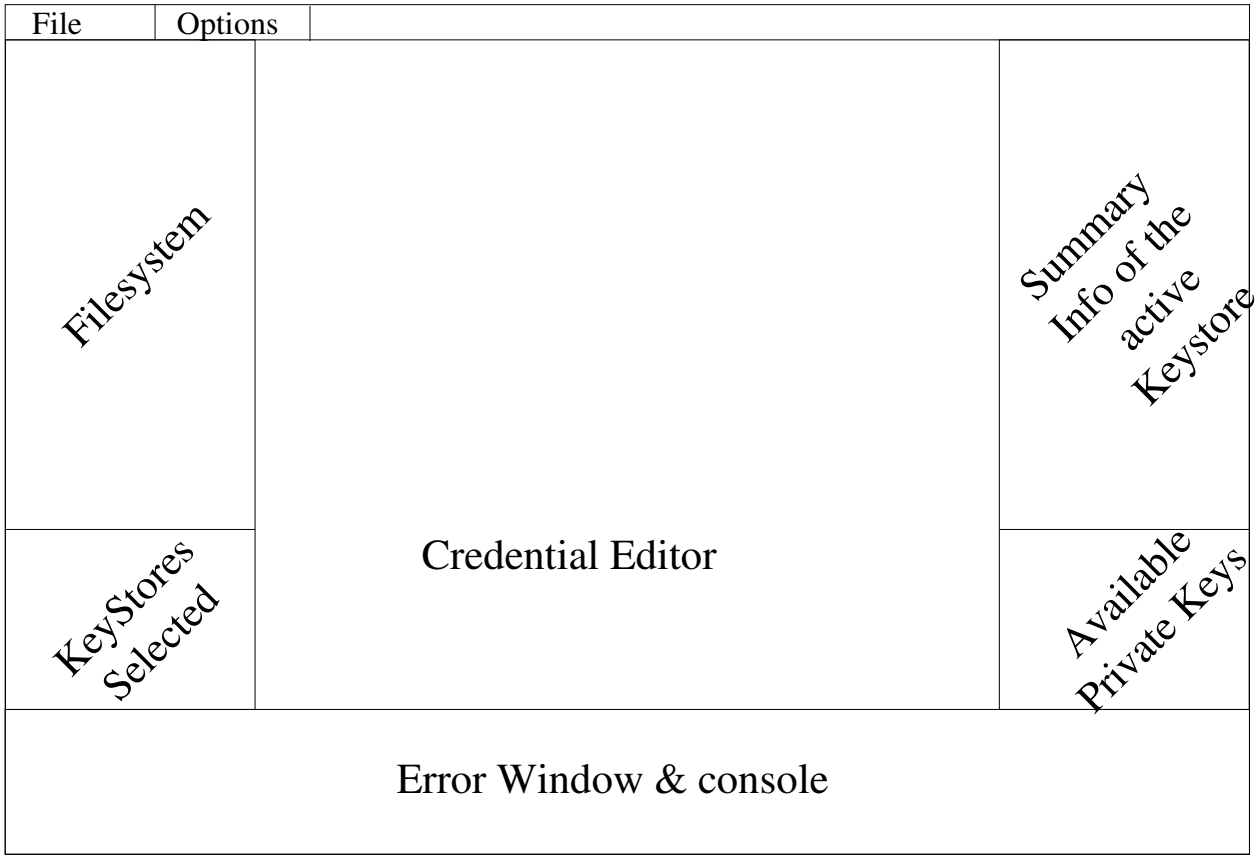
Requirements for a Credential Manager Editor:

I would assume that the screen is something like:



The idea is to have the following options:

- Filesystem: shows the current filesystem tree. It should show with a different icon (e.g. a key) to any file with extension ".ks" (representing a keystore). If clicking twice in that file, it should open a window in the editor with the information of that keystore[1]
- KeyStores selected: two ways to add keystores:
  - Right button on the filesystem window and in a context menu, one option is "Add Keystore"
  - Right button in this window and select "Add Keystore". In this case, a wizard should open (see the RCP tutorial for a definition of wizard) in which a keystore will be selected.
- Error Window & Console: trace of the operations being performed. In general, at development time, it should also include the debug messages.
- Credential Editor: this is the main window in which information is displayed and changed. If a keystore is opened, this editor shows all the information of the keystore. If the default configuration file (e.g. manager.rdf) is opened, two small tabs are shown. In the first one, we have a graphical editor with a tree and in the second, we have the RDF file for manual editing. The credentials are nodes in the tree and the branches are constructed with information from the config file. The

1 You will not deal with the keystore directly so you should make it in a way that an interface is called. Find a good abstraction so the value returned can be reused whatever the information from the credential is.

credentials are always the leaves of the tree (at the beginning it is just a disconnected graph) and the user adds parent nodes and branches in the editor. As the tree must be shown, a proper reusable function to build trees should be used.

An example, a user loads two keystores with 3 certificates: C1, C2 and C3. At the beginning, no information is available from them what means that they will be shown as individual objects in the editor. However, with a right click on the background, the user can create an abstract node. This abstract node can include some information. Then, if a right click is performed on an abstract node, a "Attach child" option appears in which it is possible to select any of the other nodes. All this relationships are stored in the RDF configuration file.

- Summary inf: this panel shows the exact information when a node in the editor is selected. It should be similar to the "outline" panel of the java view in Eclipse.
- Private Keys: this panel always shows the information of the private keys available in the selected keystores.

In the menu bar, two entries should exist:
- New Keystore: which should call a function to create a new keystore[2]
- New Credential: which should calla function to create a new credential[3]

Therefore, the RDF configuration file should include information about
- Subclass relationships between the nodes
- Keystores selected
- Credentials loaded (and to which keystore they belong to)
- Inf. from each credential

This should all be a perspective (see eclipse RCP tutorial).

2 Only call the function. You don't need to implement the function.
3 Only call the function. You don't need to implement the function.