LinksPlatform's Platform.Interfaces Class Library

./IConverter[T].cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a converter between two values of the same type.
    /// Определяет конвертер между двумя значениями одного типа.
    /// </summary>
    /// <typeparam name="T">Type of value to convert. Тип преобразуемого значения.</typeparam>
    public interface IConverter<T> : IConverter<T, T>
    {
    }
}
```

./IConverter[TSource, TTarget].cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a converter between two types (TSource and TTarget).
    /// Определяет конвертер между двумя типами (исходным TSource и целевым TTarget).
    /// </summary>
    /// <typeparam name="TSource">Source type of conversion. Исходный тип конверсии.</typeparam>
    /// <typeparam name="TTarget">Target type of conversion. Целевой тип конверсии.</typeparam>
    public interface IConverter<in TSource, out TTarget>
    {
        /// <summary>
        /// Converts the value of the source type (TSource) to the value of the target type.
        /// Конвертирует значение исходного типа (TSource) в значение целевого типа.
        /// </summary>
        /// <param name="source">The source type value (TSource). Значение исходного типа
        ↪  (TSource).</param>
        /// <returns>The value is converted to the target type (TTarget). Значение
        ↪  ковертированное в целевой тип (TTarget).</returns>
        TTarget Convert(TSource source);
    }
}
```

./ICounter[TArgument, TResult].cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a counter that requires passing an argument to perform a count.
    /// Определяет счетчик, который требует передачи аргумента для выполнения подсчёта.
    /// </summary>
    /// <typeparam name="TArgument">The argument type. Тип аргумента.</typeparam>
    /// <typeparam name="TResult">The count result type. Тип результата подсчета.</typeparam>
    public interface ICounter<in TArgument, out TResult>
    {
        /// <summary>
        /// Performs a count that requires passing an argument.
        /// Выполняет посчёт, для которого требуется передача аргумент.
        /// </summary>
        /// <param name="argument">The argument. Аргумент.</param>
        /// <returns>The count result. Результат подсчёта.</returns>
        TResult Count(TArgument argument);
    }
}
```

./ICounter[TResult].cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a counter.
    /// Определяет счетчик.
    /// </summary>
    /// <typeparam name="TResult">The count result type. Тип результата подсчета.</typeparam>
    public interface ICounter<out TResult>
    {
        /// <summary>
        /// Performs a count.
        /// Выполняет посчёт.
        /// </summary>
        /// <returns>The count result. Результат подсчёта.</returns>
        TResult Count();
    }
}
```

./ICriterionMatcher.cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a criterion matcher, that contains a specific method of determining whether the argument matches criterion or not.
    /// Определяет объект который проверяет соответствие критерию и содержащий конкретный метод определения, соответствует ли аргумент критерию или нет.
    /// </summary>
    /// <typeparam name="TArgument">Argument type. Тип аргумента.</typeparam>
    public interface ICriterionMatcher<in TArgument>
    {
        /// <summary>
        /// Determines whether the argument matches the criterion.
        /// Определяет, соответствует ли аргумент критерию.
        /// </summary>
        /// <param name="argument">The argument. Аргумент.</param>
        /// <returns>A value that determines whether the argument matches the criterion. Значение, определяющие соответствует ли аргумент критерию.</returns>
        bool IsMatched(TArgument argument);
    }
}
```

./IFactory.cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a factory that produces instances of a specific type.
    /// Определяет фабрику, которая производит экземпляры определенного типа.
    /// </summary>
    /// <typeparam name="TProduct">Type of produced instances. Тип производимых экземпляров.</typeparam>
    public interface IFactory<out TProduct>
    {
        /// <summary>
        /// Creates an instance of TProduct type.
        /// Создает экземпляр типа TProduct.
        /// </summary>
        /// <returns>The instance of TProduct type. Экземпляр типа TProduct.</returns>
        TProduct Create();
    }
}
```

./IIncrementer.cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines an incrementer that increments any number with a specific type.
    /// Определяет инкрементер, который выполняет приращение любого числа определенного типа.
    /// </summary>
    /// <typeparam name="TNumber">Type of incremented number. Тип увеличиваемого числа.</typeparam>
    public interface IIncrementer<TNumber>
    {
        /// <summary>
        /// Increments the number by a specific value.
        /// Увеличивает число на определённое значение.
        /// </summary>
        /// <param name="number">The number to be incremented.</param>
        /// <returns>The incremented number. Увеличенное число.</returns>
        TNumber Increment(TNumber number);
    }
}
```

./IPropertiesOperator.cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a properties operator that is able to get or set values of properties of a object of a specific type.
    /// Определяет оператор свойств, который может получать или установливать значения свойств объекта определенного типа.
    /// </summary>
    /// <typeparam name="TObject">Object type. Тип объекта.</typeparam>
    /// <typeparam name="TProperty">Property reference type. Тип ссылки на свойство.</typeparam>
    /// <typeparam name="TValue">Property value type. Тип значения свойства.</typeparam>
    public interface IPropertiesOperator<in TObject, in TProperty, TValue>
```

```
    {
        /// <summary>
        /// Gets the value of the property in the specified object.
        /// Получает значение свойства в указанном объекте.
        /// </summary>
        /// <param name="object">The object reference. Ссылка на объект.</param>
        /// <param name="property">The property reference. Ссылка на свойство.</param>
        /// <returns>The value of the property. Значение свойства.</returns>
        TValue GetValue(TObject @object, TProperty property);

        /// <summary>
        /// Sets the value of a property in the specified object.
        /// Устанавливает значение свойства в указанном объекте.
        /// </summary>
        /// <param name="object">The object reference. Ссылка на объект.</param>
        /// <param name="property">The property reference. Ссылка на свойство.</param>
        /// <param name="value">The value. Значение.</param>
        void SetValue(TObject @object, TProperty property, TValue value);
    }
}
```

./IPropertyOperator.cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines a specific property operator that is able to get or set values of that property.
    /// Определяет оператор определённого свойства, который может получать или установливать его значения.
    /// </summary>
    /// <typeparam name="TObject">Object type. Тип объекта.</typeparam>
    /// <typeparam name="TValue">Property value type. Тип значения свойства.</typeparam>
    public interface IPropertyOperator<in TObject, TValue> : IProvider<TValue, TObject>
    {
        /// <summary>
        /// Sets the value of a specific property in the specified object.
        /// Устанавливает значение определённого свойства в указанном объекте.
        /// </summary>
        /// <param name="object">The object reference. Ссылка на объект.</param>
        /// <param name="value">The value. Значение.</param>
        void Set(TObject @object, TValue value);
    }
}
```

./IProvider[TProvided].cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines the provider of objects.
    /// Определяет поставщика объектов.
    /// </summary>
    /// <typeparam name="TProvided">Type of provided object. Тип предоставляемого объекта.</typeparam>
    public interface IProvider<out TProvided>
    {
        /// <summary>
        /// Provides an object.
        /// </summary>
        /// <returns>The provided object.</returns>
        TProvided Get();
    }
}
```

./IProvider[TProvided, TArgument].cs

```
namespace Platform.Interfaces
{
    /// <summary>
    /// Defines the provider of objects for which an argument must be specified.
    /// Определяет поставщика объектов, для получения которых необходимо указать аргумент.
    /// </summary>
    /// <typeparam name="TProvided">Type of provided objects. Тип предоставляемых объектов.</typeparam>
    /// <typeparam name="TArgument">Argument type. Тип аргумента.</typeparam>
    public interface IProvider<out TProvided, in TArgument>
    {
        /// <summary>
        /// Provides an object.
        /// </summary>
```

```
14          /// <param name="argument">The argument required to acquire the object. Аргумент,
            ↪  необходимый для получения объекта.</param>
15          /// <returns>The object.</returns>
16          TProvided Get(TArgument argument);
17      }
18  }
```

Index