./ConcurrentQueueExtensions.cs

```
using System;
using System.Collections.Concurrent;
using System.Threading.Tasks;
using Platform.Collections.Concurrent;

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

namespace Platform.Threading
{
    public static class ConcurrentQueueExtensions
    {
        public static async Task AwaitAll(this ConcurrentQueue<Task> queue)
        {
            foreach (var item in queue.DequeueAll())
            {
                await item;
            }
        }

        public static async Task AwaitOne(this ConcurrentQueue<Task> queue)
        {
            if (queue.TryDequeue(out Task item))
            {
                await item;
            }
        }

        public static void EnqueueAsRunnedTask(this ConcurrentQueue<Task> queue, Action action)
        => queue.Enqueue(Task.Run(action));
    }
}
```

./Synchronization/ISynchronization.cs

```
using System;

namespace Platform.Threading.Synchronization
{
    /// <summary>
    /// <para>Represents a synchronization object that supports read and write operations.</para>
    /// <para>Представляет объект синхронизации с поддержкой операций чтения и записи.</para>
    /// </summary>
    public interface ISynchronization
    {
        /// <summary>
        /// <para>Executes action in read access mode.</para>
        /// <para>Выполняет действие в режиме доступа для чтения.</para>
        /// </summary>
        /// <param name="action"><para>The action.</para><para>Действие.</para></param>
        void ExecuteReadOperation(Action action);

        /// <summary>
        /// <para>Executes a function in read access mode and returns the function's result.</para>
        /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё результат.</para>
        /// </summary>
        /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип результата функции.</para></typeparam>
        /// <param name="function"><para>The function.</para><para>Функция.</para></param>
        /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
        TResult ExecuteReadOperation<TResult>(Func<TResult> function);

        /// <summary>
        /// <para>Executes action in write access mode.</para>
        /// <para>Выполняет действие в режиме доступа для записи.</para>
        /// </summary>
        /// <param name="action"><para>The action.</para><para>Действие.</para></param>
        void ExecuteWriteOperation(Action action);

        /// <summary>
        /// <para>Executes a function in write access mode and returns the function's result.</para>
        /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё результат.</para>
        /// </summary>
        /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип результата функции.</para></typeparam>
```

```
39          /// <param name="function"><para>The function.</para><para>Функция.</para></param>
40          /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
41          TResult ExecuteWriteOperation<TResult>(Func<TResult> function);
42      }
43  }
```

./Synchronization/ISynchronizationExtensions.cs

```
1   using System;
2
3   namespace Platform.Threading.Synchronization
4   {
5       /// <summary>
6       /// <para>Contains extension methods for the <see cref="ISynchronization"/> interface.</para>
7       /// <para>Содержит методы расширения для интерфейса <see cref="ISynchronization"/>.</para>
8       /// </summary>
9       public static class ISynchronizationExtensions
10      {
11          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
12          /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
            ↪  параметра.</para></typeparam>
13          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
14          /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
15          public static TResult ExecuteReadOperation<TResult, TParam>(this ISynchronization
            ↪  synchronization, TParam parameter, Func<TParam, TResult> function) =>
            ↪  synchronization.ExecuteReadOperation(() => function(parameter));
16
17          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
18          /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
            ↪  параметра.</para></typeparam>
19          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
20          /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
21          public static void ExecuteReadOperation<TParam>(this ISynchronization synchronization,
            ↪  TParam parameter, Action<TParam> action) => synchronization.ExecuteReadOperation(()
            ↪  => action(parameter));
22
23          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
24          /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
            ↪  параметра.</para></typeparam>
25          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
26          /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
27          public static TResult ExecuteWriteOperation<TResult, TParam>(this ISynchronization
            ↪  synchronization, TParam parameter, Func<TParam, TResult> function) =>
            ↪  synchronization.ExecuteWriteOperation(() => function(parameter));
28
29          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
30          /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
            ↪  параметра.</para></typeparam>
31          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
32          /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
33          public static void ExecuteWriteOperation<TParam>(this ISynchronization synchronization,
            ↪  TParam parameter, Action<TParam> action) => synchronization.ExecuteWriteOperation(()
            ↪  => action(parameter));
34
35          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
36          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
37          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
38          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
39          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
```

```
40          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
41          public static TResult ExecuteReadOperation<TResult, TParam1, TParam2>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
            ↪  Func<TParam1, TParam2, TResult> function) => synchronization.ExecuteReadOperation(()
            ↪  => function(parameter1, parameter2));
42
43          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
44          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
45          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
46          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
47          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
48          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
49          public static void ExecuteReadOperation<TParam1, TParam2>(this ISynchronization
            ↪  synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
            ↪  action) => synchronization.ExecuteReadOperation(() => action(parameter1,
            ↪  parameter2));
50
51          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
52          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
53          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
54          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
55          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
56          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
57          public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
            ↪  Func<TParam1, TParam2, TResult> function) =>
            ↪  synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2));
58
59          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
60          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
61          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
62          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
63          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
64          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
65          public static void ExecuteWriteOperation<TParam1, TParam2>(this ISynchronization
            ↪  synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
            ↪  action) => synchronization.ExecuteWriteOperation(() => action(parameter1,
            ↪  parameter2));
66
67          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
68          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
69          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
70          /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
71          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
72          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
```

```
73          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
74          /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
75          public static TResult ExecuteReadOperation<TResult, TParam1, TParam2, TParam3>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪  parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
            ↪  synchronization.ExecuteReadOperation(() => function(parameter1, parameter2,
            ↪  parameter3));
76
77          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
78          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
79          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
80          /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
81          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
82          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
83          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
84          /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
85          public static void ExecuteReadOperation<TParam1, TParam2, TParam3>(this ISynchronization
            ↪  synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
            ↪  Action<TParam1, TParam2, TParam3> action) => synchronization.ExecuteReadOperation(()
            ↪  => action(parameter1, parameter2, parameter3));
86
87          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
88          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
89          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
90          /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
91          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
92          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
93          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
94          /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
95          public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪  parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
            ↪  synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2,
            ↪  parameter3));
96
97          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
98          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
99          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
100         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
101         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
102         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
103         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
104         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
```

```
105         public static void ExecuteWriteOperation<TParam1, TParam2, TParam3>(this
            ↪ ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪ parameter3, Action<TParam1, TParam2, TParam3> action) =>
            ↪ synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
            ↪ parameter3));
106
107         /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪ n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
108         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪ параметра.</para></typeparam>
109         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪ параметра.</para></typeparam>
110         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪ параметра.</para></typeparam>
111         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪ параметра.</para></typeparam>
112         /// <param name="synchronization"><para>Synchronization
            ↪ object.</para><para>Синхронизация объекта.</para></param>
113         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪ параметр.</para></param>
114         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪ параметр.</para></param>
115         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪ параметр.</para></param>
116         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪ параметр.</para></param>
117         public static TResult ExecuteReadOperation<TResult, TParam1, TParam2, TParam3,
            ↪ TParam4>(this ISynchronization synchronization, TParam1 parameter1, TParam2
            ↪ parameter2, TParam3 parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3,
            ↪ TParam4, TResult> function) => synchronization.ExecuteReadOperation(() =>
            ↪ function(parameter1, parameter2, parameter3, parameter4));
118
119         /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪ n.ExecuteReadOperation(System.Action)"]/*'/>
120         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪ параметра.</para></typeparam>
121         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪ параметра.</para></typeparam>
122         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪ параметра.</para></typeparam>
123         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪ параметра.</para></typeparam>
124         /// <param name="synchronization"><para>Synchronization
            ↪ object.</para><para>Синхронизация объекта.</para></param>
125         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪ параметр.</para></param>
126         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪ параметр.</para></param>
127         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪ параметр.</para></param>
128         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪ параметр.</para></param>
129         public static void ExecuteReadOperation<TParam1, TParam2, TParam3, TParam4>(this
            ↪ ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪ parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action)
            ↪ => synchronization.ExecuteReadOperation(() => action(parameter1, parameter2,
            ↪ parameter3, parameter4));
130
131         /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪ path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪ n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
132         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪ параметра.</para></typeparam>
133         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪ параметра.</para></typeparam>
134         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪ параметра.</para></typeparam>
135         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪ параметра.</para></typeparam>
136         /// <param name="synchronization"><para>Synchronization
            ↪ object.</para><para>Синхронизация объекта.</para></param>
137         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪ параметр.</para></param>
```

```
138         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
139         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
140         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪  параметр.</para></param>
141         public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3,
            ↪  TParam4>(this ISynchronization synchronization, TParam1 parameter1, TParam2
            ↪  parameter2, TParam3 parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3,
            ↪  TParam4, TResult> function) => synchronization.ExecuteWriteOperation(() =>
            ↪  function(parameter1, parameter2, parameter3, parameter4));
142
143         /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
144         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
145         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
146         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
147         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪  параметра.</para></typeparam>
148         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
149         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
150         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
151         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
152         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪  параметр.</para></param>
153         public static void ExecuteWriteOperation<TParam1, TParam2, TParam3, TParam4>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
            ↪  parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action)
            ↪  => synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
            ↪  parameter3, parameter4));
154     }
155 }
```

./Synchronization/ISynchronized.cs

```
1  namespace Platform.Threading.Synchronization
2  {
3      /// <summary>
4      /// <para>Represents extendable synchronized interface access gate.</para>
5      /// <para>Представляет расширяемый интерфейс шлюза синхронизированного доступа.</para>
6      /// </summary>
7      /// <typeparam name="TInterface"><para>Synchronized interface.</para><para>Синхронизируемый
       ↪  интерфейс.</para></typeparam>
8      public interface ISynchronized<out TInterface>
9      {
10         /// <summary>
11         /// <para>Gets sychronization method.</para>
12         /// <para>Возвращает способ синхронизации.</para>
13         /// </summary>
14         ISynchronization SyncRoot { get; }
15
16         /// <summary>
17         /// <para>Get source version of <typeparamref name="TInterface"/>, that does not
           ↪  garantee thread safe access synchronization.</para>
18         /// <para>Возвращает исходную версию <typeparamref name="TInterface"/>, которая не
           ↪  гарантирует потокобезопасную синхронизацию доступа.</para>
19         /// </summary>
20         /// <remarks>
21         /// <para>It is unsafe to use it directly, unless compound context using SyncRoot is
           ↪  created.</para>
22         /// <para>Использовать напрямую небезопасно, за исключением ситуации когда создаётся
           ↪  составной контекст с использованием SyncRoot.</para>
23         /// </remarks>
24         TInterface Unsync { get; }
25
26         /// <summary>
27         /// <para>Get wrapped/decorated version of <typeparamref name="TInterface"/>, that does
           ↪  garantee thread safe access synchronization.</para>
```

```
28          /// <para>Возвращает обернутую/декорированную версию <typeparamref name="TInterface"/>,
            ↪  которая гарантирует потокобезопасную синхронизацию доступа.</para>
29          /// </summary>
30          /// <remarks>
31          /// <para>It is safe to use it directly, because it must be thread safe
            ↪  implementation.</para>
32          /// <para>Безопасно использовать напрямую, так как реализация должна быть
            ↪  потокобезопасной.</para>
33          /// </remarks>
34          TInterface Sync { get; }
35      }
36  }
```

./Synchronization/ReaderWriterLockSynchronization.cs

```
1   using System;
2   using System.Threading;
3
4   namespace Platform.Threading.Synchronization
5   {
6       /// <summary>
7       /// <para>Implementation of <see cref="ISynchronization"/> based on <see
        ↪  cref="ReaderWriterLockSlim"/>.</para>
8       /// <para>Реализация <see cref="ISynchronization"/> на основе <see
        ↪  cref="ReaderWriterLockSlim"/>.</para>
9       /// </summary>
10      public class ReaderWriterLockSynchronization : ISynchronization
11      {
12          private readonly ReaderWriterLockSlim _rwLock = new
            ↪  ReaderWriterLockSlim(LockRecursionPolicy.SupportsRecursion);
13
14          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
15          public void ExecuteReadOperation(Action action)
16          {
17              _rwLock.EnterReadLock();
18              try
19              {
20                  action();
21              }
22              finally
23              {
24                  _rwLock.ExitReadLock();
25              }
26          }
27
28          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
29          public TResult ExecuteReadOperation<TResult>(Func<TResult> function)
30          {
31              _rwLock.EnterReadLock();
32              try
33              {
34                  return function();
35              }
36              finally
37              {
38                  _rwLock.ExitReadLock();
39              }
40          }
41
42          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
43          public void ExecuteWriteOperation(Action action)
44          {
45              _rwLock.EnterWriteLock();
46              try
47              {
48                  action();
49              }
50              finally
51              {
52                  _rwLock.ExitWriteLock();
53              }
54          }
55
```

```
56          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
57          public TResult ExecuteWriteOperation<TResult>(Func<TResult> function)
58          {
59              _rwLock.EnterWriteLock();
60              try
61              {
62                  return function();
63              }
64              finally
65              {
66                  _rwLock.ExitWriteLock();
67              }
68          }
69      }
70  }
```

./Synchronization/Unsynchronization.cs

```
1   using System;
2
3   namespace Platform.Threading.Synchronization
4   {
5       /// <summary>
6       /// <para>Implementation of <see cref="ISynchronization"/> that makes no actual
            ↪  synchronization.</para>
7       /// <para>Реализация <see cref="ISynchronization"/>, которая не выполняет фактическую
            ↪  синхронизацию.</para>
8       /// </summary>
9       public class Unsynchronization : ISynchronization
10      {
11          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
12          public void ExecuteReadOperation(Action action) => action();
13
14          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
15          public TResult ExecuteReadOperation<TResult>(Func<TResult> function) => function();
16
17          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
18          public void ExecuteWriteOperation(Action action) => action();
19
20          /// <include file='bin\Release\netstandard2.0\Documentation.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
21          public TResult ExecuteWriteOperation<TResult>(Func<TResult> function) => function();
22      }
23  }
```

./TaskExtensions.cs

```
1   using System.Threading.Tasks;
2
3   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
4
5   namespace Platform.Threading
6   {
7       public static class TaskExtensions
8       {
9           public static T AwaitResult<T>(this Task<T> task) => task.GetAwaiter().GetResult();
10      }
11  }
```

./ThreadHelpers.cs

```
1   using System;
2   using System.Threading;
3
4   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6   namespace Platform.Threading
7   {
8       public static class ThreadHelpers
9       {
10          public static readonly int DefaultMaxStackSize;
11          public static readonly int ExtendedMaxStackSize = 200 * 1024 * 1024;
```

```
12          public static readonly int DefaultSleepTimeout = 1;
13
14          public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action) =>
            ↪  SyncInvokeWithExtendedStack(param, action, ExtendedMaxStackSize);
15
16          public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action, int
            ↪  maxStackSize) => StartNew(param, action, maxStackSize).Join();
17
18          public static void SyncInvokeWithExtendedStack(Action action) =>
            ↪  SyncInvokeWithExtendedStack(action, ExtendedMaxStackSize);
19
20          public static void SyncInvokeWithExtendedStack(Action action, int maxStackSize) =>
            ↪  StartNew(action, maxStackSize).Join();
21
22          public static Thread StartNew<T>(T param, Action<object> action) => StartNew(param,
            ↪  action, DefaultMaxStackSize);
23
24          public static Thread StartNew<T>(T param, Action<object> action, int maxStackSize)
25          {
26              var thread = new Thread(new ParameterizedThreadStart(action), maxStackSize);
27              thread.Start(param);
28              return thread;
29          }
30
31          public static Thread StartNew(Action action) => StartNew(action, DefaultMaxStackSize);
32
33          public static Thread StartNew(Action action, int maxStackSize)
34          {
35              var thread = new Thread(new ThreadStart(action), maxStackSize);
36              thread.Start();
37              return thread;
38          }
39
40          public static void Sleep() => Thread.Sleep(DefaultSleepTimeout);
41      }
42  }
```

Index