Spreadsheet Formulas

Cpt S 321 Homework Assignment, WSU

Submission Instructions:

Submit via Git. This project will be used in future assignments.
Directory name: Spreadsheet
Git tag for submission: Spreadsheet-v4.0

Important note: This is the framework for a spreadsheet application that you will build over the course of the semester. Almost ALL remaining homework assignments will build on top of this. So think about these instructions DEEPLY!

Assignment Instructions:

Read each step's instructions carefully before you write *any* code.

In this assignment you will finish what you completed in the previous homework assignments by implementing an arithmetic expression parser that builds a tree for an expression. This tree can then be used for evaluation of the expression. You may want to refer back to the instructions for the other assignments to make sure that everything was properly implemented there and that you are following the required naming conventions.

Recall that your expression tree class is implemented in your engine DLL and demoed in this assignment through a standalone console application that references the DLL. This is where we will make use of your expression tree implementations to handle calculating the *value* of a cell from its *text*. In this stage, you'll also make it so that variables refer to other cells.

The most interesting part of this project is getting the cells to update their value when another cell that they depend upon changes. There's two major approaches to this:

1) Spreadsheet keeps the current table of dependences between cells
2) Cells are connected directly to each other's PropertyChange of Value

Lastly, you'll need to add to your GUI interface a text box to edit a chosen cell's text, which I'll call the editor box. A text box bar across the top of the cells with your text expression in it just like Excel, LibreOffice, or Google Sheets is probably the best approach. This text box should have it's text string filled in by the cell's Text property when selected when the user clicks on the cell. When the user hits "enter" after updating the editor box's text should push that text to the selected cell's Text property, which then kicks off the expression parsing and updating.

Requirement Details:

Change your spreadsheet's calculate value to use the expression tree (6 points):

- Your Spreadsheet class should have a short function that detects if the text of a cell that's had its text changed calculate and update the value of that cell.
    - That function needs to use your expression tree to do the calculation if the text in the cell begins with "="
- VarNodes in the tree need to get their value from other cells they refer to
    - Use the string name of the VarNode to look up the *Value* of the cell it names
    - This is instead of using the Dictionary in the expression tree
    - If an invalid cell is named, then set the cell's value to "#REF!"
        - Note: this doesn't touch the cell's text, so the user can fix it

Update GUI to have a formula editing text box (4 points):

- Add a text box for the text currently being edited.
- When a cell is selected (see OnClick kinds of handlers), put it's text in the text box
    - Make sure to keep a reference or name of the currently selected cell so you can ask the spreadsheet to update the text on that cell when the user is done
- Either work out how to update the formula when the user changes focus away from the currently selected cell, or when they press enter (hint).

Cascade value changes through the cells (4 points):

- Ensure that when a given cell's value changes, any cells referring to it in their formulas are informed so they can recalculate their own values.
    - Example:
      A2 = B3 + C4
      B3 = 10
      C4 = 14
      => A2 = 24
      When B3 is changed to evaluate to 11, then A2 needs to update to: A2 = 25
- This is best done through the iPropertyChange events & delegates
    - When a cell builds an expression tree that relies upon other cell(s) values
        - The cell needs to register a delegate with the target cell's Value property
        - When the target cell's Value property changes, all cells referring to it should be informed so they can update their own Value by re-evaluating their expression tree's calculation
    - Either:
        - Add a second iPropertyChange handler to the cell to the Value property
        - Or make the other cells delegates pay attention to which property changed in the args passed.

        - Currently, this is only the “Text” property, but it can also be “Value”
        - This means the delegates would have to check the args to decide if they should care about that given event
- NOTE: Cells can make a loop of cascading updates:
    A2 = B3 + 4
    B3 = C4 + 6
    C4 = A2 + 8
    - This loop is infinite and will cause a stack overflow unless you catch it.
    - We won’t address this issue in this assignment, but we’ll talk about it class
        - There’s a relatively easy solution, and a couple more complex ones

<u>Clean, REUSABLE code (1 point):</u>

- Have clean, well-documented code and an easy to use interface for grading.
- Obey all the naming requirements outlined in the first part of this homework (as well as any added here obviously)

15 points total

** Reminder -- code that doesn’t compile for the TA is a zero. This is a 300 level class so you definitely need to be delivering code that at least compiles and runs something.