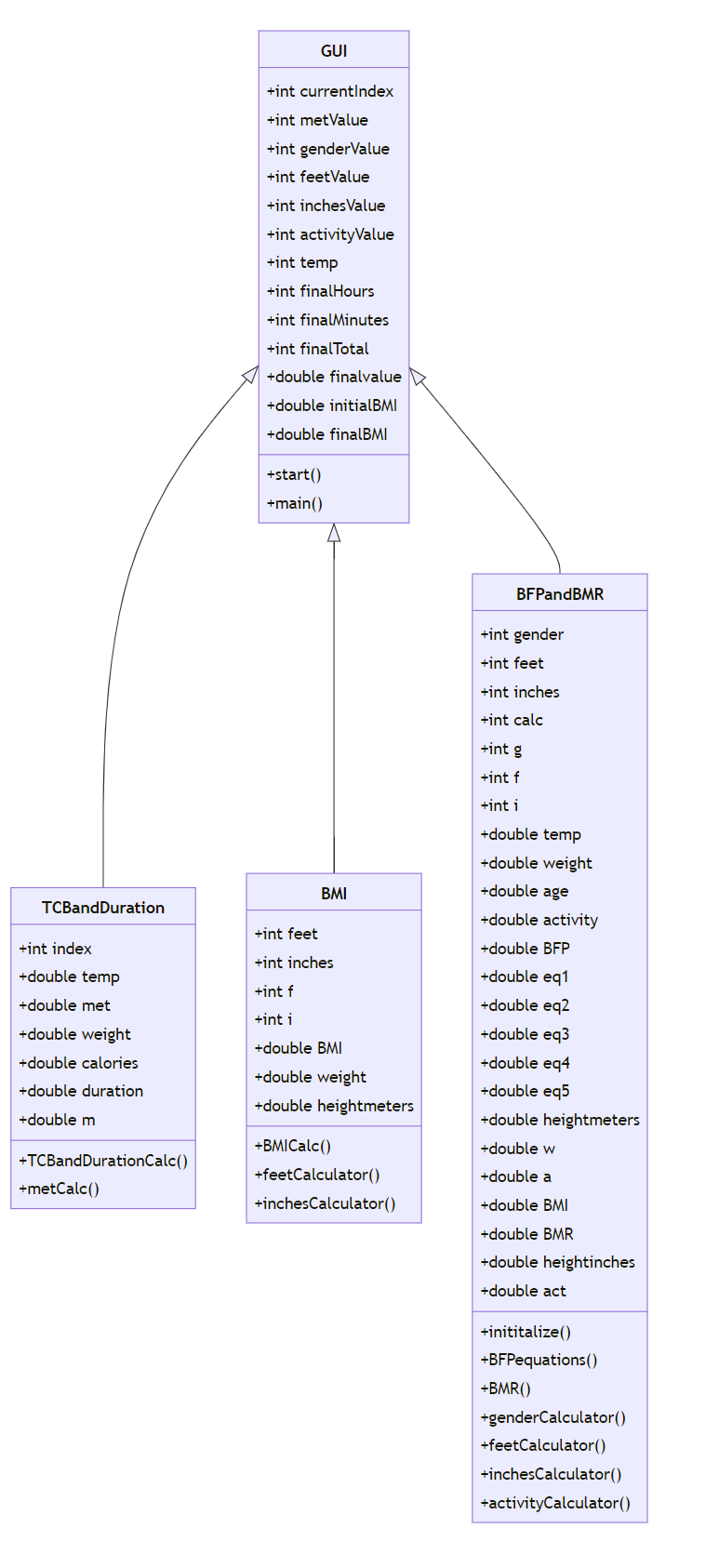
**Fitness Calculator Implementation Manual**

UML Diagram:

Description:

The Fitness Calculator has a simple, but efficient class design. All Graphical User Interface (GUI) capabilities are utilized within the “GUI” class, while all mathematical calculations are done in the subclasses “TCBandDuration”, “BMI”, and “BFPandBMR”. In this format, the “GUI” class is used as the parent class that takes user input values and invokes methods in the subclasses based on what the user decides on calculating. The “GUI” parent class is also tasked with creating the scene of the application and constantly making updates to this scene based on what the user is attempting to calculate. The scene of the “GUI” class is a BoarderPane with different HBoxes at the top and bottom, a GridPane in the center, and another GridPane and more HBoxes that are occasionally added to the right side of the scene, depending on which calculation is being completed.

More specifically, when the user begins the program, they are prompted to select which calculation they would like to compute. The “GUI” class reads whichever query is selected and adds a variety of nodes, Labels, TextFields, and ComboBoxes, according to the requirements of the chosen calculation. After the user inputs all required information, they will click a “Calculate” button, and will be presented with their results. For this step to be completed, the “GUI” parent class calls a method within one of the subclasses with the user input data. This data is then set equal to new variables in the subclass and the specific calculation is computed. Each of these subclasses return the calculated result back to the parent class “GUI”, and the results are then displayed within the scene.

The complexity of this application comes when looking closer at these different classes. A lot of the calculations that can be computed within this calculator use the same variables, but each of them has a completely different output. To account for the differences in results, more code must be written within each of the methods. For example, the Fitness Calculator will ask for a user’s height in a variety of different calculations. Some of these calculations require the value of height to be in inches, while others require meters. Since the home country of the Fitness Calculator uses imperial units, the calculator will also ask users to input all measurements in imperial units, while converting these units to metric, when needed, purely in the backend of the application. Having to convert between metric units, while simultaneously rounding between integer and double values, is just one example of what causes higher complexity within the subclasses of this application.

Even though the subclasses themselves are quite complex, they are nothing compared to the length of the “GUI” parent class. Because this application includes functionality that allows users to repeatedly run the calculations without having to close and reopen the program, the class has a higher complexity regarding the addition and deletion of nodes. The “GUI” class is tasked with not only taking user input and printing out results, but also clearing user input, altering editability of nodes, and deleting nodes when the user prompts it to. By adding these functions into the Fitness Calculator, the complexity of the classes are significantly increased, but so is the usefulness and efficiency of this application.