

# LinksPlatform's Platform.Communication.Protocol.Lino Class Library

## 1.1 ./Platform.Communication.Protocol.Lino/IListExtensions.cs

```
1 using Platform.Collections;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Runtime.CompilerServices;
6
7 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9 namespace Platform.Communication.Protocol.Lino
10 {
11     public static class IListExtensions
12     {
13         [MethodImpl(MethodImplOptions.AggressiveInlining)]
14         public static string Format(this IList<Link> links) => string.Join(Environment.NewLine,
15             ↪ links.Select(l => l.ToString()));
16
17         [MethodImpl(MethodImplOptions.AggressiveInlining)]
18         public static string Format(this IList<Link> links, bool lessParentheses)
19         {
20             if (lessParentheses == false)
21             {
22                 return links.Format();
23             }
24             else
25             {
26                 return string.Join(Environment.NewLine, links.Select(l =>
27                     ↪ l.ToString().TrimSingle('(').TrimSingle(')')));
28             }
29         }
30     }
31 }
```

## 1.2 ./Platform.Communication.Protocol.Lino/Link.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Runtime.CompilerServices;
4 using System.Text;
5 using Platform.Collections;
6 using Platform.Collections.Lists;
7
8 #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
9
10 namespace Platform.Communication.Protocol.Lino
11 {
12     public struct Link : IEquatable<Link>
13     {
14         public string Id
15         {
16             [MethodImpl(MethodImplOptions.AggressiveInlining)]
17             get;
18             [MethodImpl(MethodImplOptions.AggressiveInlining)]
19             set;
20         }
21
22         public IList<Link> Values
23         {
24             [MethodImpl(MethodImplOptions.AggressiveInlining)]
25             get;
26             [MethodImpl(MethodImplOptions.AggressiveInlining)]
27             set;
28         }
29
30         [MethodImpl(MethodImplOptions.AggressiveInlining)]
31         public Link(string id, IList<Link> values) => (Id, Values) = (id, values);
32
33         [MethodImpl(MethodImplOptions.AggressiveInlining)]
34         public Link(IList<Link> values) : this(null, values) { }
35
36         [MethodImpl(MethodImplOptions.AggressiveInlining)]
37         public Link(params Link[] values) : this(null, values) { }
38
39         [MethodImpl(MethodImplOptions.AggressiveInlining)]
40         public Link(string id) : this(id, null) { }
41
42         [MethodImpl(MethodImplOptions.AggressiveInlining)]
43         public override string ToString() => Values.IsNullOrEmpty() ? $"{Id}" :
44             ↪ GetLinkValuesString();
45     }
46 }
```

```

45 [MethodImpl(MethodImplOptions.AggressiveInlining)]
46 private string GetLinkValuesString() => Id == null ? $"{GetValuesString()}" :
    ↳ $"{Id}: {GetValuesString()}";
47
48 [MethodImpl(MethodImplOptions.AggressiveInlining)]
49 public string GetValuesString()
50 {
51     var sb = new StringBuilder();
52     for (int i = 0; i < Values.Count; i++)
53     {
54         if (i > 0)
55         {
56             sb.Append(' ');
57         }
58         sb.Append(GetValueString(Values[i]));
59     }
60     return sb.ToString();
61 }
62
63 [MethodImpl(MethodImplOptions.AggressiveInlining)]
64 public Link AddDependency(Link dependency)
65 {
66     if (Values.IsNullOrEmpty())
67     {
68         return new Link(new Link(dependency), this);
69     }
70     else
71     {
72         var firstValue = Values[0];
73         if (firstValue.Id == null)
74         {
75             var newValues = new List<Link>();
76             newValues.Add(firstValue.AddDependency(dependency));
77             newValues.AddSkipFirst(Values);
78             return new Link(newValues);
79         }
80         else
81         {
82             if (Values.Count > 1)
83             {
84                 return new Link(new Link(dependency), new Link(Values));
85             }
86             else
87             {
88                 var newValues = new List<Link>();
89                 newValues.Add(new Link(dependency));
90                 newValues.AddAll(Values);
91                 return new Link(newValues);
92             }
93         }
94     }
95 }
96
97 [MethodImpl(MethodImplOptions.AggressiveInlining)]
98 public Link Simplify()
99 {
100     if (Values.IsNullOrEmpty())
101     {
102         return this;
103     }
104     else if (Values.Count == 1)
105     {
106         return Values[0];
107     }
108     else
109     {
110         var newValues = new Link[Values.Count];
111         for (int i = 0; i < Values.Count; i++)
112         {
113             newValues[i] = Values[i].Simplify();
114         }
115         return new Link(Id, newValues);
116     }
117 }
118
119 [MethodImpl(MethodImplOptions.AggressiveInlining)]
120 public static string GetValueString(Link value) => value.ToLinkOrIdString();
121

```

```

122 [MethodImpl(MethodImplOptions.AggressiveInlining)]
123 public string ToLinkOrIdString() => Values.IsNullOrEmpty() ? Id : ToString();
124
125 [MethodImpl(MethodImplOptions.AggressiveInlining)]
126 public static implicit operator Link(string value) => new Link(value);
127
128 [MethodImpl(MethodImplOptions.AggressiveInlining)]
129 public static implicit operator Link(ValueTuple<string, IList<Link>> value) => new
    ↳ Link(value.Item1, value.Item2);
130
131 [MethodImpl(MethodImplOptions.AggressiveInlining)]
132 public override bool Equals(object obj) => obj is Link link ? Equals(link) : false;
133
134 [MethodImpl(MethodImplOptions.AggressiveInlining)]
135 public override int GetHashCode() => (Id, Values.GenerateHashCode()).GetHashCode();
136
137 [MethodImpl(MethodImplOptions.AggressiveInlining)]
138 public bool Equals(Link other) => Id == other.Id && Values.EqualTo(other.Values);
139
140 [MethodImpl(MethodImplOptions.AggressiveInlining)]
141 public static bool operator ==(Link left, Link right) => left.Equals(right);
142
143 [MethodImpl(MethodImplOptions.AggressiveInlining)]
144 public static bool operator !=(Link left, Link right) => !(left == right);
145 }
146 }

```

### 1.3 ./Platform.Communication.Protocol.Lino.Tests/ParserTests.cs

```

1 using Xunit;
2
3 namespace Platform.Communication.Protocol.Lino.Tests
4 {
5     public static class ParserTests
6     {
7         [Fact]
8         public static void ParseAndStringifyTest()
9         {
10             var source = @"(papa (lovesMama: loves mama))
11 (son lovesMama)
12 (daughter lovesMama)
13 (all (love mama))";
14             var parser = new Parser();
15             var links = parser.Parse(source);
16             var target = links.Format();
17             Assert.Equal(source, target);
18         }
19
20         [Fact]
21         public static void ParseAndStringifyWithLessParenthesesTest()
22         {
23             var source = @"lovesMama: loves mama
24 papa lovesMama
25 son lovesMama
26 daughter lovesMama
27 all (love mama)";
28             var parser = new Parser();
29             var links = parser.Parse(source);
30             var target = links.Format(lessParentheses: true);
31             Assert.Equal(source, target);
32         }
33
34         [Fact]
35         public static void SignificantWhitespaceTest()
36         {
37             var source = @"
38 users
39     user1
40         id
41             43
42         name
43             first
44                 John
45             last
46                 Williams
47         location
48             New York
49         age
50             23
51     user2
52         id

```

```

53         56
54     name
55         first
56             Igor
57         middle
58             Petrovich
59         last
60             Ivanov
61     location
62         Moscow
63     age
64         20";
65     var target = @"(users)
66 (users user1)
67 ((users user1) id)
68 (((users user1) id) 43)
69 ((users user1) name)
70 (((users user1) name) first)
71 (((users user1) name) first) John)
72 ((users user1) name) last)
73 (((users user1) name) last) Williams)
74 ((users user1) location)
75 (((users user1) location) (New York))
76 ((users user1) age)
77 (((users user1) age) 23)
78 (users user2)
79 ((users user2) id)
80 (((users user2) id) 56)
81 ((users user2) name)
82 (((users user2) name) first)
83 (((users user2) name) first) Igor)
84 ((users user2) name) middle)
85 (((users user2) name) middle) Petrovich)
86 ((users user2) name) last)
87 (((users user2) name) last) Ivanov)
88 ((users user2) location)
89 (((users user2) location) Moscow)
90 ((users user2) age)
91 (((users user2) age) 20)";
92     var parser = new Parser();
93     var links = parser.Parse(source);
94     var formattedLinks = links.Format();
95     Assert.Equal(formattedLinks, target);
96 }
97 }
98 }

```

## Index

./Platform.Communication.Protocol.Lino.Tests/ParserTests.cs, 3  
./Platform.Communication.Protocol.Lino/IListExtensions.cs, 1  
./Platform.Communication.Protocol.Lino/Link.cs, 1