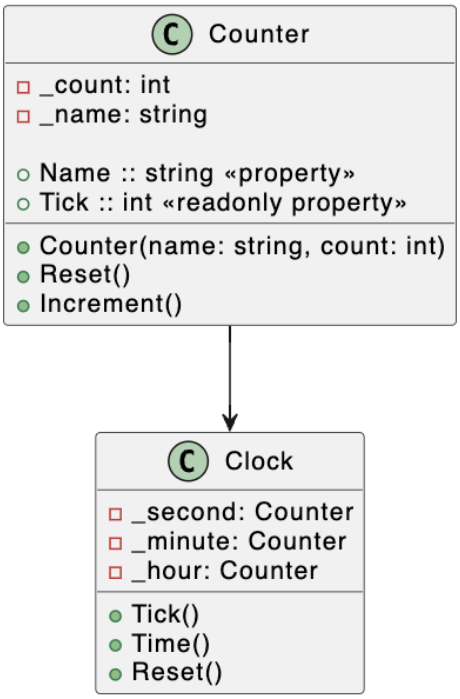
Swinburne University of Technology

COS20007 Object Oriented Programming

Clock Class

PDF generated at 00:58 on Saturday 23rd September, 2023


Counter
_count: int
_name: string
Name :: string «property»
Tick :: int «readonly property»
Counter(name: string, count: int)
Reset()
Increment()
Clock
_second: Counter
_minute: Counter
_hour: Counter
Tick()
Time()
Reset()

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


namespace ClockClass

{
    class Program
    {
        static void Main(string[] args)
        {
            Clock clock = new Clock();
            int i;

            for (i = 0; i < 86400; i++)
            {
                Thread.Sleep(1);
                Console.Clear();
                clock.Tick();
                Console.WriteLine(clock.CurrentTime());
            }
        }
    }
}
```

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

namespace ClockClass
{
    public class Clock
    {
        Counter _seccond = new Counter("seccond");
        Counter _minute = new Counter("minute");
        Counter _hour = new Counter("hour");


        public void Tick()
        {
            _seccond.Increment();
            if(_seccond.Tick>59)
            {
                _minute.Increment();
                _seccond.Reset();
                if (_minute.Tick>59)
                {
                    _hour.Increment();
                    _minute.Reset();
                    if (_hour.Tick>23)
                    {
                        Reset();
                    }
                }
            }
        }

        public void SetTime(string s)
        {
            string[] array = s.Split(":");

            _hour = new Counter("hour", int.Parse(array[0]));
            _minute = new Counter("minute", int.Parse(array[1]));
            _seccond = new Counter("second", int.Parse(array[2]));
        }

        public void Reset()
        {
            _seccond.Reset();
            _minute.Reset();
            _hour.Reset();
        }

        public string CurrentTime()
        {
            return $"{_hour.Tick:D2}:{_minute.Tick:D2}:{_seccond.Tick:D2}";
```

```
54            }
55        }
56    }
```

```
using ClockClass;
using NUnit.Framework;

namespace ClockTest
{
    [TestFixture]
    public class Tests
    {
        private Clock _clock;

        [SetUp]
        public void Setup()
        {
            _clock = new Clock();
        }

        [Test]
        public void IncrementSeconds()
        {
            _clock.Tick();
            Assert.That(_clock.CurrentTime(), Is.EqualTo("00:00:01"));
        }

        [Test]
        public void IncrementMinutes()
        {
            for (int i = 0; i < 60; i++)
            {
                _clock.Tick();
            }
            Assert.That(_clock.CurrentTime(), Is.EqualTo("00:01:00"));
        }

        [Test]
        public void IncrementHours()
        {
            for (int i = 0; i < 3600; i++)
            {
                _clock.Tick();
            }
            Assert.That(_clock.CurrentTime(), Is.EqualTo("01:00:00"));
        }

        [Test]
        public void ClockResetsToZero()
        {
            for (int i = 0; i < 86400; i++)
            {
                _clock.Tick();
            }
            Assert.That(_clock.CurrentTime(), Is.EqualTo("00:00:00"));
        }

```

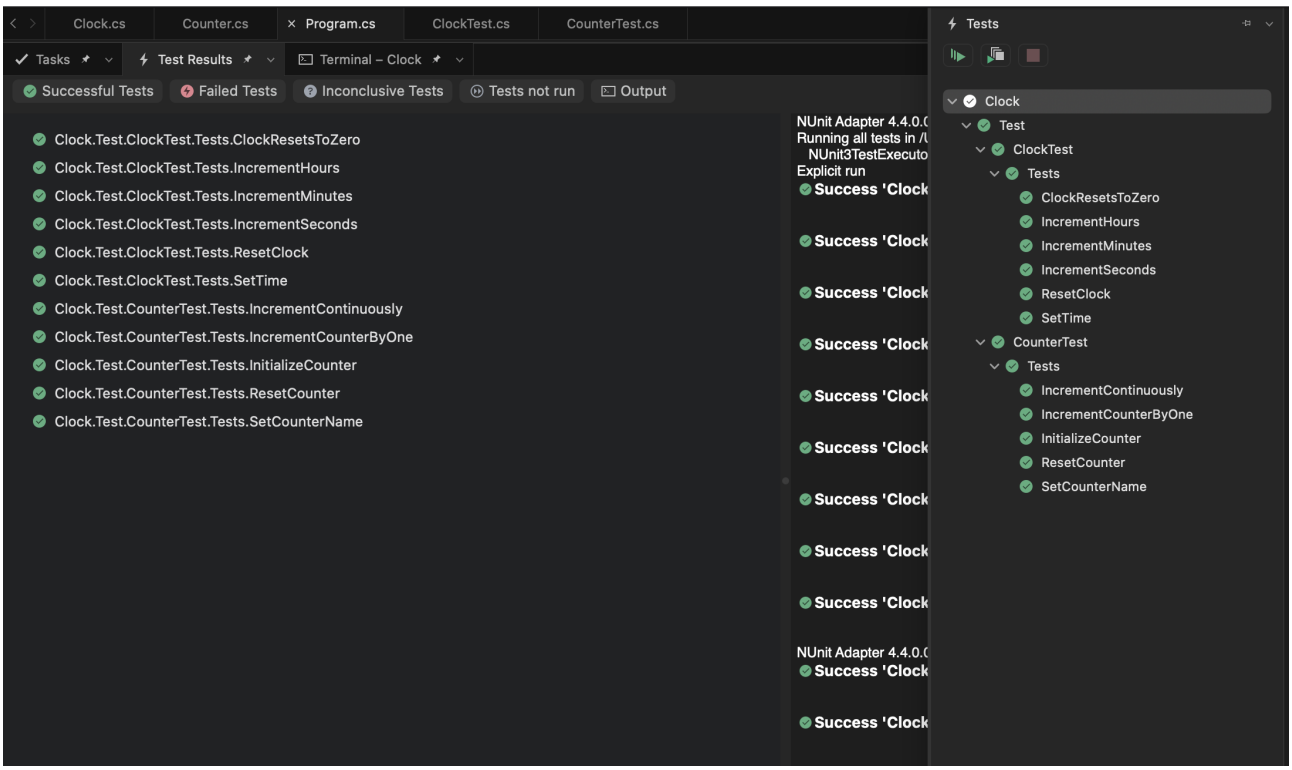
```
54          [Test]
55          public void SetTime()
56          {
57              _clock.SetTime("12:34:56");
58              Assert.That(_clock.CurrentTime(), Is.EqualTo("12:34:56"));
59          }
60
61          [Test]
62          public void ResetClock()
63          {
64              _clock.SetTime("12:34:56");
65              _clock.Reset();
66              Assert.That(_clock.CurrentTime(), Is.EqualTo("00:00:00"));
67          }
68      }
69  }
```

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

namespace ClockClass
{
    public class Counter
    {

        private int _count;
        private string _name;

        public Counter(string name)
        {
            _count = 0;
            _name = name;
        }

        public Counter(string name, int count)
        {
            _name = name;
            _count = count;
        }

        public void Reset()
        {
            _count = 0;
        }

        public void Increment()
        {
            _count++;
        }

        public string Name
        {
            get
            {
                return _name;
            }
            set
            {
                _name = value;
            }
        }

        public int Tick
        {
            get
            {
                return _count;
```

```
54                }
55            }
56        }
57    }
```

```
using ClockClass;
using NUnit.Framework;

namespace CounterTest
{
    [TestFixture]
    public class Tests
    {
        private Counter _counter;

        [SetUp]
        public void Setup()
        {
            _counter = new Counter("Test Counter");
        }

        [Test]
        public void InitializeCounter()
        {
            Assert.That(_counter.Tick, Is.EqualTo(0));
        }

        [Test]
        public void IncrementCounterByOne()
        {
            _counter.Increment();
            Assert.That(_counter.Tick, Is.EqualTo(1));
        }

        [Test]
        public void IncrementContinuously()
        {
            _counter.Increment();
            _counter.Increment();
            _counter.Increment();
            Assert.That(_counter.Tick, Is.EqualTo(3));
        }

        [Test]
        public void ResetCounter()
        {
            _counter.Increment();
            _counter.Reset();
            Assert.That(_counter.Tick, Is.EqualTo(0));
        }

        [Test]
        public void SetCounterName()
        {
            _counter.Name = "New Counter Name";
            Assert.That(_counter.Name, Is.EqualTo("New Counter Name"));
        }
    }
```

```
54 }
```


Clock.cs
Counter.cs
Program.cs
ClockTest.cs
CounterTest.cs
Tasks
Test Results
Terminal – Clock
Successful Tests
Failed Tests
Inconclusive Tests
Tests not run
Output
Clock.Test.ClockTest.Tests.ClockResetsToZero
Clock.Test.ClockTest.Tests.IncrementHours
Clock.Test.ClockTest.Tests.IncrementMinutes
Clock.Test.ClockTest.Tests.IncrementSeconds
Clock.Test.ClockTest.Tests.ResetClock
Clock.Test.ClockTest.Tests.SetTime
Clock.Test.CounterTest.Tests.IncrementContinuously
Clock.Test.CounterTest.Tests.IncrementCounterByOne
Clock.Test.CounterTest.Tests.InitializeCounter
Clock.Test.CounterTest.Tests.ResetCounter
Clock.Test.CounterTest.Tests.SetCounterName
Explicit run
Tests
Clock
Test
ClockTest
Tests
ClockResetsToZero
IncrementHours
IncrementMinutes
IncrementSeconds
ResetClock
SetTime
CounterTest
Tests
IncrementContinuously
IncrementCounterByOne
InitializeCounter
ResetCounter
SetCounterName


Terminal – Clock
04:44:40
Build successful.