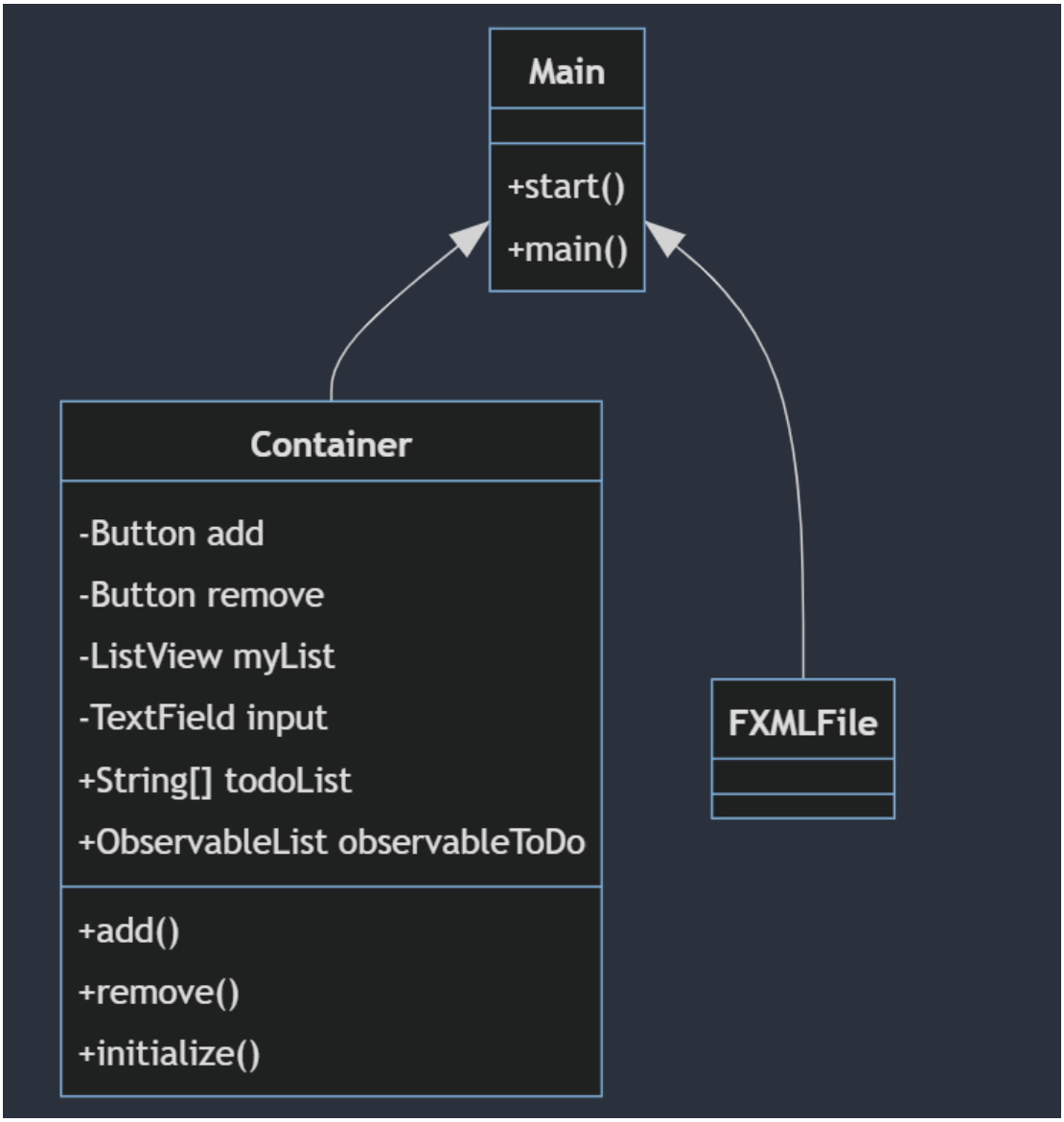
Java To-Do List Implementation Manual

By Logan Pizzurro



The majority of the code is held in the container class. This is what gives the buttons and ListView their functionality. The Main class is mainly used to setup the JavaFX stage using the FXML file created in SceneBuilder. It also sets the title of the window, as well as the dimensions.

Within the Container class, the most important methods in the class are initialize() and add(). While all of the methods work with the ObservableList object that was created, these two are arguably the most important.

The main problem with using an array for the ListView object was the list was never able to be updated. The ObservableList object is an object built into JavaFX that allows the user to dynamically update the list. This is much more efficient than using the String[] method of storing the to-do list's task.

```
public void add() {
    System.out.println(x: "Add"); // Debugging
    item = String.valueOf(input.getText()); // Gets the text from the input field
    observableToDo.add(item); // Adds the item to the ListView
    myList.setItems(observableToDo); // Updates the ListView
    input.clear();          // clear the input field
```

```
@Override
public void initialize(URL arg0, ResourceBundle arg1) {

    /* readFromFile() function currently not working.  Not in use.
     *
     * try {
     * readFromFile();
     * } catch (FileNotFoundException e) {
     *
     * e.printStackTrace();
     * }
     */

    myList.getItems().addAll(todoList);  // Adds the default list to the ListView
    myList.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>() {
        @Override
        public void changed(javafx.beans.value.ObservableValue<? extends String> observable, String oldValue,
                String newValue) { // This method is called whenever the user selects an item from the ListView
            System.out.println("Selected item: " + newValue);
            currentSelection = newValue;  // Sets the current selection to the item that was clicked
        }
    });
}
```

The initialize() method also keeps track of what is currently being selected by the user. This is used in the remove() method.

```
public void remove() {
    System.out.println(x: "remove");  // Debugging
    observableToDo.remove(currentSelection); // Removes the item from the ObservableList
    myList.setItems(observableToDo); // Updates the ListView
}
```