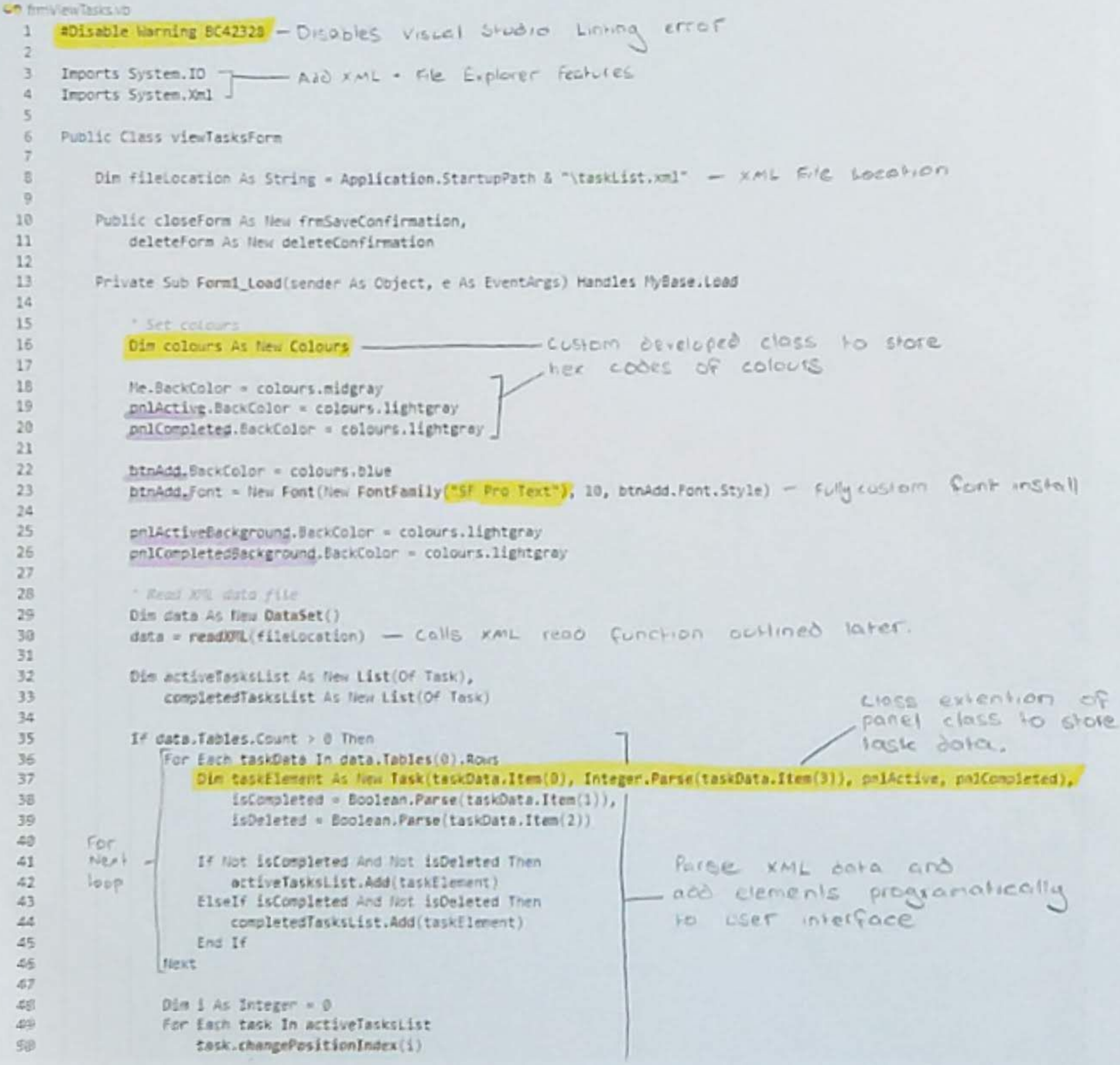
🟪 Hungarian notation usage for static controls



frmViewTasks.vb

```
#Disable Warning BC42328

Imports System.IO
Imports System.Xml

Public Class viewTasksForm

    Dim fileLocation As String = Application.StartupPath & "\taskList.xml"

    Public closeForm As New frmSaveConfirmation,
        deleteForm As New deleteConfirmation

    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        ' Set colours
        Dim colours As New Colours

        Me.BackColor = colours.midgray
        pnlActive.BackColor = colours.lightgray
        pnlCompleted.BackColor = colours.lightgrey

        btnAdd.BackColor = colours.blue
        btnAdd.Font = New Font(New FontFamily("SF Pro Text"), 10, btnAdd.Font.Style)

        pnlActiveBackground.BackColor = colours.lightgray
        pnlCompletedBackground.BackColor = colours.lightgray

        ' Read XML data file
        Dim data As New DataSet()
        data = readXML(fileLocation)

        Dim activeTasksList As New List(Of Task),
            completedTasksList As New List(Of Task)

        If data.Tables.Count > 0 Then
            For Each taskData In data.Tables(0).Rows
                Dim taskElement As New Task(taskData.Item(0), Integer.Parse(taskData.Item(3)), pnlActive, pnlCompleted),
                    isCompleted = Boolean.Parse(taskData.Item(1)),
                    isDeleted = Boolean.Parse(taskData.Item(2))

                If Not isCompleted And Not isDeleted Then
                    activeTasksList.Add(taskElement)
                ElseIf isCompleted And Not isDeleted Then
                    completedTasksList.Add(taskElement)
                End If
            Next

            Dim i As Integer = 0
            For Each task In activeTasksList
                task.changePositionIndex(i)
```

— Disables Visual Studio Linting error

— Add XML + File Explorer features

— XML File location

Custom developed class to store hex codes of colours

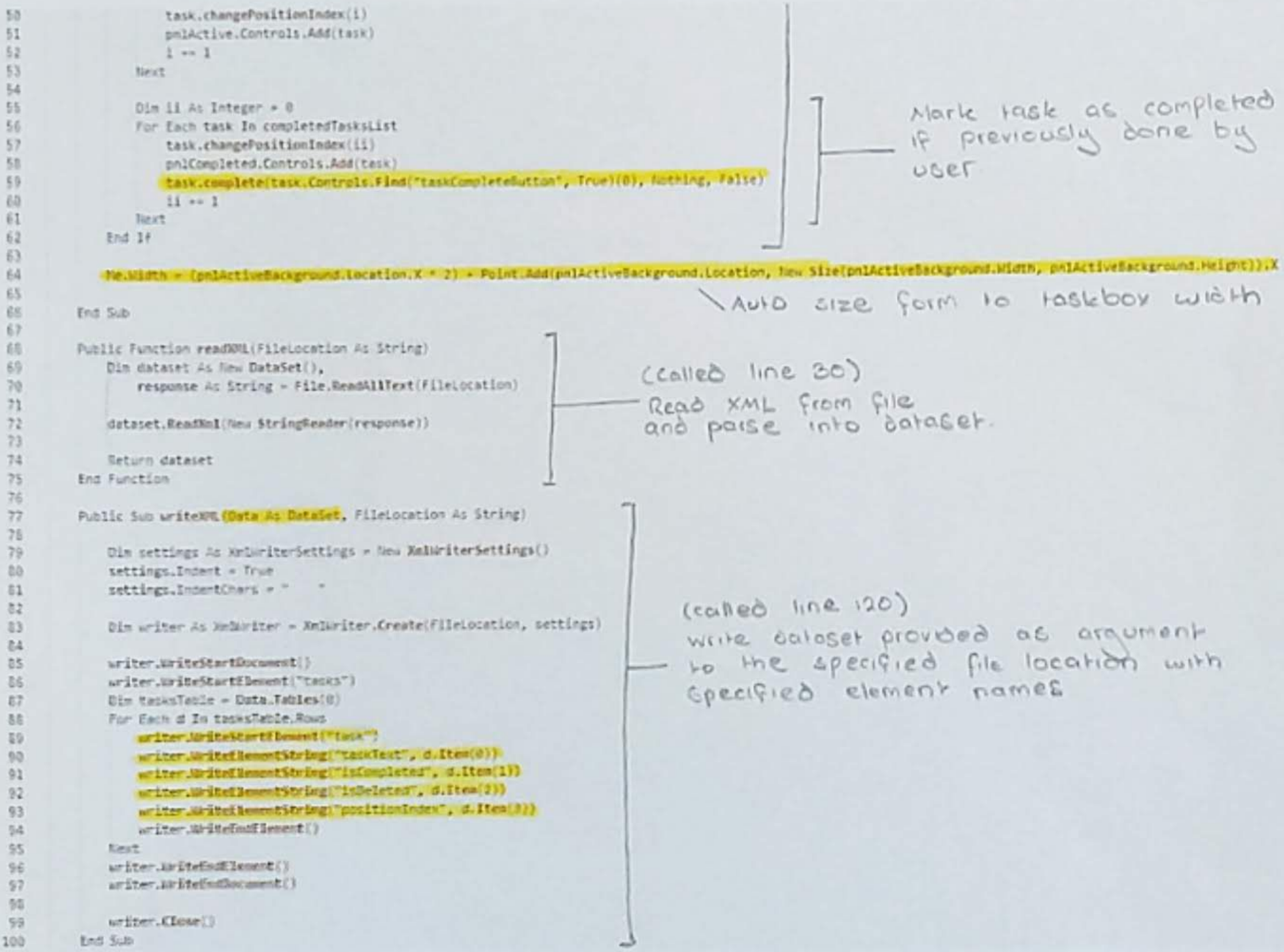
— Fully custom font install

— Calls XML read function outlined later.

Class extention of panel class to store task data.

For Next loop

Parse XML data and add elements programatically to user interface



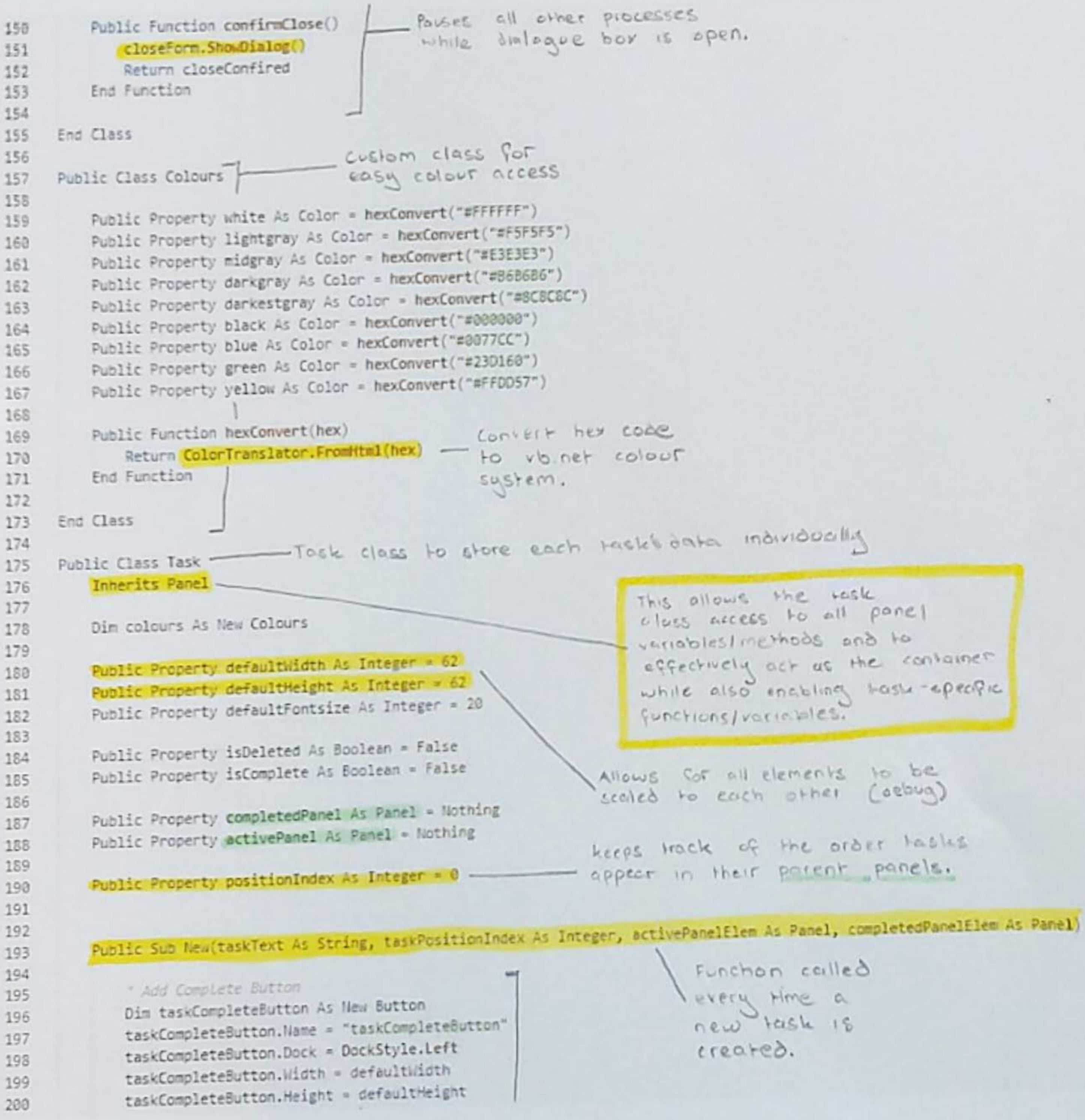
```
50                task.changePositionIndex(i)
51                pnlActive.Controls.Add(task)
52                i += 1
53            Next
54
55            Dim ii As Integer = 0
56            For Each task In completedTasksList
57                task.changePositionIndex(ii)
58                pnlCompleted.Controls.Add(task)
59                task.complete(task.Controls.Find("taskCompleteButton", True)(0), Nothing, False)
60                ii += 1
61            Next
62        End If
63
64        Me.Width = (pnlActiveBackground.Location.X * 2) + Point.Add(pnlActiveBackground.Location, New Size(pnlActiveBackground.Width, pnlActiveBackground.Height)).X
65
66    End Sub
67
68    Public Function readXML(FileLocation As String)
69        Dim dataset As New DataSet(),
70            response As String = File.ReadAllText(FileLocation)
71
72        dataset.ReadXml(New StringReader(response))
73
74        Return dataset
75    End Function
76
77    Public Sub writeXML(Data As DataSet, FileLocation As String)
78
79        Dim settings As XmlWriterSettings = New XmlWriterSettings()
80        settings.Indent = True
81        settings.IndentChars = "    "
82
83        Dim writer As XmlWriter = XmlWriter.Create(FileLocation, settings)
84
85        writer.WriteStartDocument()
86        writer.WriteStartElement("tasks")
87        Dim tasksTable = Data.Tables(0)
88        For Each d In tasksTable.Rows
89            writer.WriteStartElement("task")
90            writer.WriteElementString("taskText", d.Item(0))
91            writer.WriteElementString("isCompleted", d.Item(1))
92            writer.WriteElementString("isDeleted", d.Item(2))
93            writer.WriteElementString("positionIndex", d.Item(3))
94            writer.WriteEndElement()
95        Next
96        writer.WriteEndElement()
97        writer.WriteEndDocument()
98
99        writer.Close()
100   End Sub
```

Mark task as completed
if previously done by
user

Auto size form to taskbox width

(called line 30)
Read XML from file
and parse into dataset.

(called line 120)
write dataset provided as argument
to the specified file location with
specified element names



```
    Public Function confirmClose()
        closeForm.ShowDialog()
        Return closeConfired
    End Function

End Class
```

Pauses all other processes while dialogue box is open.

```
Public Class Colours
```

custom class for easy colour access

```
    Public Property white As Color = hexConvert("#FFFFFF")
    Public Property lightgray As Color = hexConvert("#F5F5F5")
    Public Property midgray As Color = hexConvert("#E3E3E3")
    Public Property darkgray As Color = hexConvert("#B6B6B6")
    Public Property darkestgray As Color = hexConvert("#8C8C8C")
    Public Property black As Color = hexConvert("#000000")
    Public Property blue As Color = hexConvert("#0077CC")
    Public Property green As Color = hexConvert("#23D160")
    Public Property yellow As Color = hexConvert("#FFDD57")

    Public Function hexConvert(hex)
        Return ColorTranslator.FromHtml(hex)
    End Function

End Class
```

Convert hex code to vb.net colour system.

```
Public Class Task
    Inherits Panel

    Dim colours As New Colours

    Public Property defaultWidth As Integer = 62
    Public Property defaultHeight As Integer = 62
    Public Property defaultFontsize As Integer = 20

    Public Property isDeleted As Boolean = False
    Public Property isComplete As Boolean = False

    Public Property completedPanel As Panel = Nothing
    Public Property activePanel As Panel = Nothing

    Public Property positionIndex As Integer = 0
```

Task class to store each task's data individually

This allows the task class access to all panel variables/methods and to effectively act as the container while also enabling task-specific functions/variables.

Allows for all elements to be scaled to each other (debug)

keeps track of the order tasks appear in their parent panels.

```

    Public Sub New(taskText As String, taskPositionIndex As Integer, activePanelElem As Panel, completedPanelElem As Panel)

        ' Add Complete Button
        Dim taskCompleteButton As New Button
        taskCompleteButton.Name = "taskCompleteButton"
        taskCompleteButton.Dock = DockStyle.Left
        taskCompleteButton.Width = defaultWidth
        taskCompleteButton.Height = defaultHeight
```

Function called every time a new task is created.

Form about to close handler

```
100         End Sub
101 
102         Private Sub Form1_Closing(Source As Object, e As System.Windows.Forms.FormClosingEventArgs) Handles Me.Closing
103 
```

Close validation form

```
104             If confirmClose() Then
105                 Dim ds As New DataSet()
106 
107                 ds.Tables.Add("tasks")
108                 ds.Tables(0).Columns.Add("taskName", GetType(String))
109                 ds.Tables(0).Columns.Add("isCompleted", GetType(Boolean))
110                 ds.Tables(0).Columns.Add("isDeleted", GetType(Boolean))
111                 ds.Tables(0).Columns.Add("locationIndex", GetType(Integer))
112 
113                 For Each task In pnlActive.Controls
114                     ds.Tables(0).Rows.Add(task.Controls.Find("taskTextBox", True)(0).Text, False, False, pnlActive.Controls.IndexOf(task))
115                 Next
116                 For Each task In pnlCompleted.Controls
117                     ds.Tables(0).Rows.Add(task.Controls.Find("taskTextBox", True)(0).Text, True, False, pnlCompleted.Controls.IndexOf(task))
118                 Next
```

create dataset and find data to write.

```
119 
120                 writeXML(ds, fileLocation)
121             End If
122 
123         End Sub
124 
125         Private Sub addButton_Click(sender As Object, e As EventArgs) Handles btnAdd.Click
126             Dim addTaskForm As New frmAddTask
127 
128             addTaskForm.tasksForm = Me
129             addTaskForm.Reset()
130             addTaskForm.ShowDialog()
131         End Sub
```

Show task add form as dialogue box

```
132 
133         ' Called by _addTaskForm
134         Public Sub addTask(taskName As String)
135             Dim addTaskForm As New frmAddTask
136             Dim taskPositionIndex As Integer = pnlActive.Controls.Count,
137                 newTask As New Task(taskName, taskPositionIndex, pnlActive, pnlCompleted)
138             pnlActive.Controls.Add(newTask)
139         End Sub
```

use of custom class.

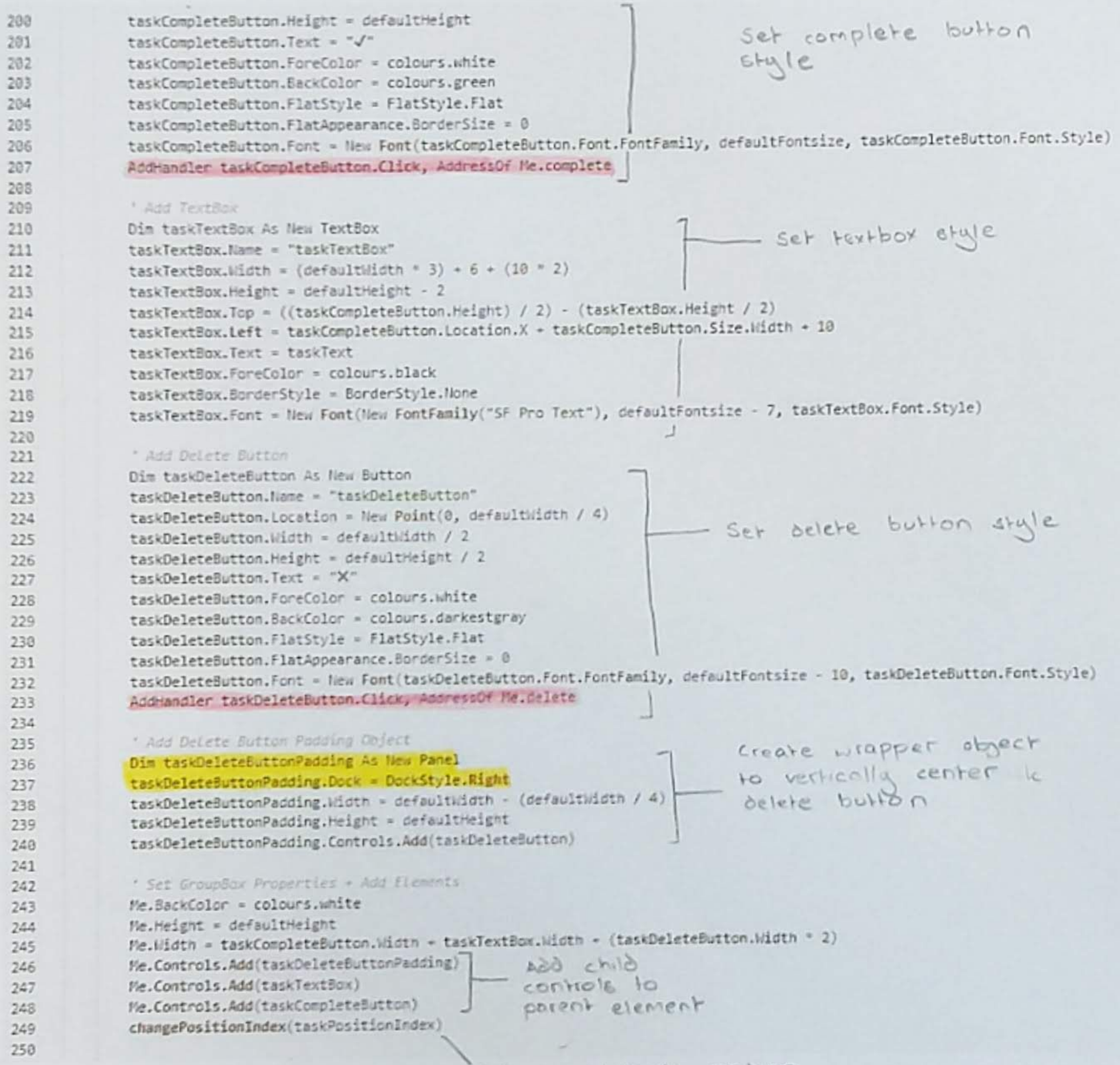
```
140 
141         ' Called by _addTaskForm
142         Public deleteConfirmed As Boolean
143         Public Function confirmDelete()
144             deleteConfirmation.ShowDialog()
145             Return deleteConfirmed
146         End Function
```

(Called line 109)

- Features public boolean accessed in the add task form.

```
147 
148         ' Called by _closeConfirmationForm
149         Public closeConfired As Boolean
150         Public Function confirmClose()
```

Second public boolean accessed by close confirm form.



```
        taskCompleteButton.Height = defaultHeight
        taskCompleteButton.Text = "✓"
        taskCompleteButton.ForeColor = colours.white
        taskCompleteButton.BackColor = colours.green
        taskCompleteButton.FlatStyle = FlatStyle.Flat
        taskCompleteButton.FlatAppearance.BorderSize = 0
        taskCompleteButton.Font = New Font(taskCompleteButton.Font.FontFamily, defaultFontsize, taskCompleteButton.Font.Style)
        AddHandler taskCompleteButton.Click, AddressOf Me.complete

        ' Add TextBox
        Dim taskTextBox As New TextBox
        taskTextBox.Name = "taskTextBox"
        taskTextBox.Width = (defaultWidth * 3) + 6 + (10 * 2)
        taskTextBox.Height = defaultHeight - 2
        taskTextBox.Top = ((taskCompleteButton.Height) / 2) - (taskTextBox.Height / 2)
        taskTextBox.Left = taskCompleteButton.Location.X + taskCompleteButton.Size.Width + 10
        taskTextBox.Text = taskText
        taskTextBox.ForeColor = colours.black
        taskTextBox.BorderStyle = BorderStyle.None
        taskTextBox.Font = New Font(New FontFamily("SF Pro Text"), defaultFontsize - 7, taskTextBox.Font.Style)

        ' Add Delete Button
        Dim taskDeleteButton As New Button
        taskDeleteButton.Name = "taskDeleteButton"
        taskDeleteButton.Location = New Point(0, defaultWidth / 4)
        taskDeleteButton.Width = defaultWidth / 2
        taskDeleteButton.Height = defaultHeight / 2
        taskDeleteButton.Text = "X"
        taskDeleteButton.ForeColor = colours.white
        taskDeleteButton.BackColor = colours.darkestgray
        taskDeleteButton.FlatStyle = FlatStyle.Flat
        taskDeleteButton.FlatAppearance.BorderSize = 0
        taskDeleteButton.Font = New Font(taskDeleteButton.Font.FontFamily, defaultFontsize - 10, taskDeleteButton.Font.Style)
        AddHandler taskDeleteButton.Click, AddressOf Me.delete

        ' Add Delete Button Padding Object
        Dim taskDeleteButtonPadding As New Panel
        taskDeleteButtonPadding.Dock = DockStyle.Right
        taskDeleteButtonPadding.Width = defaultWidth - (defaultWidth / 4)
        taskDeleteButtonPadding.Height = defaultHeight
        taskDeleteButtonPadding.Controls.Add(taskDeleteButton)

        ' Set GroupBox Properties + Add Elements
        Me.BackColor = colours.white
        Me.Height = defaultHeight
        Me.Width = taskCompleteButton.Width + taskTextBox.Width + (taskDeleteButton.Width * 2)
        Me.Controls.Add(taskDeleteButtonPadding)
        Me.Controls.Add(taskTextBox)
        Me.Controls.Add(taskCompleteButton)
        changePositionIndex(taskPositionIndex)
```

Set complete button style

Set textbox style

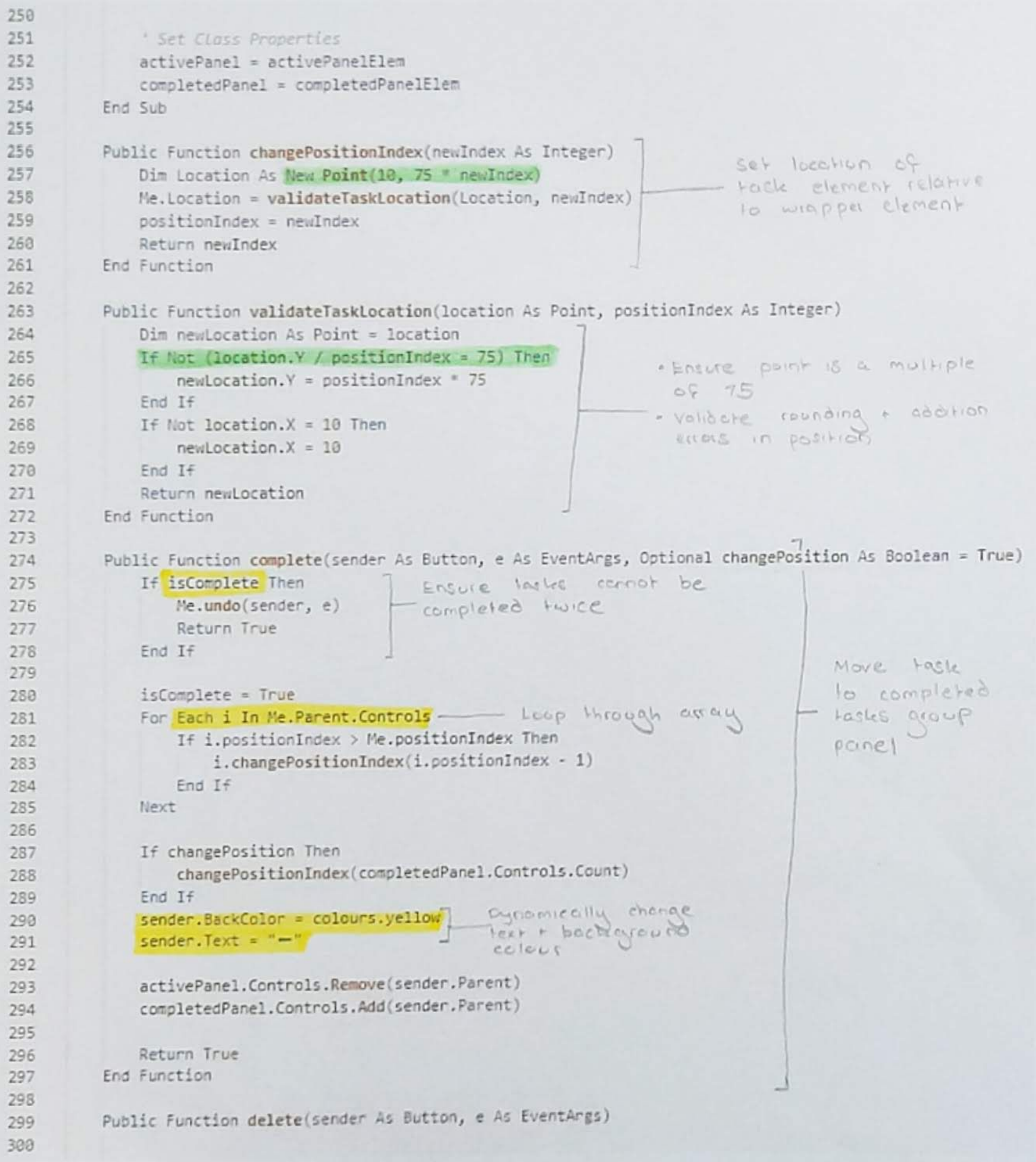
Set delete button style

Create wrapper object to vertically center k delete button

Add child controls to parent element

Set initial task position

🟥 Functions programatically added to each button



```
        ' Set Class Properties
        activePanel = activePanelElem
        completedPanel = completedPanelElem
    End Sub

    Public Function changePositionIndex(newIndex As Integer)
        Dim Location As New Point(10, 75 * newIndex)
        Me.Location = validateTaskLocation(Location, newIndex)
        positionIndex = newIndex
        Return newIndex
    End Function

    Public Function validateTaskLocation(location As Point, positionIndex As Integer)
        Dim newLocation As Point = location
        If Not (location.Y / positionIndex = 75) Then
            newLocation.Y = positionIndex * 75
        End If
        If Not location.X = 10 Then
            newLocation.X = 10
        End If
        Return newLocation
    End Function

    Public Function complete(sender As Button, e As EventArgs, Optional changePosition As Boolean = True)
        If isComplete Then
            Me.undo(sender, e)
            Return True
        End If

        isComplete = True
        For Each i In Me.Parent.Controls
            If i.positionIndex > Me.positionIndex Then
                i.changePositionIndex(i.positionIndex - 1)
            End If
        Next

        If changePosition Then
            changePositionIndex(completedPanel.Controls.Count)
        End If
        sender.BackColor = colours.yellow
        sender.Text = "—"

        activePanel.Controls.Remove(sender.Parent)
        completedPanel.Controls.Add(sender.Parent)

        Return True
    End Function

    Public Function delete(sender As Button, e As EventArgs)
```

Set location of task element relative to wrapper element

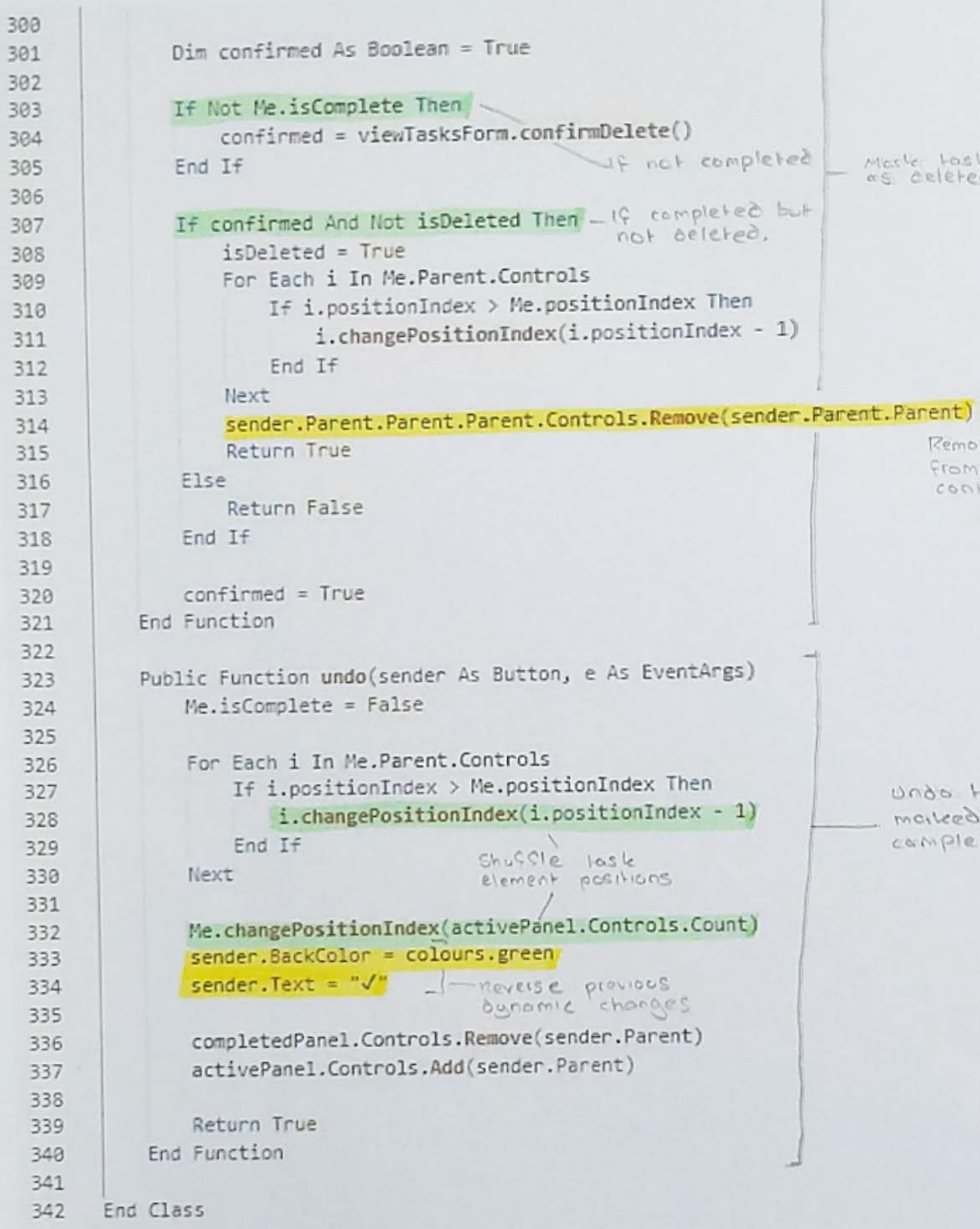
- Ensure point is a multiple of 75
- Validate rounding + addition errors in position

Ensure tasks cannot be completed twice

Move task to completed tasks group panel

Loop through array

Dynamically change text + background colour



```
        Dim confirmed As Boolean = True

        If Not Me.isComplete Then
            confirmed = viewTasksForm.confirmDelete()
        End If

        If confirmed And Not isDeleted Then
            isDeleted = True
            For Each i In Me.Parent.Controls
                If i.positionIndex > Me.positionIndex Then
                    i.changePositionIndex(i.positionIndex - 1)
                End If
            Next
            sender.Parent.Parent.Parent.Controls.Remove(sender.Parent.Parent)
            Return True
        Else
            Return False
        End If

        confirmed = True
    End Function

    Public Function undo(sender As Button, e As EventArgs)
        Me.isComplete = False

        For Each i In Me.Parent.Controls
            If i.positionIndex > Me.positionIndex Then
                i.changePositionIndex(i.positionIndex - 1)
            End If
        Next

        Me.changePositionIndex(activePanel.Controls.Count)
        sender.BackColor = colours.green
        sender.Text = "✓"

        completedPanel.Controls.Remove(sender.Parent)
        activePanel.Controls.Add(sender.Parent)

        Return True
    End Function

End Class
```

If not completed

If completed but not deleted.

Mark task as deleted

Remove elemen
from parent
control

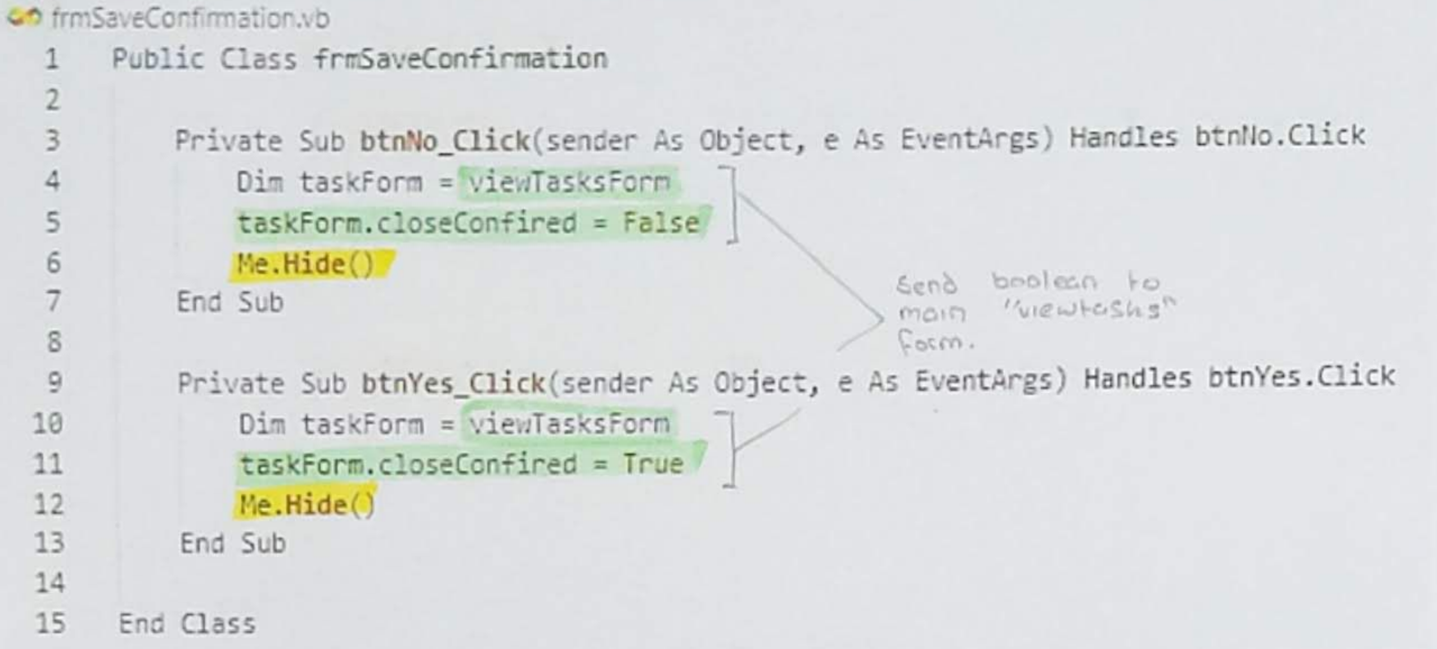
Shuffle task
element positions

reverse previous
dynamic changes

Undo task
marked as
completed. 1

1 - Deletion cannot be undone

Code For Save Confirmation Form

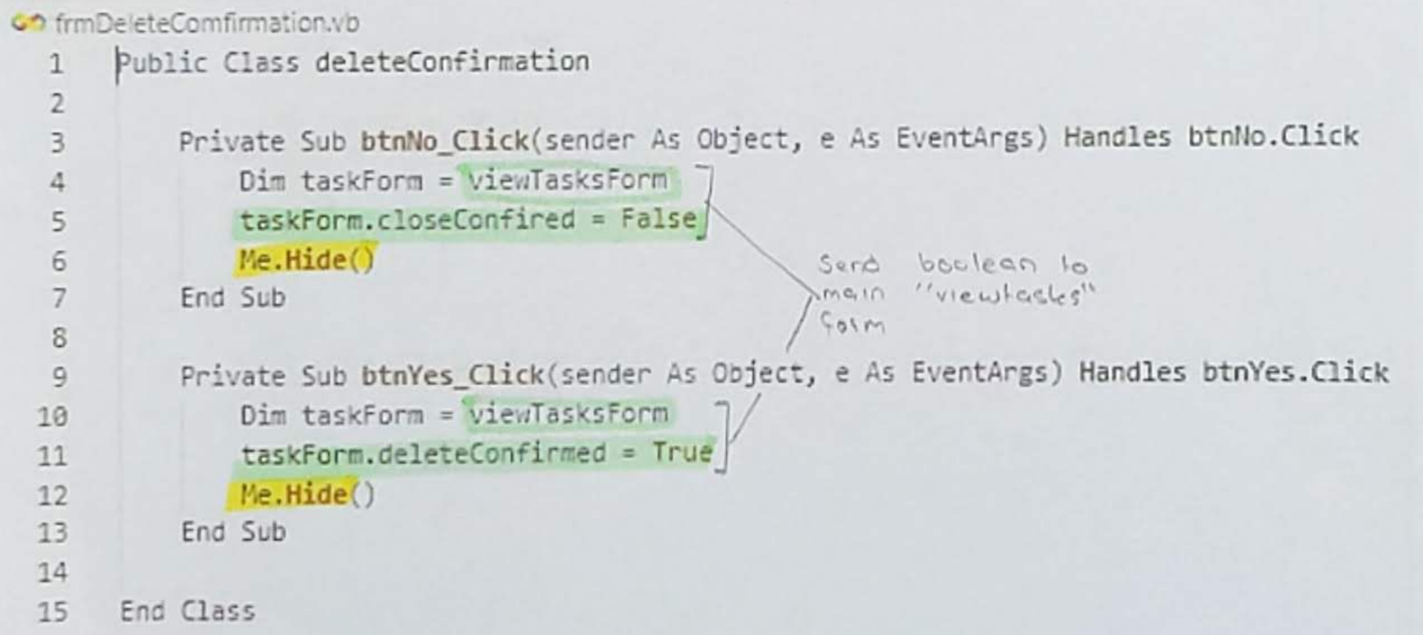


frmSaveConfirmation.vb

```
Public Class frmSaveConfirmation

    Private Sub btnNo_Click(sender As Object, e As EventArgs) Handles btnNo.Click
        Dim taskForm = viewTasksForm
        taskForm.closeConfired = False
        Me.Hide()
    End Sub

    Private Sub btnYes_Click(sender As Object, e As EventArgs) Handles btnYes.Click
        Dim taskForm = viewTasksForm
        taskForm.closeConfired = True
        Me.Hide()
    End Sub

End Class
```

Send boolean to main "viewtasks" form.

Code For Delete Confirmation Form



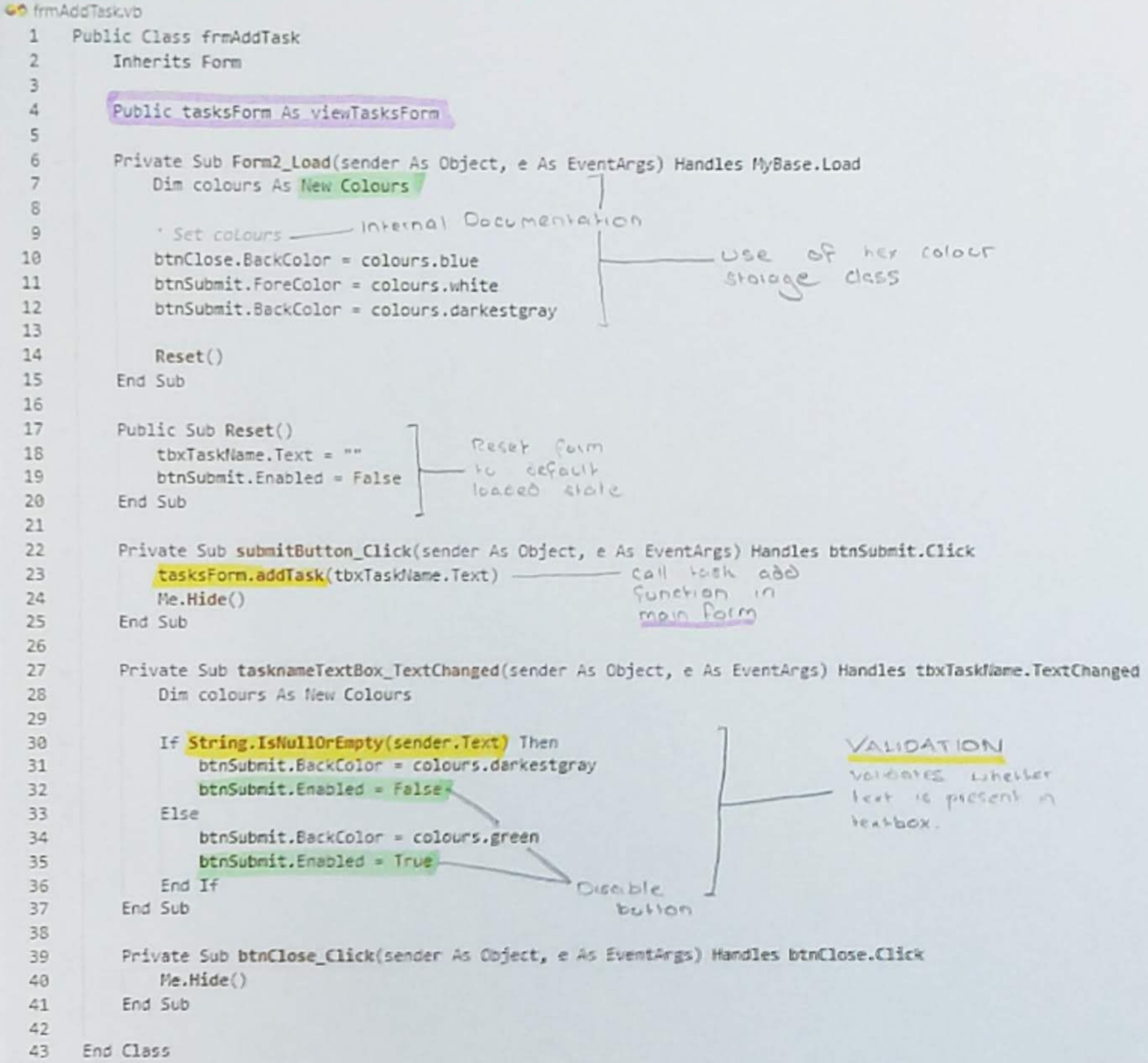
frmDeleteComfirmation.vb

```
Public Class deleteConfirmation

    Private Sub btnNo_Click(sender As Object, e As EventArgs) Handles btnNo.Click
        Dim taskForm = viewTasksForm
        taskForm.closeConfired = False
        Me.Hide()
    End Sub

    Private Sub btnYes_Click(sender As Object, e As EventArgs) Handles btnYes.Click
        Dim taskForm = viewTasksForm
        taskForm.deleteConfirmed = True
        Me.Hide()
    End Sub

End Class
```

Send boolean to main "viewtasks" form

🟨 Close form

Code For Task Add Form



frmAddTask.vb

```
Public Class frmAddTask
    Inherits Form

    Public tasksForm As viewTasksForm

    Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim colours As New Colours

        ' Set colours
        btnClose.BackColor = colours.blue
        btnSubmit.ForeColor = colours.white
        btnSubmit.BackColor = colours.darkestgray

        Reset()
    End Sub

    Public Sub Reset()
        tbxTaskName.Text = ""
        btnSubmit.Enabled = False
    End Sub

    Private Sub submitButton_Click(sender As Object, e As EventArgs) Handles btnSubmit.Click
        tasksForm.addTask(tbxTaskName.Text)
        Me.Hide()
    End Sub

    Private Sub tasknameTextBox_TextChanged(sender As Object, e As EventArgs) Handles tbxTaskName.TextChanged
        Dim colours As New Colours

        If String.IsNullOrEmpty(sender.Text) Then
            btnSubmit.BackColor = colours.darkestgray
            btnSubmit.Enabled = False
        Else
            btnSubmit.BackColor = colours.green
            btnSubmit.Enabled = True
        End If
    End Sub

    Private Sub btnClose_Click(sender As Object, e As EventArgs) Handles btnClose.Click
        Me.Hide()
    End Sub

End Class
```

Internal Documentation

Use of hex colour storage class

Reset form to default loaded state

call task add function in main form

VALIDATION

validates whether text is present in textbox.

Disable button