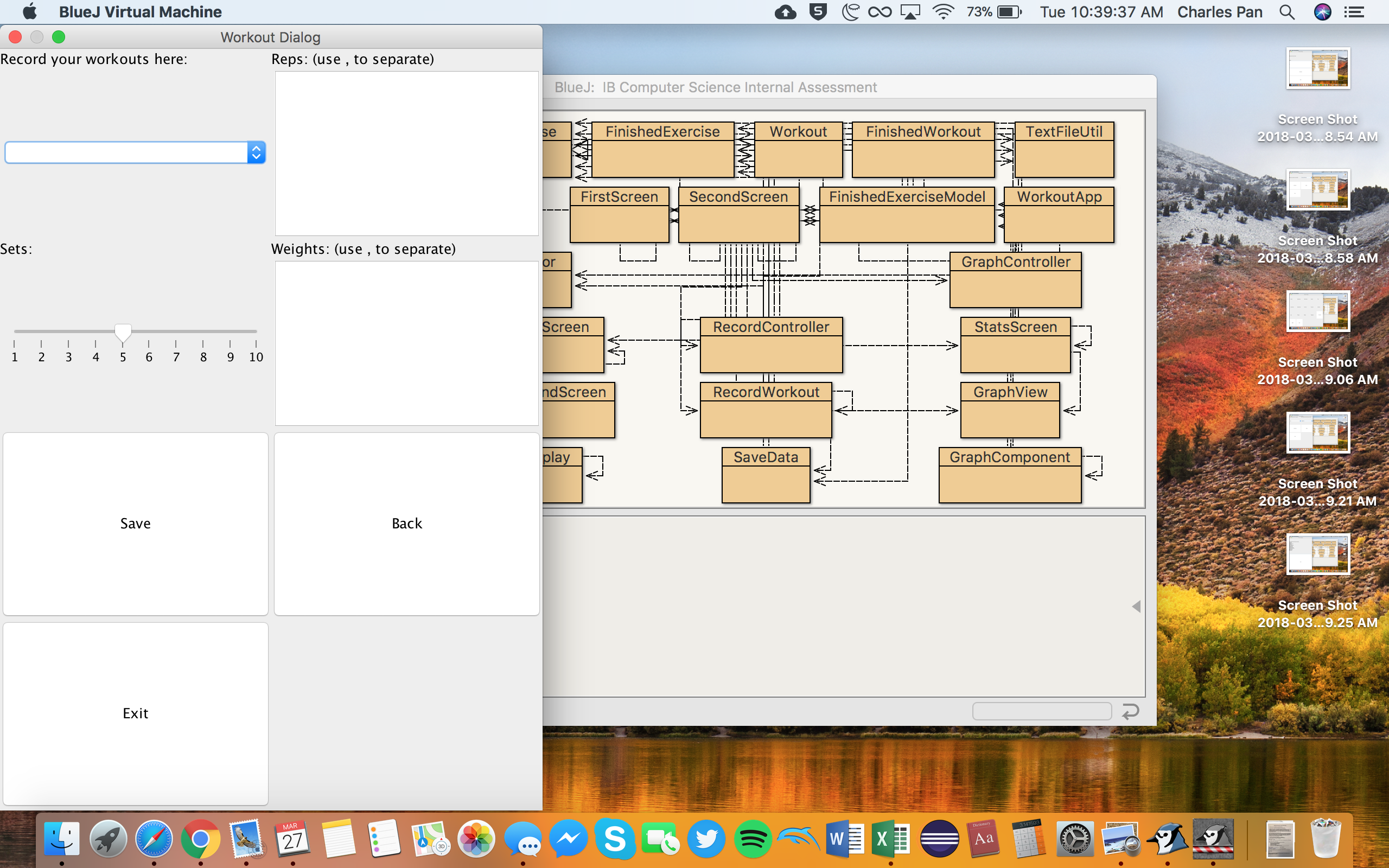
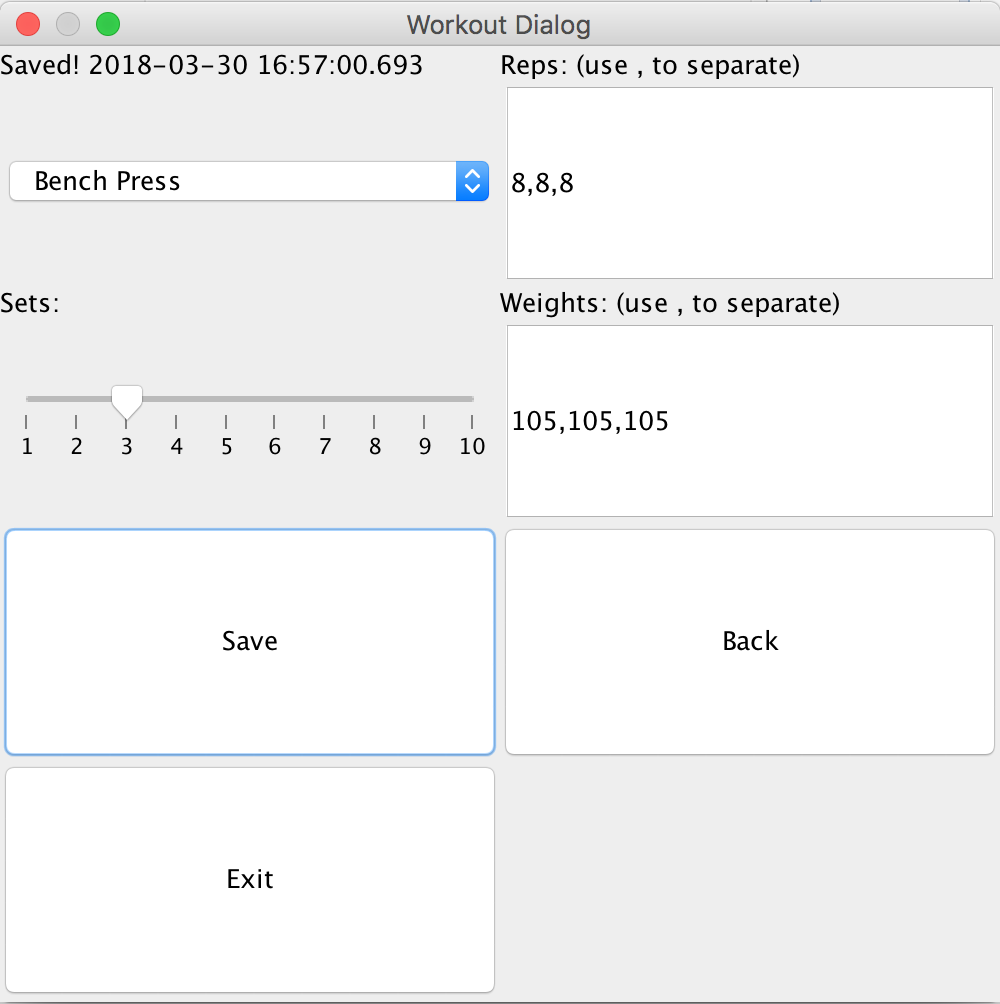
**Techniques Used:**

* MVC Pattern
* Observer/Observable
* Linear Search
* Encapsulation
* CSV and TXT files

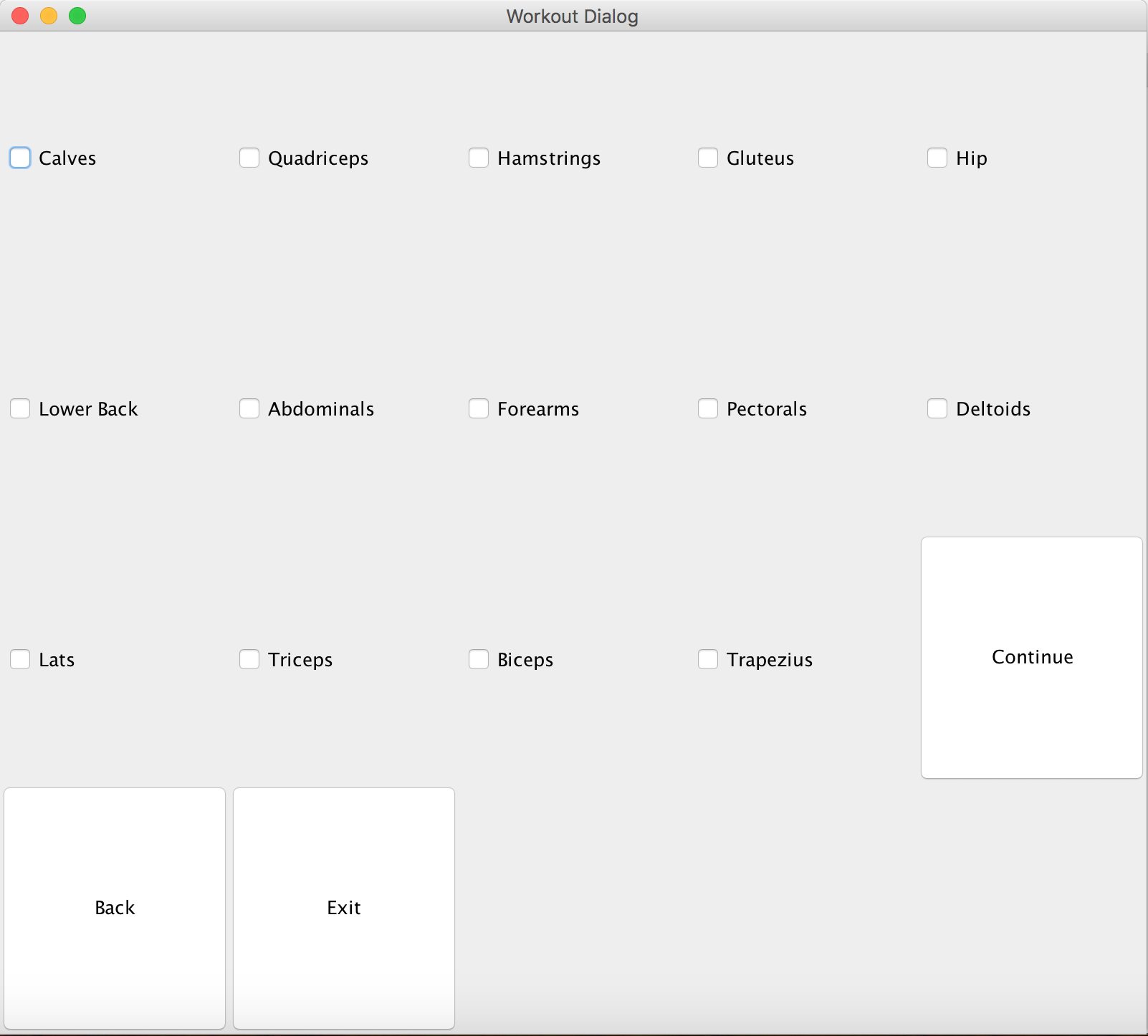
I utilized the model-view-component design pattern in my program, specifically in the record workout feature. In that feature, I have a JFrame that allows the user to input the exercise they performed, the number of sets (JSlider), the number of reps, and the weight used.

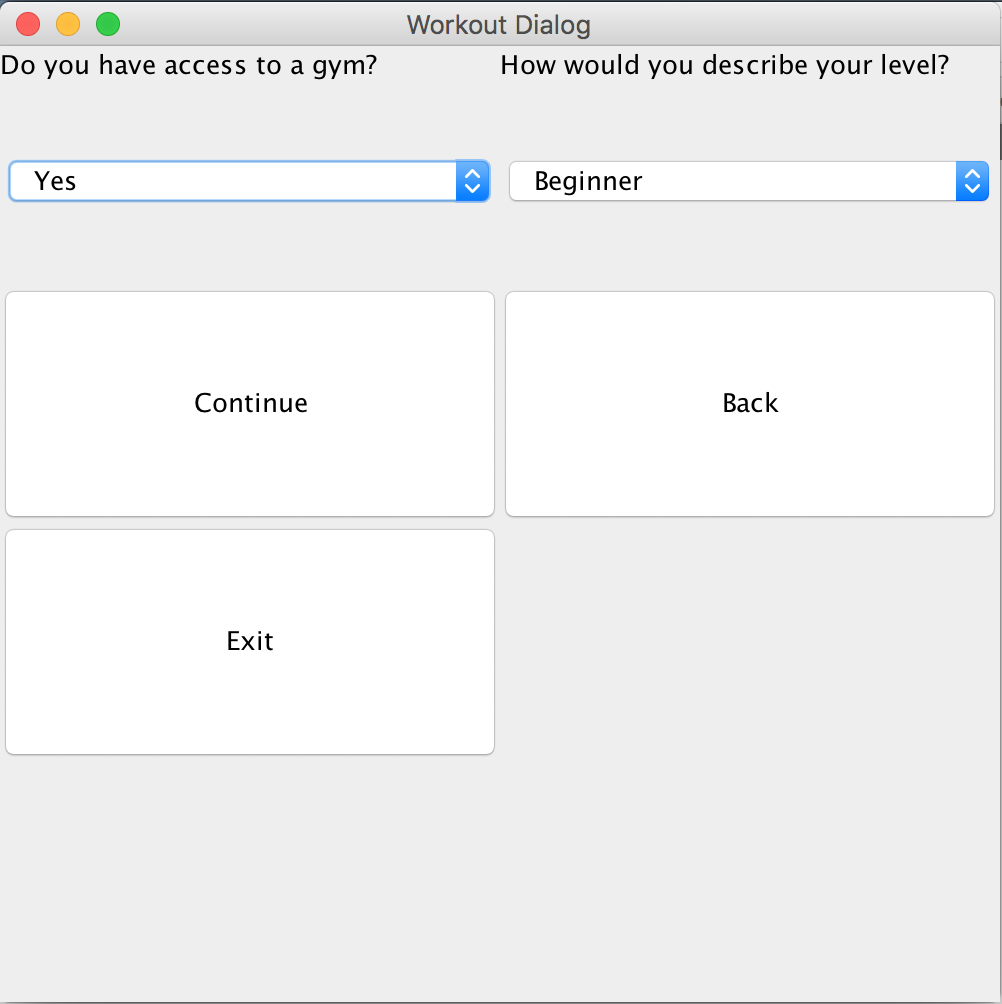
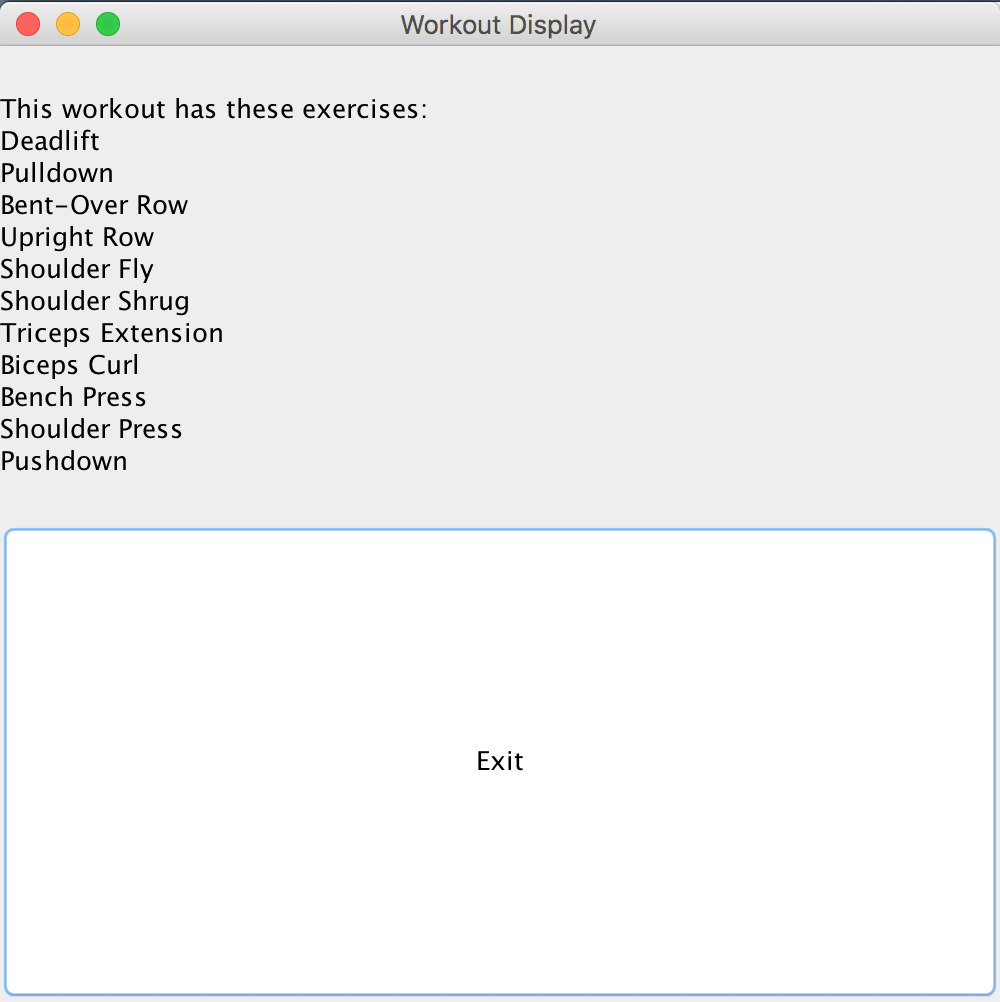


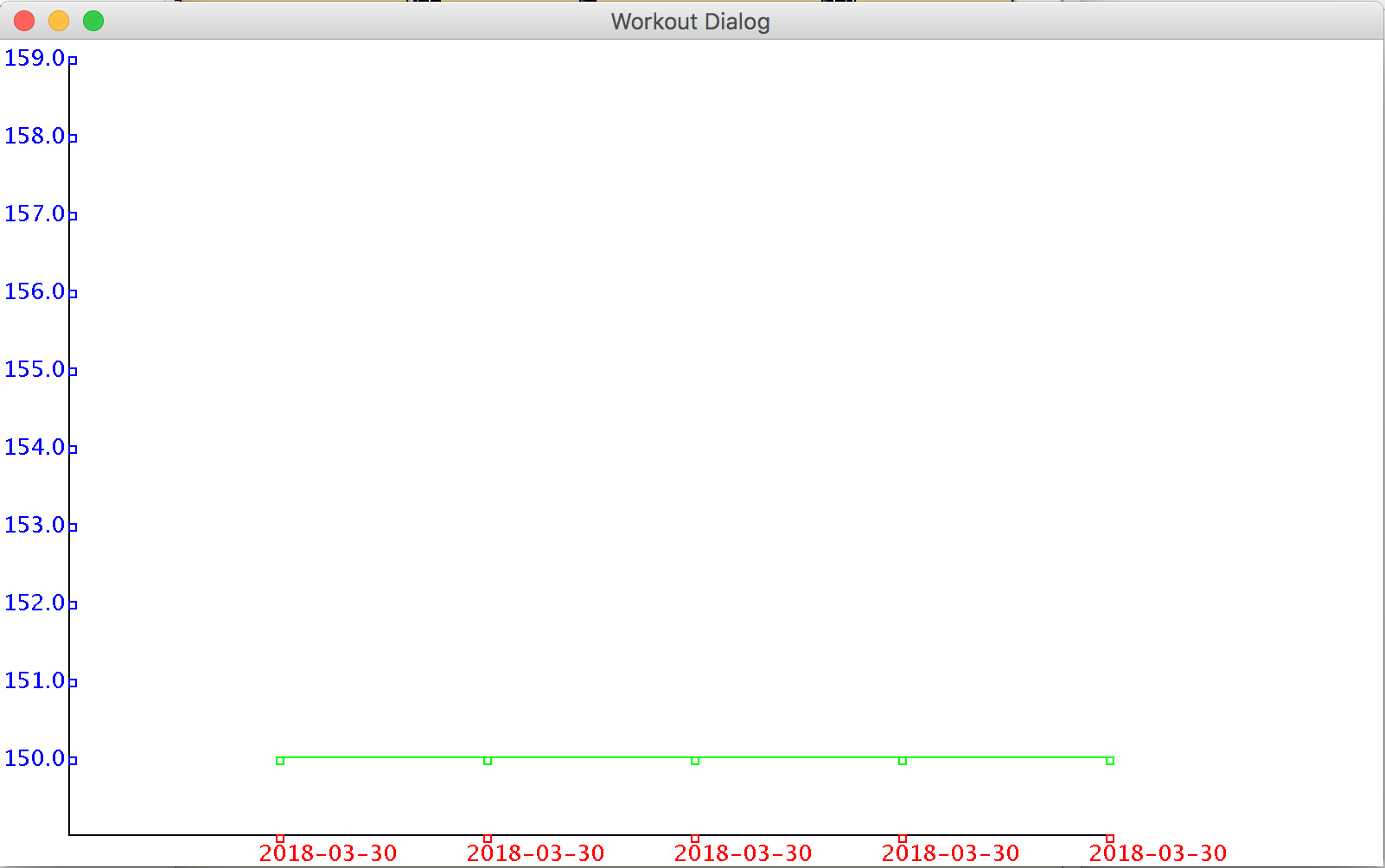
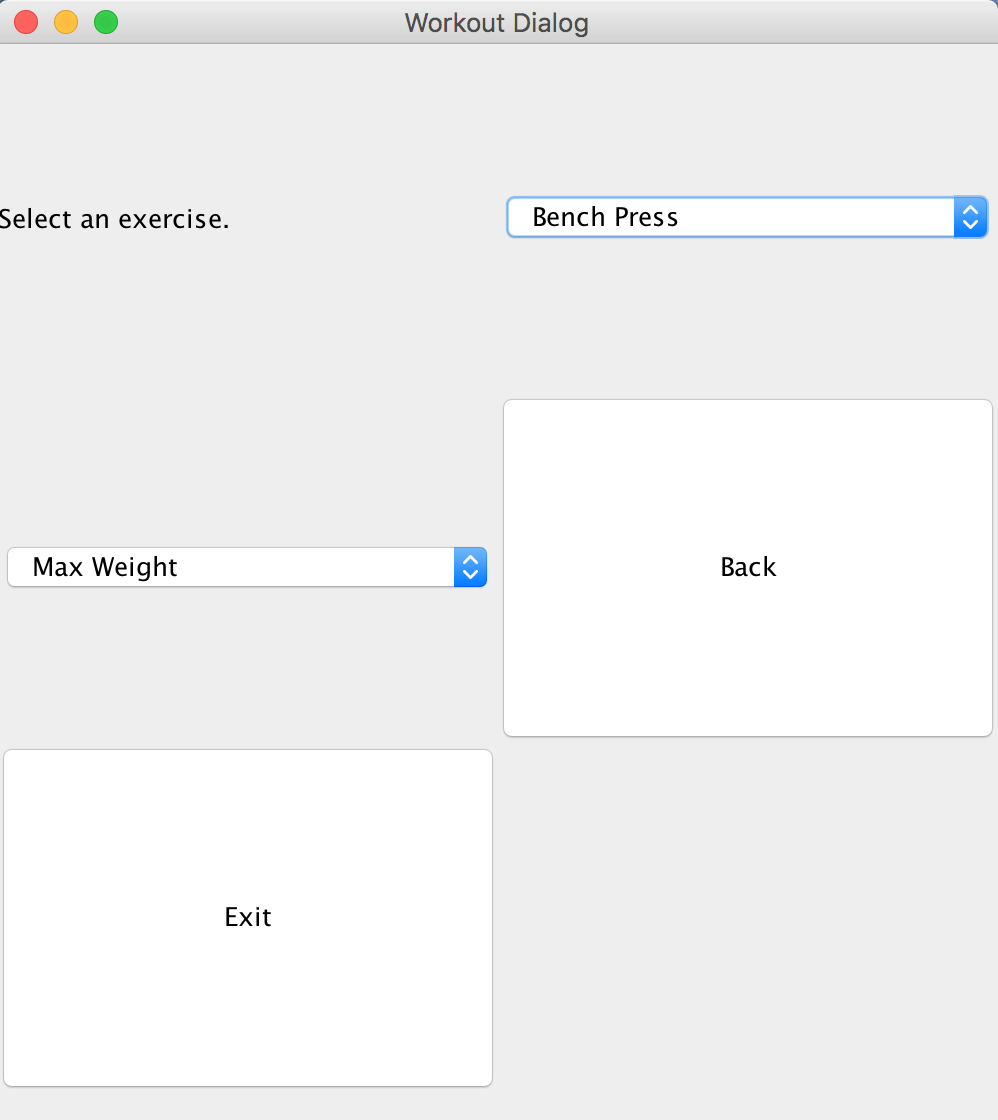
After the user presses save, the view is updated so that the user knows that the data was saved successfully. In this MVC model, the class “RecordController” serves as the controller and communicates with the view “RecordWorkout” and the model “FinishedExerciseModel.” After the model is updated, the controller updates the view with the text “Saved!” and the current timestamp.



In my first feature, generating a workout, the user is first prompted with a screen that asks them to check off muscles that they would like to target in their generated workout. After pressing the continue button, the user continues to the next screen, which allows the user to select whether or not they have access to a gym and their difficulty level in their workout. Finally, after checking the continue button, they are presented with a list of exercises for their workout.





In the third feature, viewing stats, the user is first presented with an option pane that asks which exercise they want to view and which option (max weight or total weight) they want to use. After selecting an option, a graph pops up records from the past five dates. 

In my program, my Exercise, FinishedExercise, Workout, and FinishedWorkout classes were used to keep track of the data and stats used in my program. The Exercise class was the data type used to keep track of the data in my .csv file, the FinishedExercise class was the data type used to write new data to a .TXT file.

The Workout class was the data type used to generate a list of exercises based on the user’s preferences. Generating this list involved linear search, as the exercises had to be filtered by which muscles they exercised, the difficulty level, and whether the user needed access to a gym or not. The FinishedWorkout class is essentially a list of FinishedExercise objects; although it has not been implemented in the final version program, if the program were to be expanded to view the stats of a complete day’s workout, the FinishedWorkout class could make that implementation very feasible.

In the MVC design pattern used in the program, the user interacts with the RecordWorkout view, and after they input their exercise, sets, reps, and weights, the information is passed from the view, through the RecordController controller to the FinishedExerciseModel model, where the data is written to a .TXT file. Since this set of MVC classes also implements Observer/Observable pattern, the controller then updates the view after the exercise is written to the .TXT file.

The implementation of generating a graph follows a similar design pattern to model-controller-view, but not exactly. The StatsScreen class serves as the view and allows the user to select which exercise they want to view and what y-axis option they want. This information is then passed to the GraphController class, which serves as the controller, which relays the information along to the GraphComponent class, which is the main class in charge of generating the graph. The GraphComponent class also utilizes information from the FinishedExerciseModel class.

In my program, I also utilized the TextFileUtil class provided by my teacher, Mr. Alford, which allowed me to read and write data to a .TXT file. However, I had to modify the class to include a new method that would allow me to write data to a .TXT file by appending new data to a new line, not just adding onto an existing line.

Word Count: 636