```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;               // Do not overwrite this line
using System.Data.SqlClient;     // Insert this line as per sample
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SimpleDataApp
{

    public partial class NewCustomer : Form
    {
        // VARIABLES START
        // Storage for IDENTITY values returned from database.
        private int parsedCustomerID;
        private int orderID;

        /// <summary>
        /// Verifies that the customer name text box is not empty.
        /// </summary>
        private bool IsCustomerNameValid()
        {
            if (txtCustomerName.Text == "")
            {
                MessageBox.Show("Please enter a name.");
                return false;
            }
            else
            {
                return true;
            }
        }

        /// <summary>
        /// Verifies that a customer ID and order amount have been provided.
        /// </summary>
        private bool IsOrderDataValid()
        {
            // Verify that CustomerID is present.
            if (txtCustomerID.Text == "")
            {
                MessageBox.Show("Please create customer account before placing
                  order.");
                return false;
            }
            // Verify that Amount isn't 0.
            else if ((numOrderAmount.Value < 1))
            {
                MessageBox.Show("Please specify an order amount.");
                return false;
            }
            else
            {
                // Order can be submitted.
                return true;
            }
        }

        /// <summary>
        /// Clears the form data.
        /// </summary>
        private void ClearForm()
        {
            txtCustomerName.Clear();
            txtCustomerID.Clear();
            dtpOrderDate.Value = DateTime.Now;
            numOrderAmount.Value = 0;
            this.parsedCustomerID = 0;
        }
        //VARIABLES END

        public NewCustomer()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Creates a new customer by calling the Sales.uspNewCustomer stored procedure.
        /// </summary>
        private void btnCreateAccount_Click(object sender, EventArgs e)
        {
            if (IsCustomerNameValid())
            {
                // Create the connection.
                using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
                {
                    // Create a SqlCommand, and identify it as a stored procedure.
                    using (SqlCommand sqlCommand = new SqlCommand("Sales.uspNewCustomer", connection))
                    {
                        sqlCommand.CommandType = CommandType.StoredProcedure;

                        // Add input parameter for the stored procedure and specify what to use as its value.
                        sqlCommand.Parameters.Add(new SqlParameter("@CustomerName", SqlDbType.NVarChar, 40));
                        sqlCommand.Parameters["@CustomerName"].Value = txtCustomerName.Text;

                        // Add the output parameter.
                        sqlCommand.Parameters.Add(new SqlParameter("@CustomerID", SqlDbType.Int));
                        sqlCommand.Parameters["@CustomerID"].Direction = ParameterDirection.Output;

                        try
                        {
                            connection.Open();

                            // Run the stored procedure.
                            sqlCommand.ExecuteNonQuery();

                            // Customer ID is an IDENTITY value from the database.
                            this.parsedCustomerID = (int)sqlCommand.Parameters["@CustomerID"].Value;

                            // Put the Customer ID value into the read-only text box.
                            this.txtCustomerID.Text = Convert.ToString(parsedCustomerID);
                        }
                        catch
                        {
                            MessageBox.Show("Customer ID was not returned. Account could not be created.");
                        }
                        finally
                        {
                            connection.Close();
                        }
                    }
                }
            }
        }

        /// <summary>
        /// Calls the Sales.uspPlaceNewOrder stored procedure to place an order.
        /// </summary>
        private void btnPlaceOrder_Click(object sender, EventArgs e)
        {
            // Ensure the required input is present.
            if (IsOrderDataValid())
            {
                // Create the connection.
                using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.connString))
                {
                    // Create SqlCommand and identify it as a stored procedure.
                    using (SqlCommand sqlCommand = new SqlCommand("Sales.uspPlaceNewOrder", connection))
                    {
                        sqlCommand.CommandType = CommandType.StoredProcedure;

                        // Add the @CustomerID input parameter, which was obtained from uspNewCustomer.
                        sqlCommand.Parameters.Add(new SqlParameter("@CustomerID", SqlDbType.Int));
                        sqlCommand.Parameters["@CustomerID"].Value =
this.parsedCustomerID;

                        // Add the @OrderDate input parameter.
                        sqlCommand.Parameters.Add(new SqlParameter("@OrderDate", SqlDbType.DateTime, 8));
                        sqlCommand.Parameters["@OrderDate"].Value = dtpOrderDate.Value;

                        // Add the @Amount order amount input parameter.
                        sqlCommand.Parameters.Add(new SqlParameter("@Amount", SqlDbType.Int));
                        sqlCommand.Parameters["@Amount"].Value = numOrderAmount.Value;

                        // Add the @Status order status input parameter.
                        // For a new order, the status is always O (open).
                        sqlCommand.Parameters.Add(new SqlParameter("@Status", SqlDbType.Char, 1));
                        sqlCommand.Parameters["@Status"].Value = "O";

                        // Add the return value for the stored procedure, which is  the order ID.
                        sqlCommand.Parameters.Add(new SqlParameter("@RC", SqlDbType.Int));
                        sqlCommand.Parameters["@RC"].Direction = ParameterDirection.ReturnValue;

                        try
                        {
                            //Open connection.
                            connection.Open();

                            // Run the stored procedure.
                            sqlCommand.ExecuteNonQuery();

                            // Display the order number.
                            this.orderID = (int)sqlCommand.Parameters["@RC"].Value;
                            MessageBox.Show("Order number " + this.orderID + " has been submitted.");
                        }
                        catch
                        {
                            MessageBox.Show("Order could not be placed.");
                        }
                        finally
                        {
                            connection.Close();
                        }
                    }
                }
            }
        }

        /// <summary>
        /// Clears the form data so another new account can be created.
        /// </summary>
        private void btnAddAnotherAccount_Click(object sender, EventArgs e)
        {
            this.ClearForm();
        }

        /// <summary>
        /// Closes the form/dialog box.
        /// </summary>
        private void btnAddFinish_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

```