./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>
```

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

./Synchronization/ISynchronizationExtensions.cs

```
using System;

namespace Platform.Threading.Synchronization
{
    /// <summary>
    /// <para>Contains extension methods for the <see cref="ISynchronization"/> interface.</para>
    /// <para>Содержит методы расширения для интерфейса <see cref="ISynchronization"/>.</para>
    /// </summary>
    public static class ISynchronizationExtensions
    {
        /// <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>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        /// <param name="function"><para>The function.</para><para>Функция.</para></param>
        /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
        public static TResult ExecuteReadOperation<TResult, TParam>(this ISynchronization synchronization, TParam parameter, Func<TParam, TResult> function) => synchronization.ExecuteReadOperation(() => function(parameter));

        /// <summary>
        /// <para>Executes action in read access mode.</para>
        /// <para>Выполняет действие в режиме доступа для чтения.</para>
        /// </summary>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        /// <param name="action"><para>The action.</para><para>Действие.</para></param>
        public static void ExecuteReadOperation<TParam>(this ISynchronization synchronization, TParam parameter, Action<TParam> action) => synchronization.ExecuteReadOperation(() => action(parameter));

        /// <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>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization object.</para><para>Синхронизация объекта.</para></param>
        /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
        /// <param name="function"><para>The function.</para><para>Функция.</para></param>
        /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
        public static TResult ExecuteWriteOperation<TResult, TParam>(this ISynchronization synchronization, TParam parameter, Func<TParam, TResult> function) => synchronization.ExecuteWriteOperation(() => function(parameter));

        /// <summary>
        /// <para>Executes action in write access mode.</para>
        /// <para>Выполняет действие в режиме доступа для записи.</para>
        /// </summary>
        /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип параметра.</para></typeparam>
        /// <param name="synchronization"><para>Synchronization object.</para><para>Синхронизация объекта.</para></param>
```

```
51          /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
52          /// <param name="action"><para>The action.</para><para>Действие.</para></param>
53          public static void ExecuteWriteOperation<TParam>(this ISynchronization synchronization,
            ↪  TParam parameter, Action<TParam> action) => synchronization.ExecuteWriteOperation(()
            ↪  => action(parameter));
54
55          /// <summary>
56          /// <para>Executes a function in read access mode and returns the function's
            ↪  result.</para>
57          /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
            ↪  результат.</para>
58          /// </summary>
59          /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            ↪  результата функции.</para></typeparam>
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          /// <param name="function"><para>The function.</para><para>Функция.</para></param>
66          /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
67          public static TResult ExecuteReadOperation<TResult, TParam1, TParam2>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
            ↪  Func<TParam1, TParam2, TResult> function) => synchronization.ExecuteReadOperation(()
            ↪  => function(parameter1, parameter2));
68
69          /// <summary>
70          /// <para>Executes action in read access mode.</para>
71          /// <para>Выполняет действие в режиме доступа для чтения.</para>
72          /// </summary>
73          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
74          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
75          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
76          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
77          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
78          /// <param name="action"><para>The action.</para><para>Действие.</para></param>
79          public static void ExecuteReadOperation<TParam1, TParam2>(this ISynchronization
            ↪  synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
            ↪  action) => synchronization.ExecuteReadOperation(() => action(parameter1,
            ↪  parameter2));
80
81          /// <summary>
82          /// <para>Executes a function in write access mode and returns the function's
            ↪  result.</para>
83          /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
            ↪  результат.</para>
84          /// </summary>
85          /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            ↪  результата функции.</para></typeparam>
86          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
87          /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
88          /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
89          /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
90          /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
91          /// <param name="function"><para>The function.</para><para>Функция.</para></param>
92          /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
93          public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2>(this
            ↪  ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
            ↪  Func<TParam1, TParam2, TResult> function) =>
            ↪  synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2));
94
```

```
95          /// <summary>
96          /// <para>Executes action in write access mode.</para>
97          /// <para>Выполняет действие в режиме доступа для записи.</para>
98          /// </summary>
99          /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
100         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  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="action"><para>The action.</para><para>Действие.</para></param>
105         public static void ExecuteWriteOperation<TParam1, TParam2>(this ISynchronization
            ↪  synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
            ↪  action) => synchronization.ExecuteWriteOperation(() => action(parameter1,
            ↪  parameter2));
106
107         /// <summary>
108         /// <para>Executes a function in read access mode and returns the function's
            ↪  result.</para>
109         /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
            ↪  результат.</para>
110         /// </summary>
111         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            ↪  результата функции.</para></typeparam>
112         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
113         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
114         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
115         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
116         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
117         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
118         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
119         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
120         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
121         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));
122
123         /// <summary>
124         /// <para>Executes action in read access mode.</para>
125         /// <para>Выполняет действие в режиме доступа для чтения.</para>
126         /// </summary>
127         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
128         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
129         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
130         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
131         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
132         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
133         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
134         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
135         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));
136
137         /// <summary>
```

```
138         /// <para>Executes a function in write access mode and returns the function's
            ↪  result.</para>
139         /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
            ↪  результат.</para>
140         /// </summary>
141         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            ↪  результата функции.</para></typeparam>
142         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
143         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
144         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
145         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
146         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
147         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
148         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
149         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
150         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
151         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));
152
153         /// <summary>
154         /// <para>Executes action in write access mode.</para>
155         /// <para>Выполняет действие в режиме доступа для записи.</para>
156         /// </summary>
157         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
158         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
159         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
160         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
161         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
162         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
163         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
164         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
165         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));
166
167         /// <summary>
168         /// <para>Executes a function in read access mode and returns the function's
            ↪  result.</para>
169         /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
            ↪  результат.</para>
170         /// </summary>
171         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            ↪  результата функции.</para></typeparam>
172         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
173         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
174         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
175         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪  параметра.</para></typeparam>
176         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
177         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
```

```
178         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
179         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
180         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪  параметр.</para></param>
181         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
182         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
183         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));
184
185         /// <summary>
186         /// <para>Executes action in read access mode.</para>
187         /// <para>Выполняет действие в режиме доступа для чтения.</para>
188         /// </summary>
189         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
190         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
191         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
192         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪  параметра.</para></typeparam>
193         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
194         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
195         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
196         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
197         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪  параметр.</para></param>
198         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
199         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));
200
201         /// <summary>
202         /// <para>Executes a function in write access mode and returns the function's
            ↪  result.</para>
203         /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
            ↪  результат.</para>
204         /// </summary>
205         /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            ↪  результата функции.</para></typeparam>
206         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
            ↪  параметра.</para></typeparam>
207         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
            ↪  параметра.</para></typeparam>
208         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
            ↪  параметра.</para></typeparam>
209         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
            ↪  параметра.</para></typeparam>
210         /// <param name="synchronization"><para>Synchronization
            ↪  object.</para><para>Синхронизация объекта.</para></param>
211         /// <param name="parameter1"><para>The first parameter</para><para>Первый
            ↪  параметр.</para></param>
212         /// <param name="parameter2"><para>The second parameter</para><para>Второй
            ↪  параметр.</para></param>
213         /// <param name="parameter3"><para>The third parameter</para><para>Третий
            ↪  параметр.</para></param>
214         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
            ↪  параметр.</para></param>
215         /// <param name="function"><para>The function.</para><para>Функция.</para></param>
216         /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
```

```
217         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));
218
219         /// <summary>
220         /// <para>Executes action in write access mode.</para>
221         /// <para>Выполняет действие в режиме доступа для записи.</para>
222         /// </summary>
223         /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
                ↪ параметра.</para></typeparam>
224         /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                ↪ параметра.</para></typeparam>
225         /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
                ↪ параметра.</para></typeparam>
226         /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
                ↪ параметра.</para></typeparam>
227         /// <param name="synchronization"><para>Synchronization
                ↪ object.</para><para>Синхронизация объекта.</para></param>
228         /// <param name="parameter1"><para>The first parameter</para><para>Первый
                ↪ параметр.</para></param>
229         /// <param name="parameter2"><para>The second parameter</para><para>Второй
                ↪ параметр.</para></param>
230         /// <param name="parameter3"><para>The third parameter</para><para>Третий
                ↪ параметр.</para></param>
231         /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
                ↪ параметр.</para></param>
232         /// <param name="action"><para>The action.</para><para>Действие.</para></param>
233         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));
234     }
235 }
```

./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>
```

```
        /// <para>It is safe to use it directly, because it must be thread safe implementation.</para>
        /// <para>Безопасно использовать напрямую, так как реализация должна быть потокобезопасной.</para>
        /// </remarks>
        TInterface Sync { get; }
    }
}
```

./Synchronization/ReaderWriterLockSynchronization.cs

```
using System;
using System.Threading;

namespace Platform.Threading.Synchronization
{
    /// <summary>
    /// <para>Implementation of <see cref="ISynchronization"/> based on <see cref="ReaderWriterLockSlim"/>.</para>
    /// <para>Реализация <see cref="ISynchronization"/> на основе <see cref="ReaderWriterLockSlim"/>.</para>
    /// </summary>
    public class ReaderWriterLockSynchronization : ISynchronization
    {
        private readonly ReaderWriterLockSlim _rwLock = new ReaderWriterLockSlim(LockRecursionPolicy.SupportsRecursion);

        /// <inheritdoc />
        /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml' path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.ExecuteReadOperation(System.Action)"]/*'/>
        public void ExecuteReadOperation(Action action)
        {
            _rwLock.EnterReadLock();
            try
            {
                action();
            }
            finally
            {
                _rwLock.ExitReadLock();
            }
        }

        /// <inheritdoc />
        /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml' path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
        public TResult ExecuteReadOperation<TResult>(Func<TResult> function)
        {
            _rwLock.EnterReadLock();
            try
            {
                return function();
            }
            finally
            {
                _rwLock.ExitReadLock();
            }
        }

        /// <inheritdoc />
        /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml' path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization.ExecuteWriteOperation(System.Action)"]/*'/>
        public void ExecuteWriteOperation(Action action)
        {
            _rwLock.EnterWriteLock();
            try
            {
                action();
            }
            finally
            {
                _rwLock.ExitWriteLock();
            }
        }

        /// <inheritdoc />
```

```
60          /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
61          public TResult ExecuteWriteOperation<TResult>(Func<TResult> function)
62          {
63              _rwLock.EnterWriteLock();
64              try
65              {
66                  return function();
67              }
68              finally
69              {
70                  _rwLock.ExitWriteLock();
71              }
72          }
73      }
74  }
```

./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          /// <inheritdoc />
12          /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation(System.Action)"]/*'/>
13          public void ExecuteReadOperation(Action action) => action();
14
15          /// <inheritdoc />
16          /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
17          public TResult ExecuteReadOperation<TResult>(Func<TResult> function) => function();
18
19          /// <inheritdoc />
20          /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation(System.Action)"]/*'/>
21          public void ExecuteWriteOperation(Action action) => action();
22
23          /// <inheritdoc />
24          /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            ↪  path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
            ↪  n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
25          public TResult ExecuteWriteOperation<TResult>(Func<TResult> function) => function();
26      }
27  }
```

./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
```

```
{
    public static class ThreadHelpers
    {
        public static readonly int DefaultMaxStackSize;
        public static readonly int ExtendedMaxStackSize = 200 * 1024 * 1024;
        public static readonly int DefaultSleepTimeout = 1;

        public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action) =>
        ↪  SyncInvokeWithExtendedStack(param, action, ExtendedMaxStackSize);

        public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action, int
        ↪  maxStackSize) => StartNew(param, action, maxStackSize).Join();

        public static void SyncInvokeWithExtendedStack(Action action) =>
        ↪  SyncInvokeWithExtendedStack(action, ExtendedMaxStackSize);

        public static void SyncInvokeWithExtendedStack(Action action, int maxStackSize) =>
        ↪  StartNew(action, maxStackSize).Join();

        public static Thread StartNew<T>(T param, Action<object> action) => StartNew(param,
        ↪  action, DefaultMaxStackSize);

        public static Thread StartNew<T>(T param, Action<object> action, int maxStackSize)
        {
            var thread = new Thread(new ParameterizedThreadStart(action), maxStackSize);
            thread.Start(param);
            return thread;
        }

        public static Thread StartNew(Action action) => StartNew(action, DefaultMaxStackSize);

        public static Thread StartNew(Action action, int maxStackSize)
        {
            var thread = new Thread(new ThreadStart(action), maxStackSize);
            thread.Start();
            return thread;
        }

        public static void Sleep() => Thread.Sleep(DefaultSleepTimeout);
    }
}
```

Index