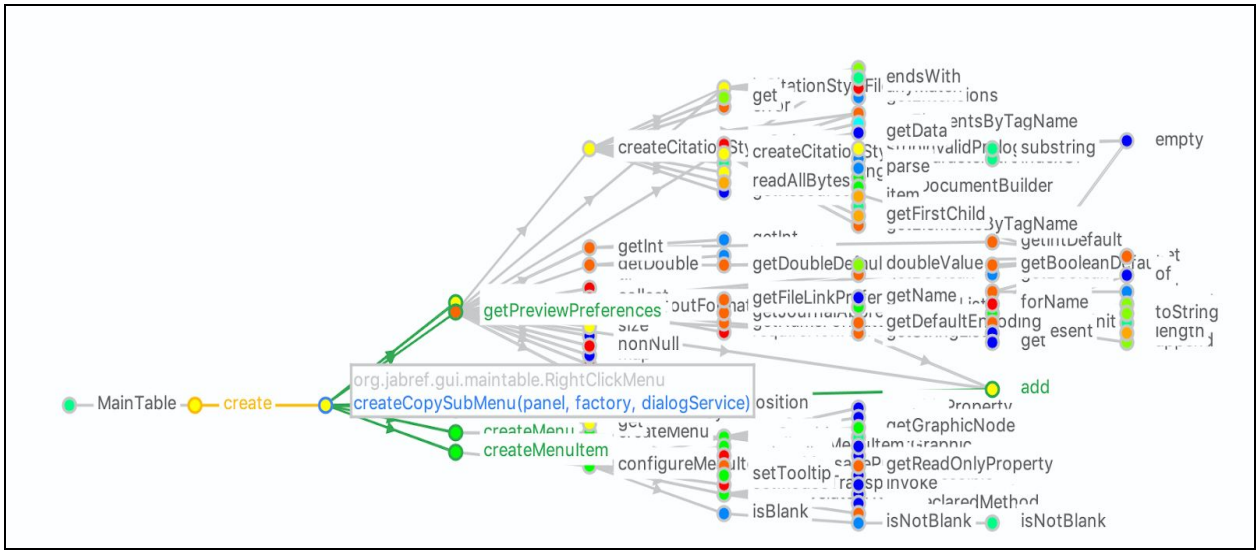
Feature 1 : MainTable

We noticed that UML diagrams were not very helpful in getting the key essence of the features. Hence, we decided to focus on Call graphs to understand the intricacy of some of these features.

To understand ViewModelTableRowFactory, we decided to view the Upstream/Downstream call graph for one of its functions, here, withOnMouseClickedEvent. As expected, it is primarily associated with MainTable and is used for binding click events.

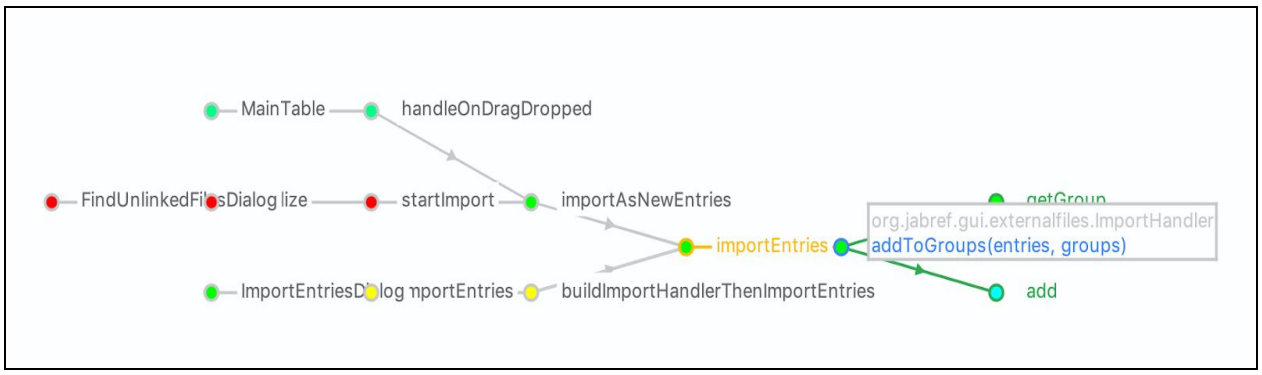
Call graph (1) : onMouseClickedEvent

To understand RightClickMenu, we decided to view the related calls for createCopySubMenu. We were surprised to notice the number of downstream relations that showed up to a seemingly simple menu item. The linkage from MainTable leads to the create() in RightClickMenu which leads to the createCopySubMenu. This accounts for the upstream. One has to trace through all these downstream calls to double-check, in case any feature needs to be implemented at a higher level.



Call graph (2) : createCopySubMenu

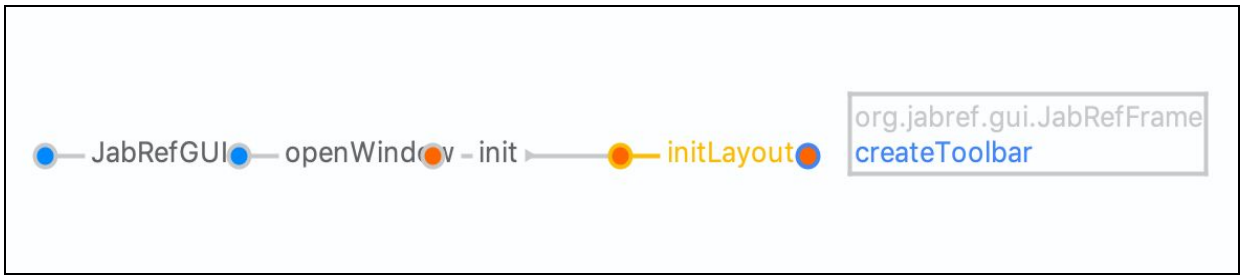
Finally, to understand importHandler, we decided to view addToGroups() which in turn calls minor functions such as getGroup and add to finish the process. The related upstream calls can be observed from ImportEntriesDialog, FindUnlinkedFilesDialog, and MainTable. Any object initialized with ImportHandler in MainTable, is linked through their handleOnDragDropped() where entry is imported as a new entry and added to the groups. This illustration is clear to understand the underlying concept.



Call graph (3) : addToGroups

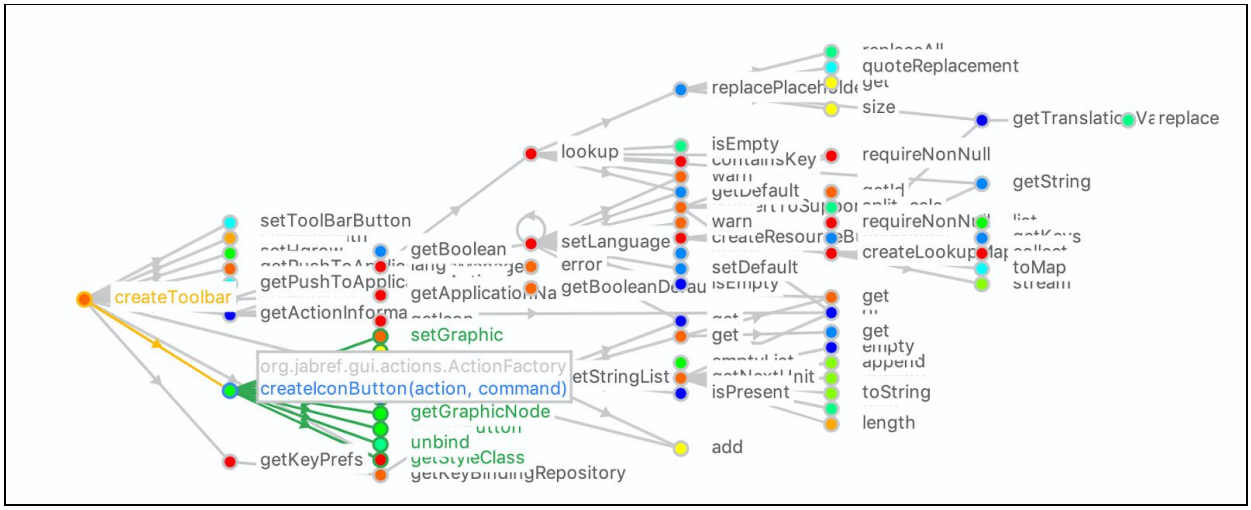
Feature 2 : Add a new article

We notice that when we need to click the plus button to add a new entry. This is essentially obvious from the createToolBar().



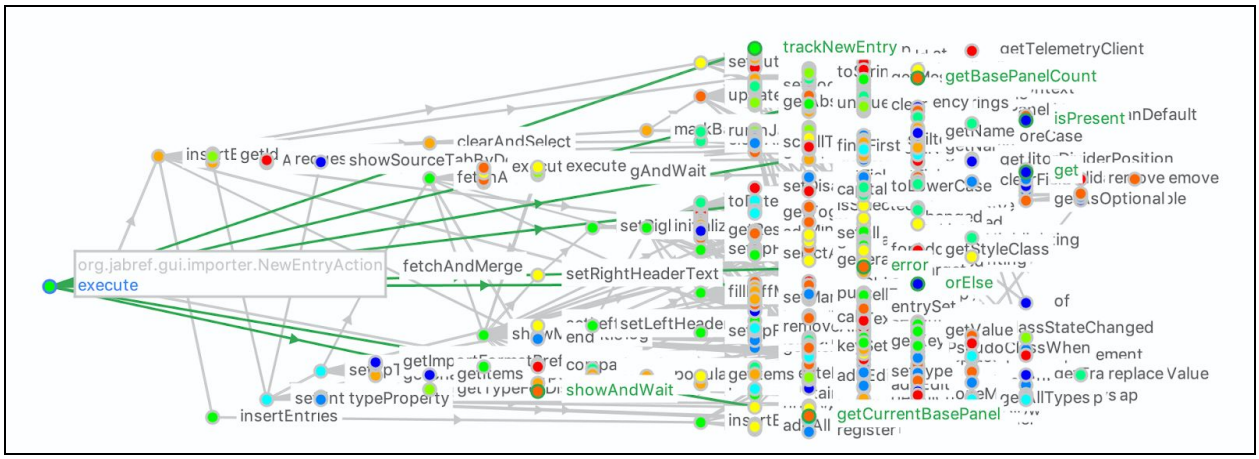
Call graph (4) : createToolBar (Upstream)

On further exploring the downstream call graph for the createToolbar, one can locate the createIconButton which is necessary to create the plus icon. We notice that the command it takes is a NewEntryAction.



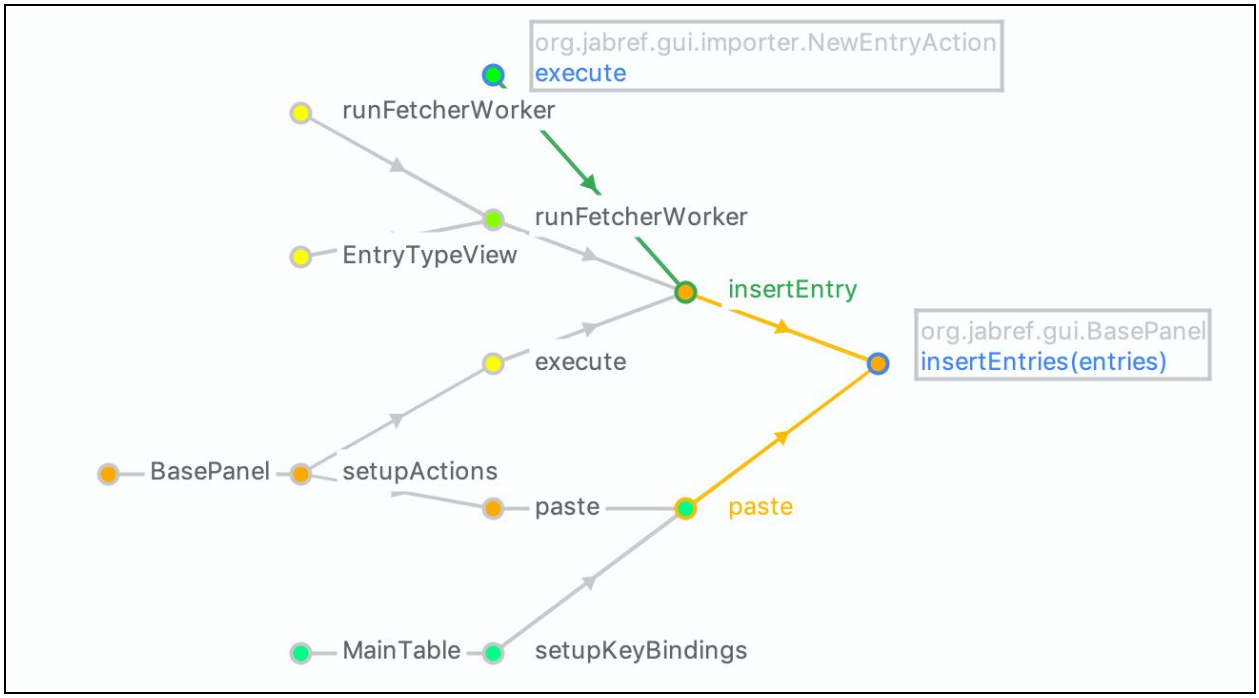
Call graph (5) : createToolBar (Downstream)

When tracing further, we realized execute() is responsible for bringing up the new entry. Therefore, one has to keep in mind about all the downstream lines that would potentially get affected while playing with this function.



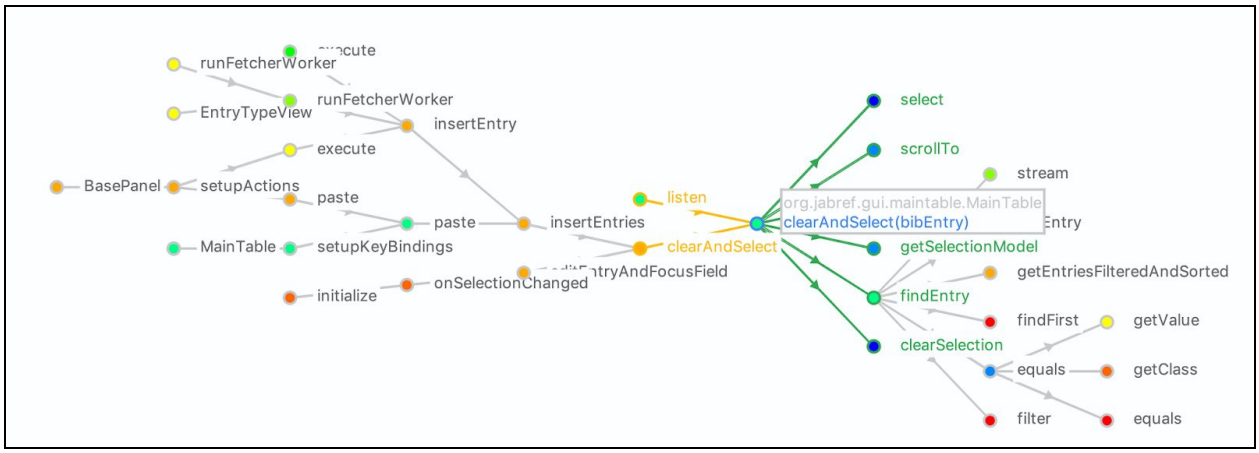
Call graph (6) : execute (Downstream)

One such downstream call from execute() is the insertEntries(), which as the name suggests is what we are looking for. When we analyze this, we noticed that paste functions usually leads to similar insertEntries call from various parts of the program.



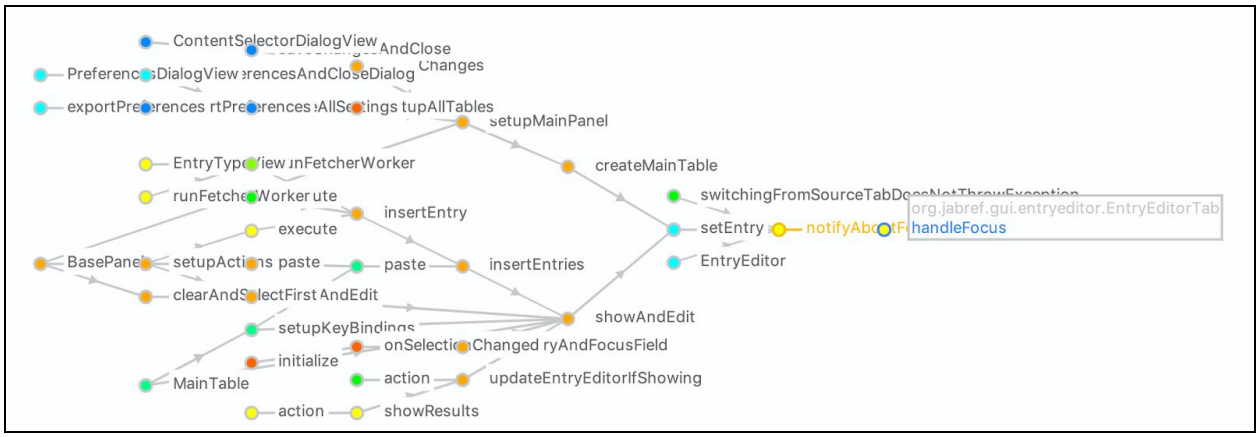
Call graph (7) : insertEntries (Upstream)

While observing the call graph for clearAndSelect(), we see that the listen method has an invocation link, suggesting the possible event bus connection. The possible downstream calls help us to realize that the selection is cleared (clearSelection).



Call graph (8) : clearAndSelect

Finally, handleFocus can be invoked in a lot of ways, as noticed there are no functions called through this method, one can modify this method to get focus and accomplish any task.



Call graph (9) : handleFocus