Clock.cs

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

namespace CounterTask2
{
    public class Clock
    {
        Counter _seconds = new Counter("seconds");
        Counter _minutes = new Counter("minutes");
        Counter _hours = new Counter("hours");

        public Clock()
        {

        }
        public void Tick()
        {
            _seconds.Increment();
            if (_seconds.Tick > 59)
            {
                _minutes.Increment();
                _seconds.Reset();
                if (_minutes.Tick > 59)
                {
                    _hours.Increment();
                    _minutes.Reset();
                    if (_hours.Tick > 23)
                    {
                        Reset();
                    }
                }
            }
        }

        public void Reset()
        {
            _seconds.Reset();
            _minutes.Reset();
            _hours.Reset();
        }

        public string Time()
        {
            return _hours.Tick.ToString("D2") + ":" +
_minutes.Tick.ToString("D2") + ":" + _seconds.Tick.ToString("D2");
        }

    }
}
```

Counter.cs

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

namespace CounterTask2
{
    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 Tick
        {
            get
            {
                return _count;
            }
        }
    }
}
```

Program.cs

```
using System;
using System.Threading;

namespace CounterTask2
{
    public class MainClass
    {
        public static void Main(string[] args)
        {
            Clock clock = new Clock();
            int i;

            for (i = 0; i < 86400; i++)
            {
                Thread.Sleep(200);
                Console.Clear();
                clock.Tick();
                Console.WriteLine(clock.Time());
            }
        }
    }
}
```

TestClock.cs

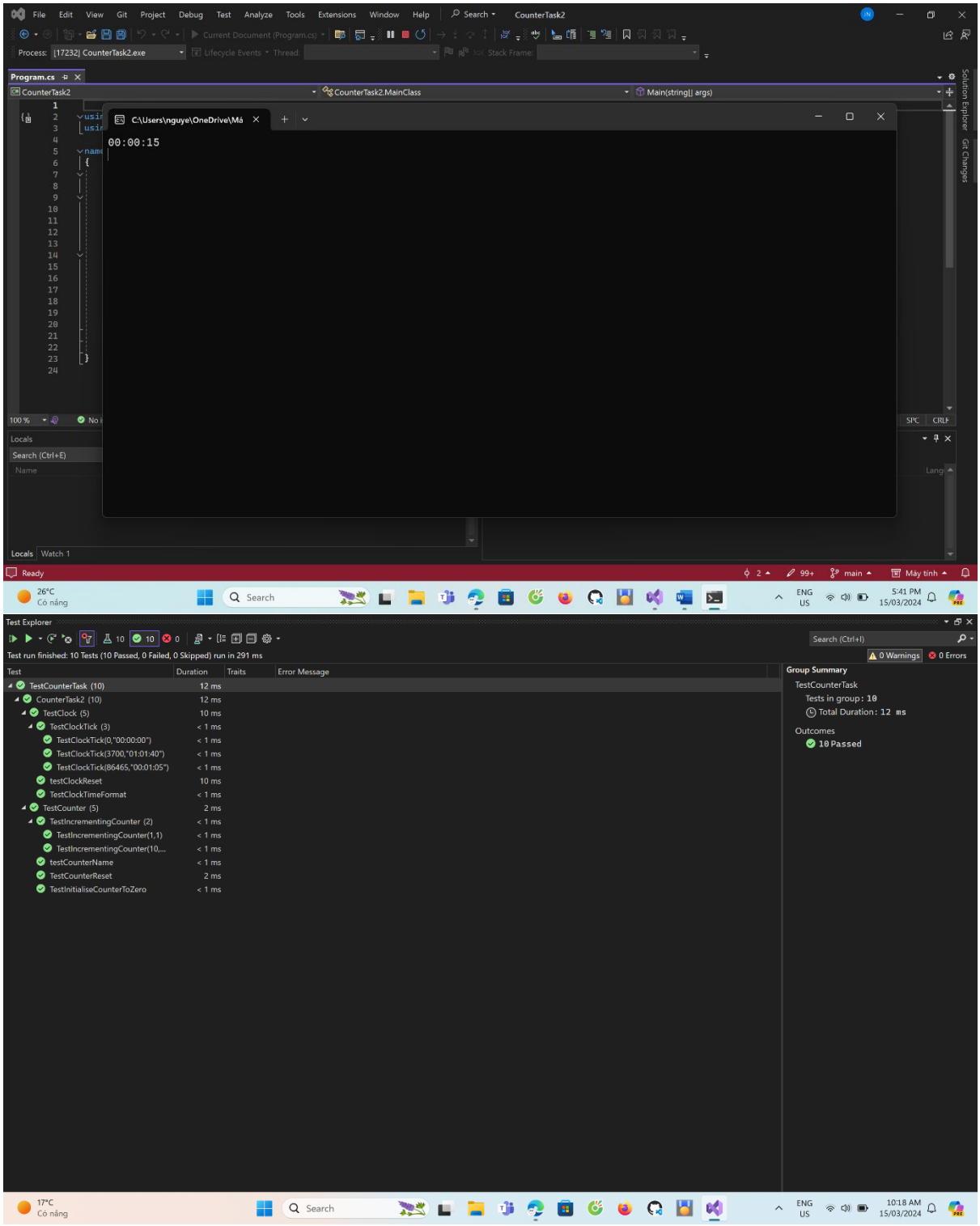
```
using CounterTask2;
using NUnit.Framework;

namespace CounterTask2
{
    public class TestClock
    {
        Clock _clock;
        [SetUp]
        public void Setup()
        {
            _clock = new Clock();
        }

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

        [TestCase(0, "00:00:00")]
        [TestCase(3700, "01:01:40")]
        [TestCase(86465, "00:01:05")]
        public void TestClockTick(int ticks, string expectedResult)
        {
            for (int i = 0; i < ticks; i++)
            {
                _clock.Tick();
            }
            Assert.That(_clock.Time(), Is.EqualTo(expectedResult), "Clock didn't
tick correctly");
        }

        [Test]
        public void testClockReset()
```

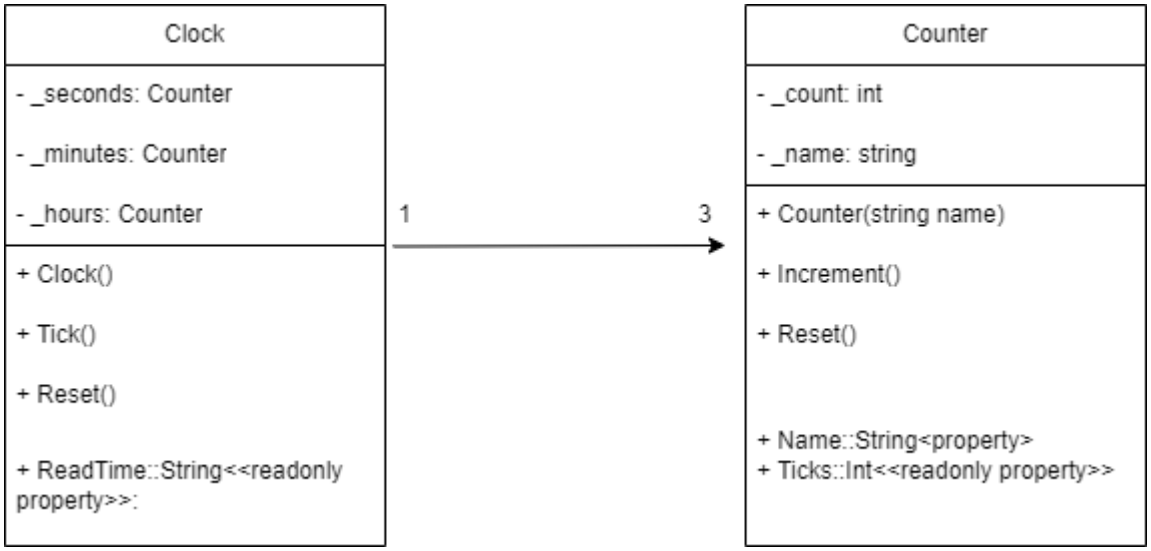
```
        {
            for (int i = 0; i < 3650; i++)
            {
                _clock.Tick();
            }
            _clock.Reset();
            Assert.That(_clock.Time(), Is.EqualTo("00:00:00"), "Clock reset
didn't reset to 0");
        }
    }
}
```

TestCounter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NUnit.Framework;
using CounterTask2;

namespace CounterTask2
{
    public class TestCounter
    {
        Counter _testCounter;
        [SetUp]
        public void Setup()
        {
            _testCounter = new Counter("TEST COUNTER");
        }

        [Test]
        public void TestInitialiseCounterToZero()
        {
            Assert.That(_testCounter.Tick, Is.EqualTo(0));
        }

        [Test]
        public void testCounterName()
        {
            Assert.That(_testCounter.Name, Is.EqualTo("TEST COUNTER"));
        }

        [TestCase(1, 1)]
        [TestCase(10, 10)]
        public void TestIncrementingCounter(int increments, int expectedResult)
        {
            for (int i = 0; i < increments; i++)
            {
                _testCounter.Increment();
            }
            Assert.That(_testCounter.Tick, Is.EqualTo(expectedResult));
        }

        [Test]
        public void TestCounterReset()
        {
            _testCounter.Increment();
            _testCounter.Reset();
```

```
            Assert.That(_testCounter.Tick, Is.EqualTo(0));
        }
    }
}
```




Clock
- _seconds: Counter
- _minutes: Counter
- _hours: Counter
+ Clock()
+ Tick()
+ Reset()
+ ReadTime::String<<readonly property>>:
1
3
Counter
- _count: int
- _name: string
+ Counter(string name)
+ Increment()
+ Reset()
+ Name::String<property>
+ Ticks::Int<<readonly property>>