```
/*--------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 8th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeTabletWindows is a program
 *   compatible with the Windows tablet. It's a
 *   program that can be use as tactil graphic
 *   interface to order the component linked with
 *   the other Raspberry Pi.
\*--------------------------------------------------*/

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Windows.Networking;
using Windows.Networking.Sockets;
using Windows.Storage.Streams;
using Windows.UI.Xaml;

namespace RaspiHomeTabletWindows
{
    public class CommunicationWithServer
    {
        #region Fields
        #region Constants
        // Default information to connect on the server
        private const int PORT = 54565;
        //// Need to be changed fo each configuration
        private const string IPSERVER = "10.134.97.117";// "192.168.2.8";

        private const string FORMATSTRING = "IPRasp={0};Location=
          {1};Component={2}";
        private const string COMMUNICATIONSEPARATOR = "@";

        // Important need to be changed if it's another room!
        private const string LOCATION = "Salon";
        private const string COMPONENT = "Tablet";
        private const string RPINAME = "Tablet_" + LOCATION;

        private const int MESSAGE_FULL_LENGHT = 512;
        #endregion

        #region Variables
        private StreamSocket _socket = new StreamSocket();
        private StreamSocketListener _listener = new StreamSocketListener();
        private List<StreamSocket> _connections = new List<StreamSocket>();
        private bool _isConnected = false;
        private bool _connecting = false;

        private Windows.Storage.ApplicationDataContainer localSettings =
    Windows.Storage.ApplicationData.Current.LocalSettings;

```

```
        private string _messageCommand = "";

        private string _nameButtonClicked = "";

        DispatcherTimer _dTimer = null;
        #endregion
        #endregion

        #region Properties

        public StreamSocket Socket
        {
            get
            {
                return _socket;
            }

            set
            {
                _socket = value;
            }
        }

        public StreamSocketListener Listener
        {
            get
            {
                return _listener;
            }

            set
            {
                _listener = value;
            }
        }

        public List<StreamSocket> Connections
        {
            get
            {
                return _connections;
            }

            set
            {
                _connections = value;
            }
        }

        public bool IsConnected
        {
            get
            {
                return _isConnected;
            }

            set
            {
                _isConnected = value;
            }
        }

        public bool Connecting
        {
            get
            {
                return _connecting;
            }

            set
            {
                _connecting = value;
            }
        }

        public string MessageCommand
        {
            get
            {
                return _messageCommand;
            }

            set
            {
                _messageCommand = value;
            }
        }

        public string NameButtonClicked
        {
            get
            {
                return _nameButtonClicked;
            }

            set
            {
                _nameButtonClicked = value;
            }
        }
        #endregion

        #region Constructors
        /// <summary>
        /// Constructor: Initializer
        /// </summary>
        public CommunicationWithServer()
        {
            Connect();

            this._dTimer = new DispatcherTimer();
            this._dTimer.Interval = new TimeSpan(10);
            this._dTimer.Tick += _dTimer_Tick;

            this._dTimer.Start();
        }
        #endregion

        #region Events
        private void _dTimer_Tick(object sender, object e)
        {
            if (localSettings.Values["SendMessageToServer"] != null)
            {
                var messageToSend = localSettings.Values
                  ["SendMessageToServer"];
                this.SendCommandToServer(messageToSend.ToString());
                localSettings.Values.Remove("SendMessageToServer");
            }
        }
        #endregion

        #region Methods
        /// <summary>
        /// Connect the raspberry to the server
        /// </summary>
        private async void Connect()
        {
            try
            {
                this.Connecting = true;
                await this.Socket.ConnectAsync(new HostName(IPSERVER),
                  PORT.ToString());
                SendForInitialize();
                this.Connecting = false;
                this.IsConnected = true;

                WaitForData(this.Socket);
            }
            catch (Exception)
            {
                this.Connecting = false;
                this.IsConnected = false;
            }
        }

        /// <summary>
        /// Listen the traffic on the port
        /// </summary>
        private async void Listen()
        {
            this.Listener.ConnectionReceived += listenerConnectionReceived;
            await this.Listener.BindServiceNameAsync(PORT.ToString());
        }

        void listenerConnectionReceived(StreamSocketListener sender,
          StreamSocketListenerConnectionReceivedEventArgs args)
        {
            this.Connections.Add(args.Socket);

            WaitForData(args.Socket);
        }

        /// <summary>
        /// Send the message in input to output
        /// </summary>
        /// <param name="socket"></param>
        /// <param name="message"></param>
        private async void SendMessage(StreamSocket socket, string message)
        {
            DataWriter dataWriter = new DataWriter(socket.OutputStream);
            var len = dataWriter.MeasureString(message); // Gets the UTF-8 string length.
            dataWriter.WriteInt32((int)len);
            dataWriter.WriteString(message);
            var ret = await dataWriter.StoreAsync();
            dataWriter.DetachStream();
        }

        /// <summary>
        /// Send to initialize the raspberry to the server
        /// </summary>
        private void SendForInitialize()
        {
            SendMessage(this.Socket, string.Format(COMMUNICATIONSEPARATOR + RPINAME + COMMUNICATIONSEPARATOR + "Connection:" + FORMATSTRING, GetHostName(), LOCATION, COMPONENT));
        }

        /// <summary>
        /// Send the command to the server
        /// </summary>
        public void SendCommandToServer(string message)
        {
            SendMessage(this.Socket, COMMUNICATIONSEPARATOR + "Send:" + message);
            this.MessageCommand = message;
        }

        /// <summary>
        /// Wait data readed if exist
        /// </summary>
        /// <param name="socket"></param>
        private async void WaitForData(StreamSocket socket)
        {
            DataReader dataReader = new DataReader(socket.InputStream);
            dataReader.InputStreamOptions = InputStreamOptions.Partial;
            var messageLenght = dataReader.UnconsumedBufferLength;
            uint stringBytes = messageLenght;

            try
            {
                // Read modification in the stream
                stringBytes = await dataReader.LoadAsync(MESSAGE_FULL_LENGHT);

```

```
                    // read message
                    string messageRead = dataReader.ReadString(stringBytes);

                    await Task.Delay(TimeSpan.FromMilliseconds(200));
                    // Store value
                    localSettings.Values["ReceiveMessageFromServer"] =
                      messageRead;
                }
                catch (Exception e)
                {
                    string output = e.Message;

                    if (messageLenght < 1)
                        return;
                }

                WaitForData(socket);
            }

            /// <summary>
            /// Get the ip of the raspberry
            /// </summary>
            /// <returns>return a string like 192.168.1.2</returns>
            public string GetHostName()
            {
                List<string> IpAddress = new List<string>();
                var Hosts =
                  Windows.Networking.Connectivity.NetworkInformation.GetHostNames
                  ().ToList();
                foreach (var Host in Hosts)
                {
                    string IP = Host.DisplayName;
                    IpAddress.Add(IP);
                }
                return IpAddress.Last();
            }
            #endregion
        }
    }
```