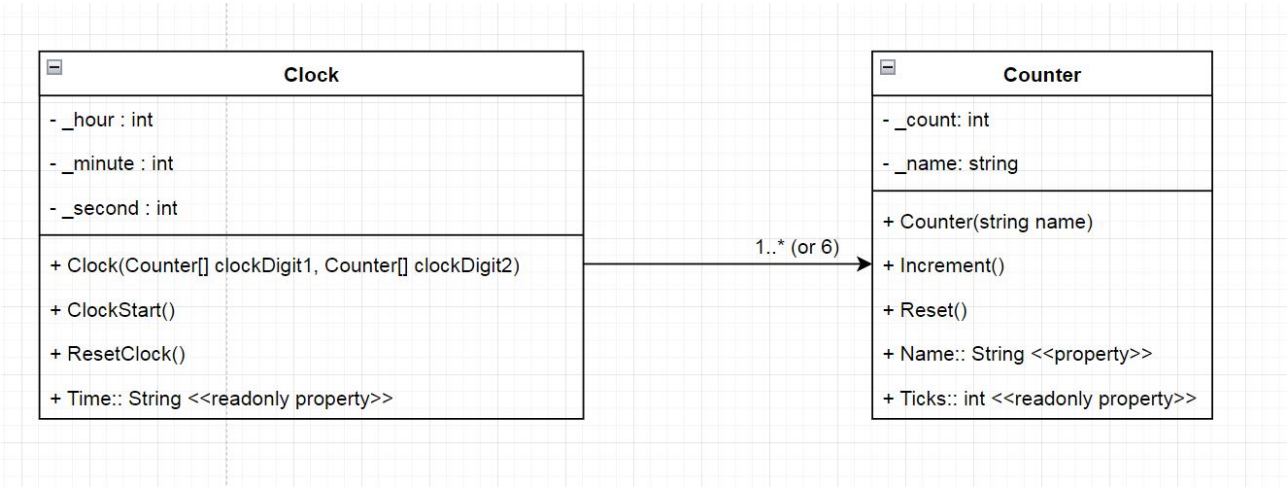
Swinburne University of Technology

COS20007 Object Oriented Programming

Clock Class

PDF generated at 20:38 on Thursday 9th November, 2023


Clock
- _hour : int
- _minute : int
- _second : int
+ Clock(Counter[] clockDigit1, Counter[] clockDigit2)
+ ClockStart()
+ ResetClock()
+ Time:: String <<readonly property>>
1..* (or 6)
Counter
- _count: int
- _name: string
+ Counter(string name)
+ Increment()
+ Reset()
+ Name:: String <<property>>
+ Ticks:: int <<readonly property>>

```
// See https://aka.ms/new-console-template for more information
using ClockTask;
using System.Diagnostics.Metrics;

Counter[] clockD1 = new Counter[3];
clockD1[0] = new Counter("Digit1 Hour ");
clockD1[1] = new Counter("Digit1 Minute ");
clockD1[2] = new Counter("Digit1 Second");

Counter[] clockD2 = new Counter[3];
clockD2[0] = new Counter("Digit2 Hour");
clockD2[1] = new Counter("Digit2 Minute");
clockD2[2] = new Counter("Digit2 Second");

Clock clock = new Clock(clockD1, clockD2);

while (true)
{
    Console.WriteLine($"Current Time is: {clock.Time}");
    Console.WriteLine("Enter Command: tick, reset or quit");
    var command = Console.ReadLine();
    if (command == "tick")
    {
        clock.ClockStart();
    }

    if (command == "reset")
    {
        clock.ClockReset();
    }

    if (command == "quit")
    {
        break;
    }
}
```

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

namespace ClockTask
{
    public class Clock
    {
        private int _hourD1;
        private int _minuteD1;
        private int _secondD1;
        private int _hourD2;
        private int _minuteD2;
        private int _secondD2;
        private Counter[] digit1;
        private Counter[] digit2;

/*        public Clock(int h, int m, int s)
        {
            _hour = h;
            _minute = m;
            _second = s;
        }*/

        public Clock(Counter[] counts1, Counter[] counts2)
        {
            /*            _hour = hour.Ticks;
                        _minute = minute.Ticks;
                        _second = second.Ticks;
                        counters = new Counter[] {hour, minute, second};*/
            //HH:MM:SS FORMAT

            //First Digit
            _hourD1 = counts1[0].Ticks;
            _minuteD1 = counts1[1].Ticks;
            _secondD1 = counts1[2].Ticks;
            digit1 = counts1;

            //Second Digit
            _hourD2 = counts2[0].Ticks;
            _minuteD2 = counts2[1].Ticks;
            _secondD2 = counts2[2].Ticks;
            digit2 = counts2;

        }

        public void ClockStart()
        {
            //Clock begin at 00:00:00
            //while loop in the main program,
```

```

            //Seconds Digit2 increment
            digit2[2].Increment();
            _secondD2 = digit2[2].Ticks;

            //Seconds Digit1 increment
            if (_secondD2 > 9)
            {
                digit2[2].Reset();
                _secondD2 = digit2[2].Ticks;
                digit1[2].Increment();
                _secondD1 = digit1[2].Ticks;
            }

            //Minutes Digit2 increment
            if (_secondD1 == 6)
            {
                digit1[2].Reset();
                _secondD1 = digit1[2].Ticks;
                digit2[1].Increment();
                _minuteD2 = digit2[1].Ticks;
            }

            //Minutes Digit1 increment
            if (_minuteD2 > 9)
            {
                digit2[1].Reset();
                _minuteD2 = digit2[1].Ticks;
                digit1[1].Increment();
                _minuteD1 = digit1[1].Ticks;
            }

            //Hours Digit2 increment
            if (_minuteD1 == 6)
            {
                digit1[1].Reset();
                _minuteD1 = digit1[1].Ticks;
                digit2[0].Increment();
                _hourD2 = digit2[0].Ticks;
            }

            //Hours Digit1 increment
            if (_hourD2 > 9)
            {
                digit2[0].Reset();
                _hourD2 = digit2[0].Ticks;
                digit1[0].Increment();
                _hourD1 = digit1[0].Ticks;
            }

            //Clock reset
            if (_hourD1 == 2 && _hourD2 == 4)
            {
```

```
                ClockReset();
            }
        }

        public string Time
        {
            get { return $"{_hourD1}{_hourD2}:{_minuteD1}{_minuteD2}:{_secondD1}{_secondD2}"; }
        }

        public void ClockReset()
        {
            for (int i = 0; i < 3; i++)
            {
                digit1[i].Reset();
                _hourD1 = digit1[0].Ticks;
                _minuteD1 = digit1[1].Ticks;
                _secondD1 = digit1[2].Ticks;

                digit2[i].Reset();
                _hourD2 = digit2[0].Ticks;
                _minuteD2 = digit2[1].Ticks;
                _secondD2 = digit2[2].Ticks;
            }

        }
    }
}
```

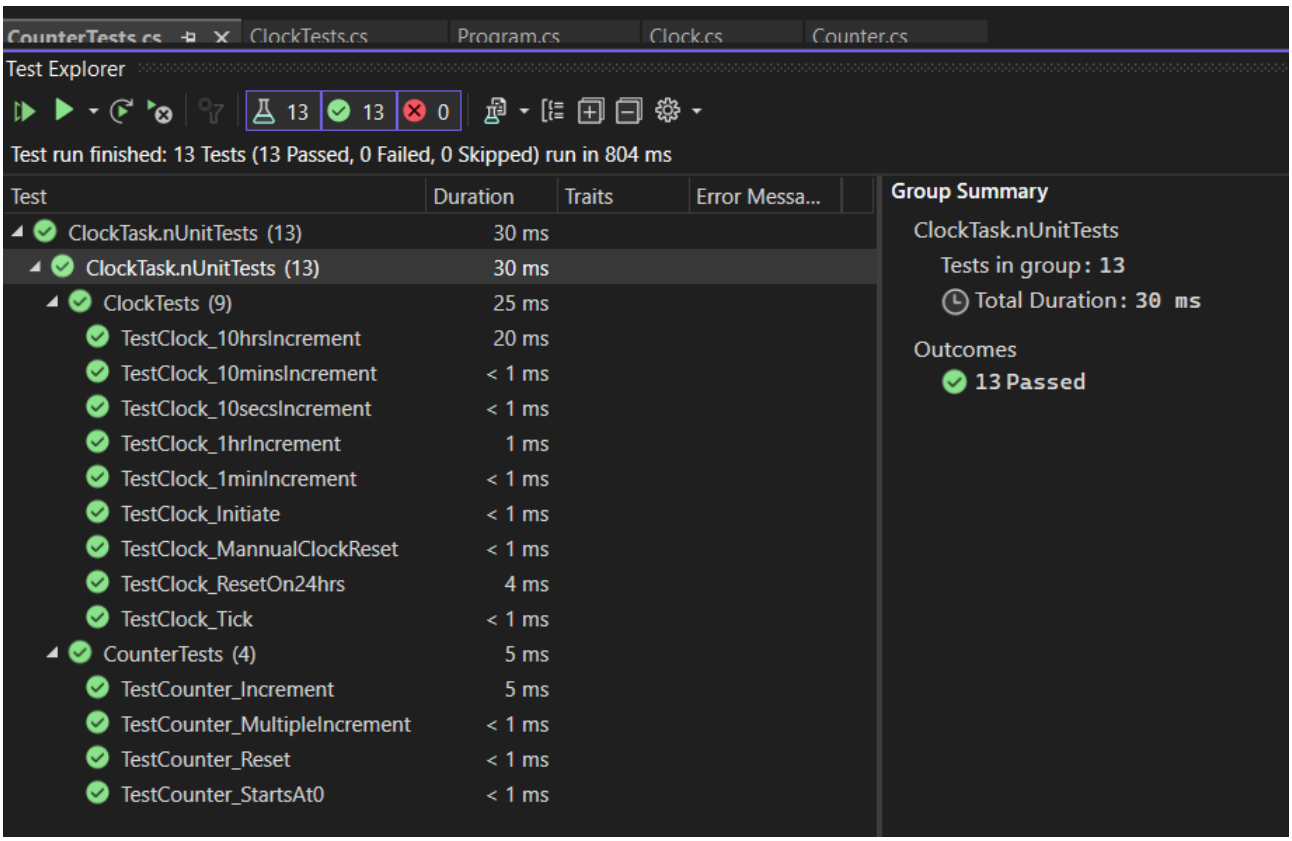
```
namespace ClockTask.nUnitTests
{
    public class ClockTests
    {
        private Clock clockOBJ { get; set; } = null!;

        /*    i represents the number of seconds in real life to test that
            digits are incrementing correctly */
        int i;

        [SetUp]
        public void Setup()
        {
            Counter hourDigit1 = new Counter("hourD1");
            Counter minuteDigit1 = new Counter("minuteD1");
            Counter secondDigit1 = new Counter("secondD1");
            Counter[] digit1 = new Counter[] { hourDigit1, minuteDigit1, secondDigit1 };

            Counter hourDigit2 = new Counter("hourD1");
            Counter minuteDigit2 = new Counter("minuteD1");
            Counter secondDigit2 = new Counter("secondD1");
            Counter[] digit2 = new Counter[] { hourDigit2, minuteDigit2, secondDigit2 };

            clockOBJ = new Clock(digit1, digit2);
        }


        //-------------------Clock Class Test-------------------//
        [Test]
        public void TestClock_Initiate()
        {
            var initialisedTime = clockOBJ.Time;
            Assert.That(initialisedTime, Is.EqualTo("00:00:00"));
            Console.WriteLine("Expected Result:00:00:00");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_Tick()
        {
            clockOBJ.ClockStart();
            var clockTicked = clockOBJ.Time;
            Assert.That(clockTicked, Is.EqualTo("00:00:01"));
            Console.WriteLine("Expected Result:00:00:01");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_10secsIncrement()
        {
            for (i = 0; i < 10; i++)
```

```
            {
                clockOBJ.ClockStart();
            }
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("00:00:10"));
            Console.WriteLine("Expected Result:00:00:10");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_1minIncrement()
        {
            for (i = 0; i < 60; i++)
            {
                clockOBJ.ClockStart();
            }
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("00:01:00"));
            Console.WriteLine("Expected Result:00:01:00");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_10minsIncrement()
        {
            for (i = 0; i < 600; i++)
            {
                clockOBJ.ClockStart();
            }
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("00:10:00"));
            Console.WriteLine("Expected Result:00:10:00");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_1hrIncrement()
        {
            for (i = 0; i < 3600; i++)
            {
                clockOBJ.ClockStart();
            }
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("01:00:00"));
            Console.WriteLine("Expected Result:01:00:00");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_10hrsIncrement()
        {
            for (i = 0; i < 36000; i++)
            {
```

```
                clockOBJ.ClockStart();
            }
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("10:00:00"));
            Console.WriteLine("Expected Result:10:00:00");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_ResetOn24hrs()
        {
            for (i = 0; i < 86400; i++)
            {
                clockOBJ.ClockStart();
                if (i == 86400)
                {
                    clockOBJ.ClockReset();
                }
            }
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("00:00:00"));
            Console.WriteLine("Expected Result:00:00:00");
            Console.WriteLine($"Given Result:{clockOBJ.Time}");
        }

        [Test]
        public void TestClock_MannualClockReset()
        {
            for (i = 0; i < 500; i++)
            {
                clockOBJ.ClockStart();
            }
            Console.WriteLine($"Before Reset:{clockOBJ.Time}");
            clockOBJ.ClockReset();
            var expected = clockOBJ.Time;
            Assert.That(expected, Is.EqualTo("00:00:00"));
            Console.WriteLine($"After Reset:{clockOBJ.Time}");
        }

    }
}
```

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

namespace ClockTask
{
    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
        {
            set { _name = value; }
            get { return _name; }
        }

        public int Ticks
        {
            get { return _count; }
        }
    }
}
```

```
namespace ClockTask.nUnitTests;

public class CounterTests
{
    private Counter counterOBJ { get; set; } = null!;

    [SetUp]
    public void Setup()
    {
        const string test = "counterTEST";
        counterOBJ = new Counter(test);
    }

    //-------------------Counter Class Test-------------------//
    [Test]
    public void TestCounter_StartsAt0()
    {
        var sut = counterOBJ.Ticks;
        Assert.That(sut, Is.EqualTo(0));
    }

    [Test]
    public void TestCounter_Increment()
    {
        counterOBJ.Increment();
        var sut = counterOBJ.Ticks;
        Assert.That(sut, Is.EqualTo(1));
    }

    [Test]
    public void TestCounter_MultipleIncrement()
    {
        int i = 0;
        while (i < 5)
        {
            counterOBJ.Increment();
            i++;
        }
        var sut = counterOBJ.Ticks;
        Assert.That(sut, Is.EqualTo(i));
    }

    [Test]
    public void TestCounter_Reset()
    {
        counterOBJ.Reset();
        var sut = counterOBJ.Ticks;
        Assert.That(sut, Is.EqualTo(0));
    }
}
```


CounterTests.cs
ClockTests.cs
Program.cs
Clock.cs
Counter.cs
Test Explorer
13
13
0
Test run finished: 13 Tests (13 Passed, 0 Failed, 0 Skipped) run in 804 ms
Test
Duration
Traits
Error Messa...
ClockTask.nUnitTests (13) 30 ms
ClockTask.nUnitTests (13) 30 ms
ClockTests (9) 25 ms
TestClock_10hrsIncrement 20 ms
TestClock_10minsIncrement < 1 ms
TestClock_10secsIncrement < 1 ms
TestClock_1hrIncrement 1 ms
TestClock_1minIncrement < 1 ms
TestClock_Initiate < 1 ms
TestClock_MannualClockReset < 1 ms
TestClock_ResetOn24hrs 4 ms
TestClock_Tick < 1 ms
CounterTests (4) 5 ms
TestCounter_Increment 5 ms
TestCounter_MultipleIncrement < 1 ms
TestCounter_Reset < 1 ms
TestCounter_StartsAt0 < 1 ms
Group Summary
ClockTask.nUnitTests
Tests in group: 13
Total Duration: 30 ms
Outcomes
13 Passed

```
Microsoft Visual Studio Debu

Current Time is: 00:00:00
Enter Command: tick, reset or quit
tick
Current Time is: 00:00:01
Enter Command: tick, reset or quit
tick
Current Time is: 00:00:02
Enter Command: tick, reset or quit
tick
Current Time is: 00:00:03
Enter Command: tick, reset or quit
reset
Current Time is: 00:00:00
Enter Command: tick, reset or quit
quit

C:\Users\anlon\OneDrive\Documents\GitHub\Swi
ess 13308) exited with code 0.
To automatically close the console when debu
le when debugging stops.
Press any key to close this window . . .
```