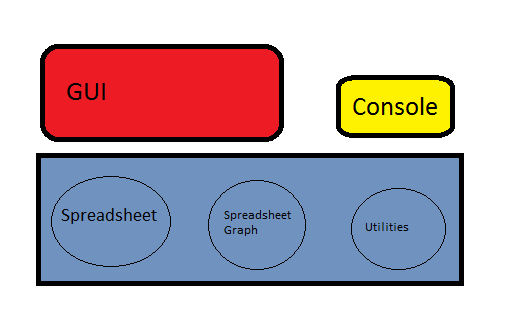
**Technical Manual:**

This Spreadsheet Application is a java application. It mainly has two larger components. The GUI or graphical user interface and the back-end supporting the actual logic of Spreadsheet application.

**GUI:**

The graphical interface is written using Java Swing framework at the Gui.java . The Gui is run with a JTable implemented with a custom DataModel called MyDataModel.java . Using the custom data model allows us to make each individual cell in the table a reference to an individual Cell object. This is the class that captures all the cell changes and repopulates the cells with the updated values. The listeners of the GUI call our back-end methods to drive the program. We have used mainly the framework provided classes like JFrame, JTable, JButton etc. We have also exposed various file menus via JMenu, JMenuIterm etc. and wired actionListener which performs the actual action. All of the Spreadsheet related logic is abstracted into the SpreadSheet class and Gui.java class does not have any code related to the actual working logic of Spreadsheet.



**Back-end:**

The back-end is the collection of classes which implement the crux of the Spreadsheet logic e.g. Setting a formula into a cell, evaluating all cell formulas into values, circular dependency. Its main class is Spreadsheet.java which is object model for the whole spreadsheet. It holds the full two-dimensional array of Cell objects and receives all the updates on the cells from the other classes. The Cell.java is object model for individual cells in a spreadsheet. It holds an int value, int inDegree, and a linked list of cells that depend on it. The inDegree variable is incremented whenever it references another cell. Whenever another cell refers to this cell, a reference to that cell is stored in the linked list of cells that depend on this cell. The Util.java class has utility methods that are used by these object model classes under various occasions. It is mainly called for the getFormula method which breaks down the formula into a stack of LiteralTokens, OperatorTokens, and CellTokens. We then pick this stack apart in our evaluation methods in the Spreadsheet class to retrieve the correct formula. The SpreadsheetGraph.java basically just holds the list of all cells that have been edited or referenced and its primary method is the topologicalSort method. It takes its list of cells and rearranges them in a new list where the inDegrees are sorted in ascending order and returns the new list that is topologically sorted. And this is the order in which we evaluate each cell.

Whenever, user sets a new value to a cell, the function in the Spreadsheet.java class changeCellFormulaAndRecalculate is invoked.

Once a cell-formula is set, it builds the new expression tree for the formula for the cell and then recursively walks the expression tree to find if the new formula introduces any cycle. If cycle is introduced it rolls-back to the old-formula, if not it in-turn calculates the dependency on other cells. It then removes the edges in the SpreadSheetGraph since its old formula is not valid anymore and then add new edges in the graph based out of the new formula. It then calls TopologicalSort routine inside Spreadsheet graph class and then based out of the topological order it evaluates the cells.

**Console-App:**

We also have written a console version of the App, where the Application can be exercised from the Console. This definitely helped us validating the initial version of the application. The Console App is implemented in SpreadSheetApp.java class and it uses the same back-end components e.g. SpreadSheet.java.