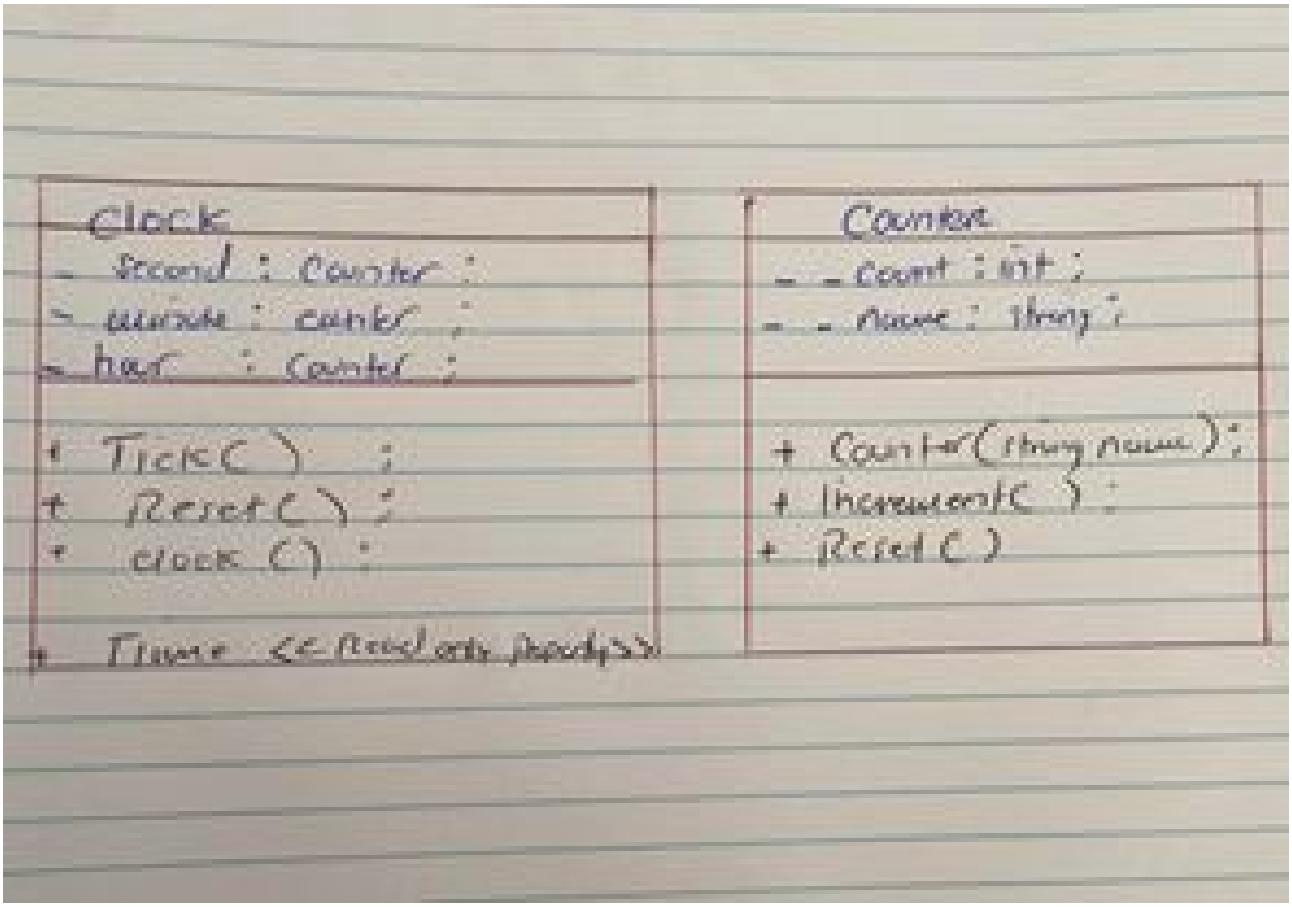
Swinburne University of Technology

COS20007 Object Oriented Programming

3.1P - Clock Class

PDF generated at 15:16 on Tuesday 28th March, 2023


Clock
- second : Counter ;
- minute : counter ;
- hour : counter ;
+ Tick ( ) ;
+ Reset ( ) ;
+ clock ( ) ;
Counter
- - count : int ;
- - name : string ;
+ Counter (string name) ;
+ Increment ( ) ;
+ Reset ( )

```
using System;

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


            for (int i = 0; i <7350; i++)
            {
            clock.Tick();

            Console.WriteLine(clock.Time);
            }

            Console.ReadLine();


        }
    }
}
```

```
using System;

namespace clock
{
    public class Clock
    {
        // each field is a counter object of type Counter
        private Counter _seconds;
        private Counter _minutes;
        private Counter _hours;

        //creates and initializes the Counter objects  to 0 and setting the name for the counter.
        public Clock()
        {
            _seconds = new Counter("seconds");
            _minutes = new Counter("minutes");
            _hours = new Counter("hours");
        }


        //Will incremenet each counter by 1 and reset once > 59 for minutes and seconds and > 23 for hours.
        public void Tick()
        {
            _seconds.Increment();
            if (_seconds.Ticks > 59)
            {
                _minutes.Increment();
                _seconds.Reset();

                if (_minutes.Ticks > 59)
                {
                    _hours.Increment();
                    _minutes.Reset();

                    if (_hours.Ticks > 23)
                    {
                        Reset();
                    }
                }
            }
        }

        //will Reset the Clock to "00:00:00"
        public void Reset()
        {
            _seconds.Reset();
            _minutes.Reset();
            _hours.Reset();
        }

        /// get propery to be read the time in "hh:mm:ss" in string format
```

```
52          public string Time
53          {
54              get { return
    ↪  $"{_hours.Ticks:D2}:{_minutes.Ticks:D2}:{_seconds.Ticks:D2}"; }
55
56          }
57      }
58  }
59
```

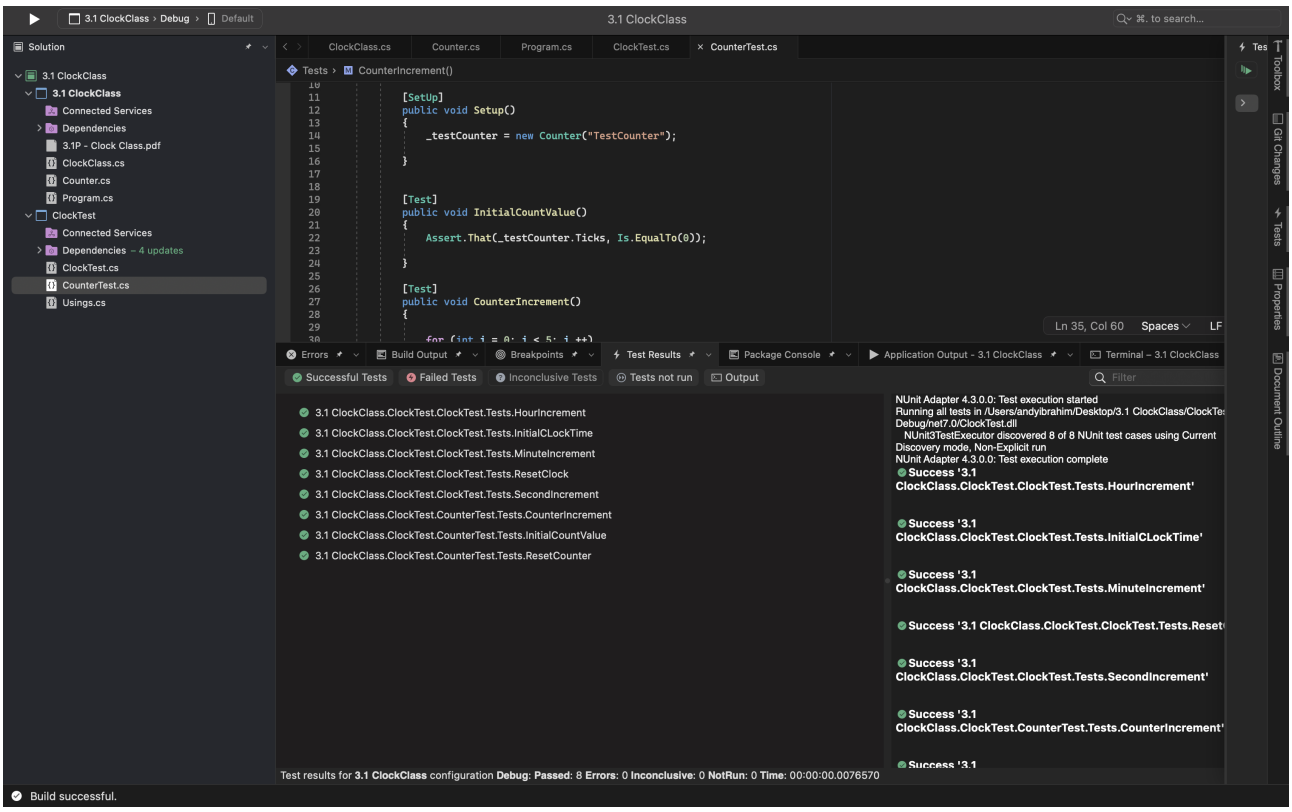
```
using NUnit.Framework;
using clock;

namespace ClockTest
{
    public class Tests
    {
        Clock _clock;

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

        [Test]
        public void InitialCLockTime()
        {
            Assert.That(_clock.Time, Is.EqualTo("00:00:00"));
        }


        [Test]
        public void ResetClock()
        {
            for (int i = 0; i < 86400; i++)
            {
                _clock.Tick();
            }

            _clock.Reset();

            Assert.That(_clock.Time, Is.EqualTo("00:00:00"));

        }

        [Test]
        public void SecondIncrement()
        {
            for (int i = 0; i < 59; i++)
            {
                _clock.Tick();

            }
            Assert.That(_clock.Time, Is.EqualTo("00:00:59"));

        }

        [Test]
        public void MinuteIncrement()
        {
            for (int i = 0; i < 60; i++)
            {
```

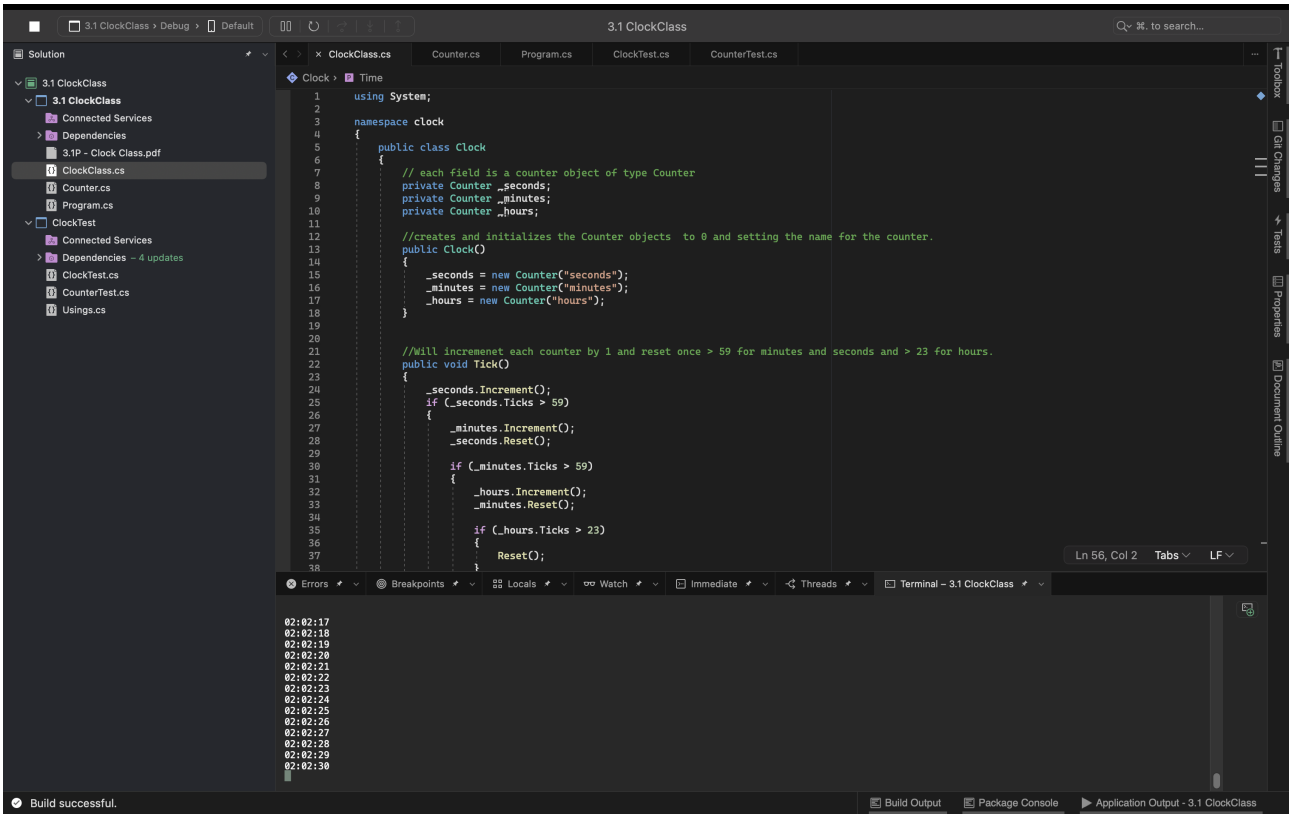
```
54                  _clock.Tick();
55              }
56
57              Assert.That(_clock.Time, Is.EqualTo("00:01:00"));
58
59          }
60
61          [Test]
62          public void HourIncrement()
63          {
64              for (int i = 0; i < 3600; i++)
65              {
66                  _clock.Tick();
67              }
68              Assert.That(_clock.Time, Is.EqualTo("01:00:00"));
69
70          }
71
72      }
73
74  }
```

```
using System;

namespace clock
{
    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 _count; }
        }
    }
}

```

```
using NUnit.Framework;
using clock;

namespace CounterTest
{
    public class Tests
    {

        public Counter _testCounter;

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

        }


        [Test]
        public void InitialCountValue()
        {
            Assert.That(_testCounter.Ticks, Is.EqualTo(0));

        }

        [Test]
        public void CounterIncrement()
        {

            for (int i = 0; i < 5; i ++)
            {
                _testCounter.Increment();
            }

            Assert.That(_testCounter.Ticks, Is.EqualTo(5));

        }


        [Test]
        public void ResetCounter()
        {
            for (int i = 0; i < 5; i ++)
            {
                _testCounter.Increment();
            }

            _testCounter.Reset();

            Assert.That(_testCounter.Ticks, Is.EqualTo(0));

        }

```

```
54
55      }
56
57  }
58
```


3.1 ClockClass
Solution
3.1 ClockClass
3.1 ClockClass
Connected Services
Dependencies
3.1P - Clock Class.pdf
ClockClass.cs
Counter.cs
Program.cs
ClockTest
Connected Services
Dependencies – 4 updates
ClockTest.cs
CounterTest.cs
Usings.cs
ClockClass.cs
Counter.cs
Program.cs
ClockTest.cs
CounterTest.cs
Tests › CounterIncrement()
[SetUp]
public void Setup()
{
_testCounter = new Counter("TestCounter");
}
[Test]
public void InitialCountValue()
{
Assert.That(_testCounter.Ticks, Is.EqualTo(0));
}
[Test]
public void CounterIncrement()
{
Ln 35, Col 60
Spaces
Errors
Build Output
Breakpoints
Test Results
Package Console
Application Output - 3.1 ClockClass
Terminal – 3.1 ClockClass
Successful Tests
Failed Tests
Inconclusive Tests
Tests not run
Output
Filter
3.1 ClockClass.ClockTest.ClockTest.Tests.HourIncrement
3.1 ClockClass.ClockTest.ClockTest.Tests.InitialCLockTime
3.1 ClockClass.ClockTest.ClockTest.Tests.MinuteIncrement
3.1 ClockClass.ClockTest.ClockTest.Tests.ResetClock
3.1 ClockClass.ClockTest.ClockTest.Tests.SecondIncrement
3.1 ClockClass.ClockTest.CounterTest.Tests.CounterIncrement
3.1 ClockClass.ClockTest.CounterTest.Tests.InitialCountValue
3.1 ClockClass.ClockTest.CounterTest.Tests.ResetCounter
NUnit Adapter 4.3.0.0: Test execution started
Running all tests in /Users/andyibrahim/Desktop/3.1 ClockClass/ClockTe
Debug/net7.0/ClockTest.dll
NUnit3TestExecutor discovered 8 of 8 NUnit test cases using Current
Discovery mode, Non-Explicit run
NUnit Adapter 4.3.0.0: Test execution complete
Success '3.1 ClockClass.ClockTest.ClockTest.Tests.HourIncrement'
Success '3.1 ClockClass.ClockTest.ClockTest.Tests.InitialCLockTime'
Success '3.1 ClockClass.ClockTest.ClockTest.Tests.MinuteIncrement'
Success '3.1 ClockClass.ClockTest.ClockTest.Tests.Reset
Success '3.1 ClockClass.ClockTest.ClockTest.Tests.SecondIncrement'
Success '3.1 ClockClass.ClockTest.CounterTest.Tests.CounterIncrement'
Test results for 3.1 ClockClass configuration Debug: Passed: 8 Errors: 0 Inconclusive: 0 NotRun: 0 Time: 00:00:00.0076570
Build successful.


3.1 ClockClass
Solution
3.1 ClockClass
Connected Services
Dependencies
3.1P - Clock Class.pdf
ClockClass.cs
Counter.cs
Program.cs
ClockTest
Connected Services
Dependencies – 4 updates
ClockTest.cs
CounterTest.cs
Usings.cs
ClockClass.cs
Counter.cs
Program.cs
ClockTest.cs
CounterTest.cs
Clock › Time
using System;
namespace clock
{
public class Clock
{
// each field is a counter object of type Counter
private Counter _seconds;
private Counter _minutes;
private Counter _hours;
//creates and initializes the Counter objects to 0 and setting the name for the counter.
public Clock()
{
_seconds = new Counter("seconds");
_minutes = new Counter("minutes");
_hours = new Counter("hours");
}
//Will incrememet each counter by 1 and reset once > 59 for minutes and seconds and > 23 for hours.
public void Tick()
{
_seconds.Increment();
if (_seconds.Ticks > 59)
{
_minutes.Increment();
_seconds.Reset();
if (_minutes.Ticks > 59)
{
_hours.Increment();
_minutes.Reset();
if (_hours.Ticks > 23)
{
Reset();
Ln 56, Col 2
Tabs
LF
Errors
Breakpoints
Locals
Watch
Immediate
Threads
Terminal – 3.1 ClockClass
02:02:17
02:02:18
02:02:19
02:02:20
02:02:21
02:02:22
02:02:23
02:02:24
02:02:25
02:02:26
02:02:27
02:02:28
02:02:29
02:02:30
Build successful.
Build Output
Package Console
Application Output - 3.1 ClockClass
Toolbox
Git Changes
Tests
Properties
Document Outline