```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using System.Windows.Forms;
6
7  namespace SKOffice
8  {
9      static class Program
10     {
11         /// <summary>
12         /// The main entry point for the application.
13         /// </summary>
14         [STAThread]
15         static void Main()
16         {
17             Application.EnableVisualStyles();
18             Application.SetCompatibleTextRenderingDefault(false);
19             Application.Run(new MainForm());
20         }
21     }
22 }
23
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Threading;
5  using System.Windows.Forms;
6  using WcfService.domain.order;
7
8  namespace SKOffice
9  {
10     public partial class MainForm : Form
11     {
12         OpenFileDialog ofd;
13         public string[] paths;
14         private List<OrderConfirmation> orderConfirmations;
15         private int updates;
16         private Thread updateThread;
17         private int updateFrequency;
18
19         /// <summary>
20         /// Instantiates the listview including the columns.
21         /// Creating and starting the thread that will auto update the list
22         /// Instantiating variables
23         /// </summary>
24         public MainForm()
25         {
26             updates = 0;
27             updateThread = new Thread(new ThreadStart(updateLoop));
28             updateFrequency = 30000;
29             updateThread.Start();
30             paths = new string[3];
31             InitializeComponent();
32
33             // Set the view to show details.
34             orderOverViewList.View = View.Details;
35
36             // Display grid lines.
37             orderOverViewList.GridLines = true;
38
39             // Create columns for the items and subitems.
40             // Width of -2 indicates auto-size.
41             orderOverViewList.Columns.Add("Order ID", 150,
                 HorizontalAlignment.Center);
42             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
43             orderOverViewList.Columns.Add("Done", 50,
                 HorizontalAlignment.Center);
44             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
45             orderOverViewList.Columns.Add("Done", 50,
                 HorizontalAlignment.Center);
46             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
47             orderOverViewList.Columns.Add("Done", 50,
                 HorizontalAlignment.Center);
48             orderOverViewList.Columns.Add("Started", 50,
                 HorizontalAlignment.Center);
```

```
49              orderOverViewList.Columns.Add("Done", 50,
                  HorizontalAlignment.Center);
50              orderOverViewList.Columns.Add("Started", 50,
                  HorizontalAlignment.Center);
51              orderOverViewList.Columns.Add("Done", 50,
                  HorizontalAlignment.Center);
52              orderOverViewList.Columns.Add("Note", -2,
                  HorizontalAlignment.Left);
53          }
54
55          private void MainForm_Load(object sender, EventArgs e)
56          {
57          }
58
59          /// <summary>
60          /// When the tabpage are clicked on, the list is updated
61          /// </summary>
62          /// <param name="sender"></param>
63          /// <param name="e"></param>
64          private void tabPage1_Click(object sender, EventArgs e)
65          {
66              updateList();
67          }
68
69
70          /// <summary>
71          /// When one of the browse buttons are clicked the user are prompted
              to browse for the e02 file.
72          /// </summary>
73          /// <param name="sender"></param>
74          /// <param name="e"></param>
75          private void browse_Click(object sender, EventArgs e)
76          {
77              ofd = new OpenFileDialog();
78
79              //fbd.RootFolder = Environment.SpecialFolder.Desktop;
80
81              Button btn = (Button)sender;
82              switch (btn.Name)
83              {
84                  case "browseE02Btn":
85                      ofd.Filter = "Text Files (.e02)|*.e02";
86                      ofd.FilterIndex = 1;
87                      ofd.Multiselect = false;
88                      ofd.ShowDialog();
89                      tb_e02.Text = ofd.FileName;
90                      paths[0] = tb_e02.Text;
91                      break;
92                  /*case "browseBlueprintBtn":
93                      ofd.Filter = "Text Files (.pdf)|*.pdf";
94                      ofd.FilterIndex = 1;
95                      ofd.Multiselect = false;
96                      ofd.ShowDialog();
97                      tb_Blueprints.Text = ofd.FileName;
98                      paths.Add(tb_Blueprints.Text);
99                      break;
```

```
100                 case "browseRequisitionBtn":
101                     ofd.Filter = "Text Files (.pdf)|*.pdf";
102                     ofd.FilterIndex = 1;
103                     ofd.Multiselect = false;
104                     ofd.ShowDialog();
105                     tb_Requisition.Text = ofd.FileName;
106                     paths.Add(tb_Requisition.Text);
107                     break;*/
108                 default:
109                     break;
110             }
111         }
112 
113         /// <summary>
114         /// When the Upload button is Clicked, it will take the file from the path and uploads it
115         /// A box will show the status for the upload, if its uploaded Green and failed Red, idle is Blue
116         /// </summary>
117         /// <param name="sender"></param>
118         /// <param name="e"></param>
119         private void uploadBtn_Click(object sender, EventArgs e)
120         {
121             feedbackE02.BackColor = System.Drawing.Color.RoyalBlue; // Feedback: idle
122             RestService.RestServiceClient rsClient = new RestService.RestServiceClient();
123 
124             //Read the content of the file into a string array
125             List<string> fileContent = new List<string>();
126             FileInfo fileInfo = new FileInfo(paths[0]);
127             fileContent.Add(fileInfo.Name);
128             using (StreamReader sReader = new StreamReader(paths[0]))
129             {
130                 while (sReader.Peek() > -1)
131                     fileContent.Add(sReader.ReadLine());
132             }
133             string msg = rsClient.addOrderConfirmation(fileContent.ToArray());
134             //Tries to add the order confirmation on the service
135             if (msg.StartsWith("OK"))
136             {
137                 feedbackE02.BackColor = System.Drawing.Color.Green; // Feedback: success
138                 tb_e02.Clear();
139                 paths[0] = "";
140             }
141             else
142                 feedbackE02.BackColor = System.Drawing.Color.Red; //Feedback: failed
143             MessageBox.Show(msg , "Service Response");
144         }
145 
146         /// <summary>
147         /// Double clicking on an order number it will open a window
148         /// </summary>
149         /// <param name="sender"></param>
```

```
150         /// <param name="e"></param>
151         private void clickedOrder(object sender, MouseEventArgs e)
152         {
153             int y = 24;
154             for (int i = 0; i < orderOverViewList.Items.Count; i++)
155             {
156                 if (e.Y >= y && e.Y <= y + 16)
157                 {
158                     Console.WriteLine(e.Y + " " + i);
159                     OrderForm orderForm = new OrderForm(orderConfirmations
                        [i]);
160                     orderForm.Show();
161                     break;
162                 }
163                 //each row is 16 pixels in height
164                 //So each element is 0-16, 17-32, 33-48
165                 y += 17;
166             }
167         }
168 
169         private delegate void UniversalVoidDelegate();
170 
171         /// <summary>
172         /// Recieves a call from the form and allows the listview to be
              updated from another thread.
173         /// </summary>
174         /// <param name="control"></param>
175         /// <param name="function"></param>
176         public static void controlInvoke(Control control, Action function)
177         {
178             if (control.IsDisposed || control.Disposing)
179                 return;
180 
181             if (control.InvokeRequired)
182             {
183                 control.Invoke(new UniversalVoidDelegate(() => controlInvoke
                      (control, function)));
184                 return;
185             }
186             function();
187         }
188 
189         /// <summary>
190         /// Updates the listview to contain orders from the database
              containing station status and notes
191         /// </summary>
192         private void updateList()
193         {
194             FormRestService.ServiceWGetClient rsClient = new
                  FormRestService.ServiceWGetClient();
195 
196             orderConfirmations = new List<OrderConfirmation>();
197             try
198             {
199                 //Clears the list
200                 orderOverViewList.Items.Clear();
```

```
201
202                 //Creates the items in the list
203                 foreach (FormRestService.OrderConfirmation oc in
                      rsClient.getAllActiveOrders())
204                 {
205                     orderConfirmations.Add((OrderConfirmation) oc);
206                     orderOverViewList.Items.Add(new ListViewItem(new[]
207                     {oc.OrderNumber + " " + oc.OrderDate.Day + "-" +
                        oc.OrderDate.Month + "-" + oc.OrderDate.Year,
208                     (oc.StationStatus.Station4 == "Active" ||
                        (oc.StationStatus.Station4 == "Done")? "X": " "),
209                     (oc.StationStatus.Station4 == "Done") ? "X" : " ",
210                     (oc.StationStatus.Station5 == "Active" ||
                        (oc.StationStatus.Station5 == "Done")? "X": " "),
211                     (oc.StationStatus.Station5 == "Done") ? "X" : " ",
212                     (oc.StationStatus.Station6 == "Active" ||
                        (oc.StationStatus.Station6 == "Done")? "X": " "),
213                     (oc.StationStatus.Station6 == "Done") ? "X" : " ",
214                     (oc.StationStatus.Station7 == "Active" ||
                        (oc.StationStatus.Station7 == "Done")? "X": " "),
215                     (oc.StationStatus.Station7 == "Done") ? "X" : " ",
216                     (oc.StationStatus.Station8 == "Active" ||
                        (oc.StationStatus.Station8 == "Done")? "X": " "),
217                     (oc.StationStatus.Station8 == "Done") ? "X" : " ",
218                     oc.Notes.Length + "" }));
219                 }
220                 //Resizes the last column to match the window.
221                 int lastIndex = orderOverViewList.Columns.Count - 1;
222                 orderOverViewList.Columns[lastIndex].AutoResize
                      (ColumnHeaderAutoResizeStyle.HeaderSize);
223             }
224             catch (Exception ex)
225             {
226                 Console.WriteLine("Error: " + ex.Message);
227             }
228         }
229
230         /// <summary>
231         /// Receives an int and return a boolean if there is an update.
232         /// </summary>
233         /// <returns></returns>
234         private bool checkServiceUpdates()
235         {
236             FormRestService.ServiceWGetClient rsClient = new
                  FormRestService.ServiceWGetClient();
237             int serviceUpdates = rsClient.getUpdates();
238             bool result = !(serviceUpdates == updates);
239             updates = serviceUpdates;
240             return result;
241         }
242
243         /// <summary>
244         /// Keep checking for an update every 30 sec
245         /// </summary>
246         private void updateLoop()
247         {
```

```
248             while (true)
249             {
250                 if (checkServiceUpdates())
251                     controlInvoke(orderOverViewList, new Action(updateList));
252                 Thread.Sleep(updateFrequency);
253             }
254         }
255 
256     }
257 }
258 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using WcfService.domain.order;
11
12 namespace SKOffice
13 {
14     public partial class OrderForm : Form
15     {
16         public OrderConfirmation OrderConfirmation { get; private set; }
17         public string OrderName { get; private set; }
18         /// <summary>
19         /// Renames the window with the OrderNumber
20         /// </summary>
21         /// <param name="orderConfirmation"></param>
22
23         public OrderForm(OrderConfirmation orderConfirmation)
24         {
25             this.OrderConfirmation = orderConfirmation;
26             InitializeComponent();
27             Text = "Order - " + orderConfirmation.OrderNumber;
28         }
29
30         /// <summary>
31         /// Makes a status check if there is a link for the Blueprint and
             Requisition button
32         /// </summary>
33         /// <param name="sender"></param>
34         /// <param name="e"></param>
35         private void OrderForm_Load(object sender, EventArgs e)
36         {
37             feedbackBp.BackColor = Color.Red;
38             feedbackReq.BackColor = Color.Red;
39             Console.WriteLine("NOTES: " + OrderConfirmation.Notes.Count);
40             foreach (OrderNote note in OrderConfirmation.Notes)
41             {
42                 addNote(note.Text);
43             }
44         }
45
46         /// <summary>
47         /// Closes the window when you press the button "Close"
48         /// </summary>
49         /// <param name="sender"></param>
50         /// <param name="e"></param>
51         private void closeBtn_Click(object sender, EventArgs e)
52         {
53             Close();
54         }
55
```

```
56          /// <summary>
57          /// Fills the text area with the note text.
58          /// </summary>
59          /// <param name="txt"></param>
60          private void addNote(string txt)
61          {
62              Console.WriteLine("Adding note: " + txt);
63              noteTxt.Text += "-------Start Note------\n";
64              noteTxt.Text += txt + "\n";
65              noteTxt.Text += "---------End Note------\n";
66          }
67      }
68  }
69
```

```
1  using WcfService.domain.order;
2  using System;
3  using System.Collections.Generic;
4  using System.IO;
5  using System.Linq;
6  using System.Net.Http;
7  using System.Runtime.Serialization;
8  using System.ServiceModel;
9  using System.ServiceModel.Web;
10 using System.Text;
11
12 namespace WcfService
13 {
14     [ServiceContract]
15     public interface IRestService
16     {
17         [OperationContract]
18         [WebInvoke(Method = "GET",
19             ResponseFormat = WebMessageFormat.Json,
20             BodyStyle = WebMessageBodyStyle.Wrapped,
21             UriTemplate = "getOrder/{orderNumber}")]
22         OrderConfirmation getOrder(string orderNumber);
23
24         [OperationContract]
25         [WebInvoke(Method="GET",
26             ResponseFormat = WebMessageFormat.Json,
27             BodyStyle = WebMessageBodyStyle.Wrapped,
28             UriTemplate = "getAllActiveOrders")]
29         List<OrderConfirmation> getAllActiveOrders();
30
31         [OperationContract]
32         [WebInvoke(Method = "GET",
33             ResponseFormat = WebMessageFormat.Json,
34             BodyStyle = WebMessageBodyStyle.Wrapped,
35             UriTemplate = "getUpdates")]
36         int getUpdates();
37
38
39         [OperationContract]
40         [WebInvoke(Method = "POST",
41             RequestFormat = WebMessageFormat.Json,
42             UriTemplate = "addOrderConfirmation")]
43         string addOrderConfirmation(List<string> fileContent);
44
45         [OperationContract]
46         [WebInvoke(Method = "POST",
47             UriTemplate = "addNote",
48             RequestFormat = WebMessageFormat.Json,
49             BodyStyle = WebMessageBodyStyle.Wrapped,
50             ResponseFormat = WebMessageFormat.Json)]
51         bool addNote(Stream stream);
52     }
53 }
54
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.Text;
7  using WcfService.domain.order;
8  using System.Net.Http;
9  using System.IO;
10 using System.Web;
11 using System.Net;
12 using System.Web.Hosting;
13 using System.ServiceModel.Web;
14 using WcfService.domain.data;
15 using WcfService.technical;
16
17 namespace WcfService
18 {
19     public class RestService : IRestService
20     {
21         public static int Updates = 0;
22
23         /// <summary>
24         /// Gets an order by the orders number
25         /// </summary>
26         /// <param name="orderNumber"></param>
27         /// <returns></returns>
28         OrderConfirmation IRestService.getOrder(string orderNumber)
29         {
30             return DBHandler.Instance.getOrder(orderNumber);
31         }
32
33         /// <summary>
34         /// Tries to create a order confirmation file from a given string
              array,
35         /// which then is parsed into an object by the OrderParser and stored
              on the DB.
36         /// </summary>
37         /// <param name="fileContent">String array of each line in the file,
              first line given is the filename</param>
38         /// <returns></returns>
39         public string addOrderConfirmation(List<string> fileContent)
40         {
41             try
42             {
43                 string fileName = fileContent[0].Split('.')[0];
44                 string fileExtension = fileContent[0].Split('.')[1];
45                 FileInfo fileInfo = new FileInfo(Path.Combine
                      (HostingEnvironment.MapPath("~/Order/" + fileName + "/"),
                      fileName + "." + fileExtension));
46
47                 if (fileInfo.Exists)
48                     return "ERROR: File already exists.";
49
50                 fileInfo.Directory.Create();
51                 using (var output = new StreamWriter(File.Create
```

```
                    (fileInfo.FullName), Encoding.Default))
52                  {
53                      for (int i = 1; i < fileContent.Count; i++)
54                      {
55                          output.WriteLine(fileContent[i]);
56                      }
57                      output.Flush();
58                      output.Close();
59                  }
60
61                  OrderConfirmation oc = OrderParser.Instance.readOrder
                      (fileInfo.FullName);
62
63                  DBHandler.Instance.createOrder(oc);
64              }
65              catch (IOException)
66              {
67                  return "ERROR: File was not uploaded correctly.";
68              }
69              Updates++;
70              return "OK: File was successfully uploaded to the service.";
71          }
72
73          /// <summary>
74          /// Returns all the active orders from the Database
75          /// </summary>
76          /// <returns>List of OrderConfirmation</returns>
77          public List<OrderConfirmation> getAllActiveOrders()
78          {
79              return DBHandler.Instance.getAllOrdersOfStatus("Active");
80          }
81
82          public int getUpdates()
83          {
84              return Updates;
85          }
86
87          /// <summary>
88          /// Saves the note from the Android and stores it in the Database
89          /// </summary>
90          /// <param name="stream">Stream containing note information</param>
91          /// <returns></returns>
92          public bool addNote(Stream stream)
93          {
94              string dataText = "";
95              try
96              {
97                  StreamReader sr = new StreamReader(stream, Encoding.UTF8);
98
99                  dataText = sr.ReadToEnd().Replace("\"", "");
100                 int index = dataText.IndexOf(';');
101                 string orderNumber = dataText.Substring(0, index).Trim();
102                 string value = dataText.Substring(index + 1).Trim();
103
104                 DBHandler.Instance.createNotes(orderNumber, new OrderNote
                      (value));
```

```
105             }
106             catch (Exception)
107             {
108                 return false;
109             }
110             Updates++;
111             return true;
112         }
113     }
114 }
115
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.ServiceModel;
6  using System.ServiceModel.Web;
7  using System.Text;
8  using System.Web.UI;
9  using WcfService.domain.order;
10
11 namespace WcfService1
12 {
13     // NOTE: You can use the "Rename" command on the "Refactor" menu to change ↵
         the interface name "IServiceWGet" in both code and config file together.
14     [ServiceContract]
15     public interface IServiceWGet
16     {
17         [OperationContract]
18         [WebInvoke(Method = "GET",
19         ResponseFormat = WebMessageFormat.Json,
20         BodyStyle = WebMessageBodyStyle.Wrapped,
21         UriTemplate = "getUpdates")]
22         int getUpdates();
23
24         [OperationContract]
25         [WebInvoke(Method = "GET",
26               ResponseFormat = WebMessageFormat.Json,
27               BodyStyle = WebMessageBodyStyle.Wrapped,
28               UriTemplate = "getAllActiveOrders")]
29         List<OrderConfirmation> getAllActiveOrders();
30     }
31 }
32
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.IO;
4  using System.Linq;
5  using System.Runtime.Serialization;
6  using System.ServiceModel;
7  using System.Text;
8  using System.Web.Hosting;
9  using WcfService;
10 using WcfService.domain.data;
11 using WcfService.domain.order;
12 using WcfService.technical;
13
14 namespace WcfService1
15 {
16     // NOTE: You can use the "Rename" command on the "Refactor" menu to change
       the class name "ServiceWGet" in code, svc and config file together.
17     // NOTE: In order to launch WCF Test Client for testing this service,
       please select ServiceWGet.svc or ServiceWGet.svc.cs at the Solution
       Explorer and start debugging.
18     public class ServiceWGet : IServiceWGet
19     {
20         public List<OrderConfirmation> getAllActiveOrders()
21         {
22             return DBHandler.Instance.getAllOrdersOfStatus("Active");
23         }
24
25         public int getUpdates()
26         {
27             return RestService.Updates;
28         }
29     }
30 }
31
```

```
 1  using WcfService.domain.order;
 2  using System;
 3  using System.Collections.Generic;
 4  using System.Data.SqlClient;
 5  using System.Data;
 6
 7  namespace WcfService.technical
 8  {
 9      class DBHandler
10      {
11
12          private static DBHandler instance;
13          private string dbName;
14          private string dbTable;
15          private string connectionString;
16
17          /// <summary>
18          /// Constructs a Singleton
19          /// </summary>
20          public static DBHandler Instance
21          {
22              get
23              {
24                  if (instance == null) instance = new DBHandler();
25                  return instance;
26              }
27              private set
28              {
29                  instance = value;
30              }
31          }
32
33          /// <summary>
34          /// Reads the Properties File for the Connection information
35          /// </summary>
36          private DBHandler() {
37              domain.Properties prop = new domain.Properties("database");
38              prop.reload();
39              if (!prop.keyExists("databaseName"))
40                  prop.set("databaseName", "localhost");
41              if (!prop.keyExists("databaseTable"))
42                  prop.set("databaseTable", "SKDB");
43              prop.Save();
44              dbName = prop.get("databaseName");
45              dbTable = prop.get("databaseTable");
46              connectionString = String.Format("Data Source={0};Initial Catalog=
                  {1};Integrated Security=True", dbName, dbTable);
47          }
48
49          // Create Methods
50
51          /// <summary>
52          /// Recieves the OrderConfirmation Object,
53          /// which then is used create an order on the database.
54          /// </summary>
55          /// <param name="orderConfirmation"></param>
```

```
56          public void createOrder(OrderConfirmation orderConfirmation)
57          {
58              try
59              {
60                  using (SqlConnection connection = new SqlConnection
                      (connectionString))
61                  {
62                      connection.Open();
63                      using (SqlCommand command = new SqlCommand("createOrder",
                        connection) { CommandType = CommandType.StoredProcedure })
64                      {
65                          //Parsing arrays of strings to a complete string
66                          string di = "";
67                          foreach (string str in orderConfirmation.DeliveryInfo)
68                              di += str + ";";
69                          di.Substring(0, di.Length - 1);
70
71                          string adi = "";
72
73                          foreach (string str in
                        orderConfirmation.AltDeliveryInfo)
74                              adi += str + ";";
75                          adi.Substring(0, adi.Length - 1);
76
77                          command.Parameters.Add(new SqlParameter
                        ("@OrderNumber", SqlDbType.VarChar, 64));
78                          command.Parameters["@OrderNumber"].Value =
                        orderConfirmation.OrderNumber;
79                          command.Parameters.Add(new SqlParameter("@OrderName",
                        SqlDbType.VarChar, 64));
80                          command.Parameters["@OrderName"].Value =
                        orderConfirmation.OrderName;
81                          command.Parameters.Add(new SqlParameter("@Delivery",
                        SqlDbType.VarChar, 256));
82                          command.Parameters["@Delivery"].Value = di;
83                          command.Parameters.Add(new SqlParameter
                        ("@AltDelivery", SqlDbType.VarChar, 256));
84                          command.Parameters["@AltDelivery"].Value = adi;
85                          command.Parameters.Add(new SqlParameter
                        ("@HousingAssociation", SqlDbType.VarChar, 64));
86                          command.Parameters["@HousingAssociation"].Value =
                        orderConfirmation.HousingAssociation;
87                          command.Parameters.Add(new SqlParameter("@StartDate",
                        SqlDbType.Date));
88                          command.Parameters["@StartDate"].Value =
                        orderConfirmation.ProducedDate.ToString("yyyy-MM-dd");
89                          command.Parameters.Add(new SqlParameter
                        ("@DeliveryDate", SqlDbType.Date));
90                          command.Parameters["@DeliveryDate"].Value =
                        orderConfirmation.OrderDate.ToString("yyyy-MM-dd");
91                          command.Parameters.Add(new SqlParameter
                        ("@DeliveryWeek", SqlDbType.VarChar, 32));
92                          command.Parameters["@DeliveryWeek"].Value =
                        orderConfirmation.Week;
93                          command.Parameters.Add(new SqlParameter
                        ("@BluePrintLink", SqlDbType.VarChar, 256));
```

```
94                          command.Parameters["@BluePrintLink"].Value = "";
95                          command.Parameters.Add(new SqlParameter
                        ("@RequisitionLink", SqlDbType.VarChar, 256));
96                          command.Parameters["@RequisitionLink"].Value = "";
97                          command.Parameters.Add(new SqlParameter
                        ("@ProgressStatus", SqlDbType.VarChar, 16));
98                          command.Parameters["@ProgressStatus"].Value =
                        orderConfirmation.Status;
99                          command.Parameters.Add(new SqlParameter("@CompanyID",
                        SqlDbType.Int));
100                         command.Parameters["@CompanyID"].Value = 1;
101                         command.ExecuteNonQuery();
102                     }
103                     connection.Close();
104
105                     foreach (OrderCategory category in
                        orderConfirmation.Categories)
106                     {
107                         createOrderCategory(orderConfirmation.OrderNumber,
                        category);
108                     }
109                 }
110             }
111             catch (SqlException ex)
112             {
113                 Console.WriteLine(ex.Message);
114                 Console.WriteLine(ex.StackTrace);
115                 throw;
116             }
117
118         }
119
120         /// <summary>
121         /// Stores a note on the database, linked to a given OrderNumber.
122         /// </summary>
123         /// <param name="orderNumber">OrderNumber of an order</param>
124         /// <param name="note">Note to store</param>
125         /// <returns></returns>
126         public List<OrderNote> createNotes(string orderNumber, OrderNote note)
127         {
128             List<OrderNote> result = new List<OrderNote>();
129             try
130             {
131                 using (SqlConnection connection = new SqlConnection
                      (connectionString))
132                 {
133                     connection.Open();
134                     using (SqlCommand command = new SqlCommand("createNotes",
                        connection) { CommandType = CommandType.StoredProcedure })
135                     {
136                         command.Parameters.Add(new SqlParameter
                        ("@OrderNumber", SqlDbType.VarChar, 64));
137                         command.Parameters["@OrderNumber"].Value =
                        orderNumber;
138                         command.Parameters.Add(new SqlParameter("@Content",
                        SqlDbType.VarChar, 1024));
```

```
139                         command.Parameters["@Content"].Value = note.Text;
140                         command.ExecuteNonQuery();
141                     }
142                     connection.Close();
143                 }
144             }
145             catch (SqlException ex)
146             {
147                 Console.WriteLine(ex.Message);
148             }
149             return result;
150         }
151 
152         /// <summary>
153         /// Stores a category from an order on the database, linked to a given
               OrderNumber.
154         /// </summary>
155         /// <param name="orderNumber">OrderNumber of an order</param>
156         /// <param name="category">Category to store</param>
157         /// <returns></returns>
158         public List<OrderCategory> createOrderCategory(string orderNumber,
              OrderCategory category)
159         {
160             List<OrderCategory> result = new List<OrderCategory>();
161             try
162             {
163                 using (SqlConnection connection = new SqlConnection
                      (connectionString))
164                 {
165                     int categoryID = -1;
166                     connection.Open();
167                     using (SqlCommand command = new SqlCommand
                          ("createOrderCategory", connection) { CommandType =
                          CommandType.StoredProcedure })
168                     {
169                         command.Parameters.Add(new SqlParameter
                          ("@OrderNumber", SqlDbType.VarChar, 64));
170                         command.Parameters["@OrderNumber"].Value =
                          orderNumber;
171                         command.Parameters.Add(new SqlParameter
                          ("@CategoryName", SqlDbType.VarChar, 64));
172                         command.Parameters["@CategoryName"].Value =
                          category.Name;
173                         //command.Parameters.Add(new SqlParameter("@new_id",
                          SqlDbType.Int, 0, "fldCategoryID").Direction =
                          ParameterDirection.Output);
174                         command.ExecuteNonQuery();
175 
176                         //categoryID = Convert.ToInt32(command.Parameters
                          ["@new_id"].Value);
177                     }
178                     connection.Close();
179 
180                     //Gets the last ID created, in this case the id for this
                      Category
181                     categoryID = getLastID("dbo.TblOrderCategory");
```

```
182
183                     //Creates all the elements for the category
184                     foreach (OrderElement element in category.Elements)
185                     {
186                         createOrderElements(element, categoryID);
187                     }
188                 }
189             }
190             catch (SqlException ex)
191             {
192                 Console.WriteLine(ex.Message);
193             }
194             return result;
195         }
196
197         /// <summary>
198         /// Stores a element from a category on the database, linked to a given CategoryID.
199         /// </summary>
200         /// <param name="element">Element to store</param>
201         /// <param name="categoryId">CategoryID to identify which category the element is linked to</param>
202         public void createOrderElements(OrderElement element, int categoryId)
203         {
204             try
205             {
206                 using (SqlConnection connection = new SqlConnection(connectionString))
207                 {
208                     connection.Open();
209                     using (SqlCommand command = new SqlCommand("createOrderElements", connection) { CommandType = CommandType.StoredProcedure })
210                     {
211                         command.Parameters.Add(new SqlParameter("@Pos", SqlDbType.VarChar, 32));
212                         command.Parameters["@Pos"].Value = element.Position;
213                         command.Parameters.Add(new SqlParameter("@Hinge", SqlDbType.VarChar, 64));
214                         command.Parameters["@Hinge"].Value = element.Hinge;
215                         command.Parameters.Add(new SqlParameter("@Finish", SqlDbType.VarChar, 64));
216                         command.Parameters["@Finish"].Value = element.Finish;
217                         command.Parameters.Add(new SqlParameter("@Amount", SqlDbType.VarChar, 32));
218                         command.Parameters["@Amount"].Value = element.Amount;
219                         command.Parameters.Add(new SqlParameter("@Unit", SqlDbType.VarChar, 32));
220                         command.Parameters["@Unit"].Value = element.Unit;
221
222                         string info = "";
223                         foreach (string str in element.ElementInfo)
224                         {
225                             info += str + ";";
226                         }
227
```

```
228                         command.Parameters.Add(new SqlParameter("@Text",
                        SqlDbType.VarChar, 256));
229                         command.Parameters["@Text"].Value = info;
230                         command.Parameters.Add(new SqlParameter
                        ("@Station4Status", SqlDbType.Bit));
231                         command.Parameters["@Station4Status"].Value =
                        element.StationStatus[0];
232                         command.Parameters.Add(new SqlParameter
                        ("@Station5Status", SqlDbType.Bit));
233                         command.Parameters["@Station5Status"].Value =
                        element.StationStatus[1];
234                         command.Parameters.Add(new SqlParameter
                        ("@Station6Status", SqlDbType.Bit));
235                         command.Parameters["@Station6Status"].Value =
                        element.StationStatus[2];
236                         command.Parameters.Add(new SqlParameter
                        ("@Station7Status", SqlDbType.Bit));
237                         command.Parameters["@Station7Status"].Value =
                        element.StationStatus[3];
238                         command.Parameters.Add(new SqlParameter
                        ("@Station8Status", SqlDbType.Bit));
239                         command.Parameters["@Station8Status"].Value =
                        element.StationStatus[4];
240                         command.Parameters.Add(new SqlParameter("@CategoryID",
                         SqlDbType.Int));
241                         command.Parameters["@CategoryID"].Value = categoryId;
242
243                         command.ExecuteNonQuery();
244                     }
245                     connection.Close();
246                 }
247             }
248             catch (SqlException ex)
249             {
250                 Console.WriteLine(ex.Message);
251             }
252         }
253
254         // Get methods
255
256         /// <summary>
257         /// Get an order from the Database by OrderNumber.
258         /// This get method also internally gets the order's categories with
              elements.
259         /// </summary>
260         /// <param name="orderNumber">OrderNumber of the order</param>
261         /// <returns></returns>
262         public OrderConfirmation getOrder(string orderNumber)
263         {
264             OrderConfirmation orderConfirmation = null;
265             try
266             {
267                 using (SqlConnection connection = new SqlConnection
                      (connectionString))
268                 {
269                     connection.Open();
```

```
270                     using (SqlCommand command = new SqlCommand("getOrder",
                        connection) { CommandType = CommandType.StoredProcedure })
271                     {
272                         command.Parameters.Add(new SqlParameter
                        ("@OrderNumber", SqlDbType.VarChar, 64));
273                         command.Parameters["@OrderNumber"].Value =
                        orderNumber;
274
275                         using (SqlDataReader dr = command.ExecuteReader())
276                         {
277                             if (dr.HasRows)
278                             {
279                                 while (dr.Read())
280                                 {
281                                     orderConfirmation = new OrderConfirmation
                                    ();
282                                     orderConfirmation.OrderNumber =
                                    dr.GetString(0);        //Order Number
283                                     orderConfirmation.OrderName = dr.GetString
                                    (1);          //Order Name
284
285                                     string[] splitName =
                                    orderConfirmation.OrderName.Split('/');
286                                     orderConfirmation.AlternativeNumber
                                    =                   //Alternative Number
287                                      splitName[0] + " " +
288                                      orderConfirmation.OrderNumber + " " +
289                                      splitName[1];
290
291                                     orderConfirmation.ProducedDate =
                                    dr.GetDateTime(2);     //Start date/produced date
292                                     orderConfirmation.OrderDate =
                                    dr.GetDateTime(3);        //Delivery date/order date
293                                     orderConfirmation.Week = dr.GetString
                                    (4);               //Delivery week
294                                     foreach (string line in dr.GetString
                                    (5).Split(';'))     //Delivery info
295                                      orderConfirmation.DeliveryInfo.Add
                                    (line);
296                                     foreach (string line in dr.GetString
                                    (6).Split(';'))     //Alt delivery info
297                                      orderConfirmation.AltDeliveryInfo.Add
                                    (line);
298                                     orderConfirmation.HousingAssociation =
                                    dr.GetString(7); //Housing Association
299                                     orderConfirmation.Status = dr.GetString
                                    (10);             //Status
300
301                                     //Company info:
302                                     orderConfirmation.CompanyInfo.Add
                                    (dr.GetString(11));    //Name
303                                     orderConfirmation.CompanyInfo.Add
                                    (dr.GetString(12));    //Address
304                                     orderConfirmation.CompanyInfo.Add
                                    (dr.GetString(13));    //ZipCode
305                                     orderConfirmation.CompanyInfo.Add
```

```
                    (dr.GetString(14));    //Phone
306                                 orderConfirmation.CompanyInfo.Add
                    (dr.GetString(15));    //FaxPhone
307                                 orderConfirmation.CompanyInfo.Add
                    (dr.GetString(16));    //CVR
308                                 orderConfirmation.CompanyInfo.Add
                    (dr.GetString(17));    //Email
309                             }
310                         }
311                     }
312                 }
313                 connection.Close();
314
315                 //Adds all the categories linked to this order
                    confirmation
316                 foreach (OrderCategory category in getOrderCategories
                    (orderNumber))
317                 {
318                     orderConfirmation.Categories.Add(category);
319                 }
320                 foreach (OrderNote note in getNotes(orderNumber))
321                 {
322                     orderConfirmation.Notes.Add(note);
323                 }
324             }
325         }
326         catch (Exception ex)
327         {
328             Console.WriteLine(ex.StackTrace);
329         }
330         return orderConfirmation;
331     }
332
333     /// <summary>
334     /// Get a list of Order Confirmations that are active.
335     /// This get method also internally gets all the orders categories
        with elements.
336     /// </summary>
337     /// <param name="status"></param>
338     /// <returns></returns>
339     public List<OrderConfirmation> getAllOrdersOfStatus(string status)
340     {
341         List<OrderConfirmation> result = new List<OrderConfirmation>();
342         try
343         {
344             using (SqlConnection connection = new SqlConnection
                    (connectionString))
345             {
346                 connection.Open();
347                 using (SqlCommand command = new SqlCommand
                    ("getAllOrdersOfStatus", connection) { CommandType =
                    CommandType.StoredProcedure })
348                 {
349                     command.Parameters.Add(new SqlParameter
                    ("@ProgressStatus", SqlDbType.VarChar, 64));
350                     command.Parameters["@ProgressStatus"].Value = status;
```

```
351
352                         using (SqlDataReader dr = command.ExecuteReader())
353                         {
354                             if (dr.HasRows)
355                             {
356                                 while (dr.Read())
357                                 {
358                                     OrderConfirmation orderConfirmation = new OrderConfirmation();
359                                     orderConfirmation.OrderNumber = dr.GetString(0);        //Order Number
360                                     orderConfirmation.OrderName = dr.GetString(1);          //Order Name
361
362                                     string[] splitName = orderConfirmation.OrderName.Split('/');
363                                     orderConfirmation.AlternativeNumber =                   //Alternative Number
364                                         splitName[0] + " " +
365                                         orderConfirmation.OrderNumber + " " +
366                                         splitName[1];
367
368                                     orderConfirmation.ProducedDate = dr.GetDateTime(2);     //Start date/produced date
369                                     orderConfirmation.OrderDate = dr.GetDateTime(3);        //Delivery date/order date
370                                     orderConfirmation.Week = dr.GetString(4);               //Delivery week
371                                     foreach (string line in dr.GetString(5).Split(';'))     //Delivery info
372                                         orderConfirmation.DeliveryInfo.Add(line);
373                                     foreach (string line in dr.GetString(6).Split(';'))     //Alt delivery info
374                                         orderConfirmation.AltDeliveryInfo.Add(line);
375                                     orderConfirmation.HousingAssociation = dr.GetString(7); //Housing Association
376                                     orderConfirmation.Status = dr.GetString(10);            //Status
377
378                                     //Company info:
379                                     orderConfirmation.CompanyInfo.Add(dr.GetString(11));    //Name
380                                     orderConfirmation.CompanyInfo.Add(dr.GetString(12));    //Address
381                                     orderConfirmation.CompanyInfo.Add(dr.GetInt32(13) + "");    //ZipCode
382                                     orderConfirmation.CompanyInfo.Add(dr.GetString(14));    //Phone
383                                     orderConfirmation.CompanyInfo.Add(dr.GetString(15));    //FaxPhone
384                                     orderConfirmation.CompanyInfo.Add(dr.GetString(16));    //CVR
385                                     orderConfirmation.CompanyInfo.Add(dr.GetString(17));    //Email
```

```
386                                     result.Add(orderConfirmation);
387                                 }
388                             }
389                         }
390                         connection.Close();
391                         //Get all categories for all orders
392                         foreach (OrderConfirmation orderConfirmation in result)
393                         {
394                             foreach (OrderCategory category in getOrderCategories(orderConfirmation.OrderNumber))
395                             {
396                                 orderConfirmation.Categories.Add(category);
397                             }
398                             foreach (OrderNote note in getNotes(orderConfirmation.OrderNumber))
399                             {
400                                 orderConfirmation.Notes.Add(note);
401                             }
402                         }
403                     }
404                 }
405             }
406             catch (Exception ex)
407             {
408                 Console.WriteLine(ex.StackTrace);
409             }
410             return result;
411         }
412 
413         /// <summary>
414         /// Gets a list of Notes by OrderNumber
415         /// </summary>
416         /// <param name="orderNumber">OrderNumber of the order</param>
417         /// <returns></returns>
418         public List<OrderNote> getNotes(string orderNumber)
419         {
420             List<OrderNote> result = new List<OrderNote>();
421             try
422             {
423                 using (SqlConnection connection = new SqlConnection(connectionString))
424                 {
425                     connection.Open();
426                     using (SqlCommand command = new SqlCommand("getNotes", connection) { CommandType = CommandType.StoredProcedure })
427                     {
428                         command.Parameters.Add(new SqlParameter("@OrderNumber", SqlDbType.VarChar, 64));
429                         command.Parameters["@OrderNumber"].Value = orderNumber;
430 
431                         using (SqlDataReader dr = command.ExecuteReader())
432                         {
433                             if (dr.HasRows)
434                             {
```

```
435                                 while (dr.Read())
436                                 {
437                                     result.Add(new OrderNote(dr.GetString
                        (2)));
438                                 }
439                             }
440                         }
441                     }
442                     connection.Close();
443                 }
444             }
445             catch (Exception ex)
446             {
447                 Console.WriteLine(ex.StackTrace);
448             }
449             return result;
450         }
451 
452         /// <summary>
453         /// Creates a list of Categories by orderNumber
454         /// </summary>
455         /// <param name="orderNumber">OrderNumber of the order</param>
456         /// <returns></returns>
457         public List<OrderCategory> getOrderCategories(string orderNumber)
458         {
459             List<OrderCategory> result = new List<OrderCategory>();
460             try
461             {
462                 using (SqlConnection connection = new SqlConnection
                      (connectionString))
463                 {
464                     connection.Open();
465                     using (SqlCommand command = new SqlCommand
                     ("getCategories", connection) { CommandType =
                     CommandType.StoredProcedure })
466                     {
467                         command.Parameters.Add(new SqlParameter
                     ("@OrderNumber", SqlDbType.VarChar, 64));
468                         command.Parameters["@OrderNumber"].Value =
                     orderNumber;
469 
470                         using (SqlDataReader dr = command.ExecuteReader())
471                         {
472                             if (dr.HasRows)
473                             {
474                                 while (dr.Read())
475                                 {
476                                     OrderCategory category = new OrderCategory
                        (dr.GetString(1));
477                                     result.Add(category);
478                                 }
479                             }
480                         }
481                     }
482                     connection.Close();
483 
```

```
484                     for (int i = 0; i < result.Count; i++)
485                     {
486                         foreach (OrderElement element in getOrderElements(i
                        +1))
487                         {
488                             result[i].Elements.Add(element);
489                         }
490                     }
491                 }
492             }
493             catch (Exception ex)
494             {
495                 Console.WriteLine(ex.StackTrace);
496             }
497             return result;
498         }
499 
500         /// <summary>
501         /// Creates a list of Elements by category ID
502         /// </summary>
503         /// <param name="categoryID">The CategoryID which the elements is
              linked to</param>
504         /// <returns></returns>
505         public List<OrderElement> getOrderElements(int categoryID)
506         {
507             List<OrderElement> elements = new List<OrderElement>();
508             try
509             {
510                 using (SqlConnection connection = new SqlConnection
                      (connectionString))
511                 {
512                     connection.Open();
513                     using (SqlCommand command = new SqlCommand
                        ("getOrderElements", connection) { CommandType =
                        CommandType.StoredProcedure })
514                     {
515                         command.Parameters.Add(new SqlParameter("@CategoryID",
                         SqlDbType.VarChar, 64));
516                         command.Parameters["@CategoryID"].Value = categoryID;
517 
518                         using (SqlDataReader dr = command.ExecuteReader())
519                         {
520                             if (dr.HasRows)
521                             {
522                                 while (dr.Read())
523                                 {
524                                     OrderElement element = new OrderElement
                        (dr.GetString(1), dr.GetString(2), dr.GetString(3),
                        dr.GetString(4), dr.GetString(5));  //(dr.GetString(0),
                        dr.GetString(1), dr.GetString(2), dr.GetString(3),
                        dr.GetString(4));
525                                     foreach (string str in dr.GetString
                        (6).Split(';'))
526                                     {
527                                         element.ElementInfo.Add(str);
528                                     }
```

```
529                                     element.StationStatus[0] = dr.GetBoolean
                        (7);
530                                     element.StationStatus[0] = dr.GetBoolean
                        (8);
531                                     element.StationStatus[0] = dr.GetBoolean
                        (9);
532                                     element.StationStatus[0] = dr.GetBoolean
                        (10);
533                                     element.StationStatus[0] = dr.GetBoolean
                        (11);
534                                     elements.Add(element);
535                                 }
536                             }
537                         }
538                     }
539                     connection.Close();
540                 }
541             }
542             catch (Exception ex)
543             {
544                 Console.WriteLine(ex.StackTrace);
545             }
546             return elements;
547         }
548 
549         /// <summary>
550         /// Retries the last created ID from a given table name.
551         /// </summary>
552         /// <param name="tableName">Table name of the desired ID</param>
553         /// <returns></returns>
554         public int getLastID(string tableName)
555         {
556             int id = -1;
557             try
558             {
559                 using (SqlConnection connection = new SqlConnection
                      (connectionString))
560                 {
561                     connection.Open();
562                     using (SqlCommand command = new SqlCommand(String.Format
                        ("SELECT IDENT_CURRENT('{0}') AS ID", tableName),
                        connection))
563                         id = Convert.ToInt32(command.ExecuteScalar());
564                     connection.Close();
565                 }
566             }
567             catch (SqlException ex)
568             {
569                 Console.WriteLine(ex.Message);
570             }
571             return id;
572         }
573     }
574 }
575 
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6  using System.IO;
7  using WcfService.domain.order;
8  using WcfService.domain.utility;
9
10 namespace WcfService.domain.data
11 {
12     public class OrderParser
13     {
14         private static OrderParser instance;
15
16         public static OrderParser Instance
17         {
18             get
19             {
20                 if (instance == null) instance = new OrderParser();
21                 return instance;
22             }
23             private set
24             {
25                 instance = value;
26             }
27         }
28
29         /// <summary>
30         /// Singleton constructor. This class can only be created from within.
31         /// </summary>
32         private OrderParser()
33         {
34
35         }
36
37         /// <summary>
38         ///
39         /// </summary>
40         /// <param name="url"></param>
41         public OrderConfirmation readOrder(string filePath)
42         {
43             OrderConfirmation result = new OrderConfirmation();
44             StreamReader stream = null;
45             try
46             {
47                 if (!filePath.ToLower().EndsWith(".e02"))
48                 {
49                     Console.Error.WriteLine("Error: Wrong file type.");
50                     return null;
51                 }
52
53                 if (!File.Exists(filePath))
54                 {
55                     Console.Error.WriteLine("Error: File doesn't exist.");
56                     return null;
```

```
57                  }
58
59                  stream = new StreamReader(filePath, Encoding.Default);
60
61                  int kitchenInfoNumber = 0;
62                  OrderElement lastElement = null;
63
64                  while (stream.Peek() > -1)
65                  {
66                      string line = stream.ReadLine();
67                      string[] lineSplit = line.Split(';');
68                      string[] date;
69
70                      if (lineSplit.Length == 0)
71                          continue;
72
73
74                      switch (lineSplit[0])
75                      {
76                          case "0": // Program header + produced date
77                              date = lineSplit[4].Split('/');
78                              string[] time = lineSplit[5].Split(':');
79
80                              DateTime dateTime = new DateTime(
81                                  ConversionUtil.stringToInt(date[2]),
82                                  ConversionUtil.stringToInt(date[1]),
83                                  ConversionUtil.stringToInt(date[0]),
84                                  ConversionUtil.stringToInt(time[0]),
85                                  ConversionUtil.stringToInt(time[1]),
86                                  ConversionUtil.stringToInt(time[2]));
87
88                              result.ProducedDate = dateTime;
89                              break;
90                          case "101": // Company Info
91                              result.CompanyInfo.Add(lineSplit[1]);
92                              result.CompanyInfo.Add(lineSplit[3]);
93                              result.CompanyInfo.Add(lineSplit[4]);
94                              result.CompanyInfo.Add(lineSplit[5]);
95                              result.CompanyInfo.Add(lineSplit[20]);
96                              result.CompanyInfo.Add(lineSplit[24]);
97                              result.CompanyInfo.Add(lineSplit[25]);
98                              result.CompanyInfo.Add(lineSplit[30]);
99                              break;
100                         case "211": // Delivery Info
101                             result.DeliveryInfo.Add(lineSplit[1]);
102                             result.DeliveryInfo.Add(lineSplit[3]);
103                             result.DeliveryInfo.Add(lineSplit[4]);
104                             result.DeliveryInfo.Add(lineSplit[5]);
105                             result.DeliveryInfo.Add(lineSplit[20]);
106                             result.DeliveryInfo.Add(lineSplit[24]);
107                             result.DeliveryInfo.Add(lineSplit[25]);
108                             break;
109                         case "212": // Alternative Delivery Address
110                             result.AltDeliveryInfo.Add(lineSplit[1]);
111                             result.AltDeliveryInfo.Add(lineSplit[3]);
112                             result.AltDeliveryInfo.Add(lineSplit[4]);
```

```
113                             result.AltDeliveryInfo.Add(lineSplit[5]);
114                             result.AltDeliveryInfo.Add(lineSplit[11]);
115                             result.AltDeliveryInfo.Add(lineSplit[19]);
116                             result.AltDeliveryInfo.Add(lineSplit[20]);
117                             result.AltDeliveryInfo.Add(lineSplit[24]);
118                             result.AltDeliveryInfo.Add(lineSplit[25]);
119                             result.AltDeliveryInfo.Add(lineSplit[26]);
120                             break;
121                         case "300": // Rightside Info
122                             result.OrderNumber = lineSplit[1];
123                             result.OrderName = lineSplit[3];
124                             date = lineSplit[4].Split('/');
125                             result.OrderDate = new DateTime(
126                                 ConversionUtil.stringToInt(date[2]),
127                                 ConversionUtil.stringToInt(date[1]),
128                                 ConversionUtil.stringToInt(date[0]), 12, 0,
                        0);
129                             result.Week = lineSplit[7];
130 
131                             string[] splitName = result.OrderName.Split('/');
132                             result.AlternativeNumber =
133                                 splitName[0] + " " +
134                                 result.OrderNumber + " " +
135                                 splitName[1];
136                             break;
137                         case "410": // Start of kitchen info
138                             result.KitchenInfo.Add("KInfo-" +
                        kitchenInfoNumber);
139                             kitchenInfoNumber++;
140                             break;
141                         case "423": // Title of the order part
142                             result.KitchenInfo.Add(lineSplit[2]);
143                             break;
144                         case "424": // Text for the order part (423)
145                             result.KitchenInfo.Add(lineSplit[2]);
146                             break;
147                         case "425": // Finish of the order part
148                             result.KitchenInfo.Add(lineSplit[2]); // Title
149                             result.KitchenInfo.Add(lineSplit[4]); //
                        Description
150                             break;
151                         case "430": // Order Category Info
152                             result.Categories.Add(new OrderCategory(lineSplit
                        [2],ConversionUtil.stringToInt(lineSplit[1])));
153                             break;
154                         case "500": // Order Element
155                             lastElement = new OrderElement(lineSplit[1], "",
                        "", lineSplit[9], lineSplit[10]);
156 
157                             int metaPos = ConversionUtil.stringToInt(lineSplit
                        [2]);
158                             if (metaPos > 0)
159                                 lastElement.Position += "." + metaPos;
160 
161                             lastElement.ElementInfo.Add(lineSplit[3]);
162                             lastElement.ElementInfo.Add(lineSplit[8]);
```

```
163                             result.findCategoryByParserID
                        (ConversionUtil.stringToInt(lineSplit[12])).Elements.Add
                        (lastElement);
164                             break;
165                         case "501": // Order Element - Hinge + Finish
166                             Console.WriteLine(line);
167                             lastElement.Hinge = lineSplit[8];
168                             lastElement.Finish = lineSplit[9];
169                             break;
170                         case "502": // Order Element - Extra info
171                             lastElement.ElementInfo.Add(lineSplit[1]);
172                             break;
173                         default:
174                             break;
175                     }
176                 }
177             }
178             catch (Exception)
179             {
180                 return null;
181             }
182             finally
183             {
184                 try
185                 {
186                     stream.Close();
187                 }
188                 catch (NullReferenceException)
189                 { }
190             }
191
192             //Instantiate the order as Active
193             result.Status = "Active";
194             return result;
195         }
196     }
197 }
198
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Runtime.Serialization;
4
5  namespace WcfService.domain.order
6  {
7      [DataContract]
8      public class OrderConfirmation
9      {
10         [DataMember]
11         public string OrderNumber { get; set; }
12         [DataMember]
13         public string AlternativeNumber { get; set; }
14         [DataMember]
15         public string OrderName { get; set; }
16         [DataMember]
17         public string Week { get; set; }
18         [DataMember]
19         public string Status { get; set; }
20         [DataMember]
21         public DateTime OrderDate { get; set; }
22         [DataMember]
23         public DateTime ProducedDate { get; set; }
24         [DataMember]
25         public string HousingAssociation { get; set; }
26         [DataMember]
27         public List<string> CompanyInfo { get; private set; }
28         [DataMember]
29         public List<string> DeliveryInfo { get; private set; }
30         [DataMember]
31         public List<string> AltDeliveryInfo { get; private set; }
32         [DataMember]
33         public List<string> KitchenInfo { get; private set; }
34         [DataMember]
35         public List<OrderCategory> Categories { get; private set; }
36         [DataMember]
37         public List<OrderNote> Notes { get; private set; }
38         [DataMember]
39         public OrderStatus StationStatus { get; private set; }
40
41         /// <summary>
42         /// instanciates the variables
43         /// </summary>
44         public OrderConfirmation()
45         {
46             OrderNumber = "";
47             AlternativeNumber = "";
48             OrderName = "";
49             Week = "";
50             Status = "";
51             HousingAssociation = "";
52             OrderDate = new DateTime();
53             ProducedDate = new DateTime();
54             CompanyInfo = new List<string>();
55             DeliveryInfo = new List<string>();
56             AltDeliveryInfo = new List<string>();
```

```
57              Categories = new List<OrderCategory>();
58              Notes = new List<OrderNote>();
59              KitchenInfo = new List<string>();
60              StationStatus = new OrderStatus();
61          }
62
63          /// <summary>
64          /// Fins the orderconfirmation with the given parserID
65          /// </summary>
66          /// <param name="parserID"></param>
67          /// <returns></returns>
68          public OrderCategory findCategoryByParserID(int parserID)
69          {
70              foreach (OrderCategory category in Categories)
71              {
72                  if (category.ParserID == parserID)
73                      return category;
74              }
75              return null;
76          }
77      }
78  }
79
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.Web;
6
7  namespace WcfService.domain.order
8  {
9      [DataContract]
10     public class OrderNote
11     {
12         [DataMember]
13         public string Text { get; set; }
14
15         public OrderNote(string text)
16         {
17             Text = text;
18         }
19     }
20 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace WcfService.domain.order
9  {
10     [DataContract]
11     public class OrderCategory
12     {
13         [DataMember]
14         public string Name { get; set; }
15         /// <summary>
16         /// Parser ID is used to assemble the categories when reading the E02
             file.
17         /// </summary>
18         public int ParserID { get; set; }
19         [DataMember]
20         public List<OrderElement> Elements { get; private set; }
21
22         public OrderCategory(string name)
23         {
24             Elements = new List<OrderElement>();
25             Name = name;
26         }
27         public OrderCategory(string name, int parserID)
28         {
29             Elements = new List<OrderElement>();
30             Name = name;
31             ParserID = parserID;
32         }
33     }
34 }
35
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace WcfService.domain.order
9  {
10     [DataContract]
11     public class OrderElement
12     {
13         [DataMember]
14         public string Position { get; set; }
15         [DataMember]
16         public List<string> ElementInfo { get; private set; }
17         [DataMember]
18         public string Hinge { get; set; }
19         [DataMember]
20         public string Finish { get; set; }
21         [DataMember]
22         public string Amount { get; set; }
23         [DataMember]
24         public string Unit { get; set; }
25         [DataMember]
26         public bool[] StationStatus { get; set; }
27
28         public OrderElement(string position, string hinge, string finish,
             string amount, string unit)
29         {
30             Position = position;
31             ElementInfo = new List<string>();
32             Hinge = hinge;
33             Finish = finish;
34             Amount = amount;
35             Unit = unit;
36             StationStatus = new bool[5];
37         }
38
39         public void updateStatus(int stationNumber, bool status)
40         {
41             StationStatus[stationNumber - 4] = status;
42         }
43
44         public bool allDone()
45         {
46             return StationStatus[0] && StationStatus[1] && StationStatus[2] &&
                 StationStatus[3] && StationStatus[4];
47         }
48     }
49 }
50
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Runtime.Serialization;
5  using System.Web;
6
7  namespace WcfService.domain.order
8  {
9      [DataContract]
10     public class OrderStatus
11     {
12         [DataMember]
13         public string Station4 { get; set; }
14         [DataMember]
15         public string Station5 { get; set; }
16         [DataMember]
17         public string Station6 { get; set; }
18         [DataMember]
19         public string Station7 { get; set; }
20         [DataMember]
21         public string Station8 { get; set; }
22         [DataMember]
23         public bool Finished { get; set; }
24
25         public OrderStatus()
26         {
27             Station4 = "NotStartet";
28             Station5 = "NotStartet";
29             Station6 = "NotStartet";
30             Station7 = "NotStartet";
31             Station8 = "NotStartet";
32             Finished = false;
33         }
34     }
35 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.IO;
5  using System.Web.Hosting;
6
7  namespace WcfService.domain
8  {
9      public class Properties
10     {
11         private Dictionary<String, String> list;
12         private FileInfo configFile;
13         private static DirectoryInfo configFolder;
14         public int Count
15         {
16             get
17             {
18                 return list.Count;
19             }
20         }
21
22         public Properties(String file)
23         {
24             configFile = new FileInfo(Path.Combine(HostingEnvironment.MapPath
                 ("~/config/"), file + ".ini"));
25             // Sets the configFolder field if needed
26             if (configFolder == null)
27                 configFolder = configFile.Directory;
28             // Creates the configFolder if needed
29             if (!configFolder.Exists)
30                 configFolder.Create();
31
32             reload(file);
33         }
34
35         public String get(String field, String defValue)
36         {
37             return (get(field) == null) ? (defValue) : (get(field));
38         }
39
40         public String get(String field)
41         {
42             return (list.ContainsKey(field)) ? (list[field]) : (null);
43         }
44
45         public void set(String field, Object value)
46         {
47             if (!list.ContainsKey(field))
48                 list.Add(field, value.ToString());
49             else
50                 list[field] = value.ToString();
51         }
52
53         public bool keyExists(string key)
54         {
55             return list.Keys.Contains(key);
```

```
56          }
57
58          public void Save()
59          {
60              Save(this.configFile);
61          }
62
63          public void Save(String filename)
64          {
65              Save(new FileInfo(configFolder.FullName + "/" + filename +
                  ".ini"));
66          }
67
68          public void Save(FileInfo fileInfo)
69          {
70              if (!fileInfo.Exists)
71                  fileInfo.Create();
72
73              StreamWriter file = new StreamWriter(fileInfo.FullName);
74
75              foreach (String prop in list.Keys.ToArray())
76              {
77                  if (!String.IsNullOrWhiteSpace(list[prop]))
78                      file.WriteLine(prop + "=" + list[prop]);
79              }
80
81              file.Close();
82          }
83
84          public void reload()
85          {
86              reload(this.configFile);
87          }
88
89          public void reload(String filename)
90          {
91              reload(new FileInfo(configFolder.FullName + "/" + filename +
                  ".ini"));
92          }
93
94          public void reload(FileInfo fileInfo)
95          {
96              list = new Dictionary<String, String>();
97
98              if (fileInfo.Exists)
99                  loadFromFile(fileInfo);
100             else
101                 fileInfo.Create();
102         }
103
104         private void loadFromFile(FileInfo fileInfo)
105         {
106             foreach (String line in File.ReadAllLines(fileInfo.FullName))
107             {
108                 if ((!String.IsNullOrEmpty(line)) &&
109                     (!line.StartsWith(";")) &&
```

```
110                     (!line.StartsWith("#")) &&
111                     (!line.StartsWith("'")) &&
112                     (line.Contains('=')))
113                 {
114                     int index = line.IndexOf('=');
115                     String key = line.Substring(0, index).Trim();
116                     String value = line.Substring(index + 1).Trim();
117 
118                     if ((value.StartsWith("\"") && value.EndsWith("\"")) ||
119                         (value.StartsWith("'") && value.EndsWith("'")))
120                     {
121                         value = value.Substring(1, value.Length - 2);
122                     }
123 
124                     try
125                     {
126                         //ignore dublicates
127                         list.Add(key, value);
128                     }
129                     catch { }
130                 }
131             }
132         }
133 
134 
135     }
136 }
137 
```