```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using O2DESNet;
using O2DESNet.Distributions;

namespace MMnQueue
{
    public class MMnQueue : State<MMnQueue.Statics>
    {
        #region Statics
        public class Statics : Scenario
        {
            /*******************************************************/
            /* All static properties shall be public,              */
            /* for both getter and setter.                         */
            /*******************************************************/
            public int ServerCapacity { get; set; }
            public double HourlyArrivalRate { get; set; }
            public double HourlyServiceRate { get; set; }
        }
        #endregion

        #region Dynamics
        /*********************************************************/
        /* All dynamic properties shall have only public getter,  */
        /* where setter should remain as private.                 */
        /*********************************************************/
        public int QueueLength { get; private set; } = 0;
        public int NInService { get; private set; } = 0;
        public HourCounter HourCounter_QueueLength { get; private set; } = new HourCounter();
        public HourCounter HourCounter_NInService { get; private set; } = new HourCounter();
        #endregion

        #region Events
        private abstract class InternalEvent : Event<MMnQueue, Statics> { } // event adapter

        private class ArriveEvent : InternalEvent
        {
            public override void Invoke()
            {
                Log("Arrive.");
                This.QueueLength++;
                Execute(new StartServiceEvent());
                Schedule(new ArriveEvent(), TimeSpan.FromHours(
                    Exponential.Sample(This.DefaultRS, 1 / This.Config.HourlyArrivalRate)));
            }
        }
        private class StartServiceEvent : InternalEvent
        {
            public override void Invoke()
            {
                if (This.QueueLength > 0 && This.NInService < This.Config.ServerCapacity)
                {
                    Log("Start Service.");
                    This.QueueLength--;
                    This.NInService++;
                    Schedule(new DepartEvent(), TimeSpan.FromHours(
                        Exponential.Sample(This.DefaultRS, 1 / This.Config.HourlyServiceRate)));
                }
                This.HourCounter_NInService.ObserveCount(This.NInService, ClockTime);
                This.HourCounter_QueueLength.ObserveCount(This.QueueLength, ClockTime);
            }
        }
        private class DepartEvent : InternalEvent
        {
            public override void Invoke()
```

```
            {
                Log("Depart.");
                This.NInService--;
                Execute(new StartServiceEvent());
            }
        }
        #endregion

        public MMnQueue(Statics config, int seed, string tag = null) : base(config, seed, tag)
        {
            Name = "MMnQueue";
            InitEvents.Add(new ArriveEvent { This = this });
        }

        public override void WarmedUp(DateTime clockTime)
        {
            HourCounter_NInService.WarmedUp(clockTime);
            HourCounter_QueueLength.WarmedUp(clockTime);
        }

        public override void WriteToConsole(DateTime? clockTime = null)
        {
            Console.WriteLine(clockTime);
            Console.WriteLine("Queue Length:\t{0}", QueueLength);
            Console.WriteLine("# in Serveice:\t{0}", NInService);
            Console.WriteLine("Avg. Q Length:\t{0:F2}", HourCounter_QueueLength.AverageCount);
            Console.WriteLine("Server Util.:\t{0:F2}", HourCounter_NInService.AverageCount /
Config.ServerCapacity);
        }
    }
}
```