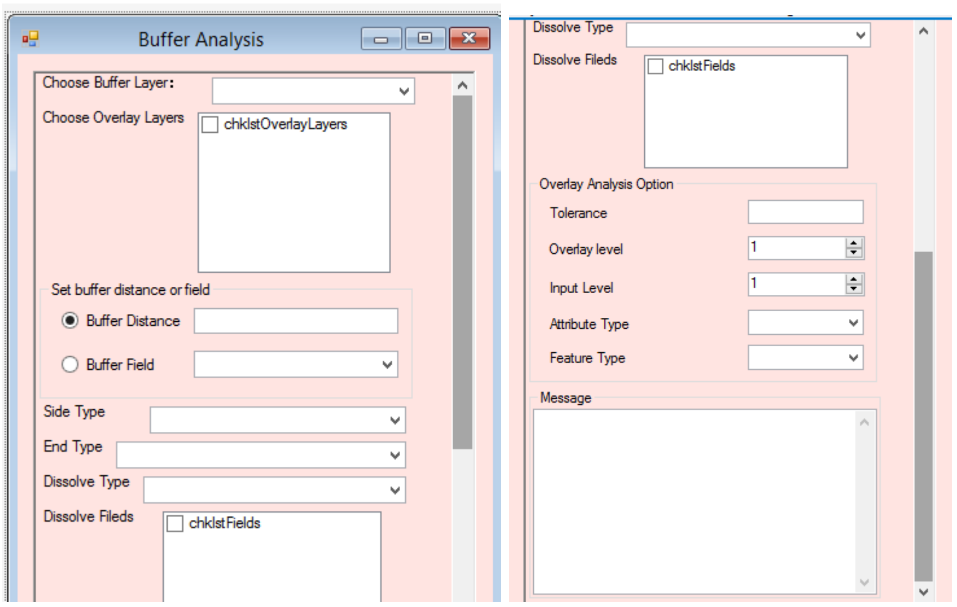
Chapter 12 Buffer Analysis

1. Add a Windows Form "BufferAnalysis" to the project:



GUI implementation detail could be found at "BufferAnalysis.Designer.cs" file.

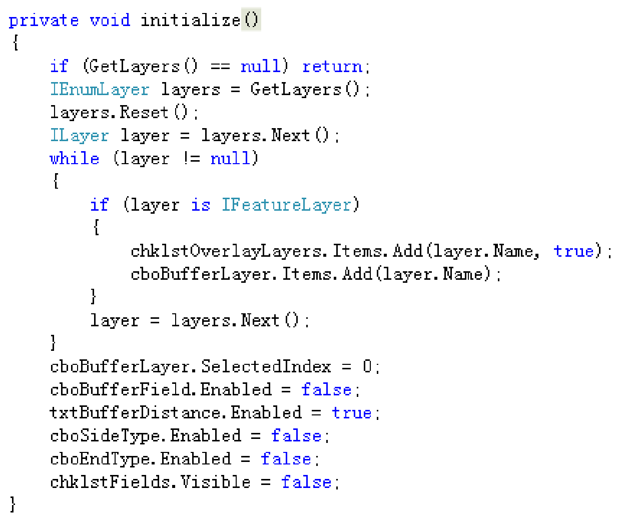
2. Implementation of BufferAnalysis:

- Class member and Construction method:

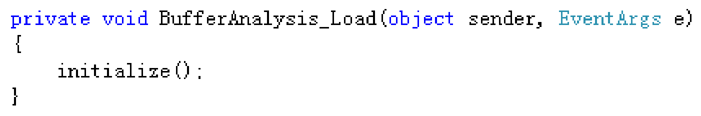
```
IHookHelper m_hookHelper = null;
IActiveView m_activeView = null;
IMap m_map = null;
public BufferAnalysis(IHookHelper hookHelper)
{
    InitializeComponent();
    if (hookHelper == null) return;
    m_hookHelper = hookHelper;
    m_activeView = m_hookHelper.ActiveView;
    m_map = m_hookHelper.FocusMap;
}
```

- Add GetLayers, initialize, and Load event of form:

```
private IEnumLayer GetLayers()
{
    UID uid = new UIDClass();
    uid.Value = "{40A9E885-5533-11d0-98BE-00805F7CED21}";
    if (m_map.LayerCount != 0)
    {
        IEnumLayer layers = m_map.get_Layers(uid, true);
        return layers;
    }
    return null;
}
```

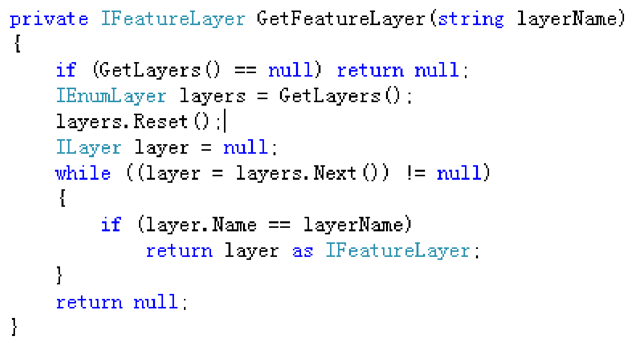


```
private void initialize()
{
    if (GetLayers() == null) return;
    IEnumLayer layers = GetLayers();
    layers.Reset();
    ILayer layer = layers.Next();
    while (layer != null)
    {
        if (layer is IFeatureLayer)
        {
            chklstOverlayLayers.Items.Add(layer.Name, true);
            cboBufferLayer.Items.Add(layer.Name);
        }
        layer = layers.Next();
    }
    cboBufferLayer.SelectedIndex = 0;
    cboBufferField.Enabled = false;
    txtBufferDistance.Enabled = true;
    cboSideType.Enabled = false;
    cboEndType.Enabled = false;
    chklstFields.Visible = false;
}
```

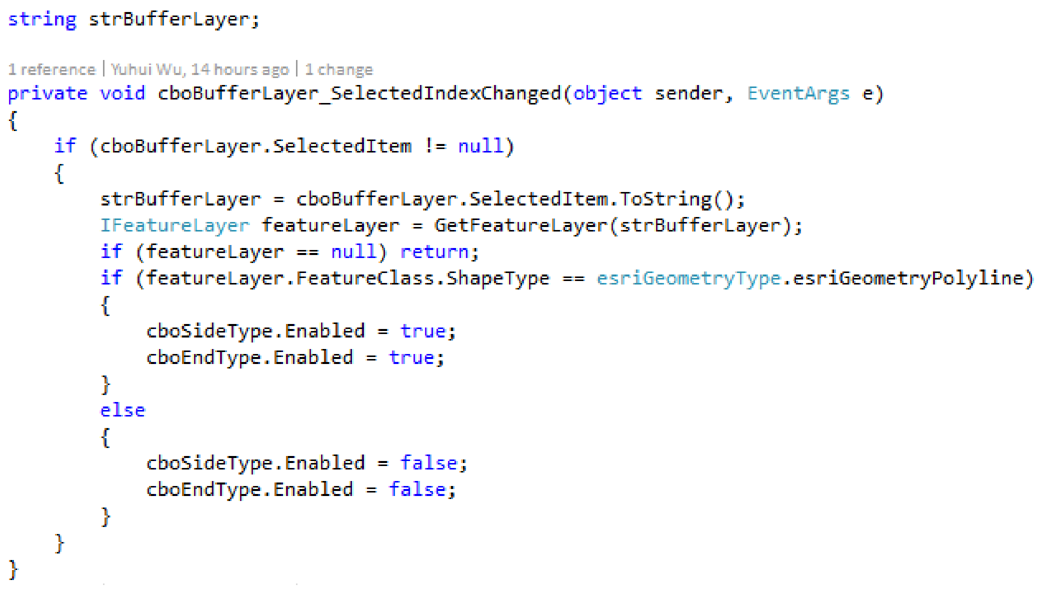


```
 private void BufferAnalysis_Load(object sender, EventArgs e)
 {
     initialize();
 }
```

- GetFeatureLayer and selectedIndexChanged event of cboBufferLayer:

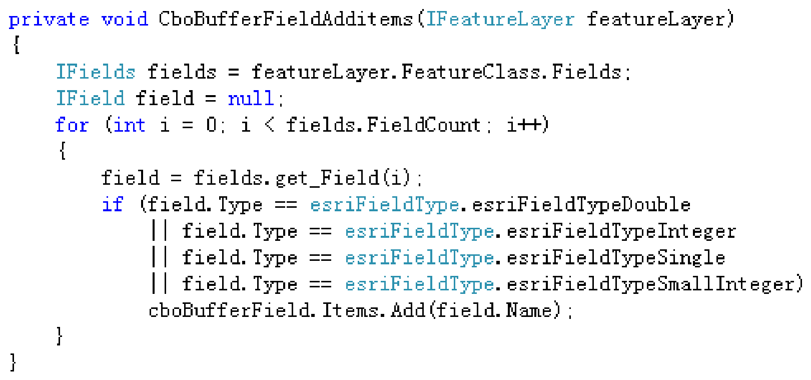


```
private IFeatureLayer GetFeatureLayer(string layerName)
{
    if (GetLayers() == null) return null;
    IEnumLayer layers = GetLayers();
    layers.Reset();
    ILayer layer = null;
    while ((layer = layers.Next()) != null)
    {
        if (layer.Name == layerName)
            return layer as IFeatureLayer;
    }
    return null;
}
```



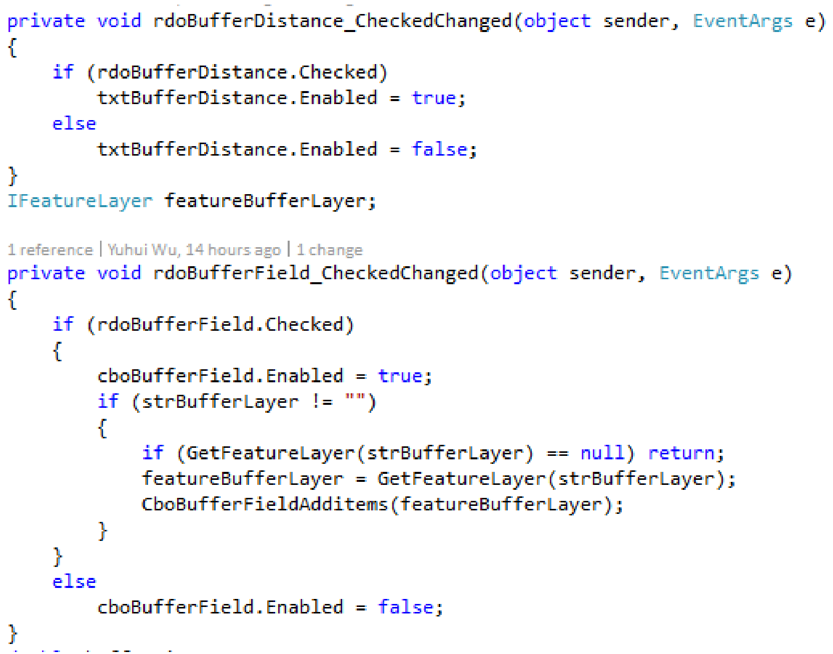
```
string strBufferLayer;

1 reference | Yuhui Wu, 14 hours ago | 1 change
private void cboBufferLayer_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cboBufferLayer.SelectedItem != null)
    {
        strBufferLayer = cboBufferLayer.SelectedItem.ToString();
        IFeatureLayer featureLayer = GetFeatureLayer(strBufferLayer);
        if (featureLayer == null) return;
        if (featureLayer.FeatureClass.ShapeType == esriGeometryType.esriGeometryPolyline)
        {
            cboSideType.Enabled = true;
            cboEndType.Enabled = true;
        }
        else
        {
            cboSideType.Enabled = false;
            cboEndType.Enabled = false;
        }
    }
}
```

- CBoBufferFieldAdditems:



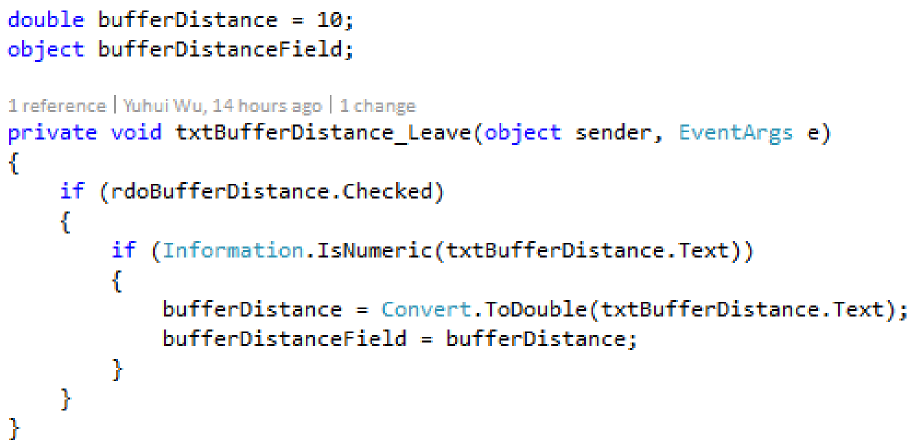
```
private void CboBufferFieldAdditems(IFeatureLayer featureLayer)
{
    IFields fields = featureLayer.FeatureClass.Fields;
    IField field = null;
    for (int i = 0; i < fields.FieldCount; i++)
    {
        field = fields.get_Field(i);
        if (field.Type == esriFieldType.esriFieldTypeDouble
            || field.Type == esriFieldType.esriFieldTypeInteger
            || field.Type == esriFieldType.esriFieldTypeSingle
            || field.Type == esriFieldType.esriFieldTypeSmallInteger)
            cboBufferField.Items.Add(field.Name);
    }
}
```

- CheckedChanged event of rdoBufferDistance and rdoBufferField:



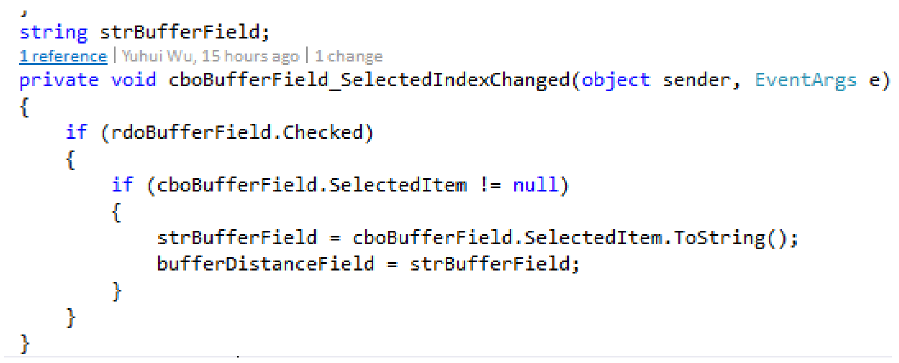
```
private void rdoBufferDistance_CheckedChanged(object sender, EventArgs e)
{
    if (rdoBufferDistance.Checked)
        txtBufferDistance.Enabled = true;
    else
        txtBufferDistance.Enabled = false;
}
IFeatureLayer featureBufferLayer;

1 reference | Yuhui Wu, 14 hours ago | 1 change
private void rdoBufferField_CheckedChanged(object sender, EventArgs e)
{
    if (rdoBufferField.Checked)
    {
        cboBufferField.Enabled = true;
        if (strBufferLayer != "")
        {
            if (GetFeatureLayer(strBufferLayer) == null) return;
            featureBufferLayer = GetFeatureLayer(strBufferLayer);
            CboBufferFieldAdditems(featureBufferLayer);
        }
    }
    else
        cboBufferField.Enabled = false;
}
```

- Leave event of txtBufferDistance:

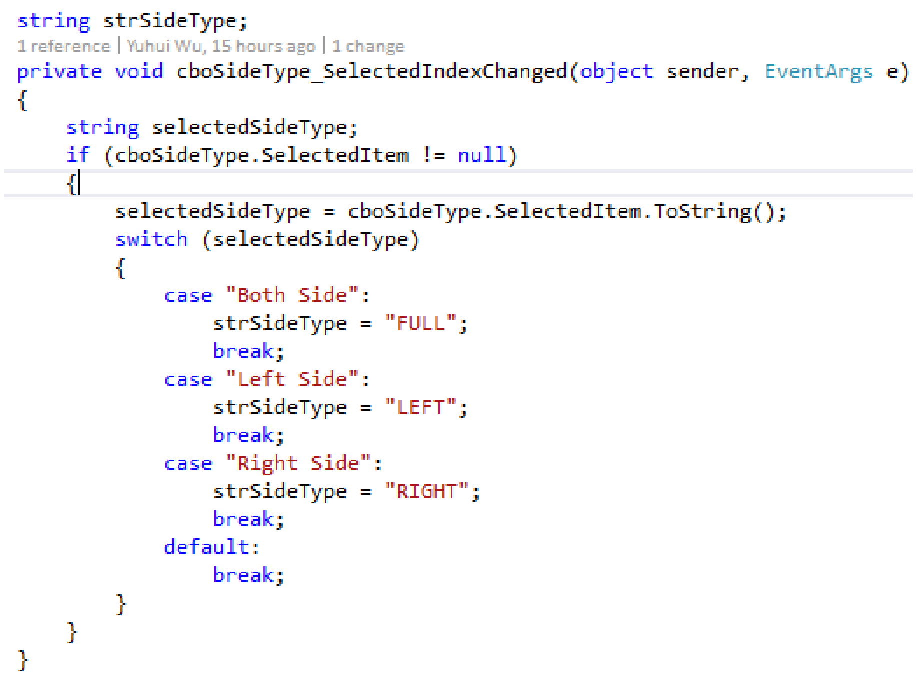


```
double bufferDistance = 10;
object bufferDistanceField;

1 reference | Yuhui Wu, 14 hours ago | 1 change
private void txtBufferDistance_Leave(object sender, EventArgs e)
{
    if (rdoBufferDistance.Checked)
    {
        if (Information.IsNumeric(txtBufferDistance.Text))
        {
            bufferDistance = Convert.ToDouble(txtBufferDistance.Text);
            bufferDistanceField = bufferDistance;
        }
    }
}
```

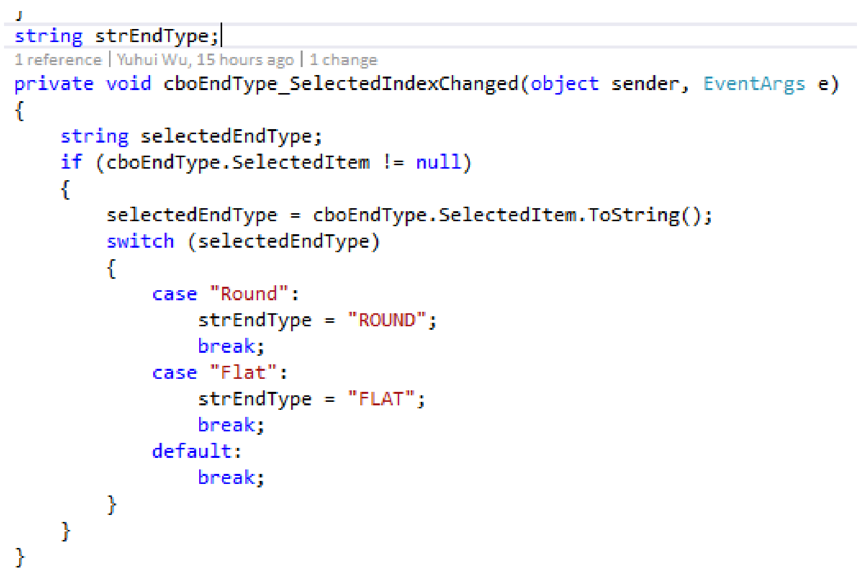
- SelectedIndexChanged event of cboBufferField, cboSideType, cboEndType:



```
string strBufferField;
1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cboBufferField_SelectedIndexChanged(object sender, EventArgs e)
{
    if (rdoBufferField.Checked)
    {
        if (cboBufferField.SelectedItem != null)
        {
            strBufferField = cboBufferField.SelectedItem.ToString();
            bufferDistanceField = strBufferField;
        }
    }
}
```

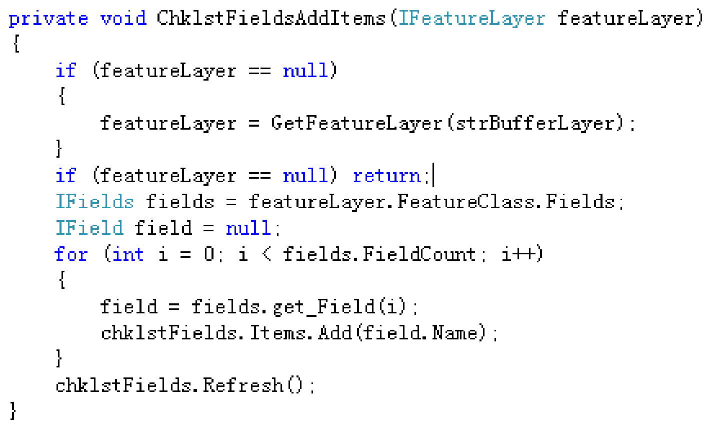


```
string strSideType;
1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cboSideType_SelectedIndexChanged(object sender, EventArgs e)
{
    string selectedSideType;
    if (cboSideType.SelectedItem != null)
    {
        selectedSideType = cboSideType.SelectedItem.ToString();
        switch (selectedSideType)
        {
            case "Both Side":
                strSideType = "FULL";
                break;
            case "Left Side":
                strSideType = "LEFT";
                break;
            case "Right Side":
                strSideType = "RIGHT";
                break;
            default:
                break;
        }
    }
}
```



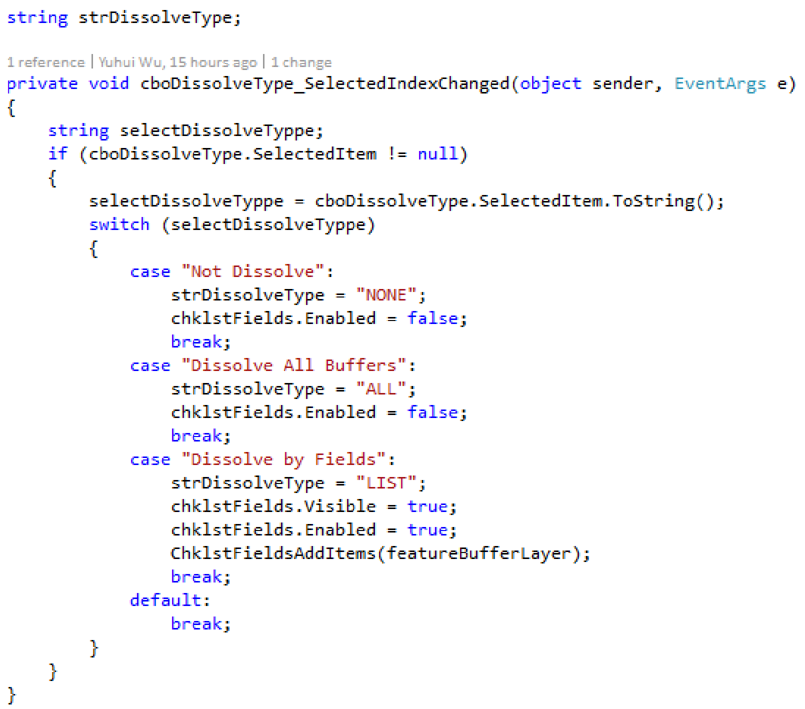
```
string strEndType;
1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cboEndType_SelectedIndexChanged(object sender, EventArgs e)
{
    string selectedEndType;
    if (cboEndType.SelectedItem != null)
    {
        selectedEndType = cboEndType.SelectedItem.ToString();
        switch (selectedEndType)
        {
            case "Round":
                strEndType = "ROUND";
                break;
            case "Flat":
                strEndType = "FLAT";
                break;
            default:
                break;
        }
    }
}
```

- ChklstFieldAddItems:



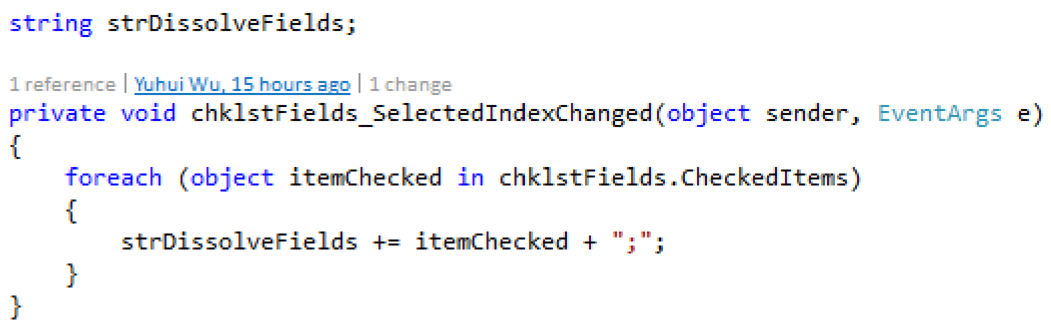
```
private void ChklstFieldsAddItems(IFeatureLayer featureLayer)
{
    if (featureLayer == null)
    {
        featureLayer = GetFeatureLayer(strBufferLayer);
    }
    if (featureLayer == null) return;
    IFields fields = featureLayer.FeatureClass.Fields;
    IField field = null;
    for (int i = 0; i < fields.FieldCount; i++)
    {
        field = fields.get_Field(i);
        chklstFields.Items.Add(field.Name);
    }
    chklstFields.Refresh();
}
```

- SelectedIndexChanged event of cboDissolveType:



```
string strDissolveType;

1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cboDissolveType_SelectedIndexChanged(object sender, EventArgs e)
{
    string selectDissolveTyppe;
    if (cboDissolveType.SelectedItem != null)
    {
        selectDissolveTyppe = cboDissolveType.SelectedItem.ToString();
        switch (selectDissolveTyppe)
        {
            case "Not Dissolve":
                strDissolveType = "NONE";
                chklstFields.Enabled = false;
                break;
            case "Dissolve All Buffers":
                strDissolveType = "ALL";
                chklstFields.Enabled = false;
                break;
            case "Dissolve by Fields":
                strDissolveType = "LIST";
                chklstFields.Visible = true;
                chklstFields.Enabled = true;
                ChklstFieldsAddItems(featureBufferLayer);
                break;
            default:
                break;
        }
    }
}
```

- SelectedIndexChanged event of chklstFields:



```
string strDissolveFields;

1 reference | Yuhui Wu, 15 hours ago | 1 change
private void chklstFields_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach (object itemChecked in chklstFields.CheckedItems)
    {
        strDissolveFields += itemChecked + ";";
    }
}
```

- Click Event of btnOutputPath:

```
string strOutputPath = System.IO.Path.GetTempPath();
private void btnOutpuPath_Click(object sender, EventArgs e)
{
    if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
    {
        strOutputPath = folderBrowserDialog1.SelectedPath;
    }
}
```

- IsDouble and Leave Event of txtTolerance:

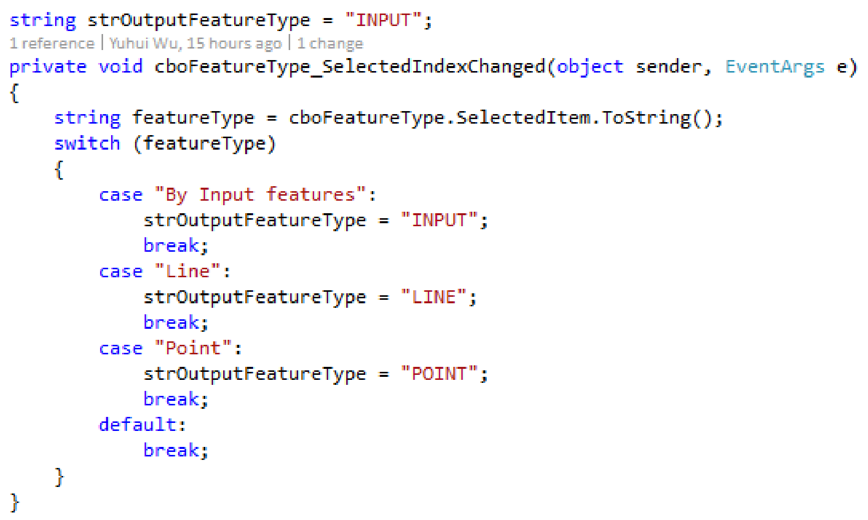
```
private bool IsDouble(string s)
{
    try
    {
        Double.Parse(s);
    }
    catch
    {
        return false;
    }
    return true;
}
double tolerance = 0.1;
private void txtTolerance_Leave(object sender, EventArgs e)
{
    if (IsDouble(txtTolerance.Text))
        tolerance = Convert.ToDouble(txtTolerance.Text);
}
```

- ValueChanged event of numUpDownOverlayLevel and numUpDownInputLevel

```
int overlayLevel = 1;
private void numUpDownOverlayLevel_ValueChanged(object sender, EventArgs e)
{
    overlayLevel = (int)numUpDownOverlayLevel.Value;
}
int inputLevel = 1;
private void numUpDownInputLevel_ValueChanged(object sender, EventArgs e)
{
    inputLevel = (int)numUpDownInputLevel.Value;
}
```

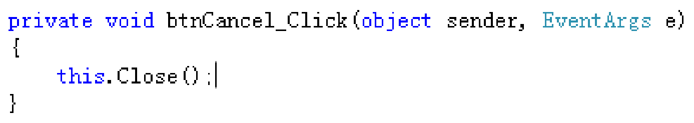
- SelectedIndexChanged Event of cboAttributeType and cboFeatureType:

```
string strJoinAttributeType = "ALL";
1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cboAttributeType_SelectedIndexChanged(object sender, EventArgs e)
{
    string attributeType = cboAttributeType.SelectedItem.ToString();
    switch (attributeType)
    {
        case "All Attributes":
            strJoinAttributeType = "ALL";
            break;
        case "Not Include FID":
            strJoinAttributeType = "NO_FID";
            break;
        case "Include only FID":
            strJoinAttributeType = "ONLY_FID";
            break;
        default:
            break;
    }
}
```



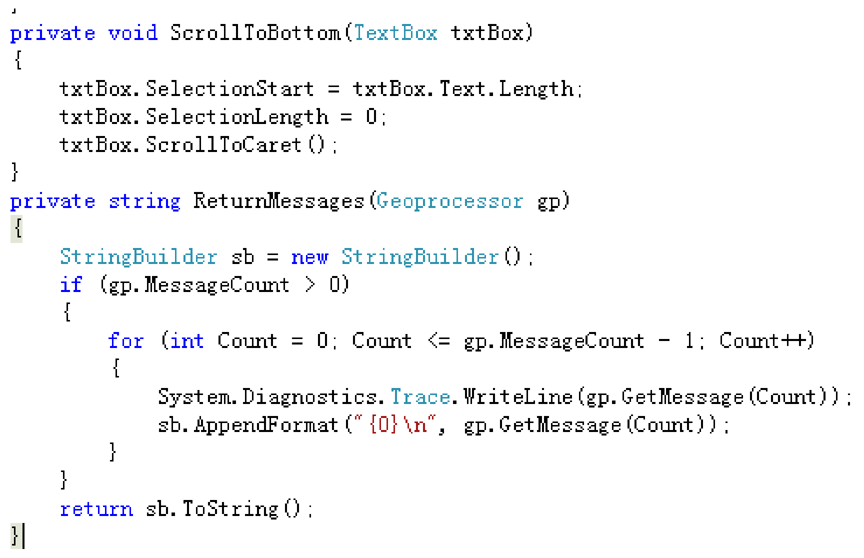
```
string strOutputFeatureType = "INPUT";
1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cboFeatureType_SelectedIndexChanged(object sender, EventArgs e)
{
    string featureType = cboFeatureType.SelectedItem.ToString();
    switch (featureType)
    {
        case "By Input features":
            strOutputFeatureType = "INPUT";
            break;
        case "Line":
            strOutputFeatureType = "LINE";
            break;
        case "Point":
            strOutputFeatureType = "POINT";
            break;
        default:
            break;
    }
}
```

- Click Event of “Cancel”:



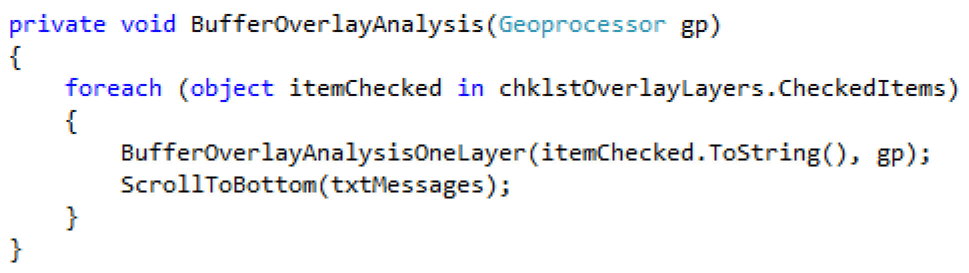
```
private void btnCancel_Click(object sender, EventArgs e)
{
    this.Close();
}
```

- ScrollToBottom and ReturnMessages

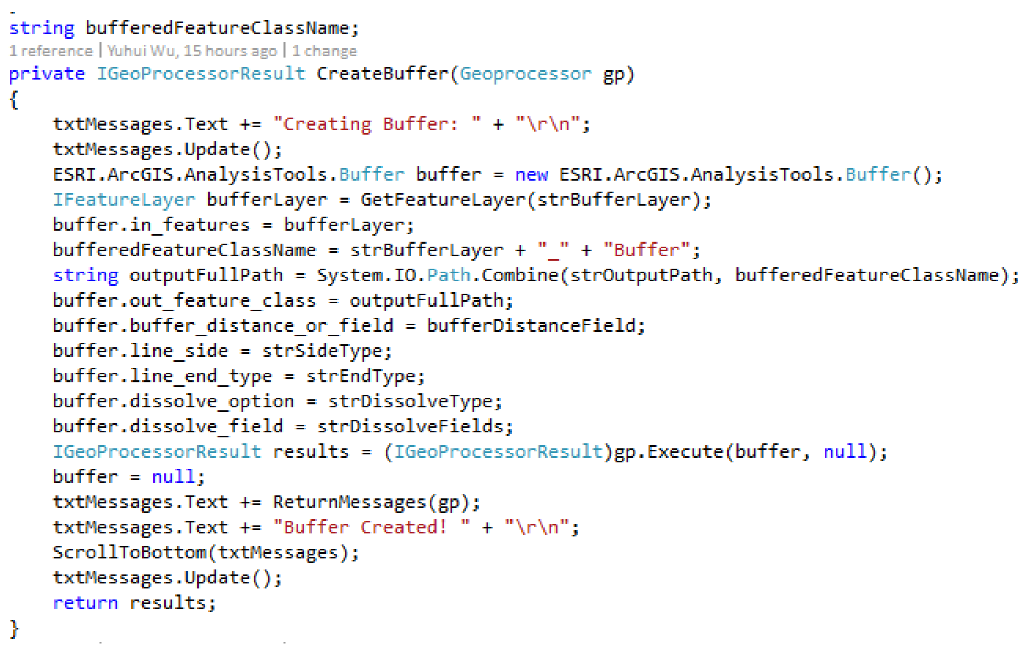


```
private void ScrollToBottom(TextBox txtBox)
{
    txtBox.SelectionStart = txtBox.Text.Length;
    txtBox.SelectionLength = 0;
    txtBox.ScrollToCaret();
}
private string ReturnMessages(Geoprocessor gp)
{
    StringBuilder sb = new StringBuilder();
    if (gp.MessageCount > 0)
    {
        for (int Count = 0; Count <= gp.MessageCount - 1; Count++)
        {
            System.Diagnostics.Trace.WriteLine(gp.GetMessage(Count));
            sb.AppendFormat("{0}\n", gp.GetMessage(Count));
        }
    }
    return sb.ToString();
}
```

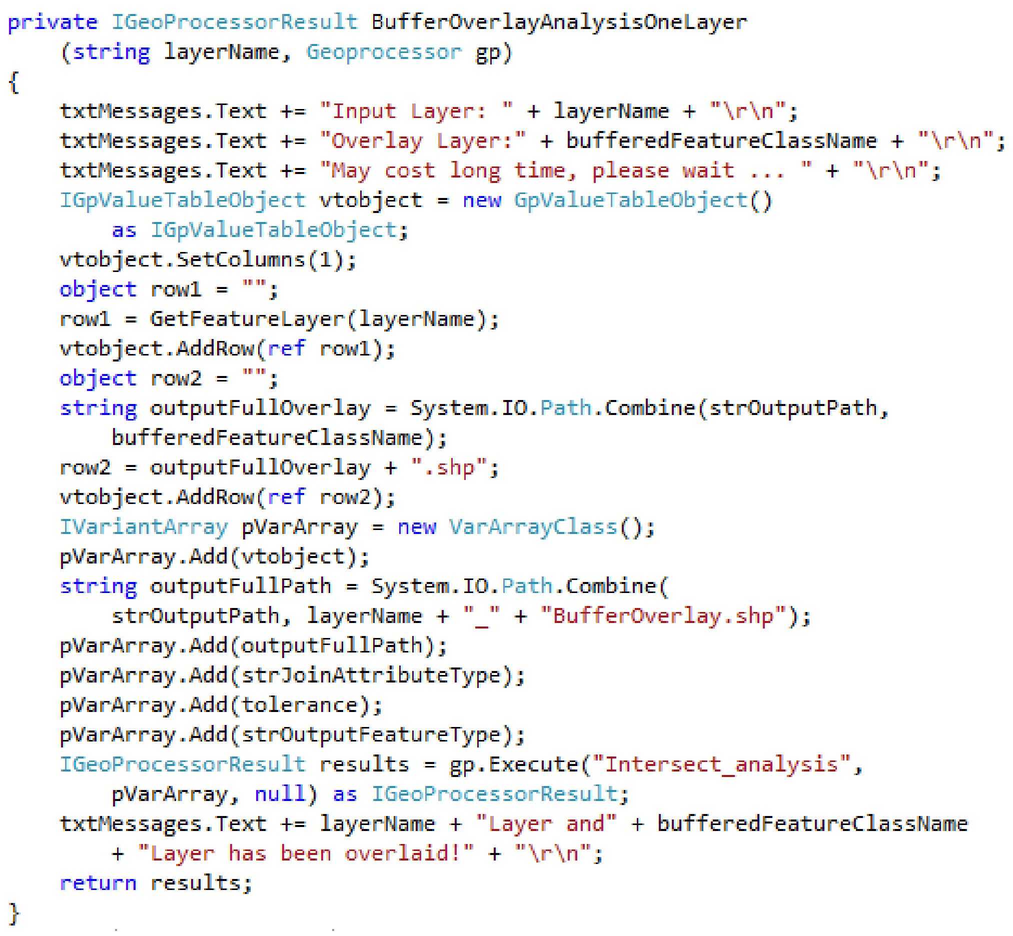
- CreateBuffer, `BufferOverlayAnalysisOneLayer, and BufferOverlayAnalysis:`



```
private void BufferOverlayAnalysis(Geoprocessor gp)
{
    foreach (object itemChecked in chklstOverlayLayers.CheckedItems)
    {
        BufferOverlayAnalysisOneLayer(itemChecked.ToString(), gp);
        ScrollToBottom(txtMessages);
    }
}
```



```
string bufferedFeatureClassName;
1 reference | Yuhui Wu, 15 hours ago | 1 change
private IGeoProcessorResult CreateBuffer(Geoprocessor gp)
{
    txtMessages.Text += "Creating Buffer: " + "\r\n";
    txtMessages.Update();
    ESRI.ArcGIS.AnalysisTools.Buffer buffer = new ESRI.ArcGIS.AnalysisTools.Buffer();
    IFeatureLayer bufferLayer = GetFeatureLayer(strBufferLayer);
    buffer.in_features = bufferLayer;
    bufferedFeatureClassName = strBufferLayer + "_" + "Buffer";
    string outputFullPath = System.IO.Path.Combine(strOutputPath, bufferedFeatureClassName);
    buffer.out_feature_class = outputFullPath;
    buffer.buffer_distance_or_field = bufferDistanceField;
    buffer.line_side = strSideType;
    buffer.line_end_type = strEndType;
    buffer.dissolve_option = strDissolveType;
    buffer.dissolve_field = strDissolveFields;
    IGeoProcessorResult results = (IGeoProcessorResult)gp.Execute(buffer, null);
    buffer = null;
    txtMessages.Text += ReturnMessages(gp);
    txtMessages.Text += "Buffer Created! " + "\r\n";
    ScrollToBottom(txtMessages);
    txtMessages.Update();
    return results;
}
```



```
private IGeoProcessorResult BufferOverlayAnalysisOneLayer
    (string layerName, Geoprocessor gp)
{
    txtMessages.Text += "Input Layer: " + layerName + "\r\n";
    txtMessages.Text += "Overlay Layer:" + bufferedFeatureClassName + "\r\n";
    txtMessages.Text += "May cost long time, please wait ... " + "\r\n";
    IGpValueTableObject vtobject = new GpValueTableObject()
        as IGpValueTableObject;
    vtobject.SetColumns(1);
    object row1 = "";
    row1 = GetFeatureLayer(layerName);
    vtobject.AddRow(ref row1);
    object row2 = "";
    string outputFullOverlay = System.IO.Path.Combine(strOutputPath,
        bufferedFeatureClassName);
    row2 = outputFullOverlay + ".shp";
    vtobject.AddRow(ref row2);
    IVariantArray pVarArray = new VarArrayClass();
    pVarArray.Add(vtobject);
    string outputFullPath = System.IO.Path.Combine(
        strOutputPath, layerName + "_" + "BufferOverlay.shp");
    pVarArray.Add(outputFullPath);
    pVarArray.Add(strJoinAttributeType);
    pVarArray.Add(tolerance);
    pVarArray.Add(strOutputFeatureType);
    IGeoProcessorResult results = gp.Execute("Intersect_analysis",
        pVarArray, null) as IGeoProcessorResult;
    txtMessages.Text += layerName + "Layer and" + bufferedFeatureClassName
        + "Layer has been overlaid!" + "\r\n";
    return results;
}
```

- Click Event of “Buffer Analysis”:

```
private void btnBufferAnalysis_Click(object sender, EventArgs e)
{
    txtMessages.Text += "Begin Buffer Analysis,Please wait... " + "\r\n";
    txtMessages.Text += DateTime.Now.ToString() + "\r\n";
    txtMessages.Update();
    Geoprocessor gp = new Geoprocessor();
    gp.OverwriteOutput = true;
    gp.AddOutputsToMap = true;
    IGeoProcessorResult results = CreateBuffer(gp);
    if ((results != null) && (results.Status == esriJobStatus.esriJobSucceeded))
    {
        BufferOverlayAnalysis(gp);
        txtMessages.Text += "Buffer Analysis Finished!" + "\r\n";
        txtMessages.Text += DateAndTime.Now.ToString() + "\r\n";
        ScrollToBottom(txtMessages);
        txtMessages.Update();
    }
    gp = null;
}
```

3. Add A Base Command Class BufferAnalysisCmd, and implement OnClick method:

```
public override void OnClick()
{
    // TODO: Add BufferAnalysisCmd.OnClick implementation
    if (m_hookHelper == null) return;
    if (m_hookHelper.FocusMap.LayerCount > 0)
    {
        BufferAnalysis bufferAnalysis = new BufferAnalysis(m_hookHelper);
        bufferAnalysis.Show(m_hookHelper as System.Windows.Forms.IWin32Window);
    }
}
```

4. Back to MainForm, add a menu content and its click event:

```
private void bufferAnalysisToolStripMenuItem_Click(object sender, EventArgs e)
{
    ICommand command = new BufferAnalysisCmd();
    command.OnCreate(m_mapControl.Object);
    command.OnClick();
}
```