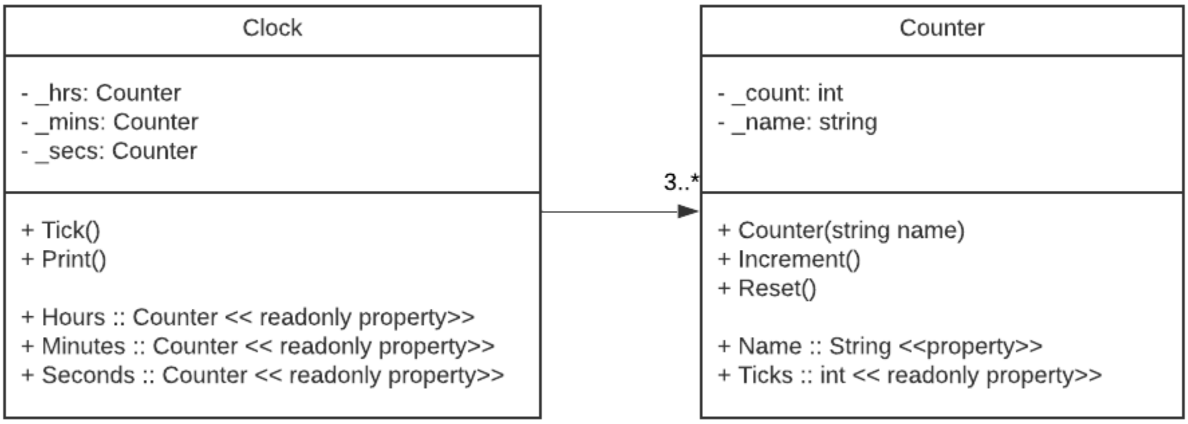
Swinburne University of Technology

COS20007 Object Oriented Programming

3.1P - Clock Class

PDF generated at 20:08 on Thursday 30th March, 2023

Produced by OOP Submission Formatter


Clock
- _hrs: Counter
- _mins: Counter
- _secs: Counter
+ Tick()
+ Print()
+ Hours :: Counter << readonly property>>
+ Minutes :: Counter << readonly property>>
+ Seconds :: Counter << readonly property>>
3..*
Counter
- _count: int
- _name: string
+ Counter(string name)
+ Increment()
+ Reset()
+ Name :: String <<property>>
+ Ticks :: int << readonly property>>

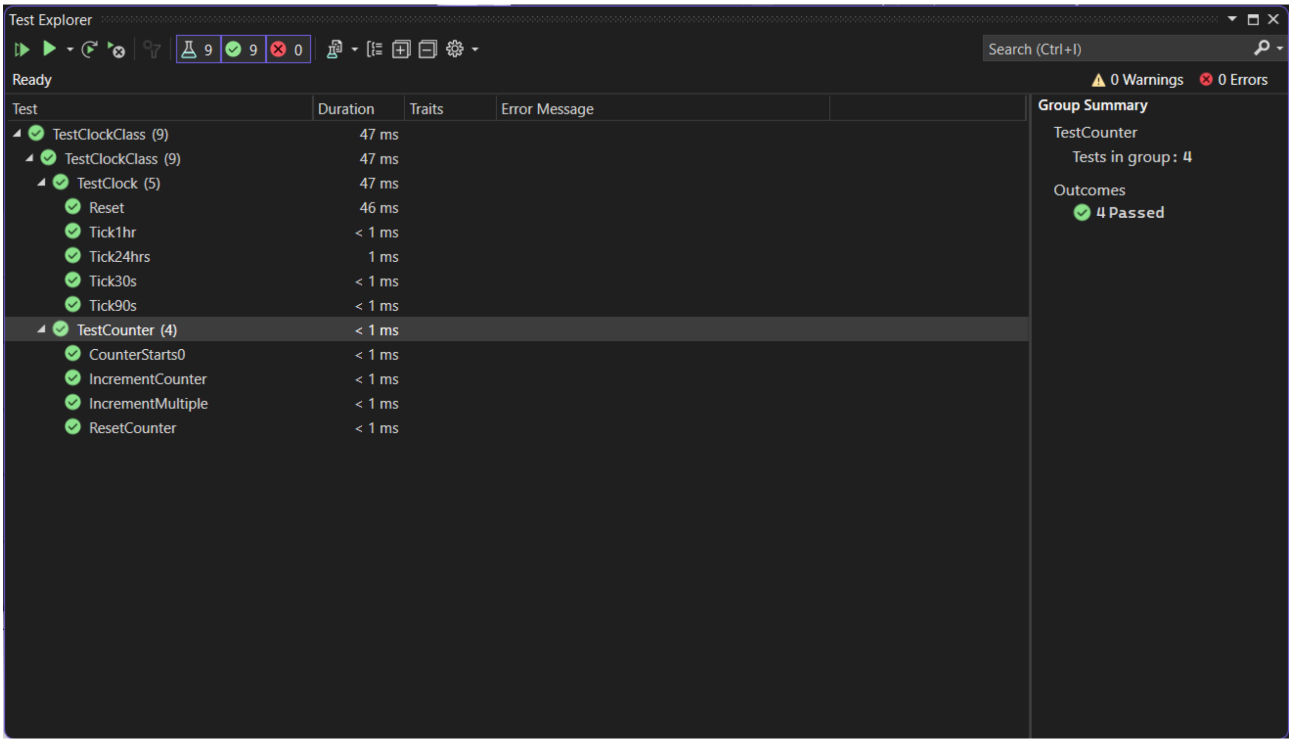
```
namespace ClockClass
{
    class MainClass
    {
        public static void Main(string[] args)
        {
            Clock myClock = new Clock();

            for (int i = 0; i < 10000; i++)
            {
                myClock.Tick();
            }
            myClock.Print();
        }
    }
}
```

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

namespace ClockClass
{
    public class Clock
    {
        private Counter _hrs = new Counter("hrs");
        private Counter _mins = new Counter("mins");
        private Counter _secs = new Counter("secs");

        public Counter Hours
        {
            get
            {
                return _hrs;
            }
        }
        public Counter Minutes
        {
            get
            {
                return _mins;
            }
        }
        public Counter Seconds
        {
            get
            {
                return _secs;
            }
        }

        public void Tick()
        {
            if (_secs.Ticks <= 58)
            {
                _secs.Increment();
            }
             else if (_mins.Ticks <= 58)
            {
                _mins.Increment();

                _secs.Reset();
            }
             else if (_hrs.Ticks <= 22)
            {
                _hrs.Increment();

                _mins.Reset();
                _secs.Reset();
            }
             else
            {
                _hrs.Reset();
                _mins.Reset();
                _secs.Reset();
            }
        }

        public void Print()
        {
            Console.WriteLine("{0}:{1}:{2}", _hrs.Ticks.ToString("00"),
↪   _mins.Ticks.ToString("00"), _secs.Ticks.ToString("00"));
        }
    }
}
```

```

using NUnit.Framework;

namespace TestClockClass
{
    public class TestClock
    {
        Clock myClock;
        [SetUp]
        public void Setup()
        {
            myClock = new Clock();
        }


        [Test]
        public void Tick30s()
        {
            for (int i = 0; i < 30; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Seconds.Ticks, Is.EqualTo(30));
            Assert.That(myClock.Minutes.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Hours.Ticks, Is.EqualTo(0));
        }

        [Test]
        public void Tick90s()
        {
            for (int i = 0; i < 90; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Seconds.Ticks, Is.EqualTo(30));
            Assert.That(myClock.Minutes.Ticks, Is.EqualTo(1));
            Assert.That(myClock.Hours.Ticks, Is.EqualTo(0));
        }

        [Test]
        public void Tick1hr()
        {
            for (int i = 0; i < 3600; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Seconds.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Minutes.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Hours.Ticks, Is.EqualTo(1));
        }

        [Test]
        public void Tick24hrs()
        {
            for (int i = 0; i < 86400; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Seconds.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Minutes.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Hours.Ticks, Is.EqualTo(0));
        }

        [Test]
        public void Reset()
        {

            for (int i = 0; i < 43200; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Seconds.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Minutes.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Hours.Ticks, Is.EqualTo(12));

            for (int i = 0; i < 43200; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Seconds.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Minutes.Ticks, Is.EqualTo(0));
            Assert.That(myClock.Hours.Ticks, Is.EqualTo(0));
        }

    }
}
```

```csharp
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 string Name
        {
            get
            {
                return _name;
            }
            set
            {
                _name = value;
            }
        }
        public int Ticks
        {
            get
            {
                return _count;
            }
        }

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

        public void Increment()
        {
            _count++;
        }

        public void Reset()
        {
            _count = 0;
        }
    }
}
```

```
namespace TestClockClass
{
    [TestFixture]
    public class TestCounter
    {
        Counter myCounter;

        [SetUp]
        public void Setup()
        {
            myCounter = new Counter("counter");
        }

        [Test]
        public void CounterStarts0()
        {
            Assert.That(myCounter.Ticks, Is.EqualTo(0));
        }

        [Test]
        public void IncrementCounter()
        {
            myCounter.Increment();
            Assert.That(myCounter.Ticks, Is.EqualTo(1));
        }

        [Test]
        public void IncrementMultiple()
        {
            for (int i = 0; i < 10; i++)
            {
                myCounter.Increment();
            }
            Assert.That(myCounter.Ticks, Is.EqualTo(10));
        }

        [Test]
        public void ResetCounter()
        {
            myCounter.Increment();
            myCounter.Reset();

            Assert.That(myCounter.Ticks, Is.EqualTo(0));
        }
    }


}
```



Test Explorer

Ready

⚠ 0 Warnings ⊗ 0 Errors

```
Test                            Duration   Traits   Error Message
◢ ✓ TestClockClass (9)             47 ms
  ◢ ✓ TestClockClass (9)           47 ms
    ◢ ✓ TestClock (5)              47 ms
        ✓ Reset                    46 ms
        ✓ Tick1hr                < 1 ms
        ✓ Tick24hrs                1 ms
        ✓ Tick30s                < 1 ms
        ✓ Tick90s                < 1 ms
    ◢ ✓ TestCounter (4)          < 1 ms
        ✓ CounterStarts0         < 1 ms
        ✓ IncrementCounter       < 1 ms
        ✓ IncrementMultiple      < 1 ms
        ✓ ResetCounter           < 1 ms
```

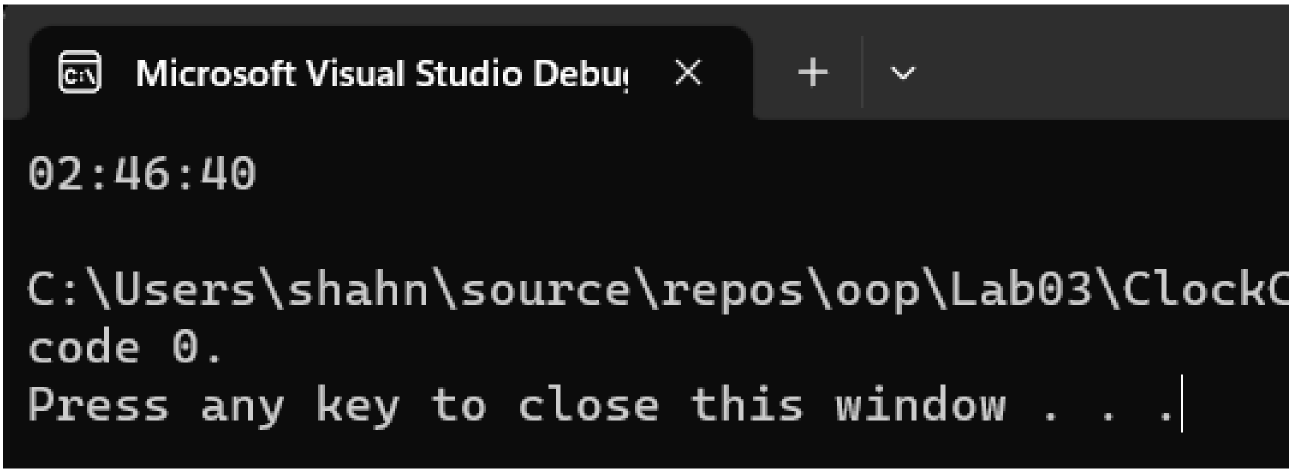
Group Summary

TestCounter

Tests in group: 4

Outcomes

✓ 4 Passed



```
02:46:40

C:\Users\shahn\source\repos\oop\Lab03\ClockC
code 0.
Press any key to close this window . . .
```