Johar "Star" Interface Interpreter Requirements Specification – Behaviour

1 Top-Level Behaviour

1. Read the IDF.
2. Create the Main Panel.
   (a) Every `CommandGroup` should correspond to a menu on the menu bar.
   (b) The menu item for a command should be the `Label` of the command, followed by "`...`" if selecting the command will result in a Command Dialog Box being created (see below).
3. Initialize the Text Area to empty.
4. Create the GemSetting and validate it.
5. Call the application engine's initialization method.
6. Refresh the tables (see below).
7. Append a horizontal line (this could be just a line of minus characters) to the bottom of the Text Area. (This will separate any initial message from the application engine from the messages that are responses to the commands.)
8. Call the *isActive* method of every command, and make each menu item of each menu active or inactive (greyed out) according to the result of the *isActive* method of the command.

2 Selecting a Command from a Menu

When the user selects a command from a menu:

1. Set a String field `lastDisplayedString` to the empty string. (Rationale: this will for displaying the last text that the program shows the user.)
2. If any table selection is *incomplete* (see below) for the command, then pop up a dialog box with an "OK" button telling the user which tables they have to select rows in in order to issue the command, and a description of the bounds of the number of rows to select (e.g., "at least 2", "between 1 and 5").
3. Otherwise, if no stage in the command has a queryable parameter (i.e., there are no parameters to any of the stages of the command, or all parameters in all stages of the command are `tableEntry` parameters corresponding to browsable tables):
   (a) Call `gemSetting.selectCurrentCommand(cmdName)`.
   (b) Call `gemSetting.selectCurrentStage(0)`.
   (c) For each stage `i` in the command, perform the Stage Loop from the IntI Specification document.
   (d) Call the application engine method.
   (e) Execute the Command Wrapup Procedure (see below) with parameter `false` (indicating command was not cancelled).

4. Otherwise (i.e., if no table selection is incomplete for the command, but there are queryable parameters):
    (a) Call `gemSetting.selectCurrentCommand(cmdName)`.
    (b) Create the Command Dialog for the command. (The Command Dialog will take over the rest of the processing of the command.)

A table selection is *incomplete for a given command* if it contains at least one stage with a parameter of type `tableEntry`, such that the `MinNumberOfReps` for that parameter is greater than the number of rows that are selected in the `SourceTable` of the parameter, and the parameter does not have a `ParentParameter`.

3 Command Wrapup Procedure

The Command Wrapup Procedure is called after the user selects a command from the menu and some processing has been done. It takes one parameter, called `commandWasCancelled`.

1. If there is a Command Dialog, delete it.
2. If `commandWasCancelled` is false (i.e., the application engine method corresponding to the command has been called):
    (a) Determine whether the application should quit, by checking the `QuitAfter` attribute and/or calling the `QuitAfterIf` method of the command.
    (b) If the application should quit:
        i. If `lastDisplayedText` is non-null and not the empty string, then create a dialog box containing `lastDisplayedText` and an “OK” button, and display it. Wait for the user to click “OK”, and then delete the dialog box. (Rationale: if the command executed a `showText` method and then Star exits, it may exit before the user has had time to read the text.)
        ii. Exit the application, e.g. using `System.exit(0)`.
    (c) Otherwise:
        i. Append a horizontal line to the Text Display Area, in order to separate the last command’s output from the output of any future commands.
        ii. Refresh the tables (see below).
3. Call the *isActive* method of every command, and make each menu item of each menu active or inactive (greyed out) according to the result of the *isActive* method of the command.

4 Refreshing the Tables

To refresh the tables:

1. For each table currently being displayed in the Table Area that is now hidden, delete the corresponding tab.

2. For each table not currently being displayed in the Table Area that is now non-hidden, create a tab in the Table Area. (After this happens, it should be the case that the only tables with tabs in the Table Area are non-hidden tables.)
3. For each non-hidden table which has been updated since the last command execution terminated, update the data on the tab to reflect the contents of the table.
4. If there is a top table now, place that table's tab on top in the tab pane.

5 The `ShowTextHandler`

The ShowTextHandler for Star handles text according to the prominence of the text.

1. 0-1999: For each line of text in the message:
   (a) Truncate the line of text, if necessary, to fit in the Status Bar. (Rationale: it should fit all right, but just in case it doesn't, we can show part of it. Because it is low-priority, it seems OK to just show part of it.)
   (b) Set the Status Bar to the text.

   [Note: This will cause the last line of the most recent message to overwrite anything that was in the Status Bar before. This is OK because these messages are "low prominence".]
2. 2000-2999: Append the text to the text being displayed in the Text Display Area. Append the text also to `lastDisplayedText`.
3. 3000 and higher: Create a dialog box containing the text and an "OK" button, and display it. When the user clicks the "OK" button, the box should be deleted.

6 The Command Dialog Box

Creating the command dialog box:

1. Call `gemSetting.selectCurrentStage(0)`.
2. Perform an Initialize Stage procedure.
3. While the current stage has no queryable parameters, perform a Next Stage procedure (see below).
4. Update the dialog box to reflect the current stage, as described in the Star GUI Specification document.

Initialize Stage procedure:

1. If the current stage is a stage that has never been initialized so far, then for each parameter in the current stage:
   (a) If the parameter has a default value, set the value of the parameter in the CommandModel to the default value. (Rationale: since the GUI widgets are linked to the CommandModel, this should have the effect of updating the value in the GUI.)
   (b) Otherwise, if the parameter has a DefaultValueMethod, then call it and set the value of the parameter in the CommandModel to the default value.

Next Stage procedure:

1. Perform a Wrap Up Stage procedure.
2. If the Wrap Up Stage procedure returns true, then (assuming that the current stage is in the variable `currentStage`):
   (a) Set the current stage to `currentStage+1`.
   (b) Call `gemSetting.selectCurrentStage(currentStage)`.
   (c) Perform an Initialize Stage procedure (see below).

Previous Stage procedure:

1. Perform a Wrap Up Stage procedure.
2. If the Wrap Up Stage procedure returns true, then (assuming that the current stage is in the variable `currentStage`):
   (a) Set the current stage to `currentStage-1`.
   (b) Call `gemSetting.selectCurrentStage(currentStage)`.
   (c) Perform an Initialize Stage procedure (see below).

Wrap Up Stage procedure:

1. Validate the current values of the current repetitions of the parameters, as indicated in requirements 26-51 of the IntI specification.
2. If any parameter or parameter repetition does not pass validation, then:
   (a) Collect information in a string about anything that does not pass validation.
   (b) Present that information to the user in a dialog box. The dialog box should have just an "OK" button.
   (c) When the user presses OK, return false.
3. Load the current values of the current repetitions of the parameters for the current stage into the Gem.
4. Call the ParameterCheckMethod of the current stage, if it has one.
5. If the ParameterCheckMethod exists and returns a non-null, non-empty string:
   (a) Show the user the string in a popup with an "OK" button.
   (b) When the user clicks OK, if the current stage has no queryable parameters, then perform the Command Wrapup procedure, with the parameter `true`. (Rationale: This might happen in some obscure situations, such as when a `tableEntry` parameter has a parent parameter in another stage which gets set to the parent value. In these situations, there is nothing we can do but cancel the command.)
   (c) Return false.
6. Otherwise, return true.

When the user presses the Next button (if it is not greyed out):

1. Perform a Next Stage procedure (see below).
2. While the current stage has no queryable parameters, perform a Next Stage procedure (see below).
3. Update the dialog box to reflect the current stage, as described in the Star GUI Specification document.

When the user presses the Previous button (if it exists and is not greyed out):

1. Perform a Previous Stage procedure (see below).
2. While the current stage has no queryable parameters, perform a Previous Stage procedure (see below).
3. Update the dialog box to reflect the current stage, as described in the Star GUI Specification document.

When the user presses the OK button (if it exists and is not greyed out):

1. Perform the Wrap Up Stage procedure.
2. For each stage `i` in the current command, from stage 0 to the last stage:
   (a) Call `gemSetting.selectCurrentStage(i)`.
   (b) Perform the Initialize Stage procedure.
   (c) Perform the Wrap Up Stage procedure.
   (d) If the Wrap Up Stage procedure returns false, then return.

   (Rationale: there may have been some previous stages which have become invalid as a result of the current stage; also, there may be stages with non-queryable parameters whose parameter values have never been loaded.)
3. Call the command method.
4. Perform the Command Wrapup procedure, with the parameter false (indicating the command was not cancelled).

When the user presses the Cancel button:

1. Perform the Command Wrapup procedure, with the parameter `true`.

7 The Parameter Section

When the user clicks the Add Another button:

- Add another repetition section to the parameter section for the parameter, at the bottom of the list. You may need to add a repetition to the model as well. (Rationale: This is safe because, if the Add Another button exists and is not greyed out, then we are not at the maximum number of repetitions for the parameter yet.)

When the user clicks the Move Up button for repetition k:

- Exchange the value in repetition k with the value in repetition $k-1$. This should be done in the model so that the changes will be automatically reflected in the GUI. (Rationale: If the Move Up button exists and is not greyed out, then there is another repetition above the kth repetition.)

When the user clicks the Move Down button for repetition k:

- Exchange the value in repetition k with the value in repetition $k+1$. This should be done in the model so that the changes will be automatically reflected in the GUI. (Rationale: If the Move Down button exists and is not greyed out, then there is another repetition below the kth repetition.)

When the user clicks the Delete button for repetition k:

- Delete repetition k. This should be done in the model as well. (Rationale: If the Delete button exists and is not greyed out, then we are not at the minimum number of repetitions for the parameter yet.)