COP-4226 Advanced Windows Programming

Prograaming Assignment 3

1 Assignment Specification

In this assignment, you will implement the "Neat Office" application with the following layout. As you see below, the main frame is divided into three main panels hosting a Calculator, a Day Counter and a Graph Operator:

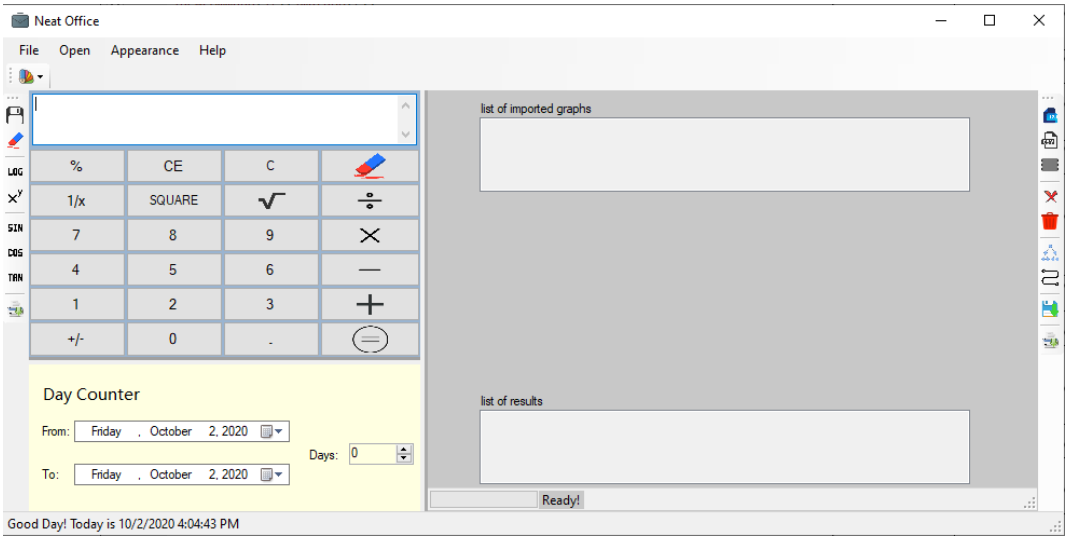

Figure 1: Main Layout of the application.

To manage the dynamic layout of main form properly, you need to add a MenuStrip (with Dock = DockStyle.Top) and a ToolStripContainer (with Dock = DockStyle.Fill) to your main Form. The ToolStripContainer has the following structure:

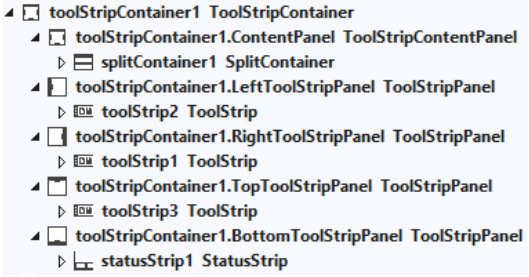

Figure 2: Content of ToolStripContainer

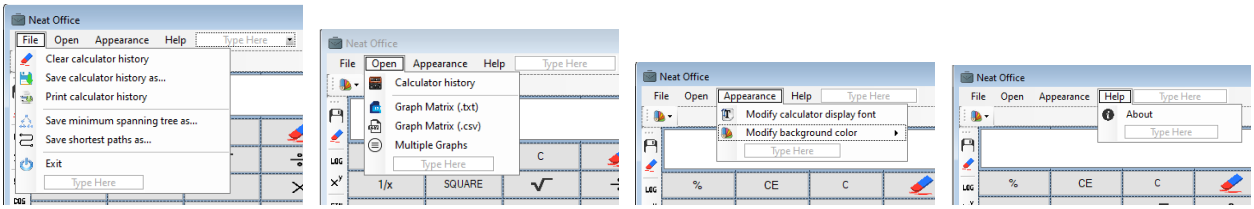

Figure 3: The MenuStrip

And the MenuStrip contains the items depicted in Figure 2. There are three ToolStrips in the ToolStripContainer. You can see two of them in the left and right edge of the main frame depicted in Figure 1. The third one is on top, below the MenuStrip and only contains a ToolStripDropDownButton that allows the user to change the background color of Day Counter, Calculator, and Graph Operator. To select a color, you need to use ColorDialog, and to modify the background color of a panel, you need to update its BackColor property.

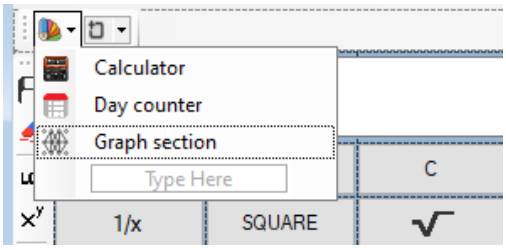

Figure 4: Content of ToolStripContainer.TopToolStripPanel

Also, there is a StatusStrip located at the bottom of the ToolStripContainer showing the current date and time (see the bottom edge of Figure 1 for illustration). To update the date and time every second, you need to handle the Tick event of a Timer with Interval = 1000. Also, the current time and date is available in DateTime.Now. You may update your StatusStrip periodically using StatusStrip.Update() method.

The ContentPanel of our ToolStripContainer is divided into three sub-panels using two SplitContainers and two Splitters:

1.1 Main Layout of the Application

In Figure 5, you can see that our ToolStripContentPanel only contains one SplitContainer called splitContainer1 with vertical (default) orientation. splitConainer1 splits the container into two panels: Panel1 (left) and Panel2 (right). splitContainer1.Panel1 contains another SplitContainer called splitContainer2 (its orientation is horizontal). However, splitContainer1.Panel2 contains the Graph Operator module made of two ListBoxes, two Labels and a StatusStrip hosting a ToolStripProgressBar and a ToolStripStatusLabel.

splitContainer2 splits the container into two panels: splitContainer2.Panel1 (top) contains the calculator which is organized with the help of a TableLayoutPanel called tableLayoutPanel1; splitContainer2.Panel2 contains the Day Counter (two DateTimePickers, four Labels and a NumericUpDown).

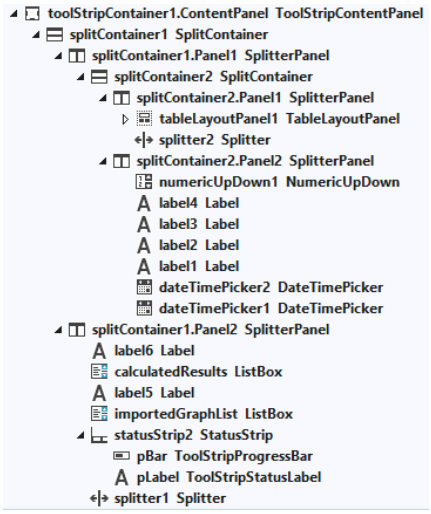

Figure 5: Content of ToolStripContainer.TopToolStripPanel

Please note that we need two Splitters: splitter1 and splitter2. splitter1 is located in splitContainer1.Panel2. It moves the vertical border separating splitContainer1.Panel1 and splitContainer1.Panel2 to the left and right. The Dock property of splitter1 must be DockStyle.Left (default value). splitter2 is located in splitContainer2.Panel1. The Dock property of splitter2 must be set to DockStyle.Bottom.

Finally, tableLayoutPanel1 can be configured as a table of seven rows and four columns (see Figure 6 for more details).

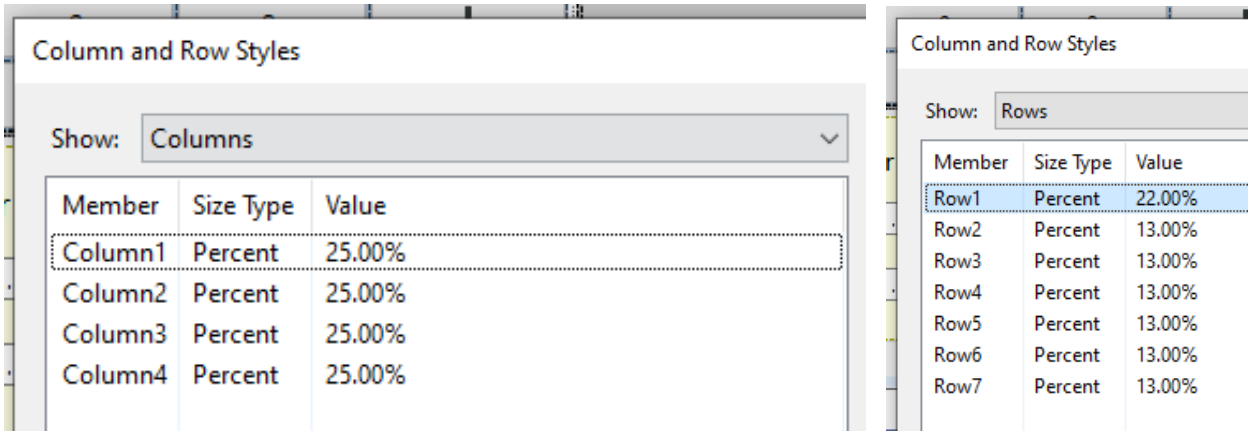

Figure 6: Configuring tableLayoutPanel1 which hosts buttons of the calculator

1.2 Functionality of Calculator

The calculator supports:

1. the following binary operators: $+, -, \times, \div, x^y$.

2. the following unary operators: $\sqrt{x}, x^2$, % (division by 100), $\frac{1}{x}$, and +/− (multiplying by -1). Each unary operator comes after its operand.

3. the following unary functions: $\log(x), \sin(x), \cos(x), \tan(x)$. To apply each one of them on value x, the user first enters the value x, and then clicks on the button specifying the function.

4. the equal button: calculates the current algebraic expression, displays the result, and adds the expression to the history of calculator.

5. backspace (top-right button of calculator in Figure 1): removes the most recently entered digit after the most recently entered operator (or function).

6. CE clears the current numerical value (either calculated or entered by user) from display and replaces it by zero.

7. C clears the algebraic expression written on the last line of display.

8. saving history of calculator in a txt file. The user can initiate the process by either clicking on the first button in the left tool strip (Figure 1) or clicking on the second item of File menu in the MenuStrip. The application must show a SaveFileDialog asking user for file path and file name.

9. clearing the history of calculator. The user can initiate the process by either clicking on the second button in the left tool strip (Figure 1) or clicking on the first item of File menu in the MenuStrip.

10. loading calculator history from a txt file. The user can initiate the process by clicking on the first item of Open menu in the MenuStrip shown in Figure 3. The application must show an OpenFileDialog so that the user can select the file after browsing the file system.

11. printing calculator history (**optional**). The user can initiate the process by clicking on the last button on the left tool strip (Figure 1), or clicking on the third item of File menu in the MenuStrip depicted in Figure 3.

1.3 Functionality of Day Counter

The functionality of Day Counter is straight-forward: The value of NumericUpDown should be always equal to ToDate − FromDate. If user modifies NumericUpDown, the application must react by updating ToDate. If user modifies either ToDate or FromDate, the application must react by updating NumericUpDown.

1.4 Functionality of Graph Operator

For implementing this section, you will use the C# code written in "PriorityQueue.cs" and "GraphAlgorithms.cs" (available on Canvas). The main Form of your application must have a instance field of type GraphAlgorithms (private GraphAlgorithms g) initialized in the

constructor of the form:

```
public mainForm()
{
    InitializeComponent();
    g = new GraphAlgorithms(pBar, pLabel, statusStrip2);
}
```

where statusStrip2 is the StatusBar docked at the Bottom of splitContainer1.Panel2 (see Figure 5), pBar is the ToolStripProgressBar located in statusStrip2, and pLabel is the ToolStripStatusLabel located in statusStrip2 as well.

Graph operator supports:

1. importing a graph by reading its matrix representation stored in a .txt file (see examples of the different files representing multiple graphs on Canvas). The user can initiate the process by clicking on either the first button in the right ToolStrip (see Figure 1) or the second item of Open menu in the MenuStrip shown in Figure 3. The application must first show an OpenFileDialog (with Filter = "txt files (*.txt)|*.txt") so that the user can select the .txt file after browsing the file system. Then, it uses GraphAlgorithms.ReadGraphFromTXTFile method to read the graph. Finally, it adds the file name to the top ListBox showing the list of imported graphs (See Figure 1).

2. importing a graph by reading its matrix representation stored in a .csv file (see examples of the different files representing multiple graphs on Canvas). The user can initiate the process by clicking on either the second button in the right ToolStrip (see Figure 1) or the third item of Open menu in the MenuStrip shown in Figure 3. The application must show an OpenFileDialog (with Filter = "csv files (*.csv)|*.csv") so that the user can select the .csv file after browsing the file system. Then, it uses GraphAlgorithms.ReadGraphFromCSVFile method to read the graph. Finally, it adds the file name to the top ListBox showing the list of imported graphs (See Figure 1).

3. importing multiple graphs by reading their matrix representation stored in either a .csv file or a .txt file. The user can initiate the process by clicking on either the third button in the right ToolStrip (see Figure 1) or the fourth (last) item of Open menu in the MenuStrip shown in Figure 3. This functionality is a mixture of the previous two and uses an OpenFileDialog to select multiple files (Multiselect = True and Filter = "all supported(*.csv,*.txt)|*.csv;*.txt|csv files (*.csv)|*.csv|txt files (*.txt)|*.txt").

4. deleting an imported graph selected from the list of imported graphs in the top ListBox. The user can initiate the process by clicking on the fourth button in the right ToolStrip (see Figure 1).

5. deleting all imported graphs listed in the top ListBox. The user can initiate the process by clicking on the fifth button in the right ToolStrip (see Figure 1).

6. applying Prim's algorithm on an imported graph selected from the list of imported graphs in the top ListBox. The user can initiate the process by clicking on the sixth button in the right ToolStrip (see Figure 1). The application reacts by first calling

GraphAlgorithms.GetMST method and then, adding the file name to the bottom ListBox showing the list of results (See Figure 1). Add "MST: " before the file name to distinguish the result of Prim's algorithm from Dijkstra's in the list of results.

7. applying Dijkstra's algorithm on an imported graph selected from the list of imported graphs in the top ListBox. The user can initiate the process by clicking on the seventh button in the right ToolStrip (see Figure 1). The application reacts by first calling GraphAlgorithms.Dijkstra method and then, adding the file name to the bottom ListBox showing the list of results (See Figure 1). Add "Shortest Paths: " before the file name to distinguish the result of Dijkstra's algorithm from Prim's in the list of results.

8. saving the result of one of the completed graph operations in a .txt file. The result must be selected from the list of results in the bottom ListBox. The user can initiate the process by either clicking on the eighth button in the right ToolStrip (see Figure 1) or clicking on the fourth/fifth item in the File menu of the MenuStrp (see Figure 3). The application reacts by first showing a SaveFileDialog asking user for file path and file name, and then, calling either GraphAlgorithms.WriteMSTSolutionTo or GraphAlgorithms.WriteSSSPSolutionTo (depending on the result type). The application should delete the result from the list of results after saving it in a file.

9. (**optional**) printing the result of one of the completed graph operations. The result must be selected from the list of results in the bottom ListBox. The user can initiate the process by clicking on the last button in the right ToolStrip (see Figure 1). The application should delete the result from the list of results after printing it.

2 20% Extra Credit

Implement the printing operations that print calculator history and Graph Operator results.

3 Deliverables

Only one member from each team should submit a zip file containing the following items: A short mp4 video (less than 4 minutes) illustrating how to interact with the application, the source files of your program (.cs files), and readme.txt containing your name (and your teammate).