```
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 Public Class NodeOutputExtract
11     Inherits GH_Component
12     ''' <summary>
13     ''' Initializes a new instance of the ElemOutputExtract class.
14     ''' </summary>
15     Public Sub New()
16         MyBase.New("Node Output Extract", "NodeOutputExtract", _
17                 "Perform Output Extract on Nodes", _
18                 "GsaComTools", "Nodes")
19     End Sub
20     Enum Output_Init_Flags
21         OP_INIT_2D_BOTTOM = &H1     ' output 2D stresses at bottom layer
22         OP_INIT_2D_MIDDLE = &H2     ' output 2D stresses at middle layer
23         OP_INIT_2D_TOP = &H4        ' output 2D stresses at top layer
24         OP_INIT_2D_BENDING = &H8    ' output 2D stresses at bending layer
25         OP_INIT_2D_AVGE = &H10      ' average 2D element stresses at nodes
26         OP_INIT_1D_AUTO_PTS = &H20  ' calculate 1D results at interesting
             points
27         OP_INIT_INFINITY = &H40     ' return infinity and NaN values as such,
             else as zero
28     End Enum
29     Enum Output_IsDataRef_Flags
30         OP_IS_AND = &H1             ' otherwise OR
31         OP_IS_PER_REC = &H2
32         OP_IS_PER_NODE = &H4
33         OP_IS_PER_ELEM = &H8
34         OP_IS_PER_MEMB = &H10
35         OP_IS_PER_1D_DISP = &H20
36         OP_IS_PER_1D_FRC = &H40
37         OP_IS_PER_TOPO = &H80
38         OP_IS_AT_CENTRE = &H100
39     End Enum
40     ''' <summary>
41     ''' Registers all the input parameters for this component.
42     ''' </summary>
43     Protected Overrides Sub RegisterInputParams(pManager As
         GH_Component.GH_InputParamManager)
44         pManager.AddGenericParameter("Gsa", "Gsa", "Gsa COM Object",
             GH_ParamAccess.item)
45         pManager.AddIntegerParameter("Nodes", "Nodes", "Node Numbers",
             GH_ParamAccess.list)
46         pManager.AddTextParameter("Axis", "Axis", "Axis (Optional)",
             GH_ParamAccess.item)
47         pManager.AddTextParameter("Load Case", "LoadCase", "Load Case",
             GH_ParamAccess.list)
48         pManager.AddIntegerParameter("Data Reference", "DataRef", "Data
             Reference", GH_ParamAccess.item)
```

```
49
50          pManager(2).Optional = True
51
52      End Sub
53
54      ''' <summary>
55      ''' Registers all the output parameters for this component.
56      ''' </summary>
57      Protected Overrides Sub RegisterOutputParams(pManager As
          GH_Component.GH_OutputParamManager)
58          pManager.AddGenericParameter("Gsa", "Gsa", "Gsa COM Object",
              GH_ParamAccess.item)
59          pManager.AddTextParameter("Data Title", "DataTitle", "Data Title",
              GH_ParamAccess.item)
60          pManager.AddTextParameter("Unit", "Unit", "Unit", GH_ParamAccess.item)
61          pManager.AddTextParameter("Node Extract Outputs", "Output", "Node
              Extract Outputs", GH_ParamAccess.tree)
62          pManager.AddTextParameter("Element Reference", "ElemRef", "Connected
              Element Reference (for elem forces on nodes)", GH_ParamAccess.tree)
63      End Sub
64
65      ''' <summary>
66      ''' This is the method that actually does the work.
67      ''' </summary>
68      ''' <param name="DA">The DA object is used to retrieve from inputs and
          store in outputs.</param>
69      Protected Overrides Sub SolveInstance(DA As IGH_DataAccess)
70          'Declare Variables
71          Dim FileName As String = ""
72          Dim NodeRefs As New List(Of Integer)
73          Dim Output As New Grasshopper.DataTree(Of String)
74          Dim ElementRefs As New Grasshopper.DataTree(Of String)
75          Dim s As Short
76          Dim Axis As String = Nothing
77          Dim LoadCase As New List(Of String)
78          Dim DataRef As Integer
79          Dim LoadCaseData As New List(Of String())
80          Dim i, j, k As Integer
81          Dim myDataCollector As New DataCollector
82          Dim myObjectWrapper As Types.GH_ObjectWrapper = Nothing
83          Dim myData As String
84          Dim ResultsExist As Short
85          Dim ConnectedElements() As Integer = Nothing
86          Dim NumElements As Integer
87          Dim UseElemRefs As Boolean
88          Dim BranchIndex As Integer
89
90          'Import GSA File Name
91          If (Not DA.GetData(0, myObjectWrapper)) Then Return
92          If (Not DA.GetDataList(1, NodeRefs)) Then Return
93          If (Not DA.GetData(2, Axis)) Then Axis = "default"
94          If (Not DA.GetDataList(3, LoadCase)) Then Return
95          If (Not DA.GetData(4, DataRef)) Then Return
96
97          '--NodeOutputExtract Output Data Structure:
98          'Level1:Load Cases
```

```
 99          'Level2:Node
100
101          'Specify whether Level 2 data organization should be engaged
102          Select Case DataRef
103              Case 12006001
104                  UseElemRefs = True
105              Case 12006002
106                  UseElemRefs = True
107              Case 12006003
108                  UseElemRefs = True
109              Case 12006004
110                  UseElemRefs = True
111              Case 12006005
112                  UseElemRefs = True
113              Case 12006006
114                  UseElemRefs = True
115              Case 12006007
116                  UseElemRefs = True
117              Case 12006008
118                  UseElemRefs = True
119              Case Else
120                  UseElemRefs = False
121          End Select
122
123          'Parse Load Case Data
124          LoadCaseData = ParseLoadCaseData(LoadCase)
125
126          Dim GsaObj As ComAuto = TryCast(myObjectWrapper.Value, ComAuto)
127
128          'Check for existing analysis results
129          BranchIndex = -1
130          i = 0
131          For Each item In LoadCaseData
132
133              'Create New Branch
134              Output.AddRange(New List(Of String), New GH_Path(i))
135              ElementRefs.AddRange(New List(Of String), New GH_Path(i))
136              BranchIndex += 1
137
138              ResultsExist = GsaObj.CaseResultsExist(item(0), CInt(item(1)), 0)
139              If ResultsExist <> 1 Then
140                  AddRuntimeMessage(GH_RuntimeMessageLevel.Error, "Analysis
                      results not detected for specified case.  Ensure the model
                      is run.")
141                  DA.SetData(0, CBool(False))
142                  Return
143              End If
144
145              'Get Data
146              s = GsaObj.Output_Init(Output_Init_Flags.OP_INIT_INFINITY, Axis,
                    LoadCase.Item(i), DataRef, 0)
147              s = GsaObj.Output_SetStage(0) ' Whole Model
148
149              j = 0
150
151              For Each ref In NodeRefs
```

```
152
153                 'Get Connected Elements
154                 GsaObj.NodeConnectedEnt(GsaEntity.ELEMENT, ref,
                      ConnectedElements)
155                 NumElements = UBound(ConnectedElements, 1)
156
157                 If UseElemRefs = True Then
158                     Output.AddRange(New List(Of String), New GH_Path(i, j))
159                     ElementRefs.AddRange(New List(Of String), New GH_Path(i,
                        j))
160                     BranchIndex += 1
161                 End If
162
163                 'Extract Data
164                 For k = 0 To NumElements
165                     myData = Nothing
166                     If (GsaObj.Output_DataExist(ref) <> 0 And
                        GsaObj.Output_IsDataRef
                        (Output_IsDataRef_Flags.OP_IS_PER_NODE) = 1) Then
167                         Try
168                             myData = CStr(GsaObj.Output_Extract(ref,
                            ConnectedElements(k)))
169                         Catch ex As Exception
170                             AddRuntimeMessage(GH_RuntimeMessageLevel.Error,
                            "Error on GsaObj.OutputExtract, Node " & ref & ". Message:
                             " & ex.Message)
171                         End Try
172                     End If
173
174                     'Write data to data tree
175                     Output.Branches.Item(BranchIndex).Add(myData)
176                     If UseElemRefs = True Then
177                         ElementRefs.Branches.Item(BranchIndex).Add(CStr
                        (ConnectedElements(k)))
178                     Else
179                         ElementRefs.Branches.Item(BranchIndex).Add("N/A")
180                         Exit For
181                     End If
182                 Next
183                 j += 1
184             Next
185             i += 1
186         Next
187         ' Assign the data to the output parameter.
188         DA.SetData(0, GsaObj)
189         DA.SetData(1, CStr(GsaObj.Output_DataTitle(1)))
190         DA.SetData(2, CStr(GsaObj.Output_UnitString))
191         DA.SetDataTree(3, Output)
192         If UseElemRefs = True Then DA.SetDataTree(4, ElementRefs)
193
194         'Send Usage Data
195         myDataCollector.UsageStatistics("NodeOutputExtract")
196
197     End Sub
198
199     ''' <summary>
```

```
200      ''' Provides an Icon for every component that will be visible in the User
           Interface.
201      ''' Icons need to be 24x24 pixels.
202      ''' </summary>
203      Protected Overrides ReadOnly Property Icon() As System.Drawing.Bitmap
204          Get
205              'You can add image files to your project resources and access them
                   like this:
206              ' return Resources.IconForThisComponent;
207              Return My.Resources.Node_Output
208          End Get
209      End Property
210
211      ''' <summary>
212      ''' Gets the unique ID for this component. Do not change this ID after
           release.
213      ''' </summary>
214      Public Overrides ReadOnly Property ComponentGuid() As Guid
215          Get
216              Return New Guid("{e6a67641-54f8-4387-bd4d-58abaa4303bb}")
217          End Get
218      End Property
219
220  End Class
```