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/SetterExtensions.cs

```
using System.Collections.Generic;

namespace Platform.Setters
{
    public static class SetterExtensions
    {
        public static TDecision SetFirstFromListAndReturnTrue<TResult, TDecision>(this Setter<TResult, TDecision> setter, IList<TResult> list)
        {
            setter.Set(list[0]);
            return setter.TrueValue;
        }

```

```
13          public static TDecision SetFirstFromFirstListAndReturnTrue<TResult, TDecision>(this
            ↪  Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
14          {
15              setter.Set(list1[0]);
16              return setter.TrueValue;
17          }
18
19          public static TDecision SetSecondFromFirstListAndReturnTrue<TResult, TDecision>(this
            ↪  Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
20          {
21              setter.Set(list1[1]);
22              return setter.TrueValue;
23          }
24
25          public static TDecision SetThirdFromFirstListAndReturnTrue<TResult, TDecision>(this
            ↪  Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
26          {
27              setter.Set(list1[2]);
28              return setter.TrueValue;
29          }
30
31          public static TDecision SetFirstFromSecondListAndReturnTrue<TResult, TDecision>(this
            ↪  Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
32          {
33              setter.Set(list2[0]);
34              return setter.TrueValue;
35          }
36
37          public static TDecision SetSecondFromSecondListAndReturnTrue<TResult, TDecision>(this
            ↪  Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
38          {
39              setter.Set(list2[1]);
40              return setter.TrueValue;
41          }
42
43          public static TDecision SetThirdFromSecondListAndReturnTrue<TResult, TDecision>(this
            ↪  Setter<TResult, TDecision> setter, IList<TResult> list1, IList<TResult> list2)
44          {
45              setter.Set(list2[2]);
46              return setter.TrueValue;
47          }
48      }
49  }
```

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

```
1   using System.Collections.Generic;
2   using System.Runtime.CompilerServices;
3
4   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6   namespace Platform.Setters
7   {
8       /// <summary>
9       /// <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          public readonly TDecision TrueValue;
17          public readonly TDecision FalseValue;
18
19          /// <summary>
20          /// <para>Initializes a new instance of the Setter class using the passed-in value as
                ↪  the default result value.</para>
21          /// <para>Инициализирует новый экземпляр класса Setter, используя переданные значения
                ↪  trueValue, falseValue, defaultValue 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)]
        public Setter(TDecision trueValue, TDecision falseValue, TResult defaultValue)
            : base(defaultValue)
        {
            TrueValue = trueValue;
            FalseValue = falseValue;
        }

        /// <summary>
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TDecision trueValue, TDecision falseValue) : this(trueValue, falseValue, default) { }

        /// <summary>
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter(TResult defaultValue) : base(defaultValue) { }

        /// <summary>
        /// <para>Gets result value.</para>
        /// <para>Возвращает результирующее значение.</para>
        /// </summary>
        [MethodImpl(MethodImplOptions.AggressiveInlining)]
        public Setter() { }

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

        /// <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;
        }
    }
}
```

1.4 ./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) { }
    }
}
```

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

```
using Xunit;

namespace Platform.Setters.Tests
{
    public class SetterTests
    {
        [Fact]
        public void ParameterlessConstructedSetterTest()
        {
            Setter<int> setter = new Setter<int>();
            Assert.Equal(default, setter.Result);
        }

        [Fact]
        public void ConstructedWithDefaultValueSetterTest()
        {
            Setter<int> setter = new Setter<int>(9);
            Assert.Equal(9, setter.Result);
        }

        [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);
        }

        [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