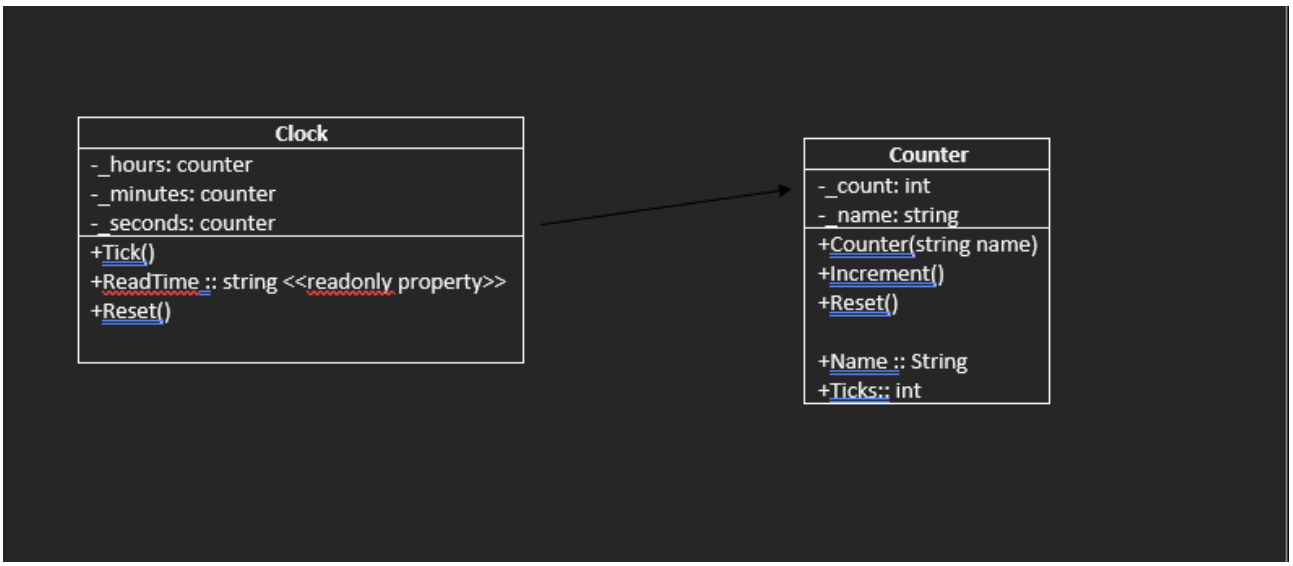
Swinburne University of Technology

COS20007 Object Oriented Programming

Clock Class

PDF generated at 03:18 on Monday 21st August, 2023


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

```
using System;


namespace ClockClass
{
    public class Program
    {
        static void Main()
        {
            Clock clock = new Clock();

            for (int i = 0; i < 86400; i++)
            {
                clock.Tick();

                Console.WriteLine(clock.Readtime);
            }

            Console.ReadLine();
            clock.Reset();
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Diagnostics.Metrics;
using System.Linq;
using System.Reflection;
using System.Text;
using System.Threading.Tasks;
using ClockClass;

namespace ClockClass
{
    public class Clock
    {
        private Counter _hours;
        private Counter _minutes;
        private Counter _seconds;

        public Clock()
        {

            _hours = new Counter("hours");
            _minutes = new Counter("minutes");
            _seconds = new Counter("seconds");
        }

        public void Tick()
        {
            _seconds.Increment();

            if (_seconds.Ticks == 60)
            {
                _seconds.reset();
                _minutes.Increment();

                if (_minutes.Ticks == 60)
                {
                _minutes.reset();
                _hours.Increment();
                    if (_hours.Ticks == 24)
                    {
                        _hours.reset();
                    }
                }

            }
        }

        public string Readtime
        {
            get
            {
                return String.Format("{0:D2}:{1:D2}:{2}", _hours.Ticks,
↪  _minutes.Ticks, _seconds.Ticks.ToString("00"));
```

```
53
54              }
55          }
56          public void Reset()
57          {
58
59
60              _hours.reset();
61              _minutes.reset();
62              _seconds.reset();
63          }
64
65      }
66  }
```

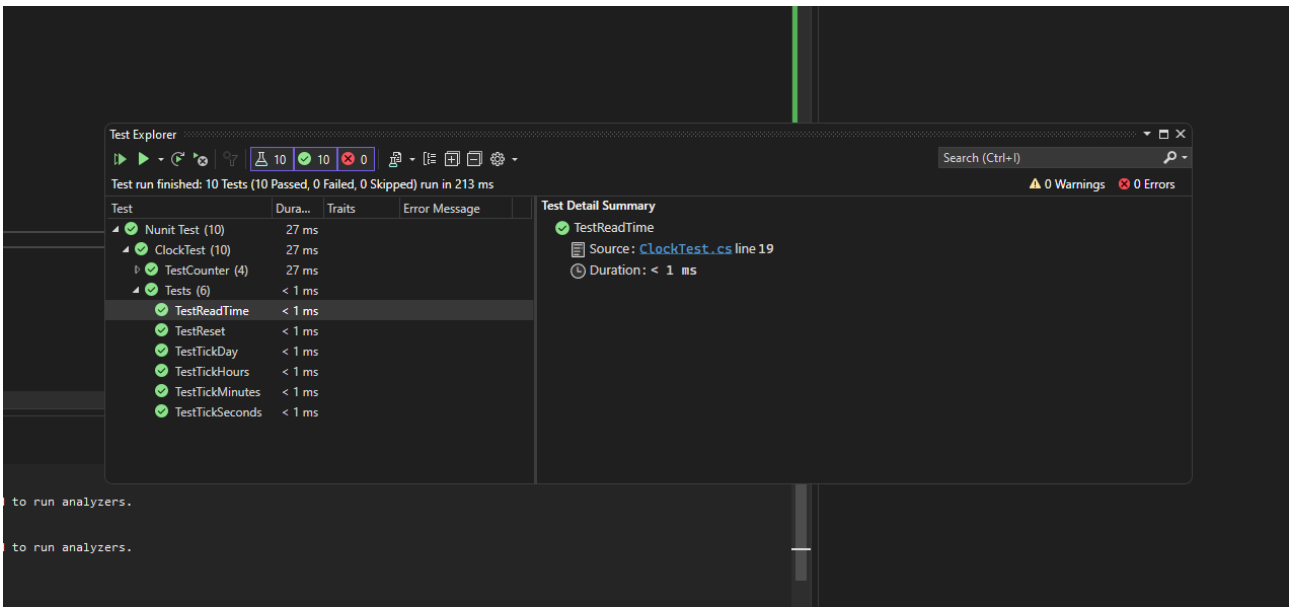
```
using System;
using NUnit.Framework;
using ClockClass;


namespace ClockTest
{
    public class Tests
    {
        private Clock _clock;

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

        [Test]
        public void TestReadTime()
        {
            Assert.AreEqual("00:00:00", _clock.Readtime);
        }

        [Test]
        public void TestTickSeconds()
        {
            _clock.Tick();
            Assert.AreEqual("00:00:01", _clock.Readtime);

        }

        [Test]
        public void TestTickMinutes()
        {
            for (int i = 0; i < 60; i++)
            {
                _clock.Tick();
            }
            Assert.AreEqual("00:01:00", _clock.Readtime);
        }
        [Test]
        public void TestTickHours()
        {
            for (int i = 0; i < 3600; i++)
            {
                _clock.Tick();
            }
            Assert.AreEqual("01:00:00", _clock.Readtime);

        }
        [Test]
        public void TestTickDay()
        {
```

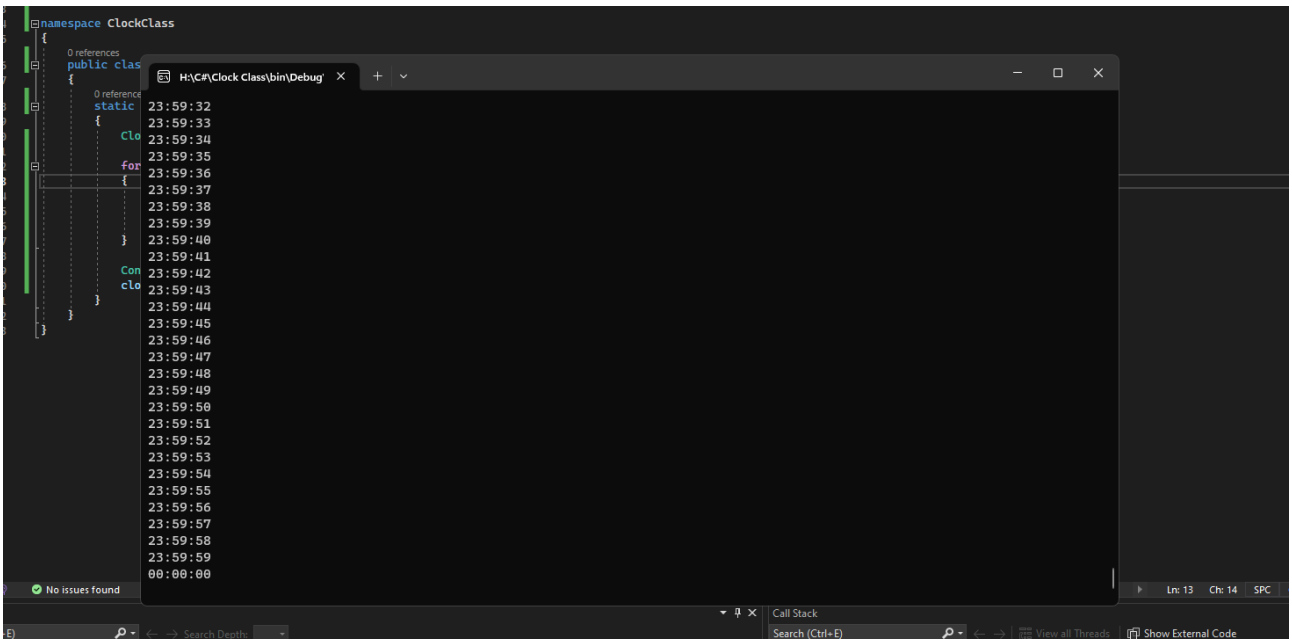
```
54              for (int i = 0; i < 86401; i++)
55              {
56                  _clock.Tick();
57              }
58              Assert.AreEqual("00:00:01", _clock.Readtime);
59
60          }
61          [Test]
62
63          public void TestReset()
64          {
65              for (int i = 0; i < 3661; i++)
66              {
67                  _clock.Tick();
68              }
69              _clock.Reset();
70              Assert.AreEqual("00:00:00", _clock.Readtime);
71
72
73
74
75          }
76      }
77  }
```

```
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;
        }
    }
}
```

```
using System;
using NUnit.Framework;
using ClockClass;



namespace ClockTest
{
    public class TestCounter
    {


        private Counter _counter;

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

        }

        [Test]
        public void Starts()
        {
            Assert.AreEqual(0, _counter.Ticks);
        }
        [Test]
        public void Increment()
        {
            _counter.Increment();
            Assert.AreEqual(1, _counter.Ticks);
        }
        [Test]

        public void Inecrement2()
        {
            for (int i = 0; i < 86400; i++)
            {
                _counter.Increment();
            }

            Assert.AreEqual(86400, _counter.Ticks);

        }
        [Test]
        public void Reset()
        {
            _counter.reset();


            Assert.AreEqual(0, _counter.Ticks);

        }
```

```
54      }
55  }
```


Test Explorer
10
10
0
Search (Ctrl+I)
Test run finished: 10 Tests (10 Passed, 0 Failed, 0 Skipped) run in 213 ms
0 Warnings
0 Errors
Test
Dura...
Traits
Error Message
Nunit Test (10) 27 ms
ClockTest (10) 27 ms
TestCounter (4) 27 ms
Tests (6) < 1 ms
TestReadTime < 1 ms
TestReset < 1 ms
TestTickDay < 1 ms
TestTickHours < 1 ms
TestTickMinutes < 1 ms
TestTickSeconds < 1 ms
Test Detail Summary
TestReadTime
Source: ClockTest.cs line 19
Duration: < 1 ms
to run analyzers.
to run analyzers.


namespace ClockClass
H:\C#\Clock Class\bin\Debug
23:59:32
23:59:33
23:59:34
23:59:35
23:59:36
23:59:37
23:59:38
23:59:39
23:59:40
23:59:41
23:59:42
23:59:43
23:59:44
23:59:45
23:59:46
23:59:47
23:59:48
23:59:49
23:59:50
23:59:51
23:59:52
23:59:53
23:59:54
23:59:55
23:59:56
23:59:57
23:59:58
23:59:59
00:00:00
No issues found
Call Stack
Search (Ctrl+E)
Show External Code