

## LinksPlatform's Platform.Interfaces Class Library

### ./IConverter[T].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a converter between two values of the same type.</para>
5     /// <para>Определяет конвертер между двумя значениями одного типа.</para>
6     /// </summary>
7     /// <typeparam name="T"><para>Type of value to convert.</para><para>Тип преобразуемого
    ↪ значения.</para></typeparam>
8     public interface IConverter<T> : IConverter<T, T>
9     {
10    }
11 }
```

### ./IConverter[TSource, TTarget].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a converter between two types (TSource and TTarget).</para>
5     /// <para>Определяет конвертер между двумя типами (исходным TSource и целевым
    ↪ TTarget).</para>
6     /// </summary>
7     /// <typeparam name="TSource"><para>Source type of conversion.</para><para>Исходный тип
    ↪ конверсии.</para></typeparam>
8     /// <typeparam name="TTarget"><para>Target type of conversion.</para><para>Целевой тип
    ↪ конверсии.</para></typeparam>
9     public interface IConverter<in TSource, out TTarget>
10    {
11        /// <summary>
12        /// <para>Converts the value of the source type (TSource) to the value of the target
    ↪ type.</para>
13        /// <para>Конвертирует значение исходного типа (TSource) в значение целевого типа.</para>
14        /// </summary>
15        /// <param name="source"><para>The source type value (TSource).</para><para>Значение
    ↪ исходного типа (TSource).</para></param>
16        /// <returns><para>The value is converted to the target type
    ↪ (TTarget).</para><para>Значение ковертированное в целевой тип
    ↪ (TTarget).</para></returns>
17        TTarget Convert(TSource source);
18    }
19 }
```

### ./ICounter[TArgument, TResult].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a counter that requires passing an argument to perform a count.</para>
5     /// <para>Определяет счетчик, который требует передачи аргумента для выполнения
    ↪ подсчёта.</para>
6     /// </summary>
7     /// <typeparam name="TArgument"><para>The argument type.</para><para>Тип
    ↪ аргумента.</para></typeparam>
8     /// <typeparam name="TResult"><para>The count result type.</para><para>Тип результата
    ↪ подсчета.</para></typeparam>
9     public interface ICounter<in TArgument, out TResult>
10    {
11        /// <summary>
12        /// <para>Performs a count that requires passing an argument.</para>
13        /// <para>Выполняет посчёт, для которого требуется передача аргумент.</para>
14        /// </summary>
15        /// <param name="argument"><para>The argument.</para><para>Аргумент.</para></param>
16        /// <returns><para>The count result.</para><para>Результат подсчёта.</para></returns>
17        TResult Count(TArgument argument);
18    }
19 }
```

### ./ICounter[TResult].cs

```
1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a counter.</para>
5     /// <para>Определяет счетчик.</para>
6     /// </summary>
7     /// <typeparam name="TResult"><para>The count result type.</para><para>Тип результата
    ↪ подсчета.</para></typeparam>
```

```

8     public interface ICounter<out TResult>
9     {
10         /// <summary>
11         /// <para>Performs a count.</para>
12         /// <para>Выполняет подсчёт.</para>
13         /// </summary>
14         /// <returns><para>The count result.</para><para>Результат подсчёта.</para></returns>
15         TResult Count();
16     }
17 }

```

#### ./ICriterionMatcher.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a criterion matcher, that contains a specific method of determining
5     /// <para>Определяет объект который проверяет соответствие критерию и содержащий конкретный
6     /// </summary>
7     /// <typeparam name="TArgument"><para>Argument type.</para><para>Тип
8     /// <para>аргумента.</para></typeparam>
9     public interface ICriterionMatcher<in TArgument>
10     {
11         /// <summary>
12         /// <para>Determines whether the argument matches the criterion.</para>
13         /// <para>Определяет, соответствует ли аргумент критерию.</para>
14         /// </summary>
15         /// <param name="argument"><para>The argument.</para><para>Аргумент.</para></param>
16         /// <returns><para>A value that determines whether the argument matches the
17         /// <para>критерию.</para><para>Значение, определяющие соответствует ли аргумент
18         /// </returns>
19         bool IsMatched(TArgument argument);
20     }
21 }

```

#### ./IFactory.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines a factory that produces instances of a specific type.</para>
5     /// <para>Определяет фабрику, которая производит экземпляры определенного типа.</para>
6     /// </summary>
7     /// <typeparam name="TProduct"><para>Type of produced instances.</para><para>Тип
8     /// <para>производимых экземпляров.</para></typeparam>
9     public interface IFactory<out TProduct>
10     {
11         /// <summary>
12         /// <para>Creates an instance of TProduct type.</para>
13         /// <para>Создает экземпляр типа TProduct.</para>
14         /// </summary>
15         /// <returns><para>The instance of TProduct type.</para><para>Экземпляр типа
16         /// <para>TProduct.</para></returns>
17         TProduct Create();
18     }
19 }

```

#### ./IIncrementer.cs

```

1 namespace Platform.Interfaces
2 {
3     /// <summary>
4     /// <para>Defines an incrementer that increments any number with a specific type.</para>
5     /// <para>Определяет инкрементер, который выполняет приращение любого числа определенного
6     /// <para>типа.</para>
7     /// </summary>
8     /// <typeparam name="TNumber"><para>Type of incremented number.</para><para>Тип
9     /// <para>увеличиваемого числа.</para></typeparam>
10     public interface IIncrementer<TNumber>
11     {
12         /// <summary>
13         /// <para>Increments the number by a specific value.</para>
14         /// <para>Увеличивает число на определённое значение.</para>
15         /// </summary>
16         /// <param name="number"><para>The number to be incremented.</para><para>Увеличиваемое
17         /// <para>число.</para></param>
18         /// <returns><para>The incremented number.</para><para>Увеличенное
19         /// <para>число.</para></returns>
20         TNumber Increment(TNumber number);
21     }
22 }

```

```

16         TNumber Increment(TNumber number);
17     }
18 }

```

## ./IPropertiesOperator.cs

```

1  namespace Platform.Interfaces
2  {
3      /// <summary>
4      /// <para>Defines a properties operator that is able to get or set values of properties of a
5      /// <para>object of a specific type.</para>
6      /// <para>Определяет оператор свойств, который может получать или устанавливать значения
7      /// <para>свойств объекта определенного типа.</para>
8      /// </summary>
9      /// <typeparam name="TObject"><para>Object type.</para><para>Тип объекта.</para></typeparam>
10     /// <typeparam name="TProperty"><para>Property reference type.</para><para>Тип ссылки на
11     /// <para>свойство.</para></typeparam>
12     /// <typeparam name="TValue"><para>Property value type.</para><para>Тип значения
13     /// <para>свойства.</para></typeparam>
14     public interface IPropertiesOperator<in TObject, in TProperty, TValue>
15     {
16         /// <summary>
17         /// <para>Gets the value of the property in the specified object.</para>
18         /// <para>Получает значение свойства в указанном объекте.</para>
19         /// </summary>
20         /// <param name="object"><para>The object reference.</para><para>Ссылка на
21         /// <para>объект.</para></param>
22         /// <param name="property"><para>The property reference.</para><para>Ссылка на
23         /// <para>свойство.</para></param>
24         /// <returns><para>The value of the property.</para><para>Значение
25         /// <para>свойства.</para></returns>
26         TValue GetValue(TObject @object, TProperty property);
27
28         /// <summary>
29         /// <para>Sets the value of a property in the specified object.</para>
30         /// <para>Устанавливает значение свойства в указанном объекте.</para>
31         /// </summary>
32         /// <param name="object"><para>The object reference.</para><para>Ссылка на
33         /// <para>объект.</para></param>
34         /// <param name="property"><para>The property reference.</para><para>Ссылка на
35         /// <para>свойство.</para></param>
36         /// <param name="value"><para>The value.</para><para>Значение.</para></param>
37         void SetValue(TObject @object, TProperty property, TValue value);
38     }
39 }

```

## ./IPropertyOperator.cs

```

1  namespace Platform.Interfaces
2  {
3      /// <summary>
4      /// <para>Defines a specific property operator that is able to get or set values of that
5      /// <para>property.</para>
6      /// <para>Определяет оператор определённого свойства, который может получать или
7      /// <para>устанавливать его значения.</para>
8      /// </summary>
9      /// <typeparam name="TObject"><para>Object type.</para><para>Тип объекта.</para></typeparam>
10     /// <typeparam name="TValue"><para>Property value type.</para><para>Тип значения
11     /// <para>свойства.</para></typeparam>
12     public interface IPropertyOperator<in TObject, TValue> : IProvider<TValue, TObject>
13     {
14         /// <summary>
15         /// <para>Sets the value of a specific property in the specified object.</para>
16         /// <para>Устанавливает значение определённого свойства в указанном объекте.</para>
17         /// </summary>
18         /// <param name="object"><para>The object reference.</para><para>Ссылка на
19         /// <para>объект.</para></param>
20         /// <param name="value"><para>The value.</para><para>Значение.</para></param>
21         void Set(TObject @object, TValue value);
22     }
23 }

```

## ./IProvider[TProvided].cs

```

1  namespace Platform.Interfaces
2  {
3      /// <summary>
4      /// <para>Defines the provider of objects.</para>
5      /// <para>Определяет поставщика объектов.</para>

```

```

6     /// </summary>
7     /// <typeparam name="TProvided"><para>Type of provided object.</para><para>Тип
    ↳ предоставляемого объекта.</para></typeparam>
8     public interface IProvider<out TProvided>
9     {
10         /// <summary>
11         /// <para>Provides an object.</para>
12         /// </summary>
13         /// <returns><para>The provided object.</para></returns>
14         TProvided Get();
15     }
16 }

```

./IProvider[TProvided, TArgument].cs

```

1     namespace Platform.Interfaces
2     {
3         /// <summary>
4         /// <para>Defines the provider of objects for which an argument must be specified.</para>
5         /// <para>Определяет поставщика объектов, для получения которых необходимо указать
    ↳ аргумент.</para>
6         /// </summary>
7         /// <typeparam name="TProvided"><para>Type of provided objects.</para><para>Тип
    ↳ предоставляемых объектов.</para></typeparam>
8         /// <typeparam name="TArgument"><para>Argument type.</para><para>Тип
    ↳ аргумента.</para></typeparam>
9         public interface IProvider<out TProvided, in TArgument>
10        {
11            /// <summary>
12            /// <para>Provides an object.</para>
13            /// <para>Предоставляет объект.</para>
14            /// </summary>
15            /// <param name="argument"><para>The argument required to acquire the
    ↳ object.</para><para>Аргумент, необходимый для получения объекта.</para></param>
16            /// <returns><para>The object.</para><para>Объект.</para></returns>
17            TProvided Get(TArgument argument);
18        }
19    }

```

## Index

./IConverter[TSource, TTarget].cs, 1  
./IConverter[T].cs, 1  
./ICounter[TArgument, TResult].cs, 1  
./ICounter[TResult].cs, 1  
./ICriterionMatcher.cs, 2  
./IFactory.cs, 2  
./IIncrementer.cs, 2  
./IPropertiesOperator.cs, 3  
./IPropertyOperator.cs, 3  
./IProvider[TProvided, TArgument].cs, 4  
./IProvider[TProvided].cs, 3