1.1 ./csharp/Platform.Setters/SetterBase.cs

```
using System.Runtime.CompilerServices;
using Platform.Interfaces;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Setters
{
    /// <summary>
    /// <para>Represents a base implementation for an setter that allows you to set a passed value as the result value.</para>
    /// <para>Представляет базовую реализацию для установщика, который позволяет установить переданное ему значение в качестве результирующего значения.</para>
    /// </summary>
    /// <typeparam name="TResult"><para>The type of result value.</para><para>Тип результирующего значения.</para></typeparam>
    /// <remarks>
    /// Must be class, not struct (in order to persist access to Result property value).
    /// </remarks>
    public abstract class SetterBase<TResult> : ISetter<TResult>
    {
        /// <summary>
        /// <para>Represents the result value.</para>
        /// <para>Представляет результирующие значение.</para>
        /// </summary>
        protected TResult _result;

        /// <summary>
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        public TResult Result => _result;

        /// <summary>
        /// <para>Initializes a new instance of the SetterBase class.</para>
        /// <para>Инициализирует новый экземпляр класса SetterBase.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        protected SetterBase() { }

        /// <summary>
        /// <para>Initializes a new instance of the SetterBase class using the passed-in value as the default result value.</para>
        /// <para>Инициализирует новый экземпляр класса SetterBase, используя переданное значение в качестве результирующего по умолчанию.</para>
        /// </summary>
        /// <param name="defaultValue"><para>The default result value.</para><para>Результирующее значение по умолчанию.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        protected SetterBase(TResult defaultValue) => _result = defaultValue;

        /// <summary>
        /// <para>Sets the passed value as the result.</para>
        /// <para>Устанавливает переданное значение в качестве результирующего.</para>
        /// </summary>
        /// <param name="value"><para>The result value.</para><para>Результирующее значение.</para></param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public void Set(TResult value) => _result = value;
    }
}
```

1.2 ./csharp/Platform.Setters/Setter[TResult, TDecision].cs

```
using System.Collections.Generic;
using System.Runtime.CompilerServices;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Setters
{
    /// <summary>
    /// <para>Represents implementation for an setter that allows you to set a passed value as the result value. This setter implementation has additional methods that, simultaneously with setting the result value, return <typeparamref name="TDecision"/> values indicating true or false.</para>
```

```
10      /// <para>Представляет реализацию для установщика, который позволяет установить переданное
            ↪ ему значение в качестве результирующего значения. В этой реализации установщика есть
            ↪ дополнительные методы, которые одновременно с установкой результирующего значения
            ↪ возвращают значения типа <typeparamref name="TDecision"/>, обозначающие истину или
            ↪ ложь.</para>
11      /// </summary>
12      /// <typeparam name="TResult"><para>The type of result value.</para><para>Тип
            ↪ результирующего значения.</para></typeparam>
13      /// <typeparam name="TDecision"><para>The type of value which will be used to make the
            ↪ decision.</para><para>Тип значения на основе которого будет приниматься
            ↪ решение.</para></typeparam>
14      public class Setter<TResult, TDecision> : SetterBase<TResult>
15      {
16          /// <summary>
17          /// <para>
18          /// The true value.
19          /// </para>
20          /// <para></para>
21          /// </summary>
22          private readonly TDecision _trueValue;
23          /// <summary>
24          /// <para>
25          /// The false value.
26          /// </para>
27          /// <para></para>
28          /// </summary>
29          private readonly TDecision _falseValue;
30
31          /// <summary>
32          /// <para>Initializes a new instance of the Setter class using the passed-in value as
                ↪ the default result value.</para>
33          /// <para>Инициализирует новый экземпляр класса Setter, используя переданные значения
                ↪ trueValue, falseValue, defaultValue в качестве результирующего по умолчанию.</para>
34          /// </summary>
35          /// <param name="defaultValue"><para>The default result
                ↪ value.</para><para>Результирующее значение по умолчанию.</para></param>
36          [MethodImpl(MethodImplOptions.AggressiveInlining)]
37          public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
38              : base(defaultValue)
39          {
40              _trueValue = trueValue;
41              _falseValue = falseValue;
42          }
43
44          /// <summary>
45          /// <para>Gets result value.</para>
46          /// <para>Возвращает результирующее значение.</para>
47          /// </summary>
48          [MethodImpl(MethodImplOptions.AggressiveInlining)]
49          public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue,
                ↪ default) { }
50
51          /// <summary>
52          /// <para>Gets result value.</para>
53          /// <para>Возвращает результирующее значение.</para>
54          /// </summary>
55          [MethodImpl(MethodImplOptions.AggressiveInlining)]
56          public Setter(TResult defaultValue) : base(defaultValue) { }
57
58          /// <summary>
59          /// <para>Gets result value.</para>
60          /// <para>Возвращает результирующее значение.</para>
61          /// </summary>
62          [MethodImpl(MethodImplOptions.AggressiveInlining)]
63          public Setter() { }
64
65          /// <summary>
66          /// <para>Gets result value.</para>
67          /// <para>Возвращает результирующее значение.</para>
68          /// </summary>
69          [MethodImpl(MethodImplOptions.AggressiveInlining)]
70          public TDecision SetAndReturnTrue(TResult value)
71          {
72              _result = value;
73              return _trueValue;
74          }
75
76          /// <summary>
```

```
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetAndReturnFalse(TResult value)
        {
            _result = value;
            return _falseValue;
        }

        /// <summary>
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetFirstAndReturnTrue(IList<TResult> list)
        {
            _result = list[0];
            return _trueValue;
        }

        /// <summary>
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public TDecision SetFirstAndReturnFalse(IList<TResult> list)
        {
            _result = list[0];
            return _falseValue;
        }
    }
}
```

1.3 ./csharp/Platform.Setters/Setter[TResult].cs

```
using System.Runtime.CompilerServices;

#pragma warning disable CS1591 // Missing XML comment for publicly visible type or member

namespace Platform.Setters
{
    /// <summary>
    /// <para>
    /// Represents the setter.
    /// </para>
    /// <para></para>
    /// </summary>
    /// <seealso cref="Setter{TResult, bool}"/>
    public class Setter<TResult> : Setter<TResult, bool>
    {
        /// <summary>
        /// <para>
        /// Initializes a new <see cref="Setter"/> instance.
        /// </para>
        /// <para></para>
        /// </summary>
        /// <param name="defaultValue">
        /// <para>A default value.</para>
        /// <para></para>
        /// </param>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TResult defaultValue) : base(true, false, defaultValue) { }

        /// <summary>
        /// <para>
        /// Initializes a new <see cref="Setter"/> instance.
        /// </para>
        /// <para></para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter() : base(true, false) { }
    }
}
```

1.4 ./csharp/Platform.Setters.Tests/SetterTests.cs

```
using Xunit;

namespace Platform.Setters.Tests
{
```

```
    /// <summary>
    /// <para>
    /// Represents the setter tests.
    /// </para>
    /// <para></para>
    /// </summary>
    public class SetterTests
    {
        /// <summary>
        /// <para>
        /// Tests that parameterless constructed setter test.
        /// </para>
        /// <para></para>
        /// </summary>
        [Fact]
        public void ParameterlessConstructedSetterTest()
        {
            Setter<int> setter = new Setter<int>();
            Assert.Equal(default, setter.Result);
        }

        /// <summary>
        /// <para>
        /// Tests that constructed with default value setter test.
        /// </para>
        /// <para></para>
        /// </summary>
        [Fact]
        public void ConstructedWithDefaultValueSetterTest()
        {
            Setter<int> setter = new Setter<int>(9);
            Assert.Equal(9, setter.Result);
        }

        /// <summary>
        /// <para>
        /// Tests that methods with boolean return type test.
        /// </para>
        /// <para></para>
        /// </summary>
        [Fact]
        public void MethodsWithBooleanReturnTypeTest()
        {
            Setter<int> setter = new Setter<int>();
            Assert.True(setter.SetAndReturnTrue(1));
            Assert.Equal(1, setter.Result);
            Assert.False(setter.SetAndReturnFalse(2));
            Assert.Equal(2, setter.Result);
            Assert.True(setter.SetFirstAndReturnTrue(new int[] { 3 }));
            Assert.Equal(3, setter.Result);
            Assert.False(setter.SetFirstAndReturnFalse(new int[] { 4 }));
            Assert.Equal(4, setter.Result);
        }

        /// <summary>
        /// <para>
        /// Tests that methods with integer return type test.
        /// </para>
        /// <para></para>
        /// </summary>
        [Fact]
        public void MethodsWithIntegerReturnTypeTest()
        {
            Setter<int, int> setter = new Setter<int, int>(1, 0);
            Assert.Equal(1, setter.SetAndReturnTrue(1));
            Assert.Equal(1, setter.Result);
            Assert.Equal(0, setter.SetAndReturnFalse(2));
            Assert.Equal(2, setter.Result);
            Assert.Equal(1, setter.SetFirstAndReturnTrue(new int[] { 3 }));
            Assert.Equal(3, setter.Result);
            Assert.Equal(0, setter.SetFirstAndReturnFalse(new int[] { 4 }));
            Assert.Equal(4, setter.Result);
        }
    }
}
```

Index

./csharp/Platform.Setters.Tests/SetterTests.cs, 3
./csharp/Platform.Setters/SetterBase.cs, 1
./csharp/Platform.Setters/Setter[TResult, TDecision].cs, 1
./csharp/Platform.Setters/Setter[TResult].cs, 3