***UML Diagram***

As seen from the package level UML diagram, we never create any Model level classes within our View. Any information that the View package needs is done through interfaces and the Controller classes. Looking into the expanded diagram, we see more details about how our system works. As we are using TableView, so we have to use the StockModel interface as a guide. We also need to keep track of a list of FolioModel interfaces as each interface defines a tab in the GUI. These are the reasons for the links between those classes.

The EditStockPopup class also needs access to the interface of StockModel. This is because it needs to pull the more stock details as well as interact with StockModel through buying and selling shares. Although this is again done with decoupling in mind. The interface is passed to it via the ObservableList of stocks created from the ArrayList of stocks in FolioModel.

We can see that our base class of StockModel is only ever created in 1 instance. This is within FolioModel, as it contains an array of stocks. In turn, there is also only 1 create instance of folio. This is done in a controller and passed back to the main for the tab system mentioned earlier. All this means that the View has as much functionality as possible with no need for Model classes. The decoupled nature of the program also means that the Model dependant aspects of View only ever see the interfaces for FolioModel and StockModel, not the inner workings of the classes.