

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;           // Do not overwrite this line
5 using System.Data.SqlClient; // Insert this line as per sample
6 using System.Drawing;
7 using System.Linq;
8 using System.Text;
9 using System.Threading.Tasks;
10 using System.Windows.Forms;
11
12 namespace SimpleDataApp
13 {
14
15     public partial class NewCustomer : Form
16     {
17         // VARIABLES START
18         // Storage for IDENTITY values returned from database.
19         private int parsedCustomerID;
20         private int orderID;
21
22         /// <summary>
23         /// Verifies that the customer name text box is not empty.
24         /// </summary>
25         private bool IsCustomerNameValid()
26         {
27             if (txtCustomerName.Text == "")
28             {
29                 MessageBox.Show("Please enter a name.");
30                 return false;
31             }
32             else
33             {
34                 return true;
35             }
36         }
37
38         /// <summary>
39         /// Verifies that a customer ID and order amount have been provided.
40         /// </summary>
41         private bool IsOrderDataValid()
42         {
43             // Verify that CustomerID is present.
44             if (txtCustomerID.Text == "")
45             {
46                 MessageBox.Show("Please create customer account before placing
47                                     order.");
48                 return false;
49             }
50             // Verify that Amount isn't 0.
51             else if ((numOrderAmount.Value < 1))
52             {
53                 MessageBox.Show("Please specify an order amount.");
54                 return false;
55             }
56             else
```

```
56         {
57             // Order can be submitted.
58             return true;
59         }
60     }
61
62     /// <summary>
63     /// Clears the form data.
64     /// </summary>
65     private void ClearForm()
66     {
67         txtCustomerName.Clear();
68         txtCustomerID.Clear();
69         dtpOrderDate.Value = DateTime.Now;
70         numOrderAmount.Value = 0;
71         this.parsedCustomerID = 0;
72     }
73     //VARIABLES END
74
75     public NewCustomer()
76     {
77         InitializeComponent();
78     }
79
80     /// <summary>
81     /// Creates a new customer by calling the Sales.uspNewCustomer stored ➤
82     /// procedure.
83     /// </summary>
84     private void btnCreateAccount_Click(object sender, EventArgs e)
85     {
86         if (IsCustomerNameValid())
87         {
88             // Create the connection.
89             using (SqlConnection connection = new SqlConnection ➤
90                 (Properties.Settings.Default.connString))
91             {
92                 // Create a SqlCommand, and identify it as a stored ➤
93                 // procedure.
94                 using (SqlCommand sqlCommand = new SqlCommand ➤
95                     ("Sales.uspNewCustomer", connection))
96                 {
97                     sqlCommand.CommandType = CommandType.StoredProcedure;
98
99                     // Add input parameter for the stored procedure and ➤
100                     // specify what to use as its value.
101                     sqlCommand.Parameters.Add(new SqlParameter ➤
102                         ("@CustomerName", SqlDbType.NVarChar, 40));
103                     sqlCommand.Parameters["@CustomerName"].Value = ➤
104                         txtCustomerName.Text;
105
106                     // Add the output parameter.
107                     sqlCommand.Parameters.Add(new SqlParameter ➤
108                         ("@CustomerID", SqlDbType.Int));
109                     sqlCommand.Parameters["@CustomerID"].Direction = ➤
110                         ParameterDirection.Output;
111                 }
112             }
113         }
114     }
115 }
```

```
103         try
104         {
105             connection.Open();
106
107             // Run the stored procedure.
108             sqlCommand.ExecuteNonQuery();
109
110             // Customer ID is an IDENTITY value from the database.
111             this.parsedCustomerID = (int)sqlCommand.Parameters["@CustomerID"].Value;
112
113             // Put the Customer ID value into the read-only text box.
114             this.txtCustomerID.Text = Convert.ToString(parsedCustomerID);
115         }
116         catch
117         {
118             MessageBox.Show("Customer ID was not returned. Account could not be created.");
119         }
120         finally
121         {
122             connection.Close();
123         }
124     }
125 }
126
127 }
128
129 /// <summary>
130 /// Calls the Sales.uspPlaceNewOrder stored procedure to place an order.
131 /// </summary>
132 private void btnPlaceOrder_Click(object sender, EventArgs e)
133 {
134     // Ensure the required input is present.
135     if (IsOrderDataValid())
136     {
137         // Create the connection.
138         using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
139         {
140             // Create SqlCommand and identify it as a stored procedure.
141             using (SqlCommand sqlCommand = new SqlCommand("Sales.uspPlaceNewOrder", connection))
142             {
143                 sqlCommand.CommandType = CommandType.StoredProcedure;
144
145                 // Add the @CustomerID input parameter, which was obtained from uspNewCustomer.
146                 sqlCommand.Parameters.Add(new SqlParameter("@CustomerID", SqlDbType.Int));
147                 sqlCommand.Parameters["@CustomerID"].Value =
```

```

        this.parsedCustomerID;

148
149         // Add the @OrderDate input parameter.
150         sqlCommand.Parameters.Add(new SqlParameter
151         ("@OrderDate", SqlDbType.DateTime, 8));
152         sqlCommand.Parameters["@OrderDate"].Value =
153         dtpOrderDate.Value;
154
155         // Add the @Amount order amount input parameter.
156         sqlCommand.Parameters.Add(new SqlParameter("@Amount",
157         SqlDbType.Int));
158         sqlCommand.Parameters["@Amount"].Value =
159         numOrderAmount.Value;
160
161         // Add the @Status order status input parameter.
162         // For a new order, the status is always 0 (open).
163         sqlCommand.Parameters.Add(new SqlParameter("@Status",
164         SqlDbType.Char, 1));
165         sqlCommand.Parameters["@Status"].Value = "0";
166
167         // Add the return value for the stored procedure,
168         which is the order ID.
169         sqlCommand.Parameters.Add(new SqlParameter("@RC",
170         SqlDbType.Int));
171         sqlCommand.Parameters["@RC"].Direction =
172         ParameterDirection.ReturnValue;
173
174         try
175         {
176             //Open connection.
177             connection.Open();
178
179             // Run the stored procedure.
180             sqlCommand.ExecuteNonQuery();
181
182             // Display the order number.
183             this.orderID = (int)sqlCommand.Parameters
184             ["@RC"].Value;
185             MessageBox.Show("Order number " + this.orderID + "
186             has been submitted.");
187         }
188         catch
189         {
190             MessageBox.Show("Order could not be placed.");
191         }
192         finally
193         {
194             connection.Close();
195         }
196     }
197 }
198
199 /// <summary>
200 /// Clears the form data so another new account can be created.

```

```
193     /// </summary>
194     private void btnAddAnotherAccount_Click(object sender, EventArgs e)
195     {
196         this.ClearForm();
197     }
198
199     /// <summary>
200     /// Closes the form/dialog box.
201     /// </summary>
202     private void btnAddFinish_Click(object sender, EventArgs e)
203     {
204         this.Close();
205     }
206 }
207 }
208
```