```
1  Imports System.Collections.Generic
2
3  Imports Grasshopper.Kernel
4  Imports Grasshopper.Kernel.Data
5  Imports Grasshopper.Kernel.Types
6  Imports Rhino.Geometry
7  Imports Interop.gsa_8_7
8  Imports UsageDataCollection
9
10
11 Public Class MiscOutputExtract
12     Inherits GH_Component
13     ''' <summary>
14     ''' Initializes a new instance of the MiscOutputExtract class.
15     ''' </summary>
16     Public Sub New()
17         MyBase.New("Misc Output Extract", "MiscOutputExtract", _
18                    "Output extract for non-node and non-element entities", _
19                    "GsaComTools", "Assemblies")
20     End Sub
21     Enum Output_Init_Flags
22         OP_INIT_2D_BOTTOM = &H1     ' output 2D stresses at bottom layer
23         OP_INIT_2D_MIDDLE = &H2     ' output 2D stresses at middle layer
24         OP_INIT_2D_TOP = &H4        ' output 2D stresses at top layer
25         OP_INIT_2D_BENDING = &H8    ' output 2D stresses at bending layer
26         OP_INIT_2D_AVGE = &H10      ' average 2D element stresses at nodes
27         OP_INIT_1D_AUTO_PTS = &H20  ' calculate 1D results at interesting
             points
28         OP_INIT_INFINITY = &H40     ' return infinity and NaN values as such,
             else as zero
29     End Enum
30     Enum Output_IsDataRef_Flags
31         OP_IS_AND = &H1             ' otherwise OR
32         OP_IS_PER_REC = &H2
33         OP_IS_PER_NODE = &H4
34         OP_IS_PER_ELEM = &H8
35         OP_IS_PER_MEMB = &H10
36         OP_IS_PER_1D_DISP = &H20
37         OP_IS_PER_1D_FRC = &H40
38         OP_IS_PER_TOPO = &H80
39         OP_IS_AT_CENTRE = &H100
40     End Enum
41
42     ''' <summary>
43     ''' Registers all the input parameters for this component.
44     ''' </summary>
45     Protected Overrides Sub RegisterInputParams(pManager As
         GH_Component.GH_InputParamManager)
46         pManager.AddGenericParameter("Gsa", "Gsa", "Gsa COM Object",
             GH_ParamAccess.item)
47         pManager.AddIntegerParameter("Record", "Record", "Record",
             GH_ParamAccess.list)
48         pManager.AddTextParameter("Axis", "Axis", "Axis (Optional)",
             GH_ParamAccess.item)
49         pManager.AddTextParameter("Load Case", "LoadCase", "Load Case
             (Optional)", GH_ParamAccess.list)
```

```
50          pManager.AddIntegerParameter("Data Reference", "DataRef", "Data
              Reference", GH_ParamAccess.item)
51
52          pManager(2).Optional = True
53          pManager(3).Optional = True
54
55      End Sub
56
57      ''' <summary>
58      ''' Registers all the output parameters for this component.
59      ''' </summary>
60      Protected Overrides Sub RegisterOutputParams(pManager As
          GH_Component.GH_OutputParamManager)
61          pManager.AddGenericParameter("Gsa", "Gsa", "Gsa COM Object",
              GH_ParamAccess.item)
62          pManager.AddTextParameter("Data Title", "DataTitle", "Data Title",
              GH_ParamAccess.item)
63          pManager.AddTextParameter("Unit", "Unit", "Unit", GH_ParamAccess.item)
64          pManager.AddTextParameter("Misc Extract Outputs", "Output", "Misc
              Extract Outputs", GH_ParamAccess.tree)
65      End Sub
66
67      ''' <summary>
68      ''' This is the method that actually does the work.
69      ''' </summary>
70      ''' <param name="DA">The DA object is used to retrieve from inputs and
          store in outputs.</param>
71      Protected Overrides Sub SolveInstance(DA As IGH_DataAccess)
72          'Declare Variables
73          Dim FileName As String = ""
74          Dim RecRefs As New List(Of Integer)
75          Dim Output As New Grasshopper.DataTree(Of String)
76          Dim s As Short
77          Dim Axis As String = Nothing
78          Dim LoadCase As New List(Of String)
79          Dim DataRef As Integer
80          Dim LoadCaseData As New List(Of String())
81          Dim i, j, k As Integer
82          Dim myDataCollector As New DataCollector
83          Dim myObjectWrapper As Types.GH_ObjectWrapper = Nothing
84          Dim myData As String
85          Dim ResultsExist As Short
86          Dim ConnectedElements() As Integer = Nothing
87          Dim NumElements As Integer
88          Dim UseElemRefs As Boolean
89          Dim BranchIndex As Integer
90
91          'Import GSA File Name
92          If (Not DA.GetData(0, myObjectWrapper)) Then Return
93          If (Not DA.GetDataList(1, RecRefs)) Then Return
94          If (Not DA.GetData(2, Axis)) Then Axis = "default"
95          If (Not DA.GetDataList(3, LoadCase)) Then LoadCase.Add("A1")
96          If (Not DA.GetData(4, DataRef)) Then Return
97
98          'Parse Load Case Data
99          LoadCaseData = ParseLoadCaseData(LoadCase)
```

```
100
101         Dim GsaObj As ComAuto = TryCast(myObjectWrapper.Value, ComAuto)
102
103         'Check for existing analysis results
104         BranchIndex = -1
105         i = 0
106         For Each item In LoadCaseData
107
108             'Create New Branch
109             Output.AddRange(New List(Of String), New GH_Path(i))
110             BranchIndex += 1
111
112             'Get Data
113             s = GsaObj.Output_Init(Output_Init_Flags.OP_INIT_INFINITY, Axis,
                  LoadCase.Item(i), DataRef, 0)
114             s = GsaObj.Output_SetStage(0) ' Whole Model
115
116             For Each ref In RecRefs
117
118                 'Extract Data
119                 myData = Nothing
120                 If (GsaObj.Output_DataExist(ref) <> 0 And
                      GsaObj.Output_IsDataRef
                      (Output_IsDataRef_Flags.OP_IS_PER_REC) = 1) Then
121                     Try
122                         myData = CStr(GsaObj.Output_Extract(ref, 0))
123                     Catch ex As Exception
124                         AddRuntimeMessage(GH_RuntimeMessageLevel.Error, "Error
                        on GsaObj.OutputExtract, Record " & ref & ". Message: " &
                        ex.Message)
125                     End Try
126                 End If
127
128                 'Write data to data tree
129                 Output.Branches.Item(BranchIndex).Add(myData)
130             Next
131
132         Next
133         ' Assign the data to the output parameter.
134         DA.SetData(0, GsaObj)
135         DA.SetData(1, CStr(GsaObj.Output_DataTitle(1)))
136         DA.SetData(2, CStr(GsaObj.Output_UnitString))
137         DA.SetDataTree(3, Output)
138
139         'Send Usage Data
140         myDataCollector.UsageStatistics("MiscOutputExtract")
141
142     End Sub
143
144     ''' <summary>
145     ''' Provides an Icon for every component that will be visible in the User
          Interface.
146     ''' Icons need to be 24x24 pixels.
147     ''' </summary>
148     Protected Overrides ReadOnly Property Icon() As System.Drawing.Bitmap
149         Get
```

```
150             Return My.Resources.Misc_Output
151         End Get
152     End Property
153
154     ''' <summary>
155     ''' Gets the unique ID for this component. Do not change this ID after release.
156     ''' </summary>
157     Public Overrides ReadOnly Property ComponentGuid() As Guid
158         Get
159             Return New Guid("{8b8247a8-9ad8-4f27-9c55-5f1cd6cf6b67}")
160         End Get
161     End Property
162 End Class
```