0.1 Graphical User Interface

The Graphical User Interface will be achieved using one the many toolkits within Java. Specifically I will be using JavaFX. JavaFX is unique in the fact that the GUI can be generated using two approaches:

1. Within the code itself, we can define how the GUI is comprised. This is done by instantiating Objects of specific component Classes, such Panes, Labels, Buttons etc. and using their methods to join them together. The advantage of this approach is that since these are simply classes, we can create child classes from them allowing us to create custom components that we can control.

2. The UI is defined using *xml* files (aptly named FXML files). A controller class is defined alongside it. This controller class is referenced within the the FXML file and when the FXML file is loaded, the controller class is initialized along with it. This controller allows us to "control" the UI components. While this approach is very simple to implement and the design of the UI looks clean, it can be somewhat difficult to allow the controller classes to interact with other controller classes.

The real strength of JavaFX however comes from the fact that these two approaches can be interwined as needed, making solutions unique and versatile.

The GUI can be decomposed into smaller parts as shown by the hierachical diagram shown below. The parts in **bold** are the parts to be implemented and designed in prototype 1 and the rest will be completed in prototype 2.

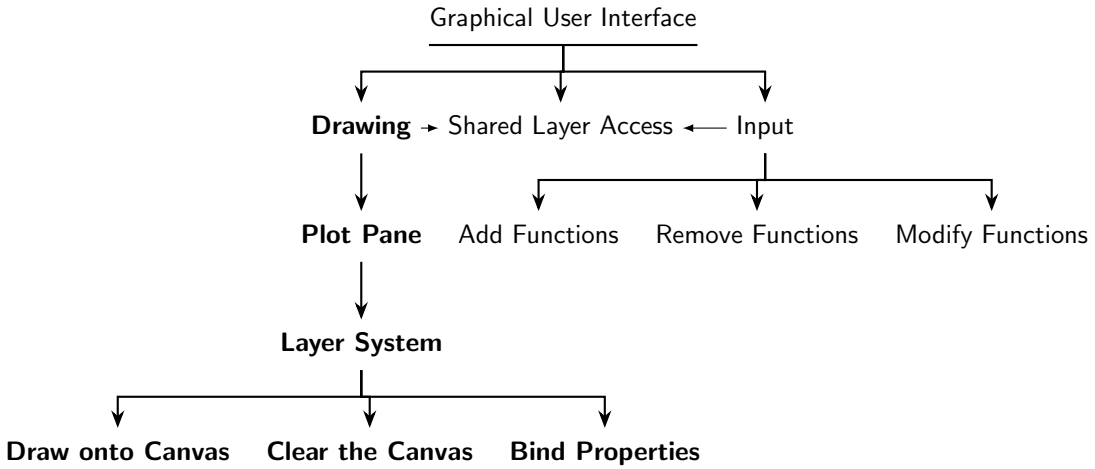

Figure 1: Graphical User Interface Hierarchical Diagram

Many of the features above will be discussed in greater detail later, however the Shared Layer Access and Properties will be touched upon now.

As mentioned before it can be hard to make two controller classes interact, as such a solution is an object that both controller classes can reference. This class will contain attributes and methods that both classes need to interact with each other.

However, how do these classes know when something changes and hence know when to update their corresponding UI elements. The answer is through Properties. Properties can be bound to each other, updating when one changes. Objects called listeners can also be added to properties. Listeners can be programmed to do a specific action when a Property is updated. This action could be a function, or updating a variable. Using properties allows for a reponsive UI, updating whenver the user does an action.