



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace ClockApplication
8 {
9     public class Counter
10    {
11        private int _count;
12
13        public Counter()
14        {
15            _count = 0;
16        }
17
18        public void Increment()
19        {
20            _count++;
21        }
22
23        public void Reset()
24        {
25            _count = 0;
26        }
27
28        public int Tick()
29        {
30            return _count;
31        }
32    }
33 }
34
```

```
1 using System;
2 using System.Collections.Generic;
3 using System.Diagnostics.Metrics;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace ClockApplication
9 {
10     public class Clock
11     {
12         Counter _second = new Counter();
13         Counter _minute = new Counter();
14         Counter _hour = new Counter();
15
16         public Clock()
17         {
18             _second.Reset();
19             _minute.Reset();
20             _hour.Reset();
21         }
22
23         public int Seconds
24         {
25             get { return _second.Tick(); }
26         }
27
28         public int Minutes
29         {
30             get { return _minute.Tick(); }
31         }
32
33         public int Hours
34         {
35             get { return _hour.Tick(); }
36         }
37
38         public void Tick()
39         {
40             _second.Increment();
41             if (_second.Tick() == 60)
42             {
43                 _second.Reset();
44                 _minute.Increment();
45             }
46             if (_minute.Tick() == 60)
47             {
48                 _minute.Reset();
49                 _hour.Increment();
50             }
51         }
52     }
53 }
```

```
50         }
51         if (_hour.Tick() == 24)
52         {
53             _hour.Reset();
54         }
55     }
56 }
57 }
58
```

```
1 namespace ClockApplication
2 {
3     internal class Program
4     {
5         static void Main(string[] args)
6         {
7             Clock clock = new Clock();
8             while (true)
9             {
10                 Console.Clear();
11                 clock.Tick();
12                 Console.WriteLine($"{clock.Hours}:{clock.Minutes}:
13                                     {clock.Seconds}");
14                 Thread.Sleep(1000);
15             }
16         }
17     }
18 }
```



D:\workspace\COS20007\COS



0:0:4

|

```
1 using ClockApplication;
2
3 namespace ClockTest
4 {
5     public class Tests
6     {
7         private Clock clock;
8
9         [SetUp]
10        public void Setup()
11        {
12            clock = new Clock();
13        }
14
15        [Test]
16        public void ClockInitializes()
17        {
18            Clock ini = new Clock();
19            Assert.IsNotNull(ini);
20            Assert.AreEqual(0, ini.Seconds);
21            Assert.AreEqual(0, ini.Minutes);
22            Assert.AreEqual(0, ini.Hours);
23        }
24
25        [Test]
26        public void TestTickShouldIncrementSecondsByOne()
27        {
28            int initialSeconds = clock.Seconds;
29
30            clock.Tick();
31
32            Assert.AreEqual(initialSeconds + 1, clock.Seconds);
33        }
34
35        [Test]
36        public void TestTickShouldIncrementMinutesByOneWhenSecondsReachSixty()
37        {
38            int initialMinutes = clock.Minutes;
39            for (int i = 0; i <= 60; i++)
40            {
41                clock.Tick();
42            }
43
44            Assert.AreEqual(initialMinutes + 1, clock.Minutes);
45        }
46
47        [Test]
48        public void TestTickShouldIncrementHoursByOneWhenMinutesReachSixty
```

```
    ()  
49     {  
50         int initialHours = clock.Hours;  
51         for (int i = 0; i <= 3600; i++)  
52         {  
53             clock.Tick();  
54         }  
55  
56         Assert.AreEqual(initialHours + 1, clock.Hours);  
57     }  
58 }  
59 }
```



```
1 using ClockApplication;
2
3 namespace ClockTest
4 {
5     public class CounterTest
6     {
7         Counter _countertest;
8
9         [SetUp]
10        public void Setup()
11        {
12            _countertest = new Counter();
13        }
14
15        [Test]
16        public void TestInitializes()
17        {
18            Counter ini = new Counter();
19            Assert.IsNotNull(ini);
20            Assert.That(ini.Tick, Is.EqualTo(0));
21        }
22
23        [Test]
24        public void TestStart()
25        {
26            Assert.That(_countertest.Tick, Is.EqualTo(0));
27        }
28
29        [Test]
30        public void TestCountReset()
31        {
32            _countertest.Increment();
33            _countertest.Reset();
34            Assert.That(_countertest.Tick, Is.EqualTo(0));
35        }
36
37        [TestCase(60, 60)]
38        [TestCase(100, 100)]
39        public void TestIncrement(int tick, int result)
40        {
41            for (int i = 0; i < tick; i++)
42            {
43                _countertest.Increment();
44            }
45            Assert.That(_countertest.Tick, Is.EqualTo(result));
46        }
47    }
48 }
49
```

test	Duration	Trails
▲ ✓ ClockTest (9)	49 ms	
▲ ✓ ClockTest (9)	49 ms	
▲ ✓ CounterTest (5)	49 ms	
▷ ✓ TestIncrement (2)	< 1 ms	
✓ TestCountReset	48 ms	
✓ TestInitializes	1 ms	
✓ TestStart	< 1 ms	
▲ ✓ Tests (4)	< 1 ms	
✓ ClockInitializes	< 1 ms	
✓ TestTickShouldIncrementHours...	< 1 ms	
✓ TestTickShouldIncrementMinut...	< 1 ms	
✓ TestTickShouldIncrementSecon...	< 1 ms	