Clock.cs

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

namespace CounterTask
{
    public class Clock
    {
        private Counter _hours;
        private Counter _minutes;
        private Counter _seconds;
        public Clock()
        {
            _hours = new Counter("");
            _minutes = new Counter("");
            _seconds = new Counter("");
        }

        public int Hours
        {
            get => _hours.Tick;
        }

        public int Minutes
        {
            get => _minutes.Tick;
        }

        public int Seconds
        {
            get => _seconds.Tick;
        }

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

```
                    _minutes.Reset();
                    _hours.Increment();
                }
                else
                {
                    _hours.Reset();
                    _minutes.Reset();
                    _seconds.Reset();
                }
            }
            public void Reset()
            {
                _hours.Reset();
                _minutes.Reset();
                _seconds.Reset();
            }
            public string PrintTime()
            {
                string currentTime = _hours.Tick.ToString("D2") + ":" + 
                  _minutes.Tick.ToString("D2") + ":" + _seconds.Tick.ToString
                  ("D2");
                Console.WriteLine(currentTime);

                return currentTime;
            }
            public void StartClock(int seconds)
            {
                for (int i = 0; i < seconds; i++)
                {
                    Thread.Sleep(1000);
                    Tick();
                    PrintTime();
                }
            }
        }
    }
```

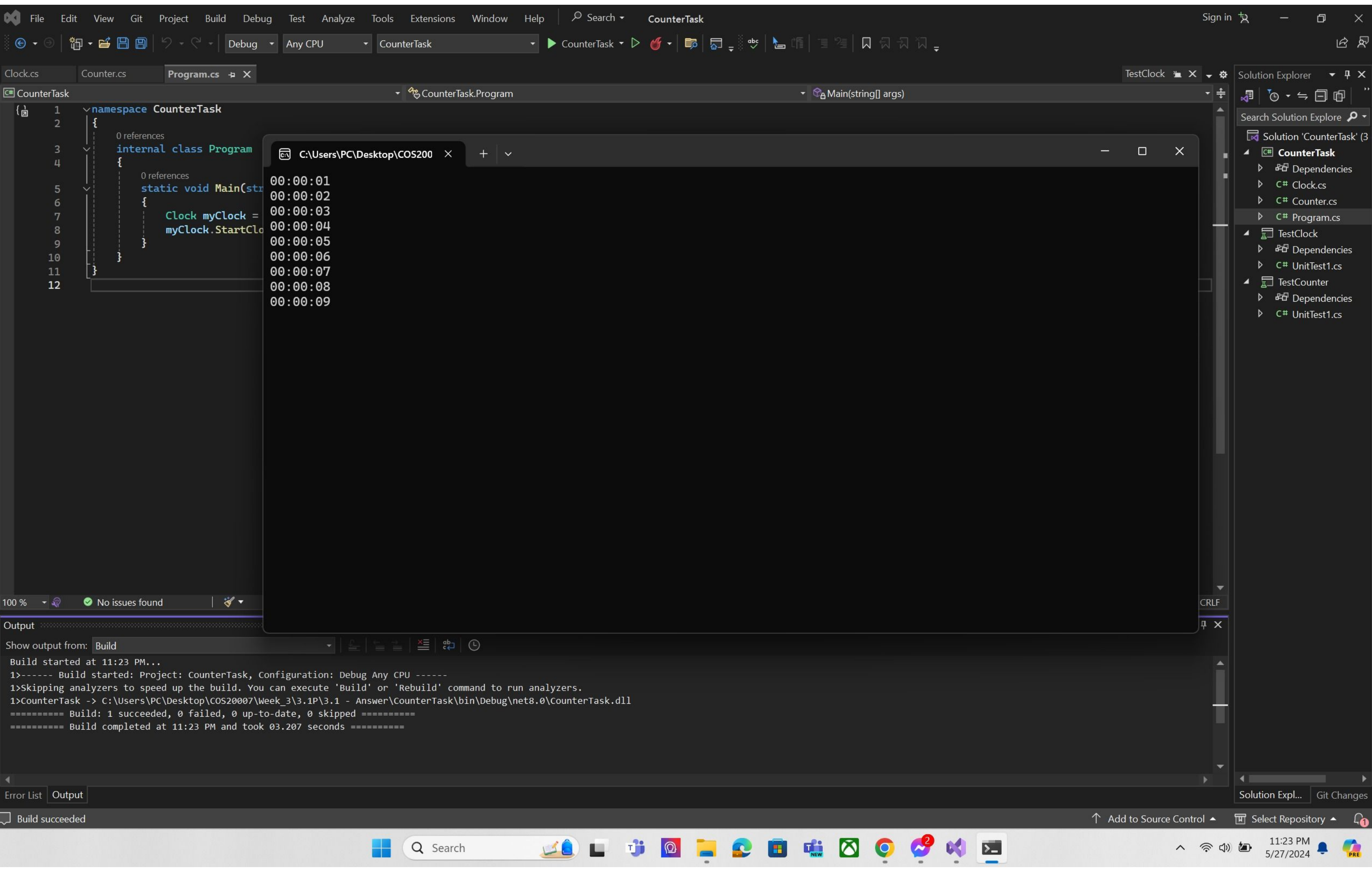
Counter.cs

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

namespace CounterTask
{
    public class Counter
    {
        private string _name;
        public string Name
        {
            get => _name;
            set => _name = value;
        }

        private int _count;
        public int Tick
        {
            get => _count;
        }

        public Counter(string name)
        {
            _name = name;
            _count = 0;
        }
        public int Increment()
        {
            return _count++;
        }
        public int Reset()
        {
            return _count = 0;
        }

    }
}

```

Program.cs

```
namespace CounterTask
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Clock myClock = new Clock();
            myClock.StartClock(56);
        }
    }
}

```

Program Output



```
namespace CounterTask
{
    0 references
    internal class Program
    {
        0 references
        static void Main(str
        {
            Clock myClock =
            myClock.StartClo
        }
    }
}
```

C:\Users\PC\Desktop\COS200

```
00:00:01
00:00:02
00:00:03
00:00:04
00:00:05
00:00:06
00:00:07
00:00:08
00:00:09
```

Show output from: Build

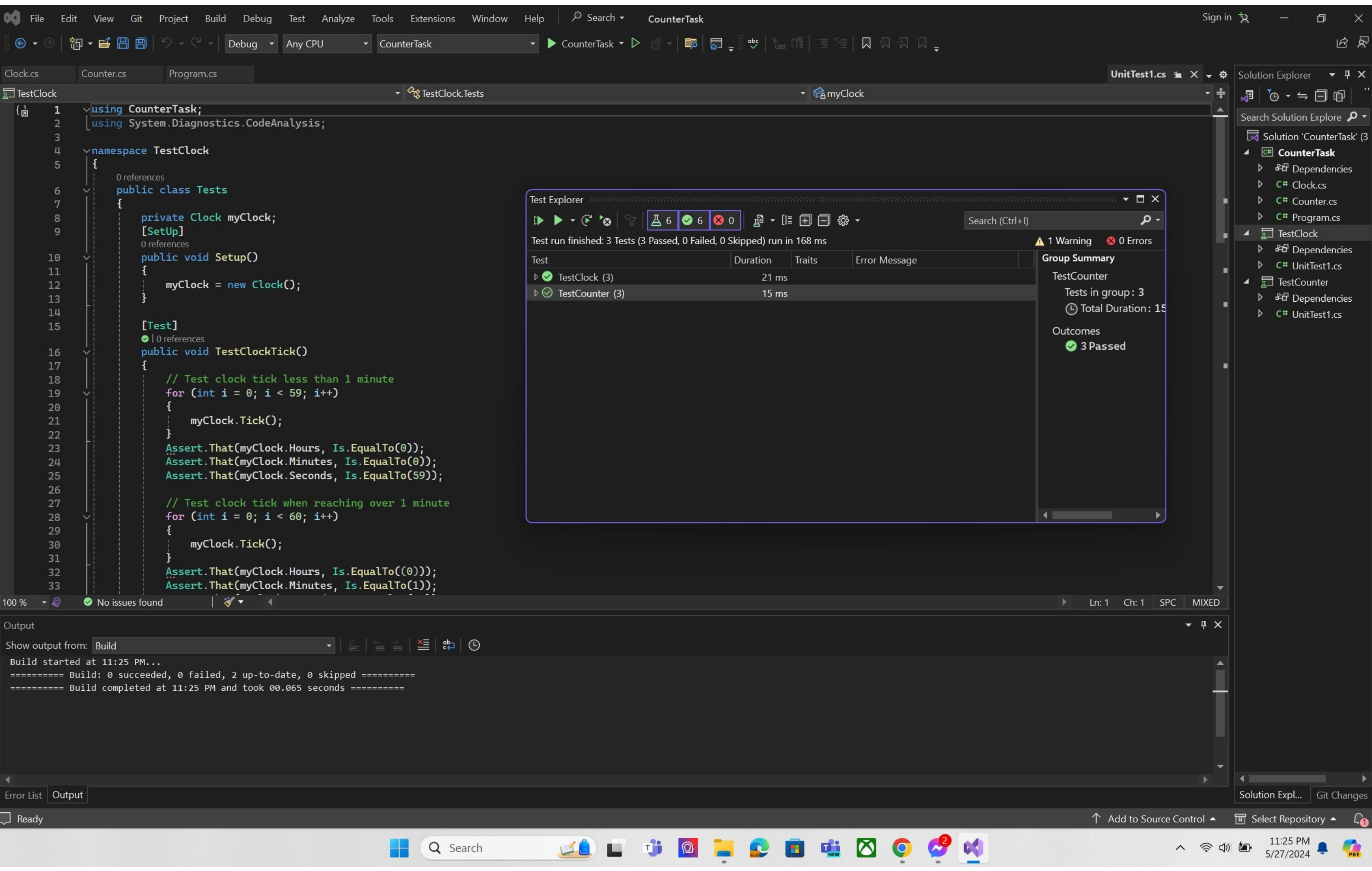
```
Build started at 11:23 PM...
1>------ Build started: Project: CounterTask, Configuration: Debug Any CPU ------
1>Skipping analyzers to speed up the build. You can execute 'Build' or 'Rebuild' command to run analyzers.
1>CounterTask -> C:\Users\PC\Desktop\COS20007\Week_3\3.1P\3.1 - Answer\CounterTask\bin\Debug\net8.0\CounterTask.dll
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 11:23 PM and took 03.207 seconds ==========
```

Build succeeded

UnitTest - Clock

```csharp
using CounterTask;
using System.Diagnostics.CodeAnalysis;

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

        [Test]
        public void TestClockTick()
        {
            // Test clock tick less than 1 minute
            for (int i = 0; i < 59; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Hours, Is.EqualTo(0));
            Assert.That(myClock.Minutes, Is.EqualTo(0));
            Assert.That(myClock.Seconds, Is.EqualTo(59));

            // Test clock tick when reaching over 1 minute
            for (int i = 0; i < 60; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Hours, Is.EqualTo((0)));
            Assert.That(myClock.Minutes, Is.EqualTo(1));
            Assert.That(myClock.Seconds, Is.EqualTo(59));

            // Test clock tick when reaching over 1 hour
            for (int i = 0; i < 3600; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Hours, Is.EqualTo(1));
            Assert.That(myClock.Minutes, Is.EqualTo(1));
            Assert.That(myClock.Seconds, Is.EqualTo(59));

            // Test clock tick when reaching over 24 hours
            for (int i = 0; i < 3600*24; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Hours, Is.EqualTo(1));
            Assert.That(myClock.Minutes, Is.EqualTo(1));
            Assert.That(myClock.Seconds, Is.EqualTo(59));

            Assert.Pass();
        }

        [Test]
        public void TestReset()
        {
            for (int i = 0; i < 3676; i++)
            {
                myClock.Tick();
            }

            myClock.Reset();
            Assert.That(myClock.Hours, Is.EqualTo(0));
            Assert.That(myClock.Minutes, Is.EqualTo(0));
            Assert.That(myClock.Seconds, Is.EqualTo(0));

            Assert.Pass();
        }

        [Test]
        public void TestTimeFormat()
        {
            for (int i = 0; i < 3676; i++)
            {
                myClock.Tick();
            }
            string a = myClock.PrintTime();
            Assert.That(a, Is.EqualTo("01:01:16"));

            Assert.Pass();
        }
    }
}
```

UnitTest Clock Output



```csharp
using CounterTask;
using System.Diagnostics.CodeAnalysis;

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

        [Test]
        public void TestClockTick()
        {
            // Test clock tick less than 1 minute
            for (int i = 0; i < 59; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Hours, Is.EqualTo(0));
            Assert.That(myClock.Minutes, Is.EqualTo(0));
            Assert.That(myClock.Seconds, Is.EqualTo(59));

            // Test clock tick when reaching over 1 minute
            for (int i = 0; i < 60; i++)
            {
                myClock.Tick();
            }
            Assert.That(myClock.Hours, Is.EqualTo((0)));
            Assert.That(myClock.Minutes, Is.EqualTo(1));
```

Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 168 ms

1 Warning 0 Errors

Test	Duration	Traits	Error Message
✓ TestClock (3)	21 ms		
✓ TestCounter (3)	15 ms		

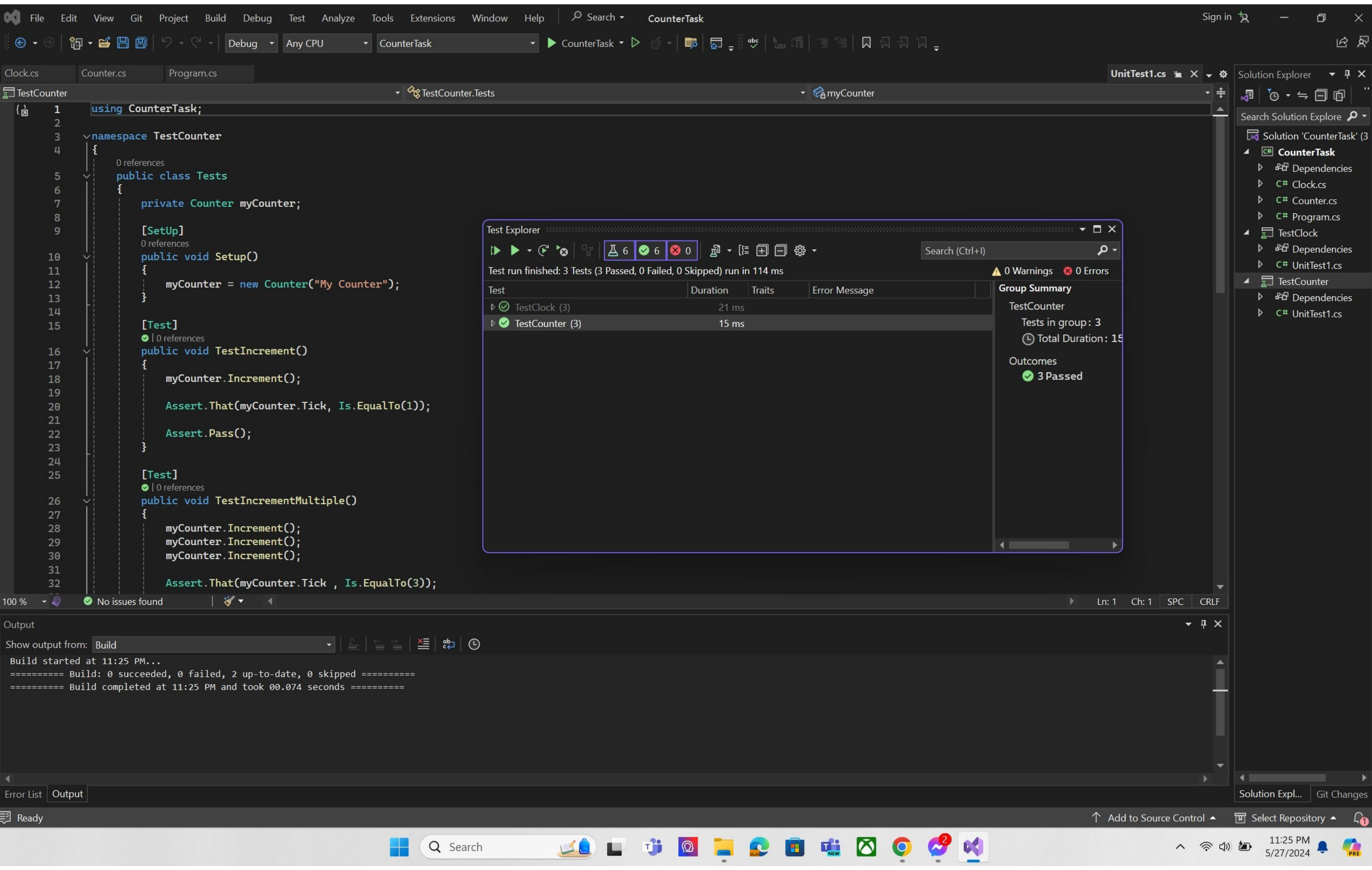
Group Summary
TestCounter
Tests in group: 3
Total Duration: 15
Outcomes
✓ 3 Passed

```
Build started at 11:25 PM...
========== Build: 0 succeeded, 0 failed, 2 up-to-date, 0 skipped ==========
========== Build completed at 11:25 PM and took 00.065 seconds ==========
```

UnitTest Counter

```
using CounterTask;

namespace TestCounter
{
    public class Tests
    {
        private Counter myCounter;

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

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

            Assert.That(myCounter.Tick, Is.EqualTo(1));

            Assert.Pass();
        }

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

            Assert.That(myCounter.Tick , Is.EqualTo(3));

            Assert.Pass();
        }

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

            myCounter.Reset();

            Assert.That(myCounter.Tick, Is.EqualTo(0));

            Assert.Pass();
        }
    }
}
```

UnitTest Counter Output



```
using CounterTask;

namespace TestCounter
{
    0 references
    public class Tests
    {
        private Counter myCounter;

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

        [Test]
        | 0 references
        public void TestIncrement()
        {
            myCounter.Increment();

            Assert.That(myCounter.Tick, Is.EqualTo(1));

            Assert.Pass();
        }

        [Test]
        | 0 references
        public void TestIncrementMultiple()
        {
            myCounter.Increment();
            myCounter.Increment();
            myCounter.Increment();

            Assert.That(myCounter.Tick , Is.EqualTo(3));
```

Test Explorer

Test run finished: 3 Tests (3 Passed, 0 Failed, 0 Skipped) run in 114 ms

0 Warnings 0 Errors

Test	Duration	Traits	Error Message
TestClock (3)	21 ms		
TestCounter (3)	15 ms		

Group Summary
TestCounter
Tests in group: 3
Total Duration: 15

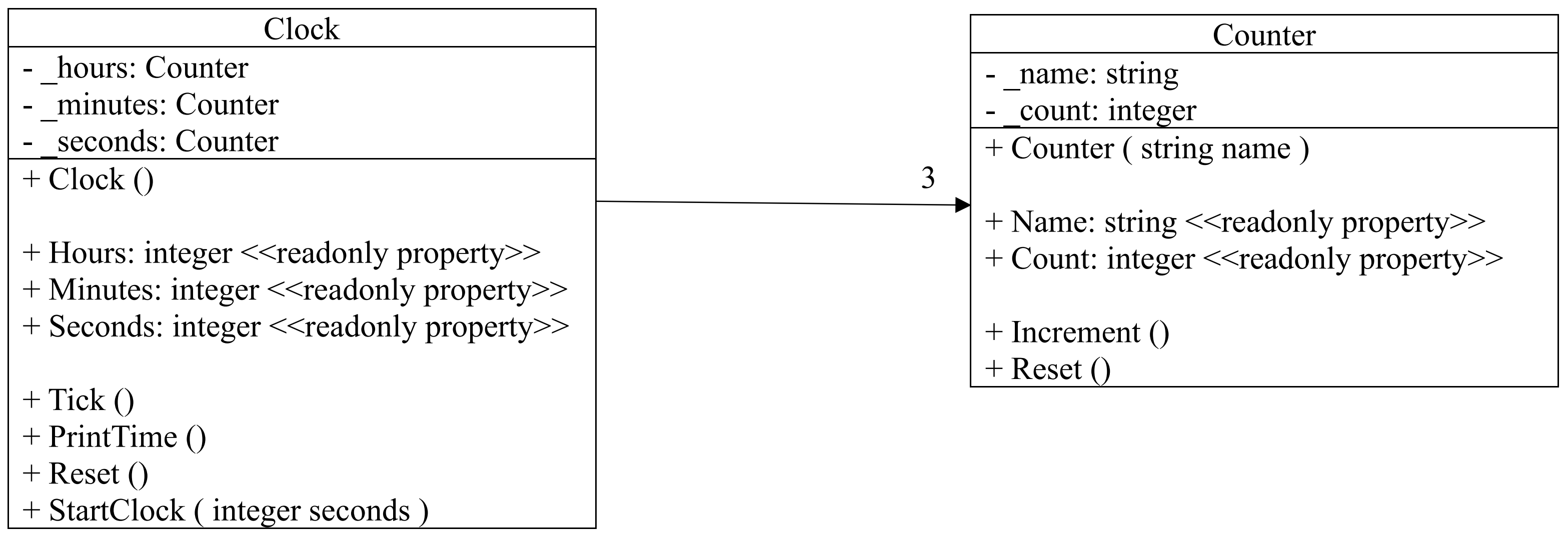
Outcomes
3 Passed

Output

Show output from: Build

```
Build started at 11:25 PM...
========== Build: 0 succeeded, 0 failed, 2 up-to-date, 0 skipped ==========
========== Build completed at 11:25 PM and took 00.074 seconds ==========
```

UML Diagram


Clock
- _hours: Counter
- _minutes: Counter
- _seconds: Counter
+ Clock ()
+ Hours: integer <<readonly property>>
+ Minutes: integer <<readonly property>>
+ Seconds: integer <<readonly property>>
+ Tick ()
+ PrintTime ()
+ Reset ()
+ StartClock (integer seconds)
3
Counter
- _name: string
- _count: integer
+ Counter (string name)
+ Name: string <<readonly property>>
+ Count: integer <<readonly property>>
+ Increment ()
+ Reset ()