Example21~23
CCR Failure Handling

Example 21: Causalities

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.Ccr.Core;

namespace Example
{
    class Program
    {
        // create dispatcher and dispatcher queue for scheduling tasks
        static Dispatcher dispatcher = new Dispatcher();
        static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);

        // Example 21: Causalities
        static void Main(string[] args)
        {
            /// <summary>
            /// 平行運作中發生的錯誤，將丟出給錯誤處理
            /// </summary>
            Port<Exception> exceptionPort = new Port<Exception>();

            // create a causality using the port instance
            Causality exampleCausality = new Causality("root cause", exceptionPort);

            // add causality to current thread
            Dispatcher.AddCausality(exampleCausality);

            // any unhandled exception from this point on, in this method or
            // any delegate that executes due to messages from this method,
            // will be posted on exceptionPort.

            Port<int> portInt = new Port<int>();
            Arbiter.Activate(_taskQueue,
                Arbiter.Receive(true, portInt, IntHandler)
            );

            // causalities flow when items are posted or Tasks are scheduled
            portInt.Post(1);
            portInt.Post(0);

```

```
            // activate a handler on the exceptionPort
            // This is the failure handler for the causality
            Arbiter.Activate(_taskQueue,
                Arbiter.Receive(true, exceptionPort,
                delegate(Exception ex)
                {
                    // deal with failure here
                    Console.WriteLine(ex);
                })
            );
        }

        static void IntHandler(int i)
        {
            // print active causalities
            foreach (Causality c in Dispatcher.ActiveCausalities)
            {
                Console.WriteLine(c.Name);
            }

            // expect DivideByZeroException that CCR will redirect to the causality
            Console.WriteLine("i: " + i);
            int k = 10 / i;
        }
    }
}
```

Example 22: Nested Causalities

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.Ccr.Core;

namespace Example
{
    class Program
    {
        //Example 15: Task.Execute
        static void Main(string[] args)
        {
            /// <summary>
            /// 與Example 16相同，但不透過port傳遞資料
            /// 而是透過Task.Execute執行任務，直接執行任務
            /// 當資料一傳入後，可馬上執行
            /// </summary>

            var dispatcher = new Dispatcher(
                0, // zero means use one thread per CPU, or 2 if only one CPU present
```

```
                "sample dispatcher" // friendly name assgined to OS threads
                );

            var taskQueue = new DispatcherQueue(
                "sample queue", // friendly name
                dispatcher // dispatcher instance
                );

            // directly enqueue a task with an inlined method plus a parameter
            taskQueue.Enqueue(
                new Task<int>(5, item => Console.WriteLine(item))
                );
        }
    }
}
```

Example 23: Joins and Causalities

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.Ccr.Core;

namespace Example
{
    class Program
    {
        // create dispatcher and dispatcher queue for scheduling tasks
        static Dispatcher dispatcher = new Dispatcher();
        static DispatcherQueue _taskQueue = new DispatcherQueue("sample queue", dispatcher);

        //Example 23: Joins and Causalities
        static void Main(string[] args)
        {
            PortSet<int, string> intPort = new PortSet<int, string>();
            //Port<int> intPort = new Port<int>();
            Port<int> leftPort = new Port<int>();
            Port<string> rightPort = new Port<string>();

            Port<Exception> leftExceptionPort = new Port<Exception>();
            Port<Exception> rightExceptionPort = new Port<Exception>();

            intPort.Post(1);
            intPort.Post("2");
            //intPort.Post(2);

            // activate two handlers that will execute concurrently and create
            // two different parallel causalities
            Arbiter.Activate(_taskQueue,
```

```
                Arbiter.Receive<int>(true, intPort,
                delegate(int i)
                {
                    Causality leftCausality = new Causality("left", leftExceptionPort);
                    Dispatcher.AddCausality(leftCausality);
                    Console.WriteLine("Left !");
                    leftPort.Post(i);
                })
            );

            Arbiter.Activate(_taskQueue,
                Arbiter.Receive<string>(true, intPort,
                delegate(string i)
                //Arbiter.Receive<int>(true, intPort,
                //delegate(int i)
                {
                    Causality rightCausality = new Causality("right", rightExceptionPort);
                    Dispatcher.AddCausality(rightCausality);
                    Console.WriteLine("Right !");
                    // post item on rightPort under the context of the right causality
                    rightPort.Post(i.ToString());
                })
            );

            // activate one join receiver that executes when items are available on
            // both leftPort and rightPort

            Arbiter.Activate(_taskQueue,
                Arbiter.JoinedReceive<int, string>(false, leftPort, rightPort,
                delegate(int i, string s)
                {
                    throw new InvalidOperationException("This exception will propagate to two peer
causalities");
                })
            );

            // activate a handler on the exceptionPort
            // This is the failure handler for the causality
            Arbiter.Activate(_taskQueue,
                Arbiter.Receive(true, leftExceptionPort,
                delegate(Exception ex)
                {
                    // deal with failure here
                    Console.WriteLine("Left causality:  " + ex);
                })
            );

            // activate a handler on the exceptionPort
            // This is the failure handler for the causality
            Arbiter.Activate(_taskQueue,
                Arbiter.Receive(true, rightExceptionPort,
                delegate(Exception ex)
```

```
190                 {
191                     // deal with failure here
192                     Console.WriteLine("Right causality:  " + ex);
193                 })
194             );
195         }
196     }
197 }
```