**Appendices J**

**Graphical User Interface**

1. **GridBag Layout Manager**



Fig x – Extract from LoginForm class, extract from layoutComponents()

As you can see from fig x, to implement grid bag layout I had to firstly set the layout of the panel in which my form is going to be within, by invoking setLayout() and passing as a parameter a new GridBagLayout object. I then had to declare and initialise a GridBagConstraints object, which is used to define the layout of any components added to the panel.

Once I have a GridBagConstrains object I then invoke methods from the GridBagConstraints class to set the constraints of any components added. The basic constraints are gridx() and gridy() which defines what position I am going to add a component on the screen, and gridwidth() and gridheight() determines what size the cell will take up on the screen.

1. **Model-View-Controller**



Fig x – Extract from HomeForm class – declaration of ListPanel object

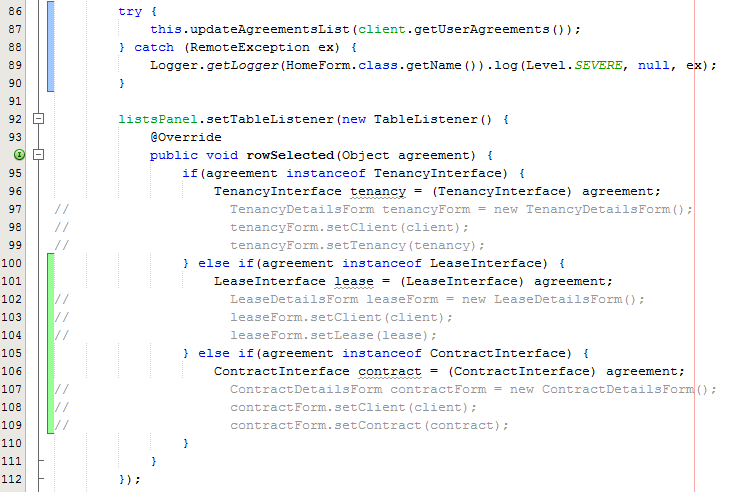


Fig x – Extract from HomeForm class – HomeForm constructor



Fig x – Extract from ListsPanel class, initialising TableListener field (action listener)



Fig x – Extract from ListsPanel class, setTableListener()

As you can see from fig x, fig x, fig x and fig x, I am ensuring the GUI makes use of MVC, by assigning the JPanels within any main frame with listeners, so if anything occurs within a panel (such as the listsPanel for the HomeForm shown above), instead of the listsPanel invoking a method from the HomeForm to notify the HomeForm of any change that has occurred within the panel, the panel is passed an action listener which listens to see if any action has been performed, and if so carries out a function within the main frame.

This ensures that the view (ListsPanel) does not know about the controller (HomeForm) and only interacts with the action listener that was passed to the ListsPanel by the HomeForm, meaning that the HomeForm can be independent, to the