```
/*-------------------------------------------------*\
 * Author    : Salvi Cyril
 * Date      : 7th juny 2017
 * Diploma   : RaspiHome
 * Classroom : T.IS-E2B
 *
 * Description:
 *      RaspiHomeServer is a server TCP. It's the m
 *      ain program, where all command pass before
 *      to be reply to the good client.
\*-------------------------------------------------*/

using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Net.Sockets;
using System.Reflection;
using System.Text;

namespace RaspiHomeServer
{
    public class CommandFilter
    {
        #region Variables
        private RaspiHomeCommands _rhCommands;
        private string _sentence = "";
        #endregion

        #region Properties
        public RaspiHomeCommands RhCommands
        {
            get
            {
                return _rhCommands;
            }

            set
            {
                _rhCommands = value;
            }
        }

        public string Sentence
        {
            get
            {
                return _sentence;
            }

            set
            {
                _sentence = value;
            }
        }
        #endregion
```

```
57
58          #region Constructor
59          /// <summary>
60          /// Constructor: Initializer
61          /// </summary>
62          public CommandFilter()
63          {
64              this.RhCommands = new RaspiHomeCommands();
65          }
66          #endregion
67
68          #region Methods
69          /// <summary>
70          /// Receive the order and treat to find raspberrys with the order
71          /// </summary>
72          /// <param name="paramSentence"> Sentence in entrence </param>
73          /// <param name="paramRaspberryClients"> List of clients informations
              </param>
74          /// <param name="paramClientsName"> Dictionnary of every clients
              registered </param>
75          /// <returns></returns>
76          public List<TcpClient> ApplyFilter(string paramSentence,
              List<RaspberryClient> paramRaspberryClients, Dictionary<string,
              Dictionary<RaspberryClient, TcpClient>> paramClientsName)
77          {
78              try
79              {
80                  // Remove characters
81                  this.Sentence = RemoveDiacritics(paramSentence);
82
83                  List<TcpClient> result = new List<TcpClient>();
84
85                  string action = "";
86                  string location = "";
87                  string componentWithoutAction = "";
88
89                  // Get the component word in the sentence
90                  string component = this.GetComponentFromSentence
                      (this.Sentence);
91
92                  Type componentType = null;
93                  string actionValue = "";
94
95                  // Different usage of the order between an component with
                      action and without one
96                  // Writes values and send to the client the information or
                      read values
97                  if (component != "")
98                  {
99                      // Get the action word with in the sentence
100                     action = this.GetActionFromSentence(this.Sentence);
101                     // Get the property name with the action word
102                     actionValue = ReadValueOfSelectedComponent(action);
103                     // Get the class type founded with the component name
104                     componentType = this.GetComponentType(component);
105                 }
```

```
106                 else
107                 {
108                     // Get the sensor component word in the sentence
109                     componentWithoutAction =
                    GetIndependantComponentFromSentence(this.Sentence);
110                     // Get the class type founded with the component name
111                     componentType = this.GetComponentType
                    (componentWithoutAction);
112                 }
113
114                 // Check every clients
115                 foreach (var rpiClient in paramRaspberryClients)
116                 {
117                     // Get the location word in the sentence
118                     location = this.GetSentenceLocationOrRaspberryLocation
                    (this.Sentence, rpiClient);
119
120                     // Check every clients at this location
121                     if (rpiClient.Location.ToLower() == location.ToLower())
122                     {
123                         // Check every clients at this location with this
                    component
124                         foreach (var itemType in rpiClient.Components)
125                         {
126                             // Check if the type is the same
127                             if (itemType.GetType() == componentType)
128                             {
129                                 if (action != "")
130                                 {
131                                     // Write the new value string informations
132                                     this.WriteValue(itemType, action,
                    itemType.GetType().GetProperty(actionValue));
133                                     foreach (var name in
                    paramClientsName.Keys)
134                                     {
135                                         if (paramClientsName[name].ContainsKey
                    (rpiClient))
136                                         {
137                                             // Add the TCPClient inside the
                    dictionnay
138                                             result.Add(paramClientsName[name]
                    [rpiClient]);
139                                         }
140                                     }
141                                 }
142                                 else
143                                 {
144                                     foreach (var name in
                    paramClientsName.Keys)
145                                     {
146                                         if (paramClientsName[name].ContainsKey
                    (rpiClient))
147                                         {
148                                             // Add the TCPClient inside the
                    dictionnay
149                                             result.Add(paramClientsName[name]
```

```
                    [rpiClient]);
150                                     }
151                                 }
152                             }
153                         }
154                     }
155                 }
156             }
157
158             return result;
159         }
160         catch (Exception ex)
161         {
162             string errorCommandFilter = ex.Message;
163             return null;
164         }
165     }
166
167     /// <summary>
168     /// Find location if exist, else all location
169     /// </summary>
170     /// <param name="sentence"></param>
171     /// <returns></returns>
172     private string GetSentenceLocationOrRaspberryLocation(string sentence, ↵
            RaspberryClient rpiClient)
173     {
174         string result = "";
175         string[] words = sentence.ToLower().Split(' ');
176
177         foreach (var word in words)
178         {
179             if (this._rhCommands.RaspiHomeLocationKnown.Contains(word))
180             {
181                 result = word;
182                 break;
183             }
184         }
185
186         if (result == "" || result == "maison")
187             result = rpiClient.Location;
188
189         return result;
190     }
191
192     /// <summary>
193     /// Get the componnent called
194     /// </summary>
195     /// <param name="sentence"></param>
196     /// <returns></returns>
197     private string GetComponentFromSentence(string sentence)
198     {
199         string result = "";
200         string[] words = sentence.ToLower().Split(' ');
201
202         foreach (var word in words)
203         {
```

```
204                 if (this._rhCommands.RaspiHomeComponentKnown.Contains(word))
205                 {
206                     result = word;
207                     break;
208                 }
209             }
210 
211             return result;
212         }
213 
214         /// <summary>
215         /// Get the componnent called without special component connected to a special action
216         /// </summary>
217         /// <param name="sentence"></param>
218         /// <returns></returns>
219         private string GetIndependantComponentFromSentence(string sentence)
220         {
221             string result = "";
222             string[] words = sentence.ToLower().Split(' ');
223 
224             foreach (var word in words)
225             {
226                 if (this._rhCommands.RaspiHomeComponentWithoutActionKnown.Contains(word))
227                 {
228                     result = word;
229                     break;
230                 }
231             }
232 
233             return result;
234         }
235 
236         /// <summary>
237         /// Find location exist
238         /// </summary>
239         /// <param name="action"></param>
240         /// <returns> the action linked to the action word </returns>
241         private string GetActionFromSentence(string sentence)
242         {
243             string result = "";
244             string[] words = sentence.ToLower().Split(' ');
245 
246             foreach (var word in words)
247             {
248                 if (this._rhCommands.RaspiHomeActionKnown.Contains(word))
249                 {
250                     result = word;
251                     break;
252                 }
253             }
254 
255             return result;
256         }
```

```
257
258         /// <summary>
259         /// Find all client who have the object in the sentence
260         /// </summary>
261         /// <param name="componentName"></param>
262         /// <returns>the object type</returns>
263         private Type GetComponentType(string componentName)
264         {
265             Type result = null;
266             Type[] types = Assembly.GetExecutingAssembly().GetTypes();
267
268             foreach (var typeOfComonent in types)
269             {
270                 if (typeOfComonent.Name ==
                      this._rhCommands.RaspiLanguageTranslation[componentName])
271                 {
272                     result = typeOfComonent;
273                     break;
274                 }
275             }
276
277             return result;
278         }
279
280         /// <summary>
281         /// Read properties value of classes
282         /// </summary>
283         /// <param name="actionName"> name used to change the good property </
              param>
284         /// <returns> return the name of the property to change the value </
              returns>
285         private string ReadValueOfSelectedComponent(string actionName)
286         {
287             string result = "";
288
289             foreach (var actionKeys in
                    this._rhCommands.RaspiBooleanCommandTranslation.Keys)
290                 if (actionKeys == actionName)
291                 {
292                     // Find the Value of the dictionary trough the inner
                        dictionary to get the first value
293                     result = this.RhCommands.RaspiBooleanCommandTranslation
                        [actionName].First().Key;
294                     break;
295                 }
296
297             return result;
298         }
299
300         /// <summary>
301         /// Search the val to change
302         /// </summary>
303         /// <param name="component"></param>
304         /// <param name="action"></param>
305         /// <param name="typeVariable"></param>
306         private void WriteValue(Component component, string action,
```

```
            PropertyInfo typeVariable)
307         {
308             switch (typeVariable.PropertyType.Name)
309             {
310                 case "Boolean":
311                     // Set the new value dynamicaly with value registered in
                        an boolean dictionary
312                     typeVariable.SetValue(component,
                        this._rhCommands.RaspiBooleanCommandTranslation[action]
                        [typeVariable.Name]);
313                     break;
314                 case "Double":
315                     break;
316                 case "Int16":
317                 case "Int32":
318                 case "Int64":
319                     break;
320             }
321         }
322
323
324
325         /// <summary>
326         /// Stack Overflow solution to delete accents in strings
327         /// http://stackoverflow.com/questions/249087/how-do-i-remove-
            diacritics-accents-from-a-string-in-net
328         /// </summary>
329         /// <param name="str"></param>
330         /// <returns></returns>
331         static string RemoveDiacritics(string sentence)
332         {
333             var normalizedString = sentence.Normalize
                  (NormalizationForm.FormD);
334             var stringBuilder = new StringBuilder();
335
336             foreach (var c in normalizedString)
337             {
338                 var unicodeCategory = CharUnicodeInfo.GetUnicodeCategory(c);
339                 if (unicodeCategory != UnicodeCategory.NonSpacingMark)
340                 {
341                     stringBuilder.Append(c);
342                 }
343             }
344
345             return stringBuilder.ToString().Normalize
                  (NormalizationForm.FormC);
346         }
347         #endregion
348     }
349 }
350
```