Answer on Question#38723 - Programming, C#

1. The development team of SoftSols Inc. has revamped the software according to the requirements of FlyHigh Airlines and is in the process of testing the software. While testing the software, the team encounters the following issues:
The operations-related data of FlyHigh Airlines is stored in a central database. The software fails to respond to user inputs, if there is a connectivity problem with the database. Add the code snippet that the development team should use to ensure that the application shows a userfriendly message, if such a situation arises in future.[5 Marks]
The application used to calculate the cost of carrying additional luggage results in erroneous amount, if the weight of the luggage is a fractional number. Help the development team modify the code snippet so that the cost of carrying additional luggage is calculated correctly. [5 Marks]

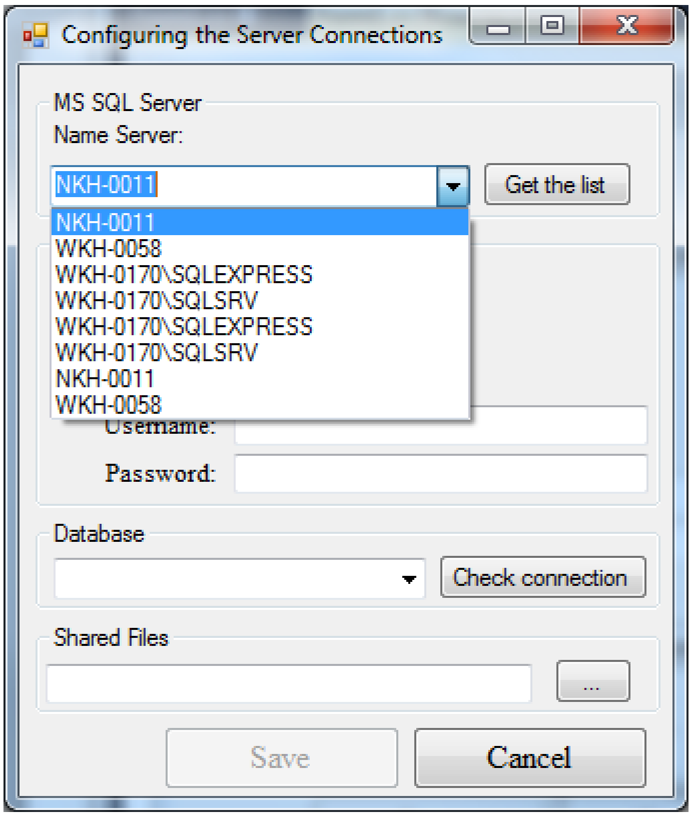
Solution.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;
using System.Data.Sql;

namespace DbProgram
{

    public partial class Form1 : Form
    {
        public Form1()
        {

            InitializeComponent();


            this.textBox4.Text = "";
            if (Trusted_connection == true)
            {
                this.radioButton1.Checked = true;
                this.radioButton2.Checked = false;
                this.textBox4.Enabled = false;
                this.textBox5.Enabled = false;
            }
            else
            {
                this.radioButton1.Checked = false;
                this.radioButton2.Checked = true;
                this.textBox4.Enabled = true;
                this.textBox5.Enabled = true;
            }
        }
```

```csharp
bool Trusted_connection;


private void button2_Click_1(object sender, EventArgs e)
{
    Close();
}

private void button3_Click(object sender, EventArgs e)
{

    if (radioButton1.Checked == true)
        Trusted_connection = true;
    else
        Trusted_connection = false;
    Close();

}

private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    button3.Enabled = false;
    if (this.radioButton1.Checked == true)
    { Trusted_connection = true; }
    else
    { Trusted_connection = false; }

    if (Trusted_connection == true)
    {
        this.textBox4.Enabled = false;
        this.textBox5.Enabled = false;
    }
    else
    {
        this.textBox4.Enabled = true;
        this.textBox5.Enabled = true;

    }
}

private void radioButton2_CheckedChanged(object sender, EventArgs e)
{
    button3.Enabled = false;
    if (this.radioButton2.Checked == false)
    { Trusted_connection = true; }
    else
    { Trusted_connection = false; }

    if (Trusted_connection == true)
    {
        this.textBox4.Enabled = false;
        this.textBox5.Enabled = false;
    }
    else
    {
        this.textBox4.Enabled = true;
        this.textBox5.Enabled = true;

    }
}

private void button5_Click_1(object sender, EventArgs e)
{
    FolderBrowserDialog openDirDialog = new FolderBrowserDialog();
    openDirDialog.SelectedPath = textBox3.Text;
```

```
            if (openDirDialog.ShowDialog() == DialogResult.OK)
            {
                try
                {

                    textBox3.Text = openDirDialog.SelectedPath.ToString();


                }
                catch (Exception ex)
                {
                    MessageBox.Show("Error opening the specified path! " + ex.Message);
                }
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            button3.Enabled = false;
            comboBox2.Enabled = false;
            comboBox2.Items.Clear();

            button6.Text = "Search...";
            button6.Enabled = false;

            #region Getting the server name of the ODBC-sources

            String[] instances = SqlLocator.GetServers();

            foreach (String element in instances)
            {
                comboBox2.Items.Add(element);
            }

            #endregion
            #region Getting a list of servers in a standard way
            DataTable t = SqlDataSourceEnumerator.Instance.GetDataSources();
            for (int i = 0; i < t.Rows.Count; i++)
            {
                string ServerName = t.Rows[i]["ServerName"].ToString();
                string InstanceName = t.Rows[i]["InstanceName"].ToString();
                string x = (InstanceName.Length > 0 ? (ServerName + "\\" + InstanceName) :
ServerName);
                comboBox2.Items.Add(x);
            }
            #endregion

            if (comboBox2.Items.Count != 0)
            {
                comboBox2.Text = comboBox2.Items[0].ToString();

                try
                {
                    GetDataBase();
                }
                catch
                {
                    MessageBox.Show("The selected server is not available. \nPlease choose
another server", "Attention", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }

            button6.Text = "Get the list";
            comboBox2.Enabled = true;
            button6.Enabled = true;

        }
        #region Getting a list of databases
        private void GetDataBase()
        {
            comboBox1.Items.Clear();
            comboBox1.Text = "";
            string Security;
            if (radioButton1.Checked == true)
            {
                Security = "Integrated Security=true";
            }
            else
            {
                Security = @"Persist Security Info=True;User ID=" + textBox5.Text +
";Password=" + textBox4.Text + "";

            }

            if (comboBox1.Text != "")
            { Security = "Initial Catalog=" + comboBox1.Text + ";" + Security; }

            SqlConnection sqlConn = new SqlConnection(@"Server=" + comboBox2.Text + ";" +
Security);

            try
            {
                sqlConn.Open();

                SqlCommand sqlCmd = new SqlCommand();
                sqlCmd.Connection = sqlConn;
                sqlCmd.CommandType = CommandType.StoredProcedure;
                sqlCmd.CommandText = "sp_helpdb";

                SqlDataAdapter da = new SqlDataAdapter(sqlCmd);
                DataSet ds = new DataSet();
                da.Fill(ds);

                foreach (DataRow row in ds.Tables[0].Rows)
                {
                    comboBox1.Items.Add(row["name"].ToString());
                }
                sqlConn.Close();



                if (comboBox2.Text == "")
                {
                    comboBox1.Enabled = false;
                    button3.Enabled = false;
                    button7.Enabled = false;
                }
                else
                {
                    comboBox1.Enabled = true;
                    button7.Enabled = true;
                }
            }
            catch
            {
                MessageBox.Show("The selected server is not available.\nSelect a different
server.", "Attention", MessageBoxButtons.OK, MessageBoxIcon.Information);
                string test = comboBox2.Text;
            }
        }
        #endregion



        private void button7_Click(object sender, EventArgs e)
        {
            String Security = "";
            if (comboBox1.Text == "")
            {
                comboBox1.Text = "master";
            }

            if (radioButton2.Checked == true)
            {
                Security = @"Persist Security Info=True;User ID=" + textBox5.Text +
";Password=" + textBox4.Text + "";
            }
            else if (radioButton1.Checked == true)
            {
                Security = "Integrated Security=true";
            }

            SqlConnection sqlConn = new SqlConnection(@"Server=" + comboBox2.Text +
";Initial Catalog=" + comboBox1.Text + ";" + Security);

            try
            {
                string _databasename = comboBox1.Text;
                sqlConn.Open();
                MessageBox.Show("Testing the connection is made.", "Connecting to a
database", MessageBoxButtons.OK, MessageBoxIcon.Information);
                SqlCommand sqlCmd = new SqlCommand();
                sqlCmd.Connection = sqlConn;
                sqlCmd.CommandType = CommandType.StoredProcedure;
                sqlCmd.CommandText = "sp_helpdb";

                SqlDataAdapter da = new SqlDataAdapter(sqlCmd);
                DataSet ds = new DataSet();
                da.Fill(ds);

                foreach (DataRow row in ds.Tables[0].Rows)
                {
                    comboBox1.Items.Add(row["name"].ToString());
                }
                sqlConn.Close();
                comboBox1.Text = _databasename;
                button3.Enabled = true;
            }
            catch
            {
                MessageBox.Show("Testing the connection failed.\n\nSQL Server is not
available.\nMaybe not correctly specify a username and password.", "Connecting to a
database", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }


        }

        private void ServerOptionsForm_Load(object sender, EventArgs e)
        {

        }
```

```
    }

    public class SqlLocator
    {
        [DllImport("odbc32.dll")]
        private static extern short SQLAllocHandle(short hType, IntPtr inputHandle, out
IntPtr outputHandle);
        [DllImport("odbc32.dll")]
        private static extern short SQLSetEnvAttr(IntPtr henv, int attribute, IntPtr
valuePtr, int strLength);
        [DllImport("odbc32.dll")]
        private static extern short SQLFreeHandle(short hType, IntPtr handle);
        [DllImport("odbc32.dll", CharSet = CharSet.Ansi)]
        private static extern short SQLBrowseConnect(IntPtr hconn, StringBuilder inString,
            short inStringLength, StringBuilder outString, short outStringLength,
            out short outLengthNeeded);

        private const short SQL_HANDLE_ENV = 1;
        private const short SQL_HANDLE_DBC = 2;
        private const int SQL_ATTR_ODBC_VERSION = 200;
        private const int SQL_OV_ODBC3 = 3;
        private const short SQL_SUCCESS = 0;

        private const short SQL_NEED_DATA = 99;
        private const short DEFAULT_RESULT_SIZE = 1024;
        private const string SQL_DRIVER_STR = "DRIVER=SQL SERVER";

        private SqlLocator() { }

        public static string[] GetServers()
        {
            string[] retval = null;
            string txt = string.Empty;
            IntPtr henv = IntPtr.Zero;
            IntPtr hconn = IntPtr.Zero;
            StringBuilder inString = new StringBuilder(SQL_DRIVER_STR);
            StringBuilder outString = new StringBuilder(DEFAULT_RESULT_SIZE);
            short inStringLength = (short)inString.Length;
            short lenNeeded = 0;

            try
            {
                if (SQL_SUCCESS == SQLAllocHandle(SQL_HANDLE_ENV, henv, out henv))
                {
                    if (SQL_SUCCESS == SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
(IntPtr)SQL_OV_ODBC3, 0))
                    {
                        if (SQL_SUCCESS == SQLAllocHandle(SQL_HANDLE_DBC, henv, out hconn))
                        {
                            if (SQL_NEED_DATA == SQLBrowseConnect(hconn, inString,
inStringLength, outString,
                                DEFAULT_RESULT_SIZE, out lenNeeded))
                            {
                                if (DEFAULT_RESULT_SIZE < lenNeeded)
                                {
                                    outString.Capacity = lenNeeded;
                                    if (SQL_NEED_DATA != SQLBrowseConnect(hconn, inString,
inStringLength, outString,
                                        lenNeeded, out lenNeeded))
                                    {
                                        throw new ApplicationException("Unabled to aquire
SQL Servers from ODBC driver.");
                                    }
                                }
                                txt = outString.ToString();
                                int start = txt.IndexOf("{") + 1;
                                int len = txt.IndexOf("}") - start;
                                if ((start > 0) && (len > 0))
                                {
                                    txt = txt.Substring(start, len);
                                }
                                else
                                {
                                    txt = string.Empty;
                                }
                            }
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                //Throw away any error if we are not in debug mode
#if (DEBUG)
                MessageBox.Show(ex.Message, "Acquire SQL Servier List Error");
#endif
                txt = string.Empty;
            }
            finally
            {
                if (hconn != IntPtr.Zero)
                {
                    SQLFreeHandle(SQL_HANDLE_DBC, hconn);
                }
                if (henv != IntPtr.Zero)
                {
                    SQLFreeHandle(SQL_HANDLE_ENV, hconn);
                }
            }

            if (txt.Length > 0)
            {
                retval = txt.Split(",".ToCharArray());
            }

            return retval;
        }
    }



}
```


Configuring the Server Connections
MS SQL Server
Name Server:
NKH-0011
Get the list
NKH-0011
WKH-0058
WKH-0170\SQLEXPRESS
WKH-0170\SQLSRV
WKH-0170\SQLEXPRESS
WKH-0170\SQLSRV
NKH-0011
WKH-0058
Username:
Password:
Database
Check connection
Shared Files
...
Save
Cancel