```vb
Imports System.Collections.Generic

Imports Grasshopper.Kernel
Imports Grasshopper.Kernel.Data
Imports Grasshopper.Kernel.Types
Imports Rhino.Geometry
Imports Interop.gsa_8_7
Imports UsageDataCollection

Public Class ElemOutputExtract
    Inherits GH_Component
    ''' <summary>
    ''' Initializes a new instance of the ElemOutputExtract class.
    ''' </summary>
    Public Sub New()
        MyBase.New("Element Output Extract", "ElemOutputExtract", _
                "Perform Output Extract on Elements", _
                "GsaComTools", "Elements")
    End Sub
    Enum Output_Init_Flags
        OP_INIT_2D_BOTTOM = &H1     ' output 2D stresses at bottom layer
        OP_INIT_2D_MIDDLE = &H2     ' output 2D stresses at middle layer
        OP_INIT_2D_TOP = &H4        ' output 2D stresses at top layer
        OP_INIT_2D_BENDING = &H8    ' output 2D stresses at bending layer
        OP_INIT_2D_AVGE = &H10      ' average 2D element stresses at nodes
        OP_INIT_1D_AUTO_PTS = &H20  ' calculate 1D results at interesting points
        OP_INIT_INFINITY = &H40     ' return infinity and NaN values as such, else as zero
    End Enum
    Enum Output_IsDataRef_Flags
        OP_IS_AND = &H1             ' otherwise OR
        OP_IS_PER_REC = &H2
        OP_IS_PER_NODE = &H4
        OP_IS_PER_ELEM = &H8
        OP_IS_PER_MEMB = &H10
        OP_IS_PER_1D_DISP = &H20
        OP_IS_PER_1D_FRC = &H40
        OP_IS_PER_TOPO = &H80
        OP_IS_AT_CENTRE = &H100
    End Enum
    ''' <summary>
    ''' Registers all the input parameters for this component.
    ''' </summary>
    Protected Overrides Sub RegisterInputParams(pManager As GH_Component.GH_InputParamManager)
        pManager.AddGenericParameter("Gsa", "Gsa", "Gsa COM Object", GH_ParamAccess.item)
        pManager.AddIntegerParameter("Elements", "Els", "Element Numbers", GH_ParamAccess.list)
        pManager.AddTextParameter("Axis", "Axis", "Axis (Optional)", GH_ParamAccess.item)
        pManager.AddTextParameter("Load Case", "LoadCase", "Load Case", GH_ParamAccess.list)
        pManager.AddIntegerParameter("Data Reference", "DataRef", "Data Reference", GH_ParamAccess.item)
        pManager.AddIntegerParameter("Number of Positions", "nPos", "Number of Equidistant Positions Along Element", GH_ParamAccess.item)

        pManager(2).Optional = True

    End Sub

    ''' <summary>
    ''' Registers all the output parameters for this component.
    ''' </summary>
    Protected Overrides Sub RegisterOutputParams(pManager As GH_Component.GH_OutputParamManager)
        pManager.AddGenericParameter("Gsa", "Gsa", "Gsa COM Object", GH_ParamAccess.item)
        pManager.AddTextParameter("Data Title", "DataTitle", "Data Title", GH_ParamAccess.item)
        pManager.AddTextParameter("Unit", "Unit", "Unit", GH_ParamAccess.item)
        pManager.AddTextParameter("Element Extract Outputs", "Output", "Element Extract Outputs {LoadCase:Position}", GH_ParamAccess.tree)
    End Sub

    ''' <summary>
    ''' This is the method that actually does the work.
    ''' </summary>
    ''' <param name="DA">The DA object is used to retrieve from inputs and store in outputs.</param>
    Protected Overrides Sub SolveInstance(DA As IGH_DataAccess)
        'Declare Variables
        Dim FileName As String = ""
        Dim ElemRefs As New List(Of Integer)
        Dim Output As New Grasshopper.DataTree(Of String)
        Dim s As Short
        Dim Axis As String
        Dim LoadCase As New List(Of String)
        Dim DataRef As Integer
        Dim RequestedPositions, nPos As Integer
        Dim myData As String
        Dim ResultsExist As Short
        Dim LoadCaseData As New List(Of String())
        Dim i, j As Integer
        Dim myDataCollector As New DataCollector
        Dim myObjectWrapper As Types.GH_ObjectWrapper

        'Import GSA File Name
        If (Not DA.GetData(0, myObjectWrapper)) Then Return
        If (Not DA.GetDataList(1, ElemRefs)) Then Return
        If (Not DA.GetData(2, Axis)) Then Axis = "default"
        If (Not DA.GetDataList(3, LoadCase)) Then Return
        If (Not DA.GetData(4, DataRef)) Then Return
        If (Not DA.GetData(5, RequestedPositions)) Then Return

        'Set up Position Data
        If RequestedPositions < 2 Then RequestedPositions = 2
        For i = 0 To LoadCase.Count - 1
            For j = 0 To RequestedPositions - 1
                Output.AddRange(New List(Of String), New GH_Path(i, j))
            Next
        Next
        RequestedPositions = RequestedPositions - 2

        'Parse Load Case Data
        LoadCaseData = ParseLoadCaseData(LoadCase)

        Dim GsaObj As ComAuto = TryCast(myObjectWrapper.Value, ComAuto)

        'Check for existing analysis results
        i = 0
        For Each item In LoadCaseData
            ResultsExist = GsaObj.CaseResultsExist(item(0), CInt(item(1)), 0)
            If ResultsExist <> 1 Then
                AddRuntimeMessage(GH_RuntimeMessageLevel.Error, "Analysis results not detected for specified case.  Ensure the model is run.")
                DA.SetData(0, CBool(False))
                Return
            End If

            'Get Data
            s = GsaObj.Output_Init(Output_Init_Flags.OP_INIT_INFINITY, Axis, LoadCase.Item(i), DataRef, RequestedPositions)
            s = GsaObj.Output_SetStage(0) ' Whole Model
            For Each Ref In ElemRefs
                nPos = GsaObj.Output_NumElemPos(Ref)
                For j = 0 To (nPos - 1)
                    If (GsaObj.Output_DataExist(Ref) <> 0 And GsaObj.Output_IsDataRef(Output_IsDataRef_Flags.OP_IS_PER_ELEM Or Output_IsDataRef_Flags.OP_IS_PER_MEMB) = 1) Then
                        myData = CStr(GsaObj.Output_Extract(Ref, j))
                        Output.Branches.Item(i * nPos + j).Add(myData)
                    Else
                        Output.Branches.Item(i * nPos + j).Add("")
                    End If
                Next
            Next
            i = i + 1
        Next

        ' Assign the data to the output parameter.
        DA.SetData(0, GsaObj)
        DA.SetData(1, CStr(GsaObj.Output_DataTitle(1)))
        DA.SetData(2, CStr(GsaObj.Output_UnitString))
        DA.SetDataTree(3, Output)

        'Send Usage Data
        myDataCollector.UsageStatistics("ElemOutputExtract")

    End Sub

    ''' <summary>
    ''' Provides an Icon for every component that will be visible in the User Interface.
    ''' Icons need to be 24x24 pixels.
    ''' </summary>
    Protected Overrides ReadOnly Property Icon() As System.Drawing.Bitmap
        Get
            'You can add image files to your project resources and access them like this:
            ' return Resources.IconForThisComponent;
            Return My.Resources.Element_ElOutput
        End Get
    End Property

    ''' <summary>
    ''' Gets the unique ID for this component. Do not change this ID after release.
    ''' </summary>
    Public Overrides ReadOnly Property ComponentGuid() As Guid
        Get
            Return New Guid("{254c8398-dea2-4d6d-a457-9e143ea69a84}")
        End Get
    End Property
End Class
```