understand.

Screen Class:

createButtons() Method Added to Screen Class

We identified that the previous Screen class had a blob of code that essentially built the entire screen. Recognizing that this is bad practice, the code responsible for the creation of the buttons was moved into its own method. This refactoring cleaned up the code nicely, made it more readable, understandable, testable, easier to maintain, and less likely to break. This method now has a particular function used by the Screen class to create buttons.

createBackground() Method Added to Screen Class

Similarly to the createButtons() method above, the code responsible for creating the background of the Screen was previously in a big blob of code in the Screen class. After refactoring it to its own method responsible for the creation of the background, it is more readable, understandable, testable, easier to maintain, and less likely to break. This method now has a particular function used by the Screen class to create the background of the screen.

Variable Names Changed

We realized that the variable names were confusing and not representative of the different types of buttons that the Screen class was responsible for making. So we changed the variable names of buttons from "resumeButton" and "exitButton" to "topButton" and "bottomButton" to reflect that the Screen class can make more than just those 2 types of buttons. Also changed the Listener names to "topListener" and "bottomListener" to reflect the new button names. Finally, we also changed the button image path names to button1ImagePath and button2ImagePath to reflect the differing types of buttons and which button they affect. This increases the readability of the code and ensures that the variable names are easily recognizable according to function.

Removed Hardcoding of Dimensions and Image File Paths

Previously to the refactoring, the dimensions and image file paths for the specified type of Screen to be created had to be hard-coded as parameters that were passed to the parent Screen class. This was bad practice as having too many parameters is dangerous, so we created a ImagePaths and Dimensions class that stores the specified values and can be passed as objects for each type of screen created. This is far more understandable, less prone to error and easier to test.

Removed Unnecessary Method

We previously had dead code that wasn't being used in the form of the preferredDimension() method that was supposed to set the dimensions of the Screen to the preferred size. This was not being used at all since we went another route for declaring the size of the screen, so we refactored and subsequently removed it.