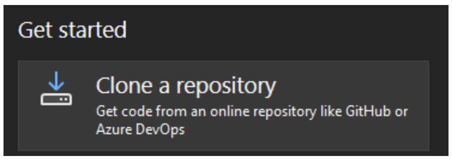
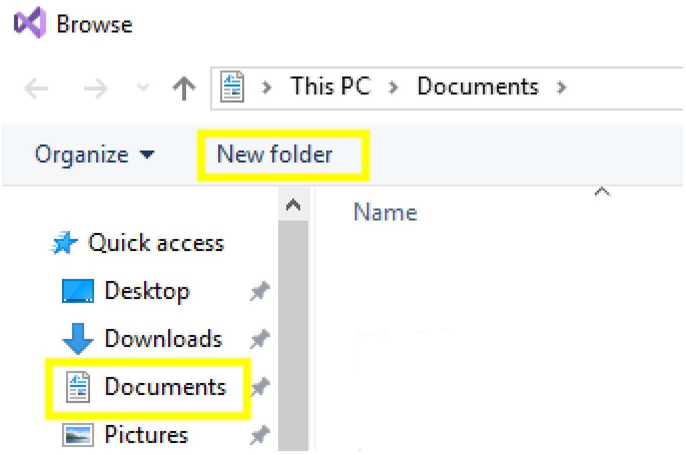
1. Open Visual Studio
2. Select 'Clone a Repository'

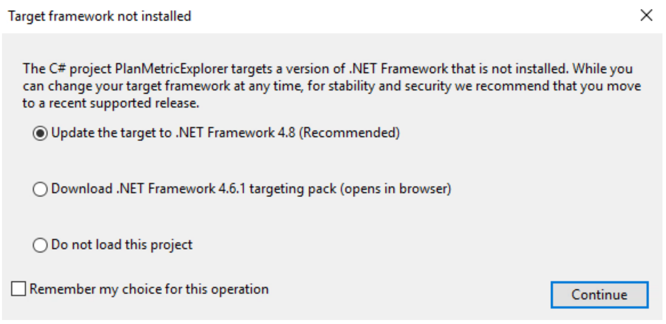

Get started

Clone a repository
Get code from an online repository like GitHub or Azure DevOps

3. Copy and paste the following URL
   https://github.com/Gateway-Scripts/PlanMetricExplorer
4. Press the Browse Button, and browse to the Documents Folder and select New Folder. Enter the name PlanStats and press Select Folder



5. Press Clone
6. Press Continue if prompted to update the target framework


Target framework not installed

The C# project PlanMetricExplorer targets a version of .NET Framework that is not installed. While you can change your target framework at any time, for stability and security we recommend that you move to a recent supported release.

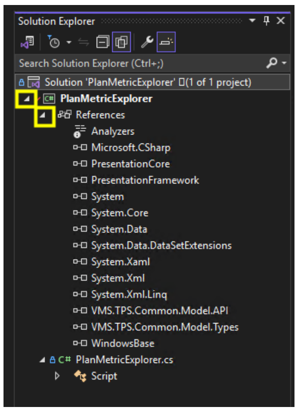
[✓ Update the target to .NET Framework 4.8 (Recommended)]

Download .NET Framework 4.6.1 targeting pack (opens in browser)

Do not load this project

☐ Remember my choice for this operation

7. In the Solution Explorer, expand the File structure to view the projects references and the PlanMetricExplorer Script file

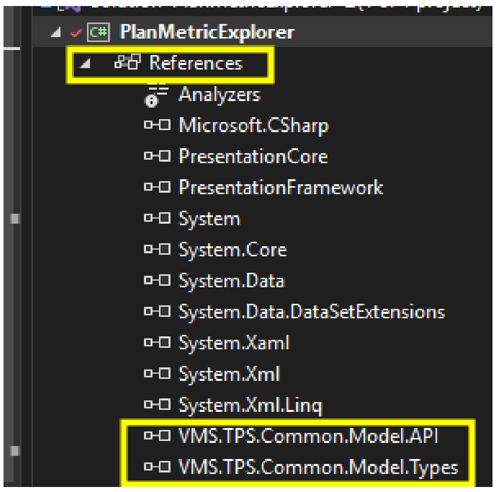


```
Solution 'PlanMetricExplorer' (1 of 1 project)
◢ PlanMetricExplorer
   ◢ References
        Analyzers
        Microsoft.CSharp
        PresentationCore
        PresentationFramework
        System
        System.Core
        System.Data
        System.Data.DataSetExtensions
        System.Xaml
        System.Xml
        System.Xml.Linq
        VMS.TPS.Common.Model.API
        VMS.TPS.Common.Model.Types
        WindowsBase
   ◢ C# PlanMetricExplorer.cs
      ▷ Script
```

8. Double click on the PlanMetricExplorer.cs file

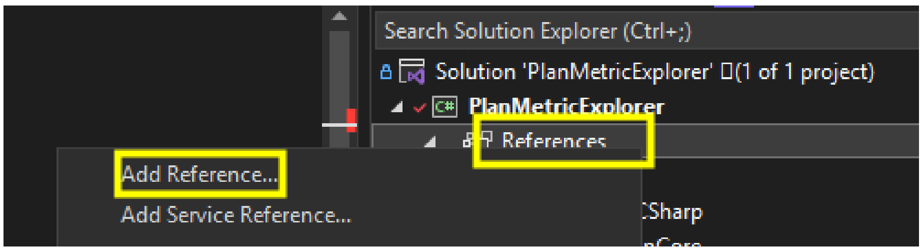
9. First, with any scipt downloaded, we need to ensure the references to the Eclipse API are for our version.

10. Expand References, and delete the references VMS.TPS.Common.Model.API and VMS.TPS.Common.Model.Types



PlanMetricExplorer
References
Analyzers
Microsoft.CSharp
PresentationCore
PresentationFramework
System
System.Core
System.Data
System.Data.DataSetExtensions
System.Xaml
System.Xml
System.Xml.Linq
VMS.TPS.Common.Model.API
VMS.TPS.Common.Model.Types

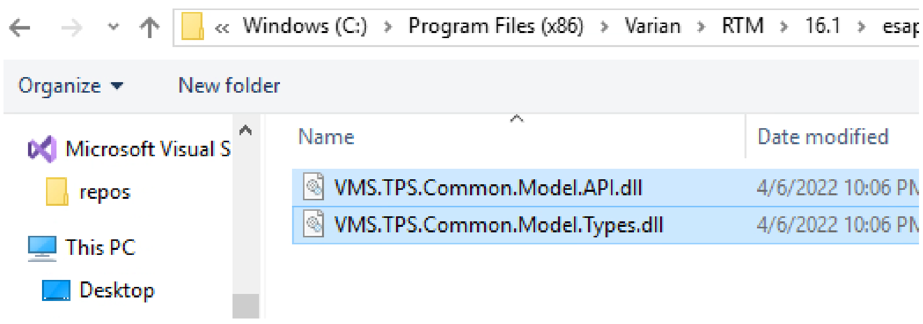
11. Rt Click on References and select Add References



Search Solution Explorer (Ctrl+;)
Solution 'PlanMetricExplorer' (1 of 1 project)
PlanMetricExplorer
References
Add Reference...
Add Service Reference...

12. Select Browse

13. Navigate to C:\Program Files (x86)\Varian\RTM\16.1\esapi\API and select "VMS.TPS.Common.Model.Types.dll" and "VMS.TPS.Common.Model.API.dll"



« Windows (C:) › Program Files (x86) › Varian › RTM › 16.1 › esap
Organize
New folder
Microsoft Visual S
repos
This PC
Desktop

Name	Date modified
VMS.TPS.Common.Model.API.dll	4/6/2022 10:06 PM
VMS.TPS.Common.Model.Types.dll	4/6/2022 10:06 PM

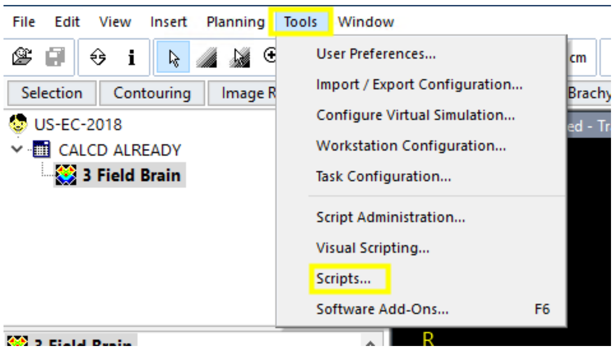
14. Click Add then Ok

15. Press Ctrl-B to "Build" the project

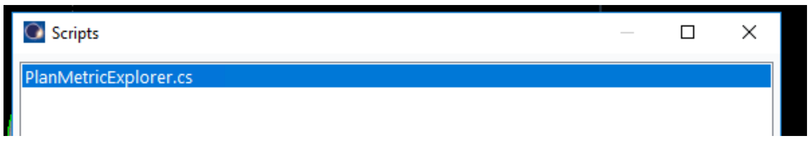
16. Open ARIA and navigate to External Beam Planning

17. Open Patient with calcualted plan and structures

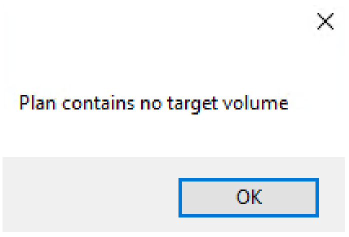
18. Select Tools, then Scripts..



19. Select Folder then Change Folder and Navigate to Documents – PlanStats and select Open. The PlanMetricExplorer.cs file should be available



20. A pop-up with Plan Contains no target volume appears…..why??Press ok and then close the next pop-up


Plan contains no target volume

21. Return to Visual Studio and scroll to line 39. It reads

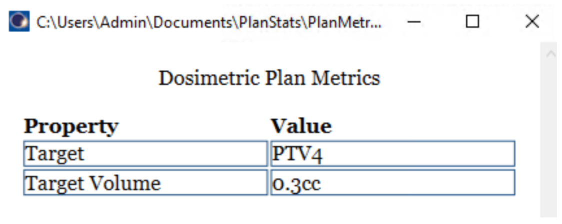
```
Structure target = plan.StructureSet.Structures.FirstOrDefault(st => st.Id.Equals(plan.TargetVolumeID));
```

22. Change the last part to st => st.Id.Contains("PTV") so that the structures we are interested in have the letters PTV in the id field.

```
Structure target = plan.StructureSet.Structures.FirstOrDefault(st => st.Id.Contains("PTV"));
```

23. Press Ctrl-B to "Build" the project. Return to Eclipse and run Script.

24. Now we get something!!!! But only one PTV is listed, why? Close the dialog box

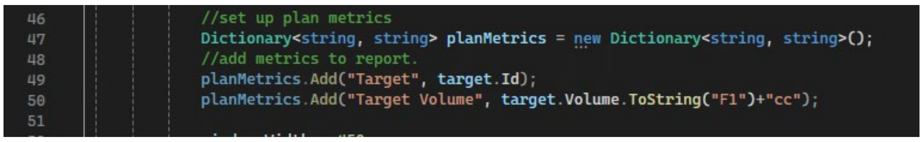


Dosimetric Plan Metrics

Property	Value
Target	PTV4
Target Volume	0.3cc

25. Return to Visual Studio and scroll to line 39, it is only looking at the FirstOrDefault structure with "PTV" in the id

26. Let’s see what else we can add to the list, scroll to line 47 -> 50, this is where the plan metrics are added to the list



```
46              //set up plan metrics
47              Dictionary<string, string> planMetrics = new Dictionary<string, string>();
48              //add metrics to report.
49              planMetrics.Add("Target", target.Id);
50              planMetrics.Add("Target Volume", target.Volume.ToString("F1")+"cc");
51
```

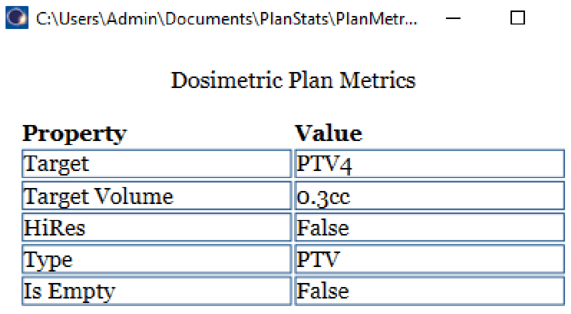
27. Add the following to the script after line 50

planMetrics.Add("HiRes", target.IsHighResolution.ToString());

planMetrics.Add("Type", target.DicomType);

planMetrics.Add("Is Empty", target.IsEmpty.ToString());

28. Press Ctrl-B to “Build” the project. Return to Eclipse and run Script



C:\Users\Admin\Documents\PlanStats\PlanMetr...

Dosimetric Plan Metrics

Property	Value
Target	PTV4
Target Volume	0.3cc
HiRes	False
Type	PTV
Is Empty	False

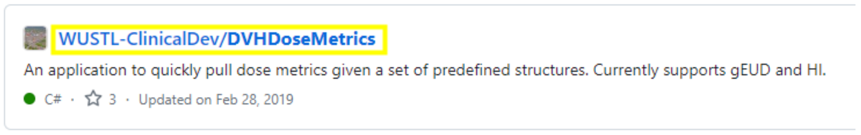
29. What about Homogeneity Index(HI)..that would be nice!! But it’s not there, what can we do?

30. Open Chrome/Edge and navigate to https://github.com/ (or follow the link)

31. In the search bar, search for dvhdosemetrics or follow this link

https://github.com/WUSTL-ClinicalDev/DVHDoseMetrics

32. Click on WUSTL-ClinicalDev/DVHMetrics



WUSTL-ClinicalDev/DVHDoseMetrics

An application to quickly pull dose metrics given a set of predefined structures. Currently supports gEUD and HI.

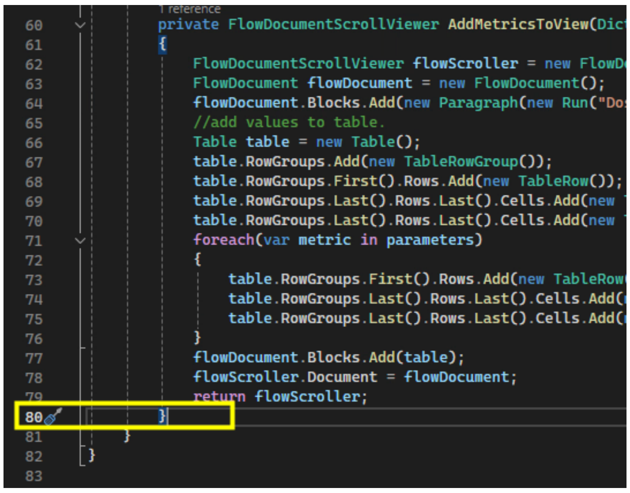
● C# · ☆ 3 · Updated on Feb 28, 2019

33. Select Plugins -> DVHDoseMetrics.cs

DVHDoseMetrics / Plugins / DVHDoseMetrics.cs

34. Scroll to line 148 and copy lines 148 -> 202. This is a Method in C# to calculate HI.

35. Return to Visual Studio and scroll to line 80(third bottom line) and press enter, then paste the previously copied text



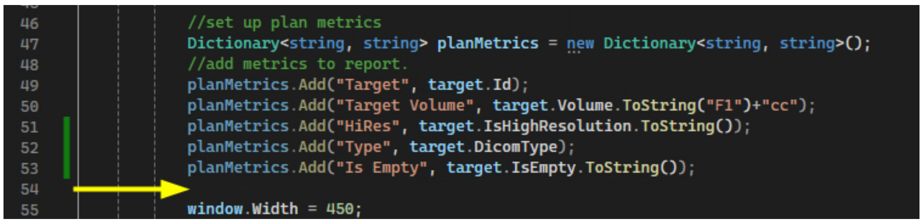
```
        1 reference
60      private FlowDocumentScrollViewer AddMetricsToView(Dic
61      {
62          FlowDocumentScrollViewer flowScroller = new FlowD
63          FlowDocument flowDocument = new FlowDocument();
64          flowDocument.Blocks.Add(new Paragraph(new Run("Do
65          //add values to table.
66          Table table = new Table();
67          table.RowGroups.Add(new TableRowGroup());
68          table.RowGroups.First().Rows.Add(new TableRow());
69          table.RowGroups.Last().Rows.Last().Cells.Add(new
70          table.RowGroups.Last().Rows.Last().Cells.Add(new
71          foreach(var metric in parameters)
72          {
73              table.RowGroups.First().Rows.Add(new TableRow
74              table.RowGroups.Last().Rows.Last().Cells.Add(
75              table.RowGroups.Last().Rows.Last().Cells.Add(
76          }
77          flowDocument.Blocks.Add(table);
78          flowScroller.Document = flowDocument;
79          return flowScroller;
80      }
81  }
82  }
83
```

36. This CalculateHI Method "looks" for 2 parmameters……

a. Structure s -> This is the structure we want to calculate the HI

b. PlanningItem pi -> This is the plan or plansum etc

37. After line 54, add the following:



```
46      //set up plan metrics
47      Dictionary<string, string> planMetrics = new Dictionary<string, string>();
48      //add metrics to report.
49      planMetrics.Add("Target", target.Id);
50      planMetrics.Add("Target Volume", target.Volume.ToString("F1")+"cc");
51      planMetrics.Add("HiRes", target.IsHighResolution.ToString());
52      planMetrics.Add("Type", target.DicomType);
53      planMetrics.Add("Is Empty", target.IsEmpty.ToString());
54
55      window.Width = 450;
```

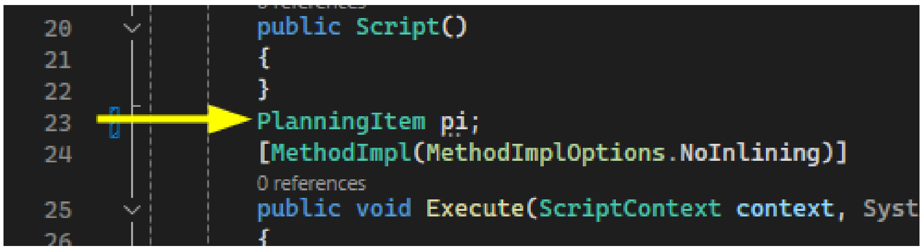
planMetrics.Add("HI", CalculateHI(target, pi).ToString());

38. Notice our variable "pi" has a red underline. Why is this? We have not told the script what "pi" is. In the Github webpage, notice on line 79 there is a line that reads:

PlanningItem pi;

39. We need to add this to our script to remove the error. In the Github script the line is added between the public Script() and public void Execute() methods. We need to do the same.

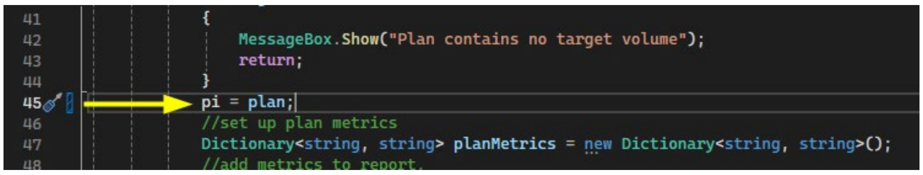
40. Navigate to line 23 and press enter then enter PlanningItem pi; Notice the red error is gone.



```
20      public Script()
21      {
22      }
23      PlanningItem pi;
24      [MethodImpl(MethodImplOptions.NoInlining)]
        0 references
25      public void Execute(ScriptContext context, Syst
26      {
```

41. We also need to assign a value to the variable "pi". Our PlanningItem will be the plan, so add the following on line 45.

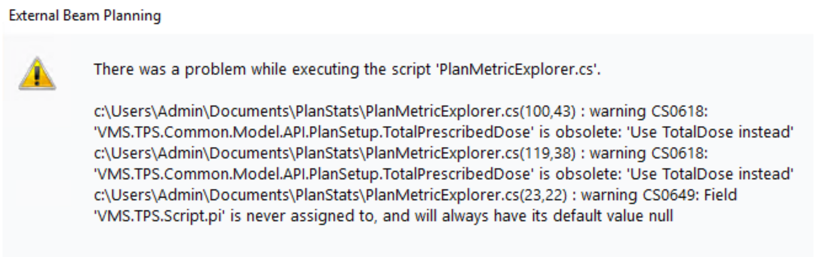
pi = plan;



```
41                  {
42                      MessageBox.Show("Plan contains no target volume");
43                      return;
44                  }
45                  pi = plan;
46                  //set up plan metrics
47                  Dictionary<string, string> planMetrics = new Dictionary<string, string>();
48                  //add metrics to report.
```

42. Press Ctrl-B to "Build" the project. Return to Eclipse and run Script

43. We get the following error…this is due to changes in the API since the creation of the original script.

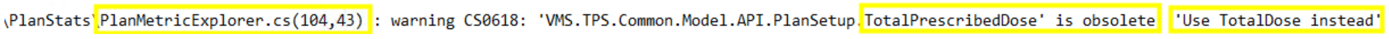


External Beam Planning

There was a problem while executing the script 'PlanMetricExplorer.cs'.

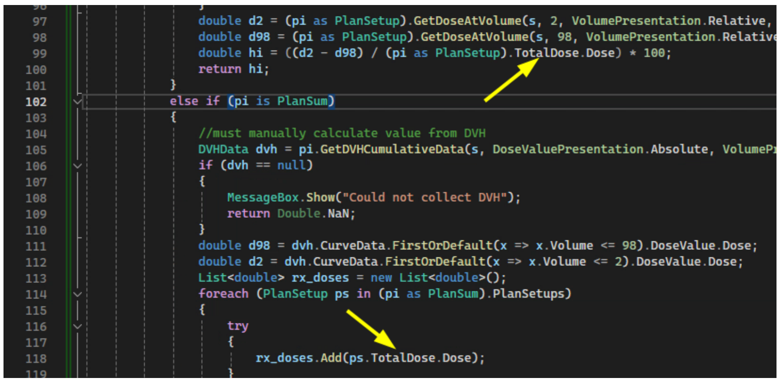
c:\Users\Admin\Documents\PlanStats\PlanMetricExplorer.cs(100,43) : warning CS0618: 'VMS.TPS.Common.Model.API.PlanSetup.TotalPrescribedDose' is obsolete: 'Use TotalDose instead'
c:\Users\Admin\Documents\PlanStats\PlanMetricExplorer.cs(119,38) : warning CS0618: 'VMS.TPS.Common.Model.API.PlanSetup.TotalPrescribedDose' is obsolete: 'Use TotalDose instead'
c:\Users\Admin\Documents\PlanStats\PlanMetricExplorer.cs(23,22) : warning CS0649: Field 'VMS.TPS.Script.pi' is never assigned to, and will always have its default value null

44. Luckily it tells us what to do, and the line/position of the error.



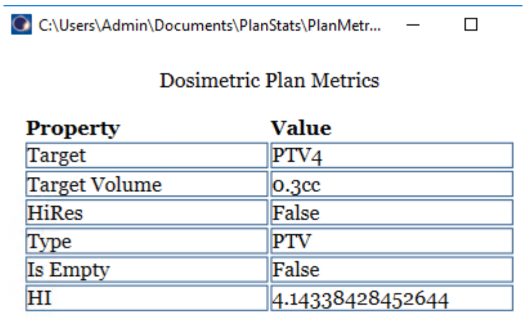
```
\PlanStats\PlanMetricExplorer.cs(104,43) : warning CS0618: 'VMS.TPS.Common.Model.API.PlanSetup.TotalPrescribedDose' is obsolete 'Use TotalDose instead'
```

45. Navigate to ~line 100 replace TotalPrescribedDose with TotalDose. Repeat for line 118.



```
97                  double d2 = (pi as PlanSetup).GetDoseAtVolume(s, 2, VolumePresentation.Relative,
98                  double d98 = (pi as PlanSetup).GetDoseAtVolume(s, 98, VolumePresentation.Relative
99                  double hi = ((d2 - d98) / (pi as PlanSetup).TotalDose.Dose) * 100;
100                 return hi;
101             }
102             else if (pi is PlanSum)
103             {
104                 //must manually calculate value from DVH
105                 DVHData dvh = pi.GetDVHCumulativeData(s, DoseValuePresentation.Absolute, VolumePr
106                 if (dvh == null)
107                 {
108                     MessageBox.Show("Could not collect DVH");
109                     return Double.NaN;
110                 }
111                 double d98 = dvh.CurveData.FirstOrDefault(x => x.Volume <= 98).DoseValue.Dose;
112                 double d2 = dvh.CurveData.FirstOrDefault(x => x.Volume <= 2).DoseValue.Dose;
113                 List<double> rx_doses = new List<double>();
114                 foreach (PlanSetup ps in (pi as PlanSum).PlanSetups)
115                 {
116                     try
117                     {
118                         rx_doses.Add(ps.TotalDose.Dose);
```

46. Press Ctrl-B to "Build" the project. Return to Eclipse and run Script.



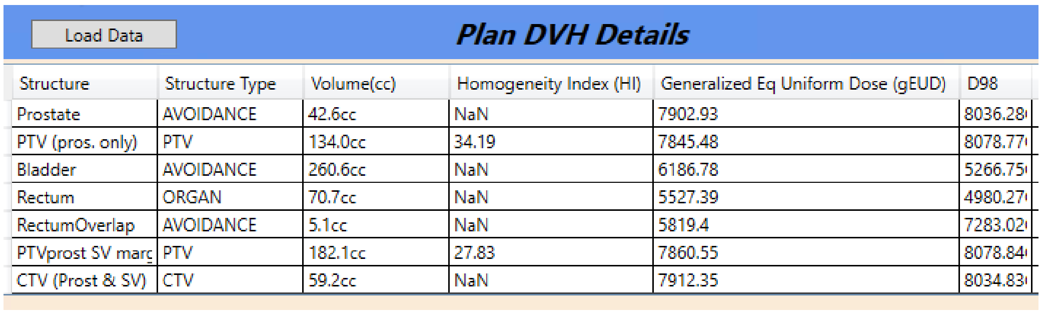
C:\Users\Admin\Documents\PlanStats\PlanMetr...

Dosimetric Plan Metrics

Property	Value
Target	PTV4
Target Volume	0.3cc
HiRes	False
Type	PTV
Is Empty	False
HI	4.14338428452644

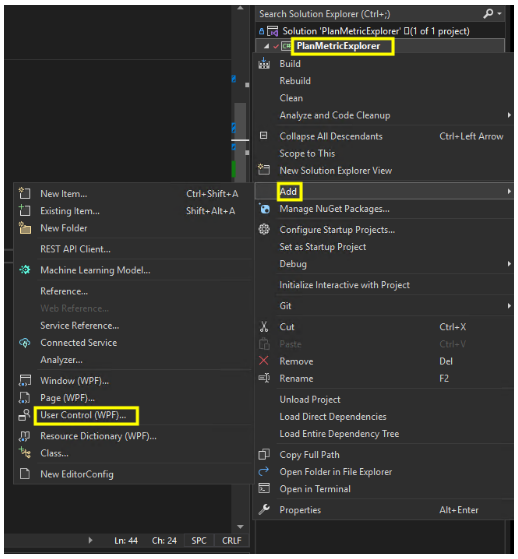
47. Great Success!!!

48. In this Script, the display on data is handled by a "Dictionary" called planMetrics (line 47)and a FlowDocumentScrollViewer Method called AddMetricsToView (line 61->81). This type of display has its limitations. Let's create a much nicer visual display, and add all the structures

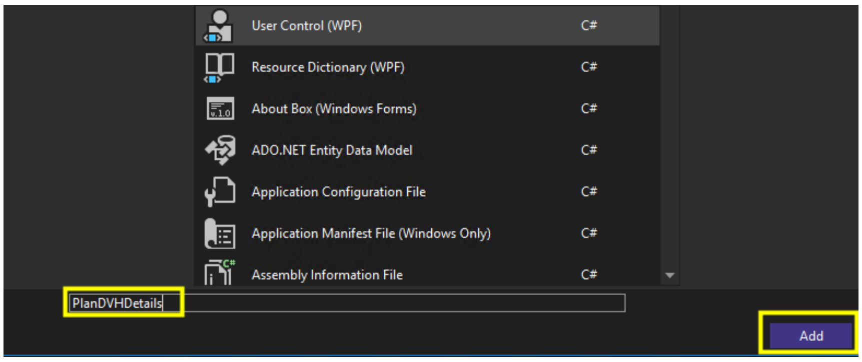

Load Data
Plan DVH Details

Structure	Structure Type	Volume(cc)	Homogeneity Index (HI)	Generalized Eq Uniform Dose (gEUD)	D98
Prostate	AVOIDANCE	42.6cc	NaN	7902.93	8036.28
PTV (pros. only)	PTV	134.0cc	34.19	7845.48	8078.77
Bladder	AVOIDANCE	260.6cc	NaN	6186.78	5266.75
Rectum	ORGAN	70.7cc	NaN	5527.39	4980.27
RectumOverlap	AVOIDANCE	5.1cc	NaN	5819.4	7283.02
PTVprost SV marg	PTV	182.1cc	27.83	7860.55	8078.84
CTV (Prost & SV)	CTV	59.2cc	NaN	7912.35	8034.83

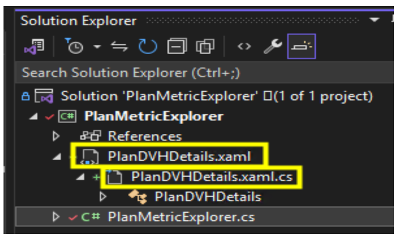
49. Right Click on the Project(PlanMetricExplorer) -> Select Add -> Select UserControl(WPF)...



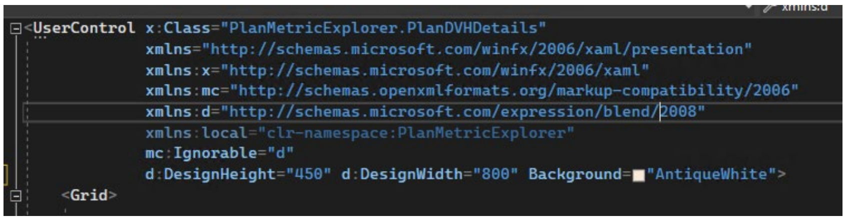
50. Enter the name "PlanDVHDetails" and press Add.


User Control (WPF) C#
Resource Dictionary (WPF) C#
About Box (Windows Forms) C#
ADO.NET Entity Data Model C#
Application Configuration File C#
Application Manifest File (Windows Only) C#
Assembly Information File C#
PlanDVHDetails
Add

51. This step Adds a Visual Display(PlanDVHDetails.xaml) and the underlining code behind it (PlanDVHDetails.xaml.cs).

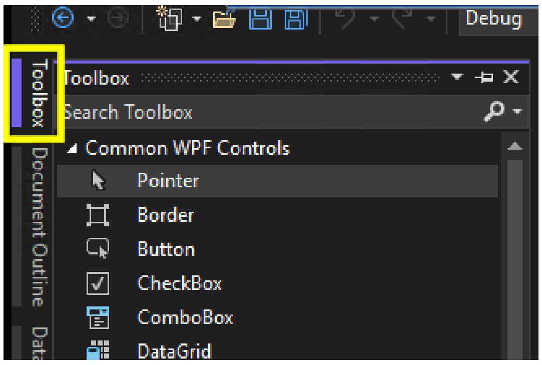


52. The Visual Display(PlanDVHDetails.xaml) can be manipulated via code.



```
<UserControl x:Class="PlanMetricExplorer.PlanDVHDetails"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:PlanMetricExplorer"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800" Background="AntiqueWhite">
    <Grid>
```

53. Or by dragging and dropping items from a "Toolbox".



54. We will do both, first let's change the background colour. On Line 8 of the code, add the following line:

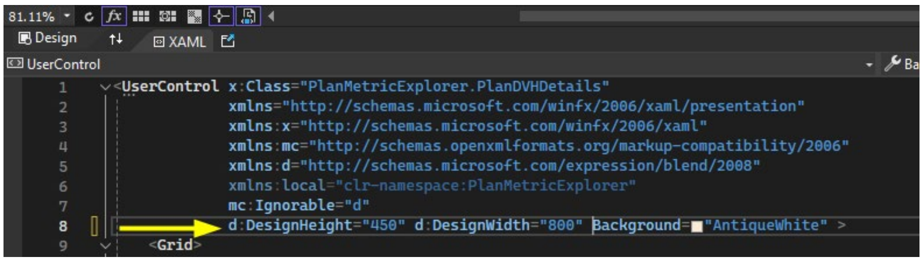
Background="AntiqueWhite"

To

d:DesignHeight="450" d:DesignWidth="800">

to get

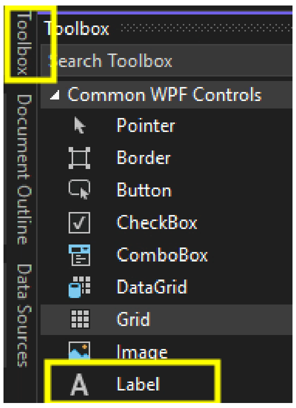
d:DesignHeight="450" d:DesignWidth="800" Background="AntiqueWhite">



```
1  <UserControl x:Class="PlanMetricExplorer.PlanDVHDetails"
2               xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3               xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4               xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
5               xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
6               xmlns:local="clr-namespace:PlanMetricExplorer"
7               mc:Ignorable="d"
8               d:DesignHeight="450" d:DesignWidth="800" Background="AntiqueWhite" >
9      <Grid>
```

55. Now from the Toolbox, drag a Label on the Grid, notice in the code, the following has been added:

<Label Content="Label" HorizontalAlignment="Left" Margin="293,43,0,0" VerticalAlignment="Top"/>



56. We can manipulate the label box by changing the code, We can change the horizontal alignment from Left to Center (American spelling…..Microsoft is American after all!!) and the margin from (293,43 ,0, 0)[which is measured from the horizontal/vertical alignment] to (0, 43, 0, 0)

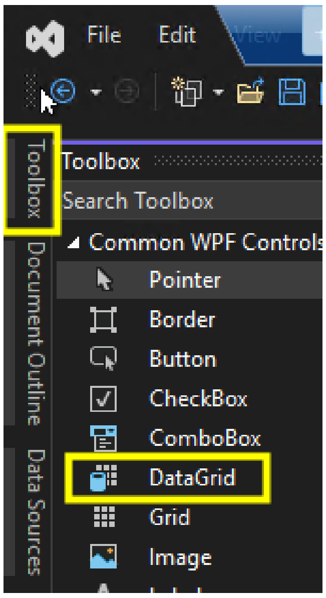
```
<Label Content="Label" HorizontalAlignment="Center" Margin="0,43,0,0" VerticalAlignment="Top"/>
```

57. We can change the Content of the label from "Label" to "Plan DVH Details", replace `Content="Label"` with `Content="Plan DVH Details"`

58. We can change many different aspects of the items in the visual display, including font, size etc

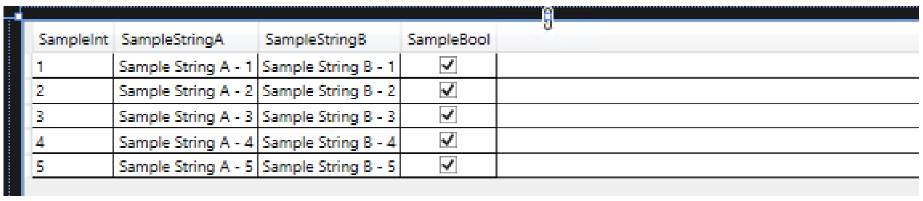
```
<Label HorizontalContentAlignment="Center"
Content="Plan DVH Details"
FontSize="20" FontWeight="Bold" FontStyle="Oblique"
HorizontalAlignment="Center"
VerticalAlignment="Top"
Margin="0,10,0,0"
Background="CornflowerBlue"
Width="780"/>
```

59. Now add the table by clicking and dragging 'DataGrid' from the toolbox.



60. In our code, we now get `<DataGrid d:ItemsSource="{d:SampleData ItemCount=5}"/>`

61. And our GUI looks like this:



SampleInt	SampleStringA	SampleStringB	SampleBool
1	Sample String A - 1	Sample String B - 1	✓
2	Sample String A - 2	Sample String B - 2	✓
3	Sample String A - 3	Sample String B - 3	✓
4	Sample String A - 4	Sample String B - 4	✓
5	Sample String A - 5	Sample String B - 5	✓

62. Click and drag the upper border and the `Margin="0, xxx,0,0"` is added to the code.

63. Now, lets match the table size/position to the Label we previously added. Remove:

```
d:ItemsSource="{d:SampleData ItemCount=5}
```

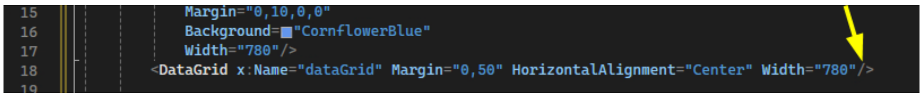
And replace it with:

```
x:Name="dataGrid" Margin="0,50" HorizontalAlignment="Center" Width="780"
```

64. This will position the grid under the label with the same width and give it the name "dataGrid". But now it only has 1 column. We need columns for Structure Id, Structure type, Volume, HI, gEUD etc.

65. The code above describes the dataGrid. We now need to add items to the datagrid, such as columns headers etc.

66. To be able to define these items, first, we need to "Start" and "End" the datagrid. Replace the `/>` at the end, with `><DataGrid.Columns></DataGrid.Columns></DataGrid>` This defines the columns in the grid.



```
        Margin="0,10,0,0"
        Background=■"CornflowerBlue"
        Width="780"/>
    <DataGrid x:Name="dataGrid" Margin="0,50" HorizontalAlignment="Center" Width="780"/>
```

67. In between the DataGrid.Columns lines, we can type the following to add headers and define the width.

```
<DataGridTextColumn Header="Structure" Width="100"/>
```

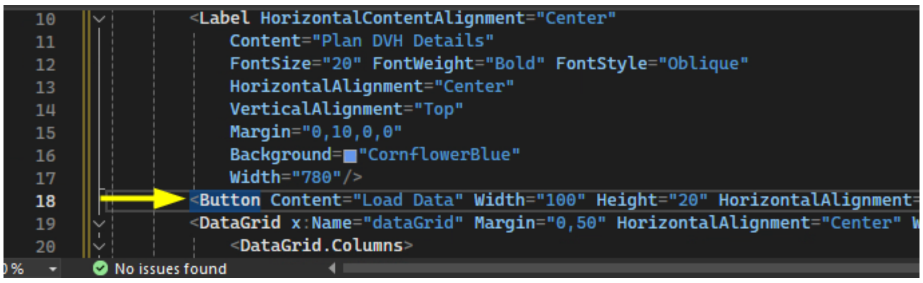
68. To add the other columns, rinse and repeat!!

```
<DataGridTextColumn Header="Structure" Width="100" />
<DataGridTextColumn Header="Structure Type" Width="100" />
<DataGridTextColumn Header="Volume(cc)" Width="100" />
<DataGridTextColumn Header="Homogeneity Index (HI)" Width="140" />
```

69. We can add a button to start the fill of data.

70. Above the definition of the dataGrid, add the following

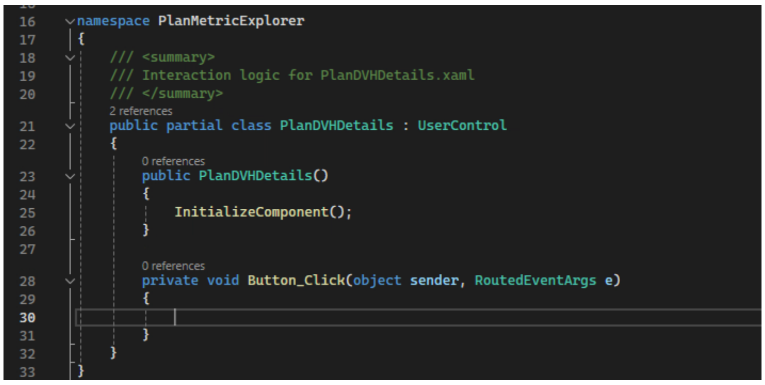
```
<Button Content="Load Data" Width="100" Height="20" HorizontalAlignment="Left" VerticalAlignment="Top"
Margin="20" Name="Button" Click="Button_Click" />
```



```
<Label HorizontalContentAlignment="Center"
    Content="Plan DVH Details"
    FontSize="20" FontWeight="Bold" FontStyle="Oblique"
    HorizontalAlignment="Center"
    VerticalAlignment="Top"
    Margin="0,10,0,0"
    Background=■"CornflowerBlue"
    Width="780"/>
<Button Content="Load Data" Width="100" Height="20" HorizontalAlignment=
<DataGrid x:Name="dataGrid" Margin="0,50" HorizontalAlignment="Center" W
    <DataGrid.Columns>
```

No issues found

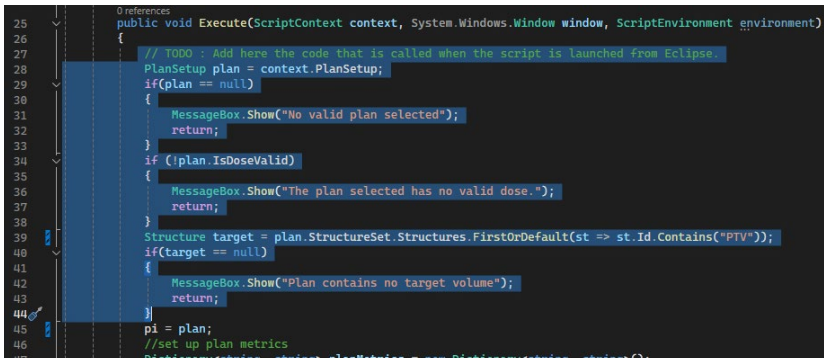
71. Now, double click on the button on the GUI and it will take us to the PlanDVHDetails.xaml.cs file……the file where we can control the things that happen with the GUI, but we need a method for what happens when we click the button.



```
namespace PlanMetricExplorer
{
    /// <summary>
    /// Interaction logic for PlanDVHDetails.xaml
    /// </summary>
    2 references
    public partial class PlanDVHDetails : UserControl
    {
        0 references
        public PlanDVHDetails()
        {
            InitializeComponent();
        }

        0 references
        private void Button_Click(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

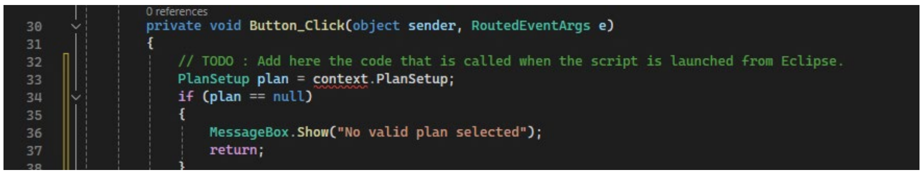
72. When we click the button, we want the dataGrid to fill we the associated data.

73. In the PlanMetricsExplorer.cs file, copy lines ~27 to 44 and paste in the button_click method of the PlanDVHDetails.xaml.cs file.



```
            0 references
25          public void Execute(ScriptContext context, System.Windows.Window window, ScriptEnvironment environment)
26          {
27              // TODO : Add here the code that is called when the script is launched from Eclipse.
28              PlanSetup plan = context.PlanSetup;
29              if(plan == null)
30              {
31                  MessageBox.Show("No valid plan selected");
32                  return;
33              }
34              if (!plan.IsDoseValid)
35              {
36                  MessageBox.Show("The plan selected has no valid dose.");
37                  return;
38              }
39              Structure target = plan.StructureSet.Structures.FirstOrDefault(st => st.Id.Contains("PTV"));
40              if(target == null)
41              {
42                  MessageBox.Show("Plan contains no target volume");
43                  return;
44              }
45              pi = plan;
46              //set up plan metrics
```

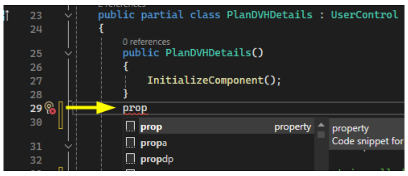
74. We now have a red underline under the word context. This because this method does not receive the context (eclipse loaded plan), so we need to tell this method what it is. We can add a property to be able to access the private field in the PlanMetricsExplorer.cs file.



```
        0 references
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            // TODO : Add here the code that is called when the script is launched from Eclipse.
            PlanSetup plan = context.PlanSetup;
            if (plan == null)
            {
                MessageBox.Show("No valid plan selected");
                return;
```

75. In the PlanDVHDetails.xaml.cs file, between the public PlanDVHDetails() method and the private void Button_Click method,

- **a.** type the word prop

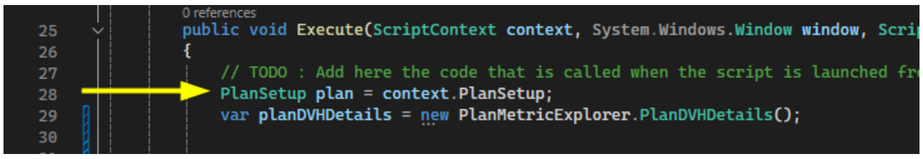


```
    public partial class PlanDVHDetails : UserControl
    {
        0 references
        public PlanDVHDetails()
        {
            InitializeComponent();
        }
        prop
```

- **b.** press tab
- **c.** type PlanSetup
- **d.** press tab tab
- **e.** type MyPlan then tab tab

76. Back in the PlanMetricsExplorer.cs file, we can now assign MyPlan the value of the plan from the PlanMetricsExplorer.cs file. But first we need to tell the PlanMetricsExplorer.cs file that the PlanDVHDetails.xaml.cs exists!!! After Line ~28 in the PlanMetricsExplorer.cs file, add the following:

```
var planDVHDetails = new PlanMetricExplorer.PlanDVHDetails();
```

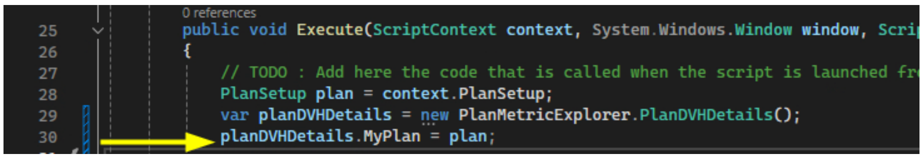


```
        0 references
25      public void Execute(ScriptContext context, System.Windows.Window window, Scri
26      {
27          // TODO : Add here the code that is called when the script is launched fr
28          PlanSetup plan = context.PlanSetup;
29          var planDVHDetails = new PlanMetricExplorer.PlanDVHDetails();
30
```

77. This assigns the variable planDVHDetails (notice the upper and lower casing) in the new GUI.

78. We can now "see" public items in the PlanDVHDetails.xaml.cs file and assign them values. We can assign the property we just created, the value of the PlanSetup in the PlanMetricsExplorer.cs file. After Line~29 add the following

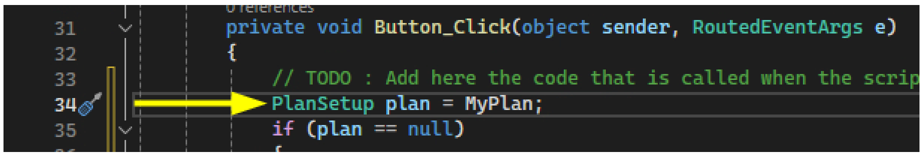
```
planDVHDetails.MyPlan = plan;
```



```
        0 references
25      public void Execute(ScriptContext context, System.Windows.Window window, Scri
26      {
27          // TODO : Add here the code that is called when the script is launched fr
28          PlanSetup plan = context.PlanSetup;
29          var planDVHDetails = new PlanMetricExplorer.PlanDVHDetails();
30          planDVHDetails.MyPlan = plan;
```

79. Back in the PlanDVHDetails.xaml.cs file, we can now replace the line:

`PlanSetup plan = context.PlanSetup;` with `PlanSetup plan = MyPlan;`

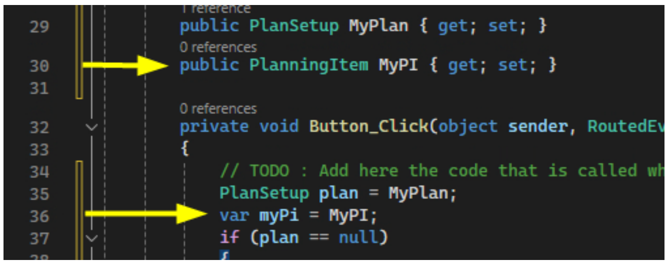


```
31      private void Button_Click(object sender, RoutedEventArgs e)
32      {
33          // TODO : Add here the code that is called when the scrip
34          PlanSetup plan = MyPlan;
35          if (plan == null)
```

80. All the red underlining should now go away!!!

81. We can do the same for the PlanningItem, as we will need that later for the HI calculation.

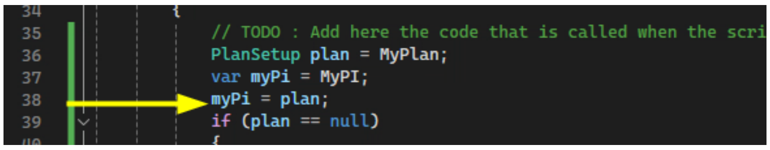
82. Add `public PlanningItem MyPI { get; set; }` and `var myPi = MyPI;` to the PlanDVHDetails.xaml.cs file



```
        1 reference
29      public PlanSetup MyPlan { get; set; }
        0 references
30      public PlanningItem MyPI { get; set; }
31
        0 references
32      private void Button_Click(object sender, RoutedEv
33      {
34          // TODO : Add here the code that is called wh
35          PlanSetup plan = MyPlan;
36          var myPi = MyPI;
37          if (plan == null)
```

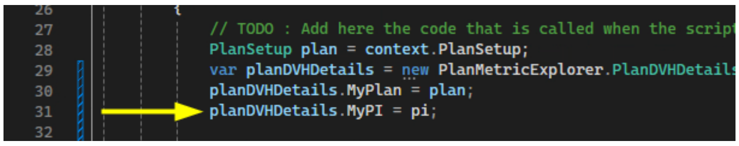
83. We can give myPi a value which is the plan, add the following

```
myPi = plan;
```



```
34      {
35          // TODO : Add here the code that is called when the scri
36          PlanSetup plan = MyPlan;
37          var myPi = MyPI;
38          myPi = plan;
39          if (plan == null)
```

84. Add planDVHDetails.MyPI = pi; to the PlanMetricsExplorer.cs file



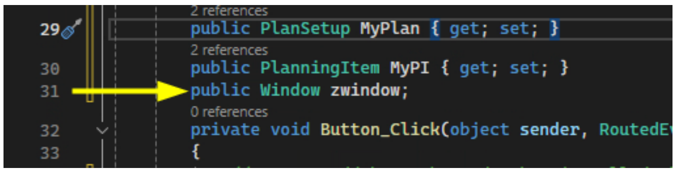
```
{
    // TODO : Add here the code that is called when the script
    PlanSetup plan = context.PlanSetup;
    var planDVHDetails = new PlanMetricExplorer.PlanDVHDetails
    planDVHDetails.MyPlan = plan;
    planDVHDetails.MyPI = pi;
```

85. We now need to tell the PlanMetricsExplorer.cs file to open the GUI. Add the following line under the previous line:

planDVHDetails.Visibility = Visibility.Visible;

86. We can also define where the window opens. First in the PlanDVHDetails.xaml.cs define a new method:

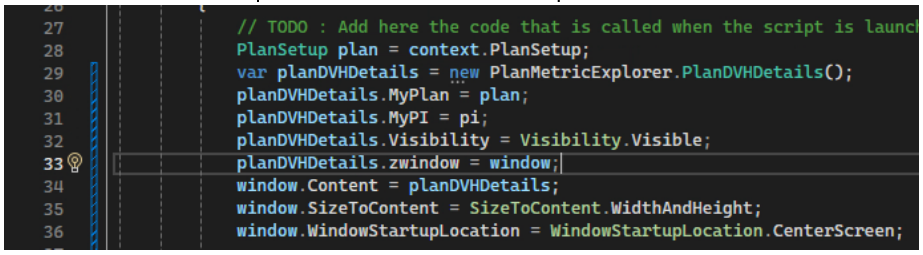
public Window zwindow;



```
2 references
public PlanSetup MyPlan { get; set; }
2 references
public PlanningItem MyPI { get; set; }
public Window zwindow;
0 references
private void Button_Click(object sender, RoutedE
{
```

87. In the PlanMetricsExplorer.cs file, add the following:

planDVHDetails.zwindow = window;
window.Content = planDVHDetails;
window.SizeToContent = SizeToContent.WidthAndHeight;
window.WindowStartupLocation = WindowStartupLocation.CenterScreen;

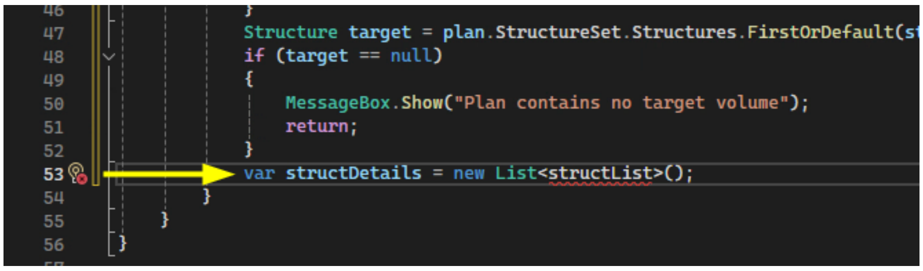


```
{
    // TODO : Add here the code that is called when the script is launch
    PlanSetup plan = context.PlanSetup;
    var planDVHDetails = new PlanMetricExplorer.PlanDVHDetails();
    planDVHDetails.MyPlan = plan;
    planDVHDetails.MyPI = pi;
    planDVHDetails.Visibility = Visibility.Visible;
    planDVHDetails.zwindow = window;
    window.Content = planDVHDetails;
    window.SizeToContent = SizeToContent.WidthAndHeight;
    window.WindowStartupLocation = WindowStartupLocation.CenterScreen;
```

88. Now the window will open in the centre of the screen.

89. Now we need to create a list of our structures and their associated properties(Id, Type, volume, HI, gEUD).

90. Back in the PlanDVHDetails.xaml.cs file, to create a new list, after the "Plan contains no target volume" if statement, hit enter and add the following var structDetails = new List<structList>();

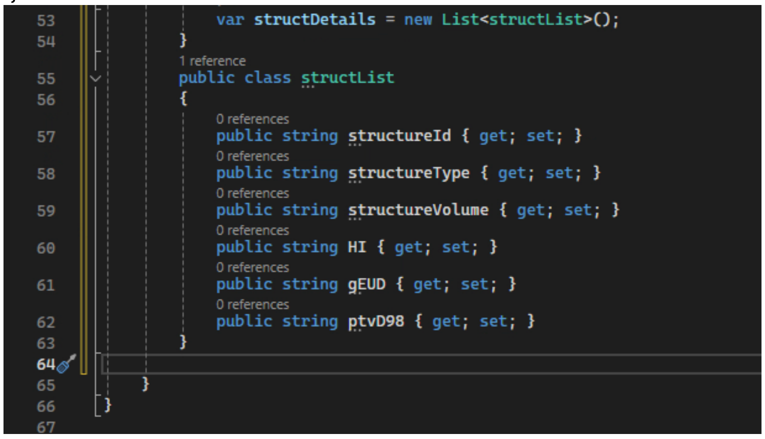


```
    }
    Structure target = plan.StructureSet.Structures.FirstOrDefault(s
    if (target == null)
    {
        MessageBox.Show("Plan contains no target volume");
        return;
    }
    var structDetails = new List<structList>();
}
    }
}
```

91. This assigns a variable structDetails to a list called structList. We now need to create a method for the list that contains all the properties.

92. On the 3rd last line, hit enter and add the following:

```
public class structList
 {
    public string structureId { get; set; }
    public string structureType { get; set; }
    public string structureVolume { get; set; }
    public string HI { get; set; }
    public string gEUD { get; set; }
    public string ptvD98 { get; set; }
}
```



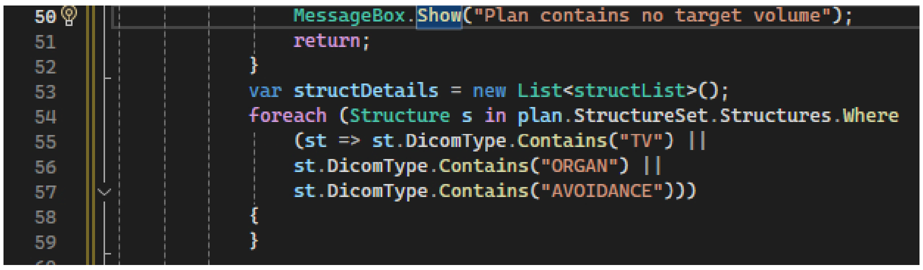
```
            var structDetails = new List<structList>();
        }
        1 reference
        public class structList
        {
            0 references
            public string structureId { get; set; }
            0 references
            public string structureType { get; set; }
            0 references
            public string structureVolume { get; set; }
            0 references
            public string HI { get; set; }
            0 references
            public string gEUD { get; set; }
            0 references
            public string ptvD98 { get; set; }
        }

    }
}
```

93. Like we did earlier this lets us gain access to private fields in the script to assign and values to the properties.

94. We can now add a loop to add the values to the correct property.

95. Under the List declaration, add the following:

```
foreach (Structure s in plan.StructureSet.Structures.Where
              (st => st.DicomType.Contains("TV") ||
              st.DicomType.Contains("ORGAN") ||
         st.DicomType.Contains("AVOIDANCE")))

   {

 }
```

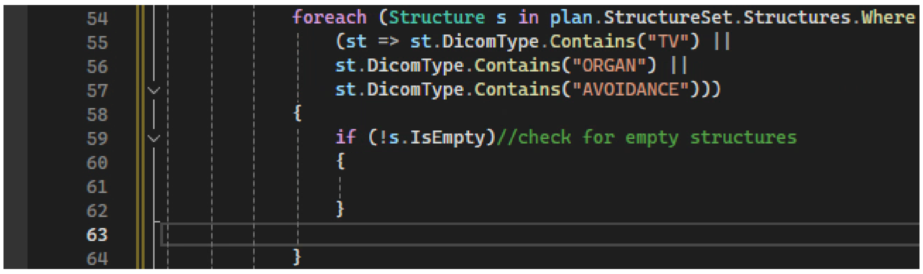


```
                MessageBox.Show("Plan contains no target volume");
                return;
            }
            var structDetails = new List<structList>();
            foreach (Structure s in plan.StructureSet.Structures.Where
                (st => st.DicomType.Contains("TV") ||
                st.DicomType.Contains("ORGAN") ||
                st.DicomType.Contains("AVOIDANCE")))
            {
            }
```

96. This will loop through the structures that contain the characters “TV” or “ORGAN” or “AVOIDANCE” in their Dicom Type.

97. In between the curly brackets, we can first check if the structure is empty(try not to shout at the planner!!) by adding:

```
if (!s.IsEmpty)//check for empty structures

{


}
```

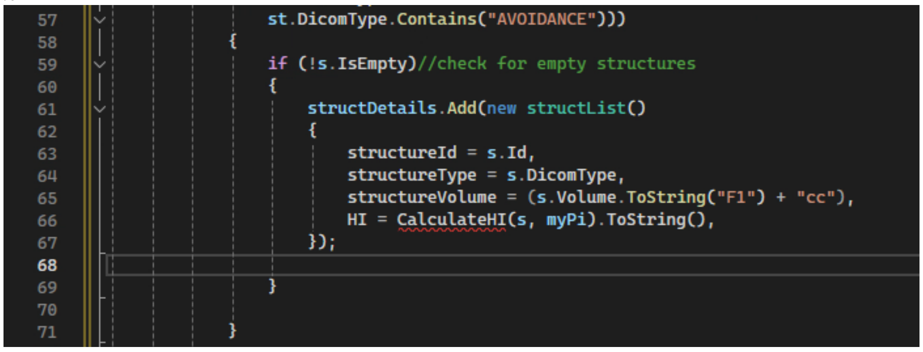


```
foreach (Structure s in plan.StructureSet.Structures.Where
    (st => st.DicomType.Contains("TV") ||
    st.DicomType.Contains("ORGAN") ||
    st.DicomType.Contains("AVOIDANCE")))
{
    if (!s.IsEmpty)//check for empty structures
    {

    }

}
```

98. Note the "//" in the above line, allows us to add comments to explain our actions.

99. Now we add the details for each structure to the list:

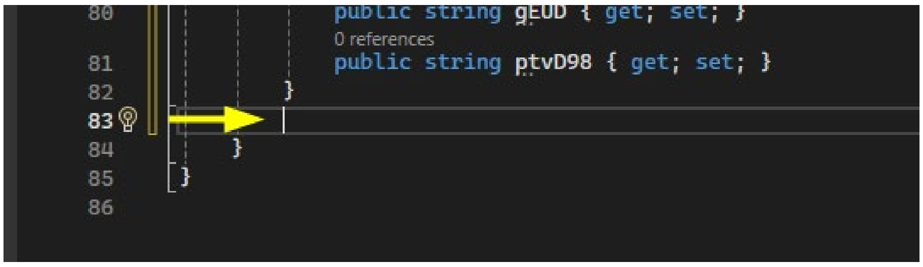
```
structDetails.Add(new structList()
{
    structureId = s.Id,
    structureType = s.DicomType,
    structureVolume = (s.Volume.ToString("F1") + "cc"),
    HI = CalculateHI(s, myPi).ToString(),
});
```



```
    st.DicomType.Contains("AVOIDANCE")))
{
    if (!s.IsEmpty)//check for empty structures
    {
        structDetails.Add(new structList()
        {
            structureId = s.Id,
            structureType = s.DicomType,
            structureVolume = (s.Volume.ToString("F1") + "cc"),
            HI = CalculateHI(s, myPi).ToString(),
        });

    }

}
```

100. Now our only red line should be the CalculateHI, as we haven't got that method in this class. We can copy the method from the PlanMetricsExplorer.cs file or the appendix of this document.

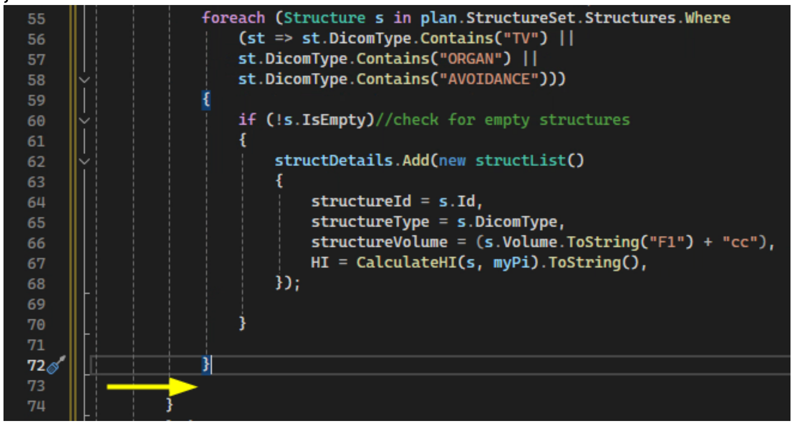
101. On the 3rd last line, paste the CalculateHI method and the red line should disappear.



```
80              public string gEUD { get; set; }
                0 references
81              public string ptvD98 { get; set; }
82          }
83
84      }
85  }
86
```

102. Now all we need to do is add the List to the dataGrid. Again, we will use a loop, this time a For loop. After the last ForEach loop, (approx. line 74) add the following:

```
for (int i = 0; i < structDetails.Count; i++)
{
this.dataGrid.Items.Add(new structList() {
structureId = structDetails[i].structureId.ToString(),
structureType = structDetails[i].structureType.ToString(),
structureVolume = structDetails[i].structureVolume.ToString(),
HI = structDetails[i].HI.ToString()});
}
```



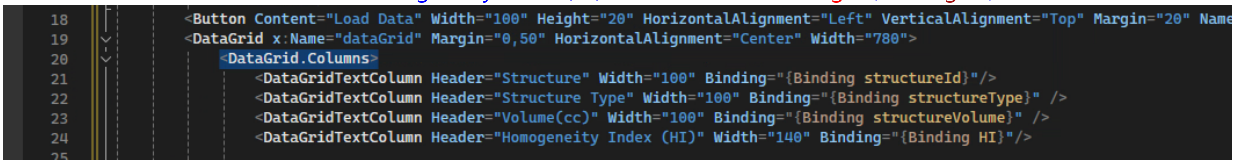
```
55              foreach (Structure s in plan.StructureSet.Structures.Where
56                  (st => st.DicomType.Contains("TV") ||
57                  st.DicomType.Contains("ORGAN") ||
58                  st.DicomType.Contains("AVOIDANCE")))
59              {
60                  if (!s.IsEmpty)//check for empty structures
61                  {
62                      structDetails.Add(new structList()
63                      {
64                          structureId = s.Id,
65                          structureType = s.DicomType,
66                          structureVolume = (s.Volume.ToString("F1") + "cc"),
67                          HI = CalculateHI(s, myPi).ToString(),
68                      });
69
70                  }
71
72              }
73
74          }
```

103. This assigns a variable 'i' for each row in the table and for each structure. It then adds the correct property to the dataGrid.

104. We can now go to our PlanDVHDetails.xaml file and "Bind" the list to the properties.

105. For each DataGridTextColumn we can add, Binding="{Binding structureId}, volume, type etc so they look like:

```
<DataGridTextColumn Header="Structure" Width="100" Binding="{Binding structureId}"/>
<DataGridTextColumn Header="Structure Type" Width="100" Binding="{Binding structureType}" />
<DataGridTextColumn Header="Volume(cc)" Width="100" Binding="{Binding structureVolume}" />
<DataGridTextColumn Header="Homogeneity Index (HI)" Width="140" Binding="{Binding HI}"/>
```



```
        <Button Content="Load Data" Width="100" Height="20" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="20" Name
        <DataGrid x:Name="dataGrid" Margin="0,50" HorizontalAlignment="Center" Width="780">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Structure" Width="100" Binding="{Binding structureId}"/>
                <DataGridTextColumn Header="Structure Type" Width="100" Binding="{Binding structureType}" />
                <DataGridTextColumn Header="Volume(cc)" Width="100" Binding="{Binding structureVolume}" />
                <DataGridTextColumn Header="Homogeneity Index (HI)" Width="140" Binding="{Binding HI}"/>
```

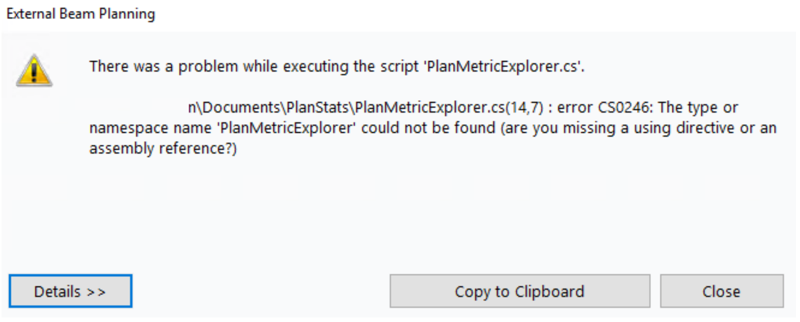
106. And lastly, we need to remove the lines in the PlanMetricsExplorer.cs file. that we have added to the PlanDVHDetails.xaml.cs file.

107. Our PlanMetricsExplorer.cs file should now be:

```
namespace VMS.TPS
{
  public class Script
  {
    public Script()
    {
    }
    PlanningItem pi;
    [MethodImpl(MethodImplOptions.NoInlining)]
    public void Execute(ScriptContext context, System.Windows.Window window, ScriptEnvironment environment)
    {
      // TODO : Add here the code that is called when the script is launched from Eclipse.
      PlanSetup plan = context.PlanSetup;
      var planDVHDetails = new PlanMetricExplorer.PlanDVHDetails();
      planDVHDetails.MyPlan = plan;
      planDVHDetails.MyPI = pi;
      planDVHDetails.Visibility = Visibility.Visible;
      planDVHDetails.zwindow = window;
      window.Content = planDVHDetails;
      window.SizeToContent = SizeToContent.WidthAndHeight;
      window.WindowStartupLocation = WindowStartupLocation.CenterScreen;
    }
  }
}
```

108. Let's build it to see how we went press ctrl-b -> Great Success (Hopefully!!)

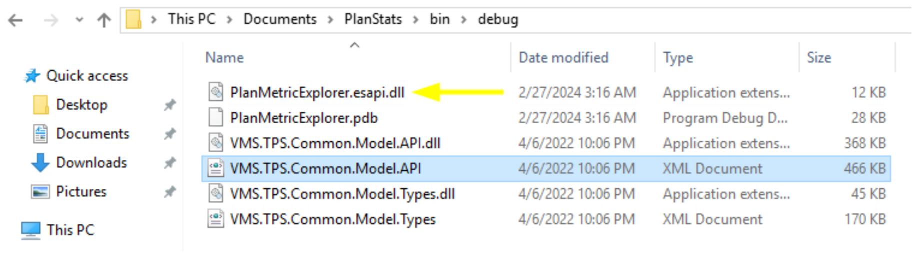
109. Go to Eclipse and run it. Choose the PlanMetricExplorer.cs file and press Run…….Hang on, it built correctly in Visual Studio, so why is it not working in Eclipse?


External Beam Planning

There was a problem while executing the script 'PlanMetricExplorer.cs'.

n\Documents\PlanStats\PlanMetricExplorer.cs(14,7) : error CS0246: The type or namespace name 'PlanMetricExplorer' could not be found (are you missing a using directive or an assembly reference?)

110. Go to the \Documents\PlanStats\ directory and select the bin then debug folder. Rename the file called PlanMetricExplorer.dll to PlanMetricExplorer.esapi.dll. Now in Eclipse navigate to the \Documents\PlanStats\bin\debug folder and now run the new file.


This PC > Documents > PlanStats > bin > debug

Name	Date modified	Type	Size
PlanMetricExplorer.esapi.dll	2/27/2024 3:16 AM	Application extens...	12 KB
PlanMetricExplorer.pdb	2/27/2024 3:16 AM	Program Debug D...	28 KB
VMS.TPS.Common.Model.API.dll	4/6/2022 10:06 PM	Application extens...	368 KB
VMS.TPS.Common.Model.API	4/6/2022 10:06 PM	XML Document	466 KB
VMS.TPS.Common.Model.Types.dll	4/6/2022 10:06 PM	Application extens...	45 KB
VMS.TPS.Common.Model.Types	4/6/2022 10:06 PM	XML Document	170 KB

Extension – Add gEUD and d98

1. The calculation for the HI has the d98 calculation in it. We can use this to create another method that returns the d98. Copy the following line from the HI method:

```
double d98 = (pi as PlanSetup).GetDoseAtVolume(s, 98, VolumePresentation.Relative,
DoseValuePresentation.Absolute).Dose;
```

2. Above the HI method, create a new method called Calculated98

```
private double Calculated98(Structure s, PlanningItem pi)

{}
```

3. Like the HI method, this method "looks" for a structure, s, and a planning item, pi. It also returns a double, like the HI method.

4. Paste the copied line into the method and add the following line to return the double:

```
return d98;
```

5. We can also round the value to 2 decimal places. In the line above the return add the following:

```
d98 = Math.Round(d98, 2);
```

6. In our strucutList class , we can add the following property:

```
public string ptvD98 { get; set; }
```

7. In our formula to add the items to the list (structDetails.Add(new structList()), we can add the following 2 lines (we will work out the gEUD later):

```
gEUD = CalculateGEUD(s, myPi).ToString(),

ptvD98 = Calculated98(s, myPi).ToString() + "Gy",
```

8. In our this.dataGrid.Items.Add we can add the following:

```
, gEUD = structDetails[i].gEUD.ToString(), ptvD98 = structDetails[i].ptvD98.ToString()
```

It should now look like this:

```
for (int i = 0; i < structDetails.Count; i++)
{
    this.dataGrid.Items.Add(new structList() {
        structureId = structDetails[i].structureId.ToString(),
        structureType = structDetails[i].structureType.ToString(),
        structureVolume = structDetails[i].structureVolume.ToString(),
        HI = structDetails[i].HI.ToString(),
        gEUD = structDetails[i].gEUD.ToString(),
        ptvD98 = structDetails[i].ptvD98.ToString() });
}
```

9. We can add the following 2 lines to the DataGrid.Columns in the PlanDVHDetails.xaml file:

```
<DataGridTextColumn Header="Generalized Eq Uniform Dose (gEUD)" Width="210" Binding="{Binding gEUD}"/>
<DataGridTextColumn Header="D98" Width="50" Binding="{Binding ptvD98}"/>
```

10. In between the Calculated98 and CalculateHI method, create a new method, called: CalculateGEUD

```
private double CalculateGEUD(Structure s, PlanningItem pi)
```

11. Copy and paste appendix 2 gEUD Method and a_value Properties. This method/property creates a "library" of structures and assigns the a_value to each.

Appendix 1: HI Method

```
private double CalculateHI(Structure s, PlanningItem pi)
{
    //hi only needs to be calculated for ptv, so filter those out here.
    if (!s.DicomType.ToUpper().Contains("PTV"))
    {
        return Double.NaN;
    }
    //now check if the planning item is a plansetup or sum.
    if (pi is PlanSetup)
    {
        //plansetups have a method called GetDoseAtVolume.
        if (pi.Dose == null)
        {
            MessageBox.Show("Plan has no dose");
            return Double.NaN;
        }
        double d2 = (pi as PlanSetup).GetDoseAtVolume(s, 2, VolumePresentation.Relative,
                                                        DoseValuePresentation.Absolute).Dose;
        double d98 = (pi as PlanSetup).GetDoseAtVolume(s, 98, VolumePresentation.Relative,
                                                        DoseValuePresentation.Absolute).Dose;
        double hi = ((d2 - d98) / (pi as PlanSetup).TotalDose.Dose) * 100;
        hi = Math.Round(hi, 2);
        return hi;
    }
    else if (pi is PlanSum)
    {
        //must manually calculate value from DVH
        DVHData dvh = pi.GetDVHCumulativeData(s, DoseValuePresentation.Absolute, VolumePresentation.Relative, 0.1);
        if (dvh == null)
        {
            MessageBox.Show("Could not collect DVH");
            return Double.NaN;
        }
        double d98 = dvh.CurveData.FirstOrDefault(x => x.Volume <= 98).DoseValue.Dose;
        double d2 = dvh.CurveData.FirstOrDefault(x => x.Volume <= 2).DoseValue.Dose;
        List<double> rx_doses = new List<double>();
        foreach (PlanSetup ps in (pi as PlanSum).PlanSetups)
        {
            try
            {
                rx_doses.Add(ps.TotalDose.Dose);
            }
            catch
            {
                MessageBox.Show("One of the prescriptions for the plansum is not defined");
                return Double.NaN;
            }
        }
        double rx = rx_doses.Sum();
        double hi = ((d2 - d98) / rx) * 100;
        hi = Math.Round(hi, 2);
        return hi;
    }
    else
    {
        MessageBox.Show("Plan not handled correctly");
        return Double.NaN;
    }
}
```

Appendix 2: gEUD Method and a_value Properties

```
private double CalculateGEUD(Structure s, PlanningItem pi)
{
    //collect the DVH
    //if volume is not relative, make sure to normalize over the total volume during geud calculation.
    //double volume = s.Volume;
    //remember plansums must be absolute dose.

    List<geudValues> structgeud = new List<geudValues>();
    structgeud.Add(new geudValues { struc = "Heart", value = 0.5 });
    structgeud.Add(new geudValues { struc = "Cord", value = 20 });
    structgeud.Add(new geudValues { struc = "Parotid", value = 0.5 });
    structgeud.Add(new geudValues { struc = "Bladder", value = 10 });
    structgeud.Add(new geudValues { struc = "Rectum", value = 10 });
    structgeud.Add(new geudValues { struc = "TV", value = 0.5 });
    structgeud.Add(new geudValues { struc = "Stem", value = 20 });
    structgeud.Add(new geudValues { struc = "Brain", value = 20 });
    structgeud.Add(new geudValues { struc = "Eye", value = 0.5 });
    structgeud.Add(new geudValues { struc = "Rectum", value = 10 });

    Structure geudStruct = s;
    double a_value = 1;
    for (int i = 0; i < structgeud.Count; i++) { if (structgeud[i].struc.Contains(s.Id)) { a_value = structgeud[i].value; } }


    DVHData dvh = pi.GetDVHCumulativeData(s, DoseValuePresentation.Absolute, VolumePresentation.Relative, 0.1);
    if (dvh == null)
    {
        MessageBox.Show("Could not calculate DVH");
        return Double.NaN;
    }
    //we need to get the differential volume from the definition. Loop through Volumes and take the difference with the
        previous dvhpoint
    double running_sum = 0;
    int counter = 0;
    foreach (DVHPoint dvhp in dvh.CurveData.Skip(1))
    {
        //volume units are in % (divide by 100)
        double vol_diff = Math.Abs(dvhp.Volume - dvh.CurveData[counter].Volume) / 100;
        double dose = dvhp.DoseValue.Dose;
        running_sum += vol_diff * Math.Pow(dose, a_value);
        counter++;
    }
    double geud = Math.Pow(running_sum, 1 / a_value);
    geud = Math.Round(geud, 2);
    return geud;
}

public class geudValues
{
    public string struc { get; set; }
    public double value { get; set; }
}
```