```
LinksPlatform's Platform.Communication.Protocol.Lino Class Library
     ./csharp/Platform.Communication.Protocol.Lino/ILinksGroupListExtensions.cs
   using System.Collections.Generic;
   using System.Runtime.CompilerServices;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
   namespace Platform.Communication.Protocol.Lino
6
        /// <summary>
        /// <para>
9
        /// Represents the links group list extensions.
10
11
        /// </para>
        /// <para></para>
12
        /// </summary>
13
        public static class ILinksGroupListExtensions
15
            /// <summary>
16
            /// <para>
17
            /// Returns the links list using the specified groups.
18
            /// </para>
19
            /// <para></para>
20
            /// </summary>
            /// <param name="groups">
22
            /// <para>The groups.</para>
23
            /// <para></para>
            /// </param>
25
            /// <returns>
26
            /// <para>The list.</para>
            /// <para></para>
            /// </returns>
29
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
30
31
            public static List<Link> ToLinksList(this IList<LinksGroup> groups)
32
                var list = new List<Link>();
33
                for (var i = 0; i < groups.Count; i++)</pre>
                {
                     groups[i].AppendToLinksList(list);
36
37
                return list;
38
            }
39
        }
   }
41
    ./csharp/Platform.Communication.Protocol.Lino/IListExtensions.cs
   using Platform.Collections;
   using System;
using System.Collections.Generic;
2
   using System.Linq;
   using System.Runtime.CompilerServices;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Lino
9
10
        /// <summary>
11
        /// <para>
12
        /// Represents the list extensions.
13
        /// </para>
        /// <para></para>
15
        /// </summary>
16
        public static class IListExtensions
17
18
            /// <summary>
19
            /// <para>
            /// Formats the links.
21
            /// </para>
22
            /// <para></para>
23
            /// </summary>
^{24}
            /// <param name="links">
25
            /// <para>The links.</para>
26
            /// <para></para>
            /// </param>
28
            /// <returns>
29
            /// <para>The string</para>
30
            /// <para></para>
            /// </returns>
32
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```
public static string Format(this IList<Link> links) => string.Join(Environment.NewLine,
34
                links.Select(1 => 1.ToString()));
35
            /// <summary>
36
            /// <para>
37
            /// Formats the links.
            /// </para>
39
            /// <para></para>
40
            /// </summary>
41
            /// <param name="links">
42
            /// <para>The links.</para>
43
            /// <para></para>
44
            /// </param>
            /// <param name="lessParentheses">
46
            /// <para>The less parentheses.</para>
47
            /// <para></para>
            /// </param>
49
            /// <returns>
50
            /// <para>The string</para>
            /// <para></para>
            /// </returns>
53
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
54
            public static string Format(this IList<Link> links, bool lessParentheses)
55
56
                 if (lessParentheses == false)
57
                     return links.Format();
59
                 }
60
                 else
                 {
62
                     return string.Join(Environment.NewLine, links.Select(1 =>
63
                      → 1.ToString().TrimSingle('(').TrimSingle(')')));
                 }
            }
65
        }
66
   }
67
    ./csharp/Platform.Communication.Protocol.Lino/Link.cs
   using System;
   using System.Collections.Generic;
   using System.Runtime.CompilerServices;
   using System. Text;
   using Platform.Collections;
   using Platform.Collections.Lists;
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
   namespace Platform.Communication.Protocol.Lino
10
11
12
        /// <summary>
        /// <para>
13
        /// The link
14
        /// </para>
15
        /// <para></para>
16
        /// </summary>
17
        public struct Link : IEquatable<Link>
18
19
            /// <summary>
20
            /// <para>
            /// The id.
            /// </para>
23
            /// <para></para>
/// </summary>
24
25
            public readonly string Id;
26
            /// <summary>
28
            /// <para>
29
            ^{\prime\prime\prime} The values.
30
            /// </para>
31
            /// <para></para>
32
            /// </summary>
33
            public readonly IList<Link> Values;
34
35
            /// <summary>
36
            /// <para>
37
            /// Initializes a new <see cref="Link"/> instance.
38
            /// </para>
            /// <para></para>
40
            /// </summary>
```

```
/// <param name="id">
42
             /// <para>A id.</para>
43
             /// <para></para>
44
             /// </param>
45
             /// <param name="values">
             /// <para>A values.</para>
47
             /// <para></para>
48
             /// </param>
49
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
             public Link(string id, ÎList<Link> values) => (Ĭd, Values) = (id, values);
51
             /// <summary>
             /// <para> /// Initializes a new <see cref="Link"/> instance.
54
55
             /// </para>
             /// <para></para>
57
             /// </summary>
58
             /// <param name="values">
             /// <para>A values.</para>
60
             /// <para></para>
61
             /// </param>
62
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
63
             public Link(IList<Link> values) : this(null, values) { }
64
             /// <summary>
66
             /// <para>
67
             /// Initializes a new <see cref="Link"/> instance.
68
             /// </para>
69
             /// <para></para>
70
             /// </summary>
71
             /// <param name="values">
             /// <para>A values.</para>
73
             /// <para></para>
74
             /// </param>
75
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
76
             public Link(params Link[] values) : this(null, values) { }
77
             /// <summary>
/// <para>
79
80
             /// Initializes a new <see cref="Link"/> instance.
81
             /// </para>
82
             /// <para></para>
83
             /// </summary>
84
             /// <param name="id">
             /// <para>A id.</para>
86
             /// <para></para>
87
             /// </param>
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
89
             public Link(string id) : this(id, null) { }
90
             /// <summary>
/// <para>
92
93
             /// Returns the string.
94
             /// </para>
95
             /// <para></para>
96
             /// </summary>
             /// <returns>
             /// <para>The string</para>
99
             /// <para></para>
100
             /// </returns>
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
102
             public override string ToString() => Values.IsNullOrEmpty() ? $\frac{\$"({Id})"}{!}":
103

    GetLinkValuesString();

             /// <summary>
105
             /// <para>
106
             /// Gets the link values string.
107
             /// </para>
             /// <para></para>
109
             /// </summary>
110
             /// <returns>
111
             /// <para>The string</para>
             /// <para></para>
113
             /// </returns>
114
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
             private string GetLinkValuesString() => Id == null ? $\|\big|\|(\{\text{GetValuesString()}}\)\] :
116

    $\"({Id}: {GetValuesString()})";
```

117

```
/// <summary>
118
             /// <para>
             /// Gets the values string.
120
             /// </para>
121
             /// <para></para>
             /// </summary>
123
             /// <returns>
124
             /// <para>The string</para>
125
             /// <para></para>
             /// </returns>
127
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
128
             public string GetValuesString()
130
                  var sb = new StringBuilder();
131
132
                  for (int i = 0; i < Values.Count; i++)</pre>
133
                       if (i > 0)
134
                       {
135
                           sb.Append(' ');
136
137
                      sb.Append(GetValueString(Values[i]));
138
139
                  return sb.ToString();
140
             }
141
             /// <summary>
143
             /// <para>
144
             /// Simplifies this instance.
145
             /// </para>
             /// <para></para>
147
             /// </summary>
148
             /// <returns>
             /// <para>The link</para>
150
             /// <para></para>
151
             /// </returns>
152
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
153
             public Link Simplify()
154
155
                  if (Values.IsNullOrEmpty())
                  {
157
158
                      return this;
                  }
159
                  else if (Values.Count == 1)
160
161
                      return Values[0];
                  }
163
                  else
                  {
165
                      var newValues = new Link[Values.Count];
166
                      for (int i = 0; i < Values.Count; i++)</pre>
167
                           newValues[i] = Values[i].Simplify();
169
170
                      return new Link(Id, newValues);
171
                  }
172
             }
173
             /// <summary>
175
             /// <para>
176
             /// Combines the other.
177
             /// </para>
178
             /// <para></para>
179
             /// </summary>
180
             /// <param name="other">
181
             /// <para>The other.</para>
/// <para></para>
182
183
             /// </param>
             /// <returns>
185
             /// <para>The link</para>
186
             /// <para></para>
187
             /// </returns>
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
189
             public Link Combine(Link other) => new Link(this, other);
190
191
             /// <summary>
192
             /// <para>
             /// Gets the value string using the specified value.
194
             /// </para>
195
```

```
/// <para></para>
196
             /// </summary>
             /// <param name="value">
198
             /// <para>The value.</para>
199
             /// <para></para>
             /// </param>
201
             /// <returns>
202
             /// <para>The string</para>
203
             /// <para></para>
             /// </returns>
205
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
206
            public static string GetValueString(Link value) => value.ToLinkOrIdString();
208
             /// <summary>
209
             /// <para>
210
             /// Returns the link or id string.
211
             /// </para>
212
             /// <para></para>
             /// </summary>
214
             /// <returns>
215
             /// <para>The string</para>
216
             /// <para></para>
217
             /// </returns>
218
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
219
            public string ToLinkOrIdString() => Values.IsNullOrEmpty() ? Id : ToString();
221
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
223
            public static implicit operator Link(string value) => new Link(value);
224
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static implicit operator Link((string, IList<Link>) value) => new
226
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
228
            public static implicit operator Link((Link, Link) value) => new Link(value.Item1,
229

    value.Item2);
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
231
            public static implicit operator Link((Link, Link, Link) value) => new Link(value.Item1,
232

    value.Item2, value.Item3);
233
             /// <summary>
234
             /// <para>
235
             /// Determines whether this instance equals.
237
             /// </para>
             /// <para></para>
238
             /// </summary>
239
             /// <param name="obj">
^{240}
            /// <para>The obj.</para>
241
            /// <para></para>
242
             /// </param>
             /// <returns>
244
             /// <para>The bool</para>
245
             /// <para></para>
246
             /// </returns>
247
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
248
            public override bool Equals(object obj) => obj is Link link ? Equals(link) : false;
249
250
             /// <summary>
251
             /// <para>
             /// Gets the hash code.
253
             /// </para>
254
             /// <para></para>
255
             /// </summary>
             /// <returns>
257
             /// <para>The int</para>
258
             /// <para></para>
             /// </returns>
260
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
261
            public override int GetHashCode() => (Id, Values.GenerateHashCode()).GetHashCode();
262
263
             /// <summary>
264
             /// <para>
            /// Determines whether this instance equals.
266
             /// </para>
267
             /// <para></para>
268
             /// </summary>
269
             /// <param name="other">
270
```

```
/// <para>The other.</para>
271
             /// <para></para>
272
             /// </param>
273
             /// <returns>
274
             /// <para>The bool</para>
             /// <para></para>
276
             /// </returns>
277
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
278
             public bool Equals(Link other) => Id == other.Id && Values.EqualTo(other.Values);
279
280
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
             public static bool operator ==(Link left, Link right) => left.Equals(right);
282
283
284
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
             public static bool operator !=(Link left, Link right) => !(left == right);
285
        }
286
    }
287
     ./csharp/Platform.Communication.Protocol.Lino/LinksGroup.cs
    using System;
    using System.Collections.Generic;
 2
    using System.Runtime.CompilerServices;
    using Platform.Collections;
 4
    using Platform.Collections.Lists;
    #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
 9
    namespace Platform.Communication.Protocol.Lino
10
        /// <summary>
11
        /// <para>
12
        ^{\prime\prime\prime} The links group.
13
        /// </para>
14
        /// <para></para>
15
        /// </summary>
16
        public struct LinksGroup : IEquatable<LinksGroup>
17
18
             /// <summary>
19
             /// <para>
20
             /// Gets or sets the link value.
             /// </para>
22
             /// <para></para>
23
             /// </summary>
24
            public Link Link
25
26
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
28
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
                 set:
30
             }
32
             /// <summary>
33
             /// <para>
34
             /// Gets or sets the groups value.
35
             /// </para>
36
             /// <para></para>
             /// </summary>
38
             public IList<LinksGroup> Groups
39
40
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
41
                 [MethodImpl(MethodImplOptions.AggressiveInlining)]
43
44
             }
45
             /// <summary>
47
             /// <para>
48
             /// Initializes a new <see cref="LinksGroup"/> instance.
49
             /// </para>
50
             /// <para></para>
51
             /// </summary>
52
             /// <param name="link">
             /// <para>A link.</para>
             /// <para></para>
55
             /// </param>
56
             /// <param name="groups">
             /// <para>A groups.</para>
58
             /// <para></para>
59
             /// </param>
```

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
61
             public LinksGroup(Link link, IList<LinksGroup> groups)
63
                 Link = link;
                 Groups = groups;
65
             }
66
67
             /// <summary>
68
             /// <para>
             /// Initializes a new <see cref="LinksGroup"/> instance.
70
             /// </para>
71
             /// <para></para>
72
             /// </summary>
73
             /// <param name="link">
74
             /// <para>A link.</para>
75
             /// <para></para>
             /// </param>
77
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
78
             public LinksGroup(Link link) : this(link, null) { }
79
80
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
81
             public static implicit operator List<Link>(LinksGroup value) => value.ToLinksList();
83
             /// <summary>
             /// <para>
85
             /// Returns the links list.
86
             /// </para>
87
             /// <para></para>
             /// </summary>
89
             /// <returns>
90
             /// <para>The list.</para>
91
             /// <para></para>
92
             /// </returns>
93
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
             public List<Link> ToLinksList()
96
                 var list = new List<Link>();
97
                 AppendToLinksList(list);
                 return list;
99
             }
101
             /// <summary>
102
             /// <para>
103
             /// Appends the to links list using the specified list.
104
             /// </para>
105
             /// <para></para>
106
             /// </summary>
107
             /// <param name="list">
108
             /// <para>The list.</para>
109
             /// <para></para>
110
             /// </param>
111
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
112
             public void AppendToLinksList(List<Link> list) => AppendToLinksList(list, Link, this);
114
             /// <summary>
115
             /// <para>
116
             /// Appends the to links list using the specified list.
117
             /// </para>
118
             /// <para></para>
             /// </summary>
120
             /// <param name="list">
121
             /// <para>The list.</para>
122
             /// <para></para>
123
             /// </param>
124
             /// <param name="dependency">
125
             /// <para>The dependency.</para>
             /// <para></para>
127
             /// </param>
128
             /// <param name="group">
129
             /// <para>The group.</para>
130
             /// <para></para>
131
             /// </param>
132
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
             public static void AppendToLinksList(List<Link> list, Link dependency, LinksGroup group)
134
135
                 list.Add(dependency)
136
                 var groups = group.Groups;
137
                 if (!groups.IsNullOrEmpty())
138
```

```
139
                     for (int i = 0; i < groups.Count; i++)</pre>
140
141
                         var innerGroup = groups[i];
142
                         AppendToLinksList(list, dependency.Combine(innerGroup.Link), innerGroup);
144
                 }
145
            }
146
147
             /// <summary>
148
            /// <para>
149
             /// Determines whether this instance equals.
             /// </para>
151
             /// <para></para>
152
             /// </summary>
            /// <param name="obj">
154
            /// <para>The obj.</para>
155
             /// <para></para>
156
             /// </param>
157
             /// <returns>
158
             /// <para>The bool</para>
159
             /// <para></para>
160
             /// </returns>
161
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
162
            public override bool Equals(object obj) => obj is LinksGroup linksGroup ?
             164
             /// <summary>
165
             /// <para>
            /// Gets the hash code.
167
            /// </para>
168
            /// <para></para>
169
            /// </summary>
170
            /// <returns>
171
            /// <para>The int</para>
172
             /// <para></para>
173
             /// </returns>
174
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
175
            public override int GetHashCode() => (Link, Groups.GenerateHashCode()).GetHashCode();
177
             /// <summary>
178
             /// <para>
179
            /// Determines whether this instance equals.
180
            /// </para>
181
             /// <para></para>
            /// </summary>
183
            /// <param name="other">
184
             /// <para>The other.</para>
185
             /// <para></para>
186
            /// </param>
187
            /// <returns>
188
             /// <para>The bool</para>
             /// <para></para>
190
             /// </returns>
191
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
192
            public bool Equals(LinksGroup other) => Link == other.Link &&
193
                Groups.EqualTo(other.Groups);
194
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static bool operator ==(LinksGroup left, LinksGroup right) => left.Equals(right);
196
197
             [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static bool operator !=(LinksGroup left, LinksGroup right) => !(left == right);
199
        }
200
201
1.5
     ./csharp/Platform.Communication.Protocol.Lino/ .cs
    #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
    using System.Runtime.CompilerServices;
    namespace Platform.Communication.Protocol.Lino
 5
        /// <summary>
        /// <para>
        /// The .
        /// </para>
10
        /// <para></para>
```

```
/// </summary>
12
13
        public struct
14
            /// <summary>
            /// <para>
            /// The link.
17
            /// </para>
18
            /// <para></para>
19
            /// </summary>
20
            public readonly Link Link;
22
            /// <summary>
23
            /// <para>
^{24}
            /// Initializes a new <see cref="_"/> instance.
25
            /// </para>
26
            /// <para></para>
            /// </summary>
            /// <param name="id">
29
            /// <para>A id.</para>
30
            /// <para></para>
31
            /// </param>
32
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
33
            public _(Link id) => Link = id;
35
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
            public static implicit operator _(Link value) => new _(value);
37
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
39
40
            public static implicit operator _(string id) => new _(id);
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
42
            public static implicit operator _((string, Link) value) => new Link(value.Item1, new
43

    Link[] { value.Item2 });
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
45
            public static implicit operator _((string, Link, Link) value) => new Link(value.Item1,
46
            → new Link[] { value.Item2, value.Item3 });
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
48
            public static implicit operator _((string, Link, Link, Link) value) => new
49

→ Link(value.Item1, new Link[] { value.Item2, value.Item3, value.Item4 });

50
            [MethodImpl(MethodImplOptions.AggressiveInlining)]
51
            public static implicit operator Link(_ value) => value.Link;
       }
53
54
1.6
    ./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs
   using Xunit;
2
   namespace Platform.Communication.Protocol.Lino.Tests
4
        /// <summary>
5
        /// <para>
        /// Represents the parser tests.
       /// </para>
        /// <para></para>
        /// </summary>
10
        public static class ParserTests
11
12
            /// <summary>
13
            /// <para>
14
            /// Tests that single link test.
            /// </para>
            /// <para></para>
/// </summary>
17
18
            [Fact]
19
            public static void SingleLinkTest()
20
21
                var source = 0"(address: source target)";
22
                var parser = new Parser();
                var links = parser.Parse(source);
24
                var target = links.Format();
25
                Assert.Equal(source, target);
            }
27
            /// <summary>
            /// <para>
30
            /// Tests that two links test.
```

```
/// </para>
32
             /// <para></para>
33
             /// </summary>
34
             [Fact]
35
             public static void TwoLinksTest()
37
                 var source = 0"(first: x y)
38
    (second: a b)";
39
                 var parser = new Parser();
40
                 var links = parser.Parse(source);
41
                 var target = links.Format();
42
                 Assert.Equal(source, target);
43
             }
45
             /// <summary>
             /// <para>
47
             /// Tests that parse and stringify test.
48
             /// </para>
             /// <para></para>
50
             /// </summary>
51
             [Fact]
52
             public static void ParseAndStringifyTest()
53
54
                 var source = @"(papa (lovesMama: loves mama))
55
    (son lovesMama)
56
    (daughter lovesMama)
57
    (all (love mama))";
58
59
                 var parser = new Parser();
                 var links = parser.Parse(source);
60
                 var target = links.Format();
61
                 Assert. Equal(source, target);
62
             }
64
             /// <summary>
65
             /// <para>
66
             /// Tests that bug test.
67
             /// </para>
68
             /// <para></para>
             /// </summary>
70
             [Fact]
71
72
             public static void BugTest()
73
                 var source = 0"(ignore conan-center-index repository)";
74
                 var links = (new Platform.Communication.Protocol.Lino.Parser()).Parse(source);
75
                 var target = links.Format();
                 Assert.Equal(source, target);
77
             }
78
79
             /// <summary>
80
             /// <para>
81
             /// Tests that parse and stringify with less parentheses test.
             /// </para>
83
             /// <para></para>
84
             /// </summary>
85
             [Fact]
86
             public static void ParseAndStringifyWithLessParenthesesTest()
87
88
                 var source = 0"lovesMama: loves mama
89
    papa lovesMama
    son lovesMama
91
    daughter lovesMama
92
    all (love mama)";
93
                 var parser = new Parser();
94
                 var links = parser.Parse(source);
95
                 var target = links.Format(lessParentheses: true);
96
                 Assert. Equal(source, target);
97
             }
99
             /// <summary>
100
             /// <para>
101
             \ensuremath{///} \bar{\text{Tests}} that significant whitespace test.
102
             /// </para>
103
             /// <para></para>
             /// </summary>
105
             [Fact]
106
             public static void SignificantWhitespaceTest()
108
             {
                 var source = @"
109
```

```
users
110
111
        user1
                  43
113
114
             name
                  first
                      John
116
                  last
117
                      Williams
118
             location
119
                  New York
120
121
                  23
         user2
123
124
             id
                  56
             name
126
                  first
127
128
                      Igor
                  middle
129
130
                      Petrovich
                  last
131
                      Ivanov
132
             location
133
                  Moscow
134
             age
                  20";
136
                  var target = 0"(users)
137
     (users user1)
138
    ((users user1) id)
139
    (((users user1) id) 43)
140
    ((users user1) name)
    (((users user1) name) first)
142
    ((((users user1) name) first) John)
143
     (((users user1) name) last)
144
    ((((users user1) name) last) Williams)
145
     ((users user1) location)
146
     (((users user1) location) (New York))
    ((users user1) age)
    (((users user1) age) 23)
149
    (users user2)
150
     ((users user2) id)
    (((users user2) id) 56)
152
    ((users user2) name)
153
    (((users user2) name) first)
154
    ((((users user2) name) first) Igor)
155
    (((users user2) name) middle)
156
     ((((users user2) name) middle) Petrovich)
157
    (((users user2) name) last)
    ((((users user2) name) last) Ivanov)
159
    ((users user2) location)
160
    (((users user2) location) Moscow)
    ((users user2) age)
162
    (((users user2) age) 20)";
163
                  var parser = new Parser();
164
                  var links = parser.Parse(source);
                  var formattedLinks = links.Format();
166
                  Assert.Equal(target, formattedLinks);
167
             }
168
         }
169
170
     ./csharp/Platform.Communication.Protocol.Lino.Tests/TupleTests.cs
    using System.Collections.Generic;
using Xunit;
    namespace Platform.Communication.Protocol.Lino.Tests
 5
         /// <summary>
 6
         /// <para>
         /// Represents the tuple tests.
         /// </para>
 9
         /// <para></para>
10
         /// </summary>
         public class TupleTests
12
13
             /// <summary>
14
             /// <para>
15
             /// Tests that tuple to link test.
16
             /// </para>
```

```
/// <para></para>
/// </summary>
18
19
              [Fact]
^{20}
              public void TupleToLinkTest()
21
                   var source = 0"(papa (lovesMama: loves mama))
23
    (son lovesMama)
24
    (daughter lovesMama)
25
    (all (love mama))";
26
                   var parser = new Parser();
27
                   var links = parser.Parse(source);
28
                   var targetFromString = links.Format();
30
31
                   IList<Link> constructedLinks = new List<Link>()
32
                        ("papa", (_)("lovesMama", "loves", "mama")), ("son", "lovesMama"), ("daughter", "lovesMama"), ("all", ("love", "mama")),
33
34
36
                   };
37
                   var targetFromTuples = constructedLinks.Format();
38
                   Assert.Equal(targetFromString, targetFromTuples);
39
              }
40
         }
41
    }
42
```

Index

- ./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs, 9 ./csharp/Platform.Communication.Protocol.Lino.Tests/ParserTests.cs, 9
 ./csharp/Platform.Communication.Protocol.Lino.Tests/TupleTests.cs, 11
 ./csharp/Platform.Communication.Protocol.Lino/ILinksGroupListExtensions.cs, 1
 ./csharp/Platform.Communication.Protocol.Lino/IListExtensions.cs, 1
 ./csharp/Platform.Communication.Protocol.Lino/Link.cs, 2

- ./csharp/Platform.Communication.Protocol.Lino/LinksGroup.cs, 6 ./csharp/Platform.Communication.Protocol.Lino/_.cs, 8