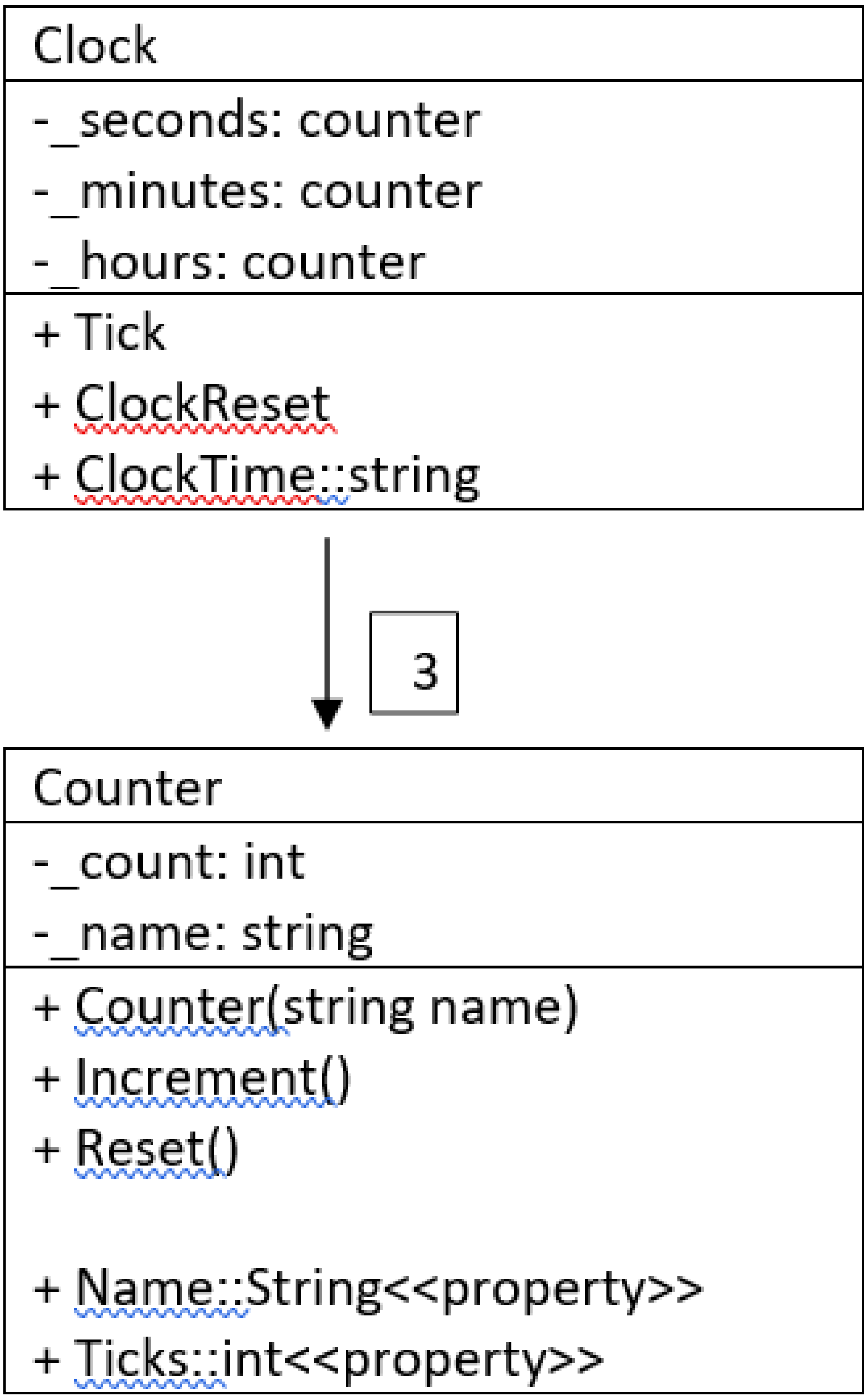
Swinburne University of Technology

COS20007 Object Oriented Programming

3.1P - Clock Class

PDF generated at 12:26 on Thursday 23rd March, 2023


Clock
-_seconds: counter
-_minutes: counter
-_hours: counter
+ Tick
+ ClockReset
+ ClockTime::string
3
Counter
-_count: int
-_name: string
+ Counter(string name)
+ Increment()
+ Reset()
+ Name::String<<property>>
+ Ticks::int<<property>>

```
using Microsoft.VisualBasic;
using System;

namespace CounterTask
{
    class Program

    {

        static void PrintCounters(Counter[] counters)
        {


            foreach (Counter counter in counters)
            {
                Console.WriteLine("{0} is {1},", counter.Name, counter.Ticks);
            }

        }



        static void Main(string[] args)

        {

            Counter[] myCounters = new Counter[3];


            myCounters[0] = new Counter("Counter 1");
            myCounters[1] = new Counter("Counter 2");
            myCounters[2] = myCounters[0];

            Clock _Clock = new Clock();

            for (int i = 0; i < 9; i++)
            {
                myCounters[0].Increment();
            }
            for (int i = 0;i < 14; i++)
            {
                myCounters[1].Increment();
            }

            PrintCounters(myCounters);

            myCounters[2].Reset();

            PrintCounters(myCounters);

            for (int i = 0; i < 120 ; i++)
            {
                Console.WriteLine(_Clock.ClockTime());
                _Clock.Tick();

            }
            Console.WriteLine("The time is enough!");


        }

    }

}

```

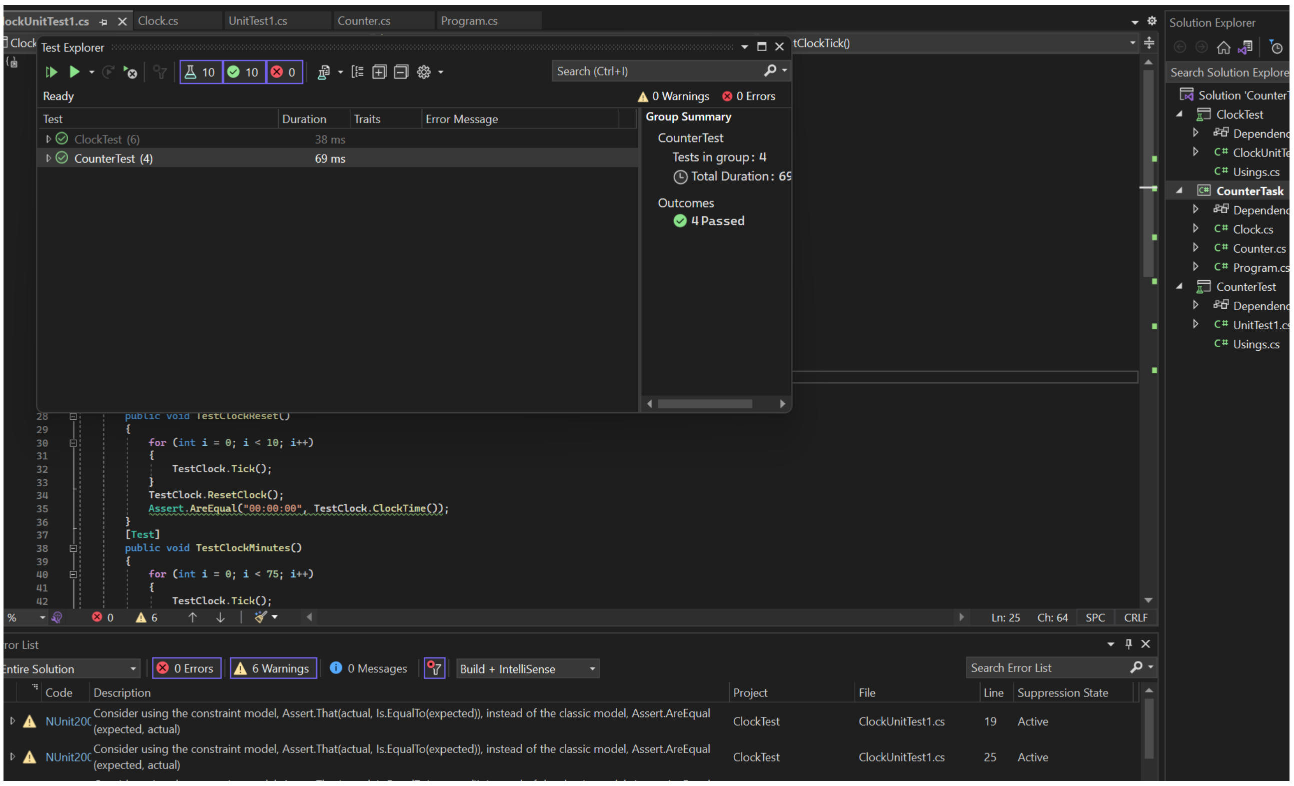
```
using CounterTask;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CounterTask
{


    public class Clock
    {

        private Counter _Seconds = new Counter("Seconds");
        private Counter _Minutes = new Counter("Minutes");
        private Counter _Hours = new Counter("Hours");


        public void Tick()
        {
            _Seconds.Increment();
            if (_Seconds.Ticks > 59)
            {
                _Seconds.Reset();
                _Minutes.Increment();
                if (_Minutes.Ticks > 59)
                {
                    _Minutes.Reset();
                    _Hours.Increment();
                    if (_Hours.Ticks > 23)
                    {
                        ResetClock();
                    }

                }
            }

        }

        public void ResetClock()
        {
            _Seconds.Reset();
            _Minutes.Reset();
            _Hours.Reset();

        }
        public string ClockTime()
        {

            return  $"{_Hours.Ticks:D2}" + ":" + $"{_Minutes.Ticks:D2}" + ":" +
 ↪  $"{_Seconds.Ticks:D2}";

```

```
            }

        }


}
```

```
using CounterTask;
namespace ClockUnitTest
{
    public class CLockTest
    {
        Clock TestClock;

        [SetUp]

        public void Setup()
        {
            TestClock = new Clock();

        }

        [Test]
        public void TestClockInitialValue()
        {
            Assert.AreEqual("00:00:00", TestClock.ClockTime());
        }
        [Test]
        public void TestClockTick()
        {
            TestClock.Tick();
            Assert.AreEqual("00:00:01", TestClock.ClockTime());
        }
        [Test]
        public void TestClockReset()
        {
            for (int i = 0; i < 10; i++)
            {
                TestClock.Tick();
            }
            TestClock.ResetClock();
            Assert.AreEqual("00:00:00", TestClock.ClockTime());
        }
        [Test]
        public void TestClockMinutes()
        {
            for (int i = 0; i < 75; i++)
            {
                TestClock.Tick();
            }
            Assert.AreEqual("00:01:15", TestClock.ClockTime());
        }
        [Test]
        public void TestClockHours()
        {
            for (int i = 0; i < 45001; i++) //45001 seconds/ticks = 12h 30m 1s
            {
                TestClock.Tick();
            }
            Assert.AreEqual("12:30:01", TestClock.ClockTime());
        }
        [Test]
        public void TestClock24HReset()
        {
            for (int i = 0; i < 86400; i++) //86400 seconds/ticks = 24h
            {
                TestClock.Tick();
            }
            Assert.AreEqual("00:00:00", TestClock.ClockTime());
        }
    }
}
```

```
using System.Security.Cryptography.X509Certificates;

namespace CounterTask
{
    public class Counter
    {
        private int _count;
        private string _name;

        public Counter (string name)

        {
            _name = name;
            _count = 0;
        }
        public void Increment()
        {
            _count++;

        }
        public void Reset()
        {
            _count = 0;
        }
        public string Name
        {
            get
            {
                return _name;
            }
            set
            {
                _name = value;
            }
        }
        public int Ticks
        {
            get
            {
                return (int)_count;
            }
        }



    }





}
```

```
using CounterTask;
namespace CounterTest
{
    public class Tests
    {
        private Counter _TestCounter;

        [SetUp]
        public void Setup()
        {
            _TestCounter = new Counter("TestCounter");

        }

        [Test]
        public void Intialising0()
        {
            Assert.AreEqual(0, _TestCounter.Ticks);
        }
        [Test]
        public void IncrementCounter()
        {
            _TestCounter.Increment();
            Assert.AreEqual(1, _TestCounter.Ticks);
        }

        [TestCase (10, 10)]
        public void IncrementMatchCounter(int Ticks, int Increm )
        {

            int index;
            for ( index = 0; index < Ticks; index++ )
            {
                _TestCounter.Increment();
            }

            Assert.AreEqual(Increm, _TestCounter.Ticks);
        }
        [Test]
        public void TestReset()
        {
            _TestCounter.Reset();
            Assert.AreEqual(0, _TestCounter.Ticks);
        }







    }
}
```



10 | 10 | 0

Ready

0 Warnings 0 Errors

Test	Duration	Traits	Error Message
ClockTest (6)	38 ms		
CounterTest (4)	69 ms		

Group Summary
CounterTest
Tests in group: 4
Total Duration: 69
Outcomes
4 Passed

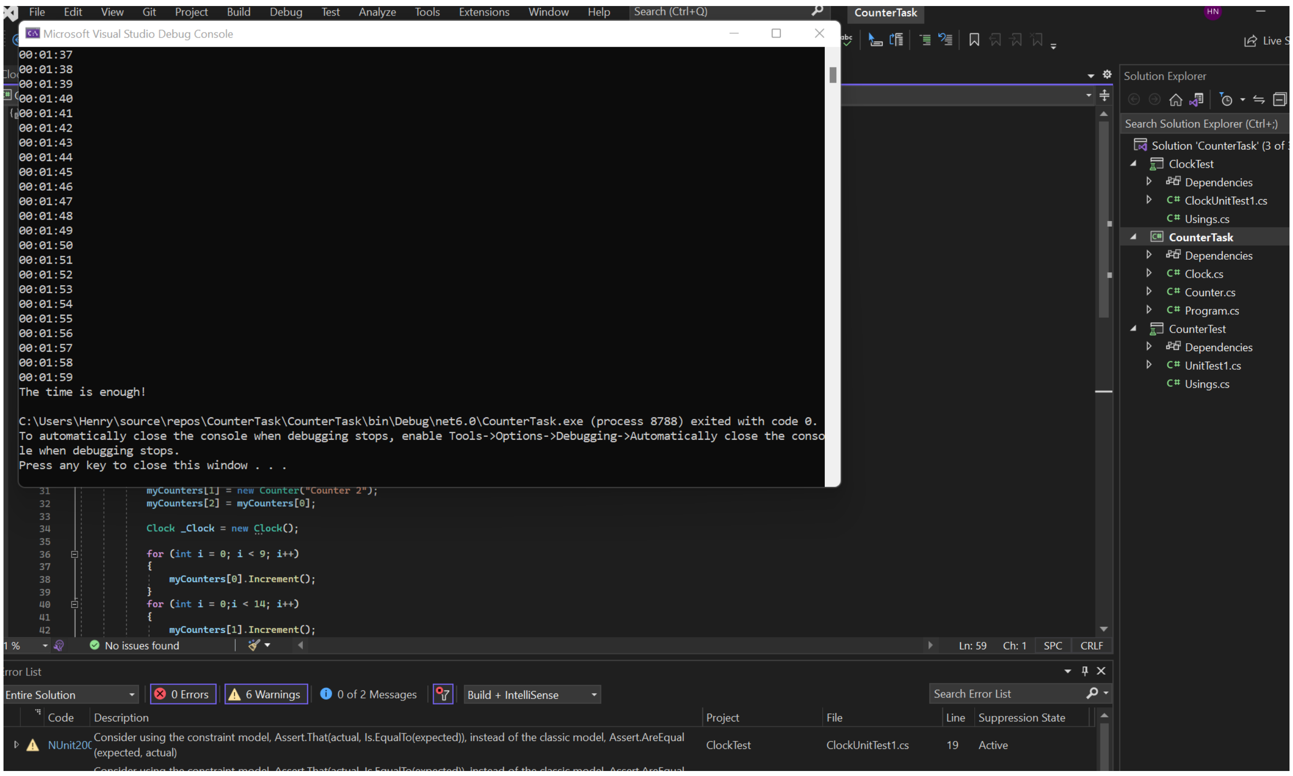
```
        public void TestClockReset()
        {
            for (int i = 0; i < 10; i++)
            {
                TestClock.Tick();
            }
            TestClock.ResetClock();
            Assert.AreEqual("00:00:00", TestClock.ClockTime());
        }
        [Test]
        public void TestClockMinutes()
        {
            for (int i = 0; i < 75; i++)
            {
                TestClock.Tick();
```

0 | 6 | Ln: 25 | Ch: 64 | SPC | CRLF

Error List

0 Errors | 6 Warnings | 0 Messages

Code	Description	Project	File	Line	Suppression State
NUnit200	Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual (expected, actual)	ClockTest	ClockUnitTest1.cs	19	Active
NUnit200	Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual (expected, actual)	ClockTest	ClockUnitTest1.cs	25	Active



```
00:01:37
00:01:38
00:01:39
00:01:40
00:01:41
00:01:42
00:01:43
00:01:44
00:01:45
00:01:46
00:01:47
00:01:48
00:01:49
00:01:50
00:01:51
00:01:52
00:01:53
00:01:54
00:01:55
00:01:56
00:01:57
00:01:58
00:01:59
The time is enough!

C:\Users\Henry\source\repos\CounterTask\CounterTask\bin\Debug\net6.0\CounterTask.exe (process 8788) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

```
            myCounters[1] = new Counter("Counter 2");
            myCounters[2] = myCounters[0];

            Clock _Clock = new Clock();

            for (int i = 0; i < 9; i++)
            {
                myCounters[0].Increment();
            }
            for (int i = 0;i < 14; i++)
            {
                myCounters[1].Increment();
```

No issues found — Ln: 59 Ch: 1 SPC CRLF

Error List — Entire Solution — 0 Errors — 6 Warnings — 0 of 2 Messages — Build + IntelliSense

Code	Description	Project	File	Line	Suppression State
NUnit20(	Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual (expected, actual)	ClockTest	ClockUnitTest1.cs	19	Active