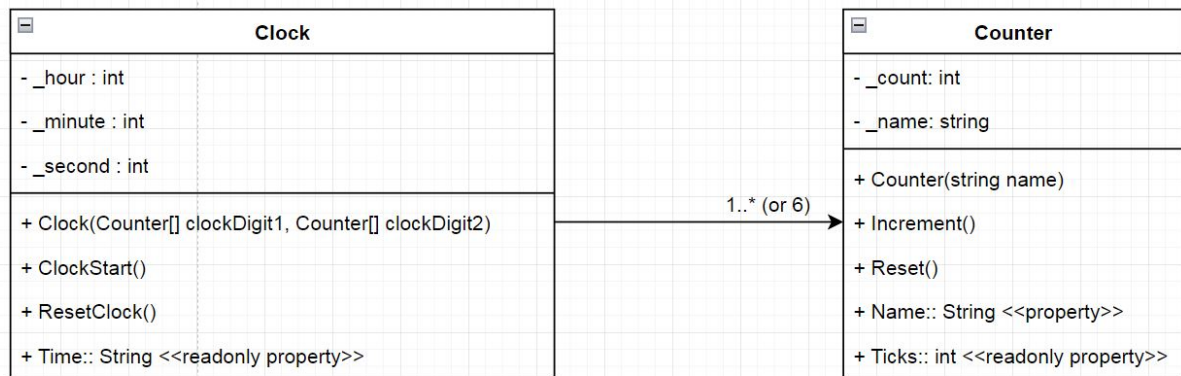


SWINBURNE UNIVERSITY OF TECHNOLOGY

COS20007 OBJECT ORIENTED PROGRAMMING

Clock Class

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



```
1  // See https://aka.ms/new-console-template for more information
2  using ClockTask;
3  using System.Diagnostics.Metrics;
4
5  Counter[] clockD1 = new Counter[3];
6  clockD1[0] = new Counter("Digit1 Hour ");
7  clockD1[1] = new Counter("Digit1 Minute ");
8  clockD1[2] = new Counter("Digit1 Second");
9
10 Counter[] clockD2 = new Counter[3];
11 clockD2[0] = new Counter("Digit2 Hour");
12 clockD2[1] = new Counter("Digit2 Minute");
13 clockD2[2] = new Counter("Digit2 Second");
14
15 Clock clock = new Clock(clockD1, clockD2);
16
17 while (true)
18 {
19     Console.WriteLine($"Current Time is: {clock.Time}");
20     Console.WriteLine("Enter Command: tick, reset or quit");
21     var command = Console.ReadLine();
22     if (command == "tick")
23     {
24         clock.ClockStart();
25     }
26
27     if (command == "reset")
28     {
29         clock.ClockReset();
30     }
31
32     if (command == "quit")
33     {
34         break;
35     }
36 }
```

```
1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
8  namespace ClockTask
9  {
10     public class Clock
11     {
12         private int _hourD1;
13         private int _minuteD1;
14         private int _secondD1;
15         private int _hourD2;
16         private int _minuteD2;
17         private int _secondD2;
18         private Counter[] digit1;
19         private Counter[] digit2;
20
21         /*      public Clock(int h, int m, int s)
22             {
23                 _hour = h;
24                 _minute = m;
25                 _second = s;
26             }*/
27
28         public Clock(Counter[] counts1, Counter[] counts2)
29         {
30             /*      _hour = hour.Ticks;
31                 _minute = minute.Ticks;
32                 _second = second.Ticks;
33                 counters = new Counter[] {hour, minute, second};*/
34             //HH:MM:SS FORMAT
35
36             //First Digit
37             _hourD1 = counts1[0].Ticks;
38             _minuteD1 = counts1[1].Ticks;
39             _secondD1 = counts1[2].Ticks;
40             digit1 = counts1;
41
42             //Second Digit
43             _hourD2 = counts2[0].Ticks;
44             _minuteD2 = counts2[1].Ticks;
45             _secondD2 = counts2[2].Ticks;
46             digit2 = counts2;
47
48         }
49
50         public void ClockStart()
51         {
52             //Clock begin at 00:00:00
53             //while loop in the main program,
```

```
54
55     //Seconds Digit2 increment
56     digit2[2].Increment();
57     _secondD2 = digit2[2].Ticks;
58
59     //Seconds Digit1 increment
60     if (_secondD2 > 9)
61     {
62         digit2[2].Reset();
63         _secondD2 = digit2[2].Ticks;
64         digit1[2].Increment();
65         _secondD1 = digit1[2].Ticks;
66     }
67
68     //Minutes Digit2 increment
69     if (_secondD1 == 6)
70     {
71         digit1[2].Reset();
72         _secondD1 = digit1[2].Ticks;
73         digit2[1].Increment();
74         _minuteD2 = digit2[1].Ticks;
75     }
76
77     //Minutes Digit1 increment
78     if (_minuteD2 > 9)
79     {
80         digit2[1].Reset();
81         _minuteD2 = digit2[1].Ticks;
82         digit1[1].Increment();
83         _minuteD1 = digit1[1].Ticks;
84     }
85
86     //Hours Digit2 increment
87     if (_minuteD1 == 6)
88     {
89         digit1[1].Reset();
90         _minuteD1 = digit1[1].Ticks;
91         digit2[0].Increment();
92         _hourD2 = digit2[0].Ticks;
93     }
94
95     //Hours Digit1 increment
96     if (_hourD2 > 9)
97     {
98         digit2[0].Reset();
99         _hourD2 = digit2[0].Ticks;
100        digit1[0].Increment();
101        _hourD1 = digit1[0].Ticks;
102    }
103
104    //Clock reset
105    if (_hourD1 == 2 && _hourD2 == 4)
106    {
```

```
107         ClockReset();
108     }
109 }
110
111     public string Time
112     {
113         get { return
↪     $"{_hourD1}{_hourD2}:{_minuteD1}{_minuteD2}:{_secondD1}{_secondD2}"; }
114     }
115
116     public void ClockReset()
117     {
118         for (int i = 0; i < 3; i++)
119         {
120             digit1[i].Reset();
121             _hourD1 = digit1[0].Ticks;
122             _minuteD1 = digit1[1].Ticks;
123             _secondD1 = digit1[2].Ticks;
124
125             digit2[i].Reset();
126             _hourD2 = digit2[0].Ticks;
127             _minuteD2 = digit2[1].Ticks;
128             _secondD2 = digit2[2].Ticks;
129         }
130     }
131 }
132 }
133 }
```

```

1 namespace ClockTask.nUnitTests
2 {
3     public class ClockTests
4     {
5         private Clock clockOBJ { get; set; } = null!;
6
7         /* i represents the number of seconds in real life to test that
8         digits are incrementing correctly */
9         int i;
10
11         [SetUp]
12         public void Setup()
13         {
14             Counter hourDigit1 = new Counter("hourD1");
15             Counter minuteDigit1 = new Counter("minuteD1");
16             Counter secondDigit1 = new Counter("secondD1");
17             Counter[] digit1 = new Counter[] { hourDigit1, minuteDigit1, secondDigit1
↵ };
18
19             Counter hourDigit2 = new Counter("hourD1");
20             Counter minuteDigit2 = new Counter("minuteD1");
21             Counter secondDigit2 = new Counter("secondD1");
22             Counter[] digit2 = new Counter[] { hourDigit2, minuteDigit2, secondDigit2
↵ };
23
24             clockOBJ = new Clock(digit1, digit2);
25         }
26
27         //-----Clock Class Test-----//
28         [Test]
29         public void TestClock_Initiate()
30         {
31             var initialisedTime = clockOBJ.Time;
32             Assert.That(initialisedTime, Is.EqualTo("00:00:00"));
33             Console.WriteLine("Expected Result:00:00:00");
34             Console.WriteLine($"Given Result:{clockOBJ.Time}");
35         }
36
37         [Test]
38         public void TestClock_Tick()
39         {
40             clockOBJ.ClockStart();
41             var clockTicked = clockOBJ.Time;
42             Assert.That(clockTicked, Is.EqualTo("00:00:01"));
43             Console.WriteLine("Expected Result:00:00:01");
44             Console.WriteLine($"Given Result:{clockOBJ.Time}");
45         }
46
47         [Test]
48         public void TestClock_10secsIncrement()
49         {
50             for (i = 0; i < 10; i++)
51

```

```
52         {
53             clockOBJ.ClockStart();
54         }
55         var expected = clockOBJ.Time;
56         Assert.That(expected, Is.EqualTo("00:00:10"));
57         Console.WriteLine("Expected Result:00:00:10");
58         Console.WriteLine($"Given Result:{clockOBJ.Time}");
59     }
60
61     [Test]
62     public void TestClock_1minIncrement()
63     {
64         for (i = 0; i < 60; i++)
65         {
66             clockOBJ.ClockStart();
67         }
68         var expected = clockOBJ.Time;
69         Assert.That(expected, Is.EqualTo("00:01:00"));
70         Console.WriteLine("Expected Result:00:01:00");
71         Console.WriteLine($"Given Result:{clockOBJ.Time}");
72     }
73
74     [Test]
75     public void TestClock_10minsIncrement()
76     {
77         for (i = 0; i < 600; i++)
78         {
79             clockOBJ.ClockStart();
80         }
81         var expected = clockOBJ.Time;
82         Assert.That(expected, Is.EqualTo("00:10:00"));
83         Console.WriteLine("Expected Result:00:10:00");
84         Console.WriteLine($"Given Result:{clockOBJ.Time}");
85     }
86
87     [Test]
88     public void TestClock_1hrIncrement()
89     {
90         for (i = 0; i < 3600; i++)
91         {
92             clockOBJ.ClockStart();
93         }
94         var expected = clockOBJ.Time;
95         Assert.That(expected, Is.EqualTo("01:00:00"));
96         Console.WriteLine("Expected Result:01:00:00");
97         Console.WriteLine($"Given Result:{clockOBJ.Time}");
98     }
99
100    [Test]
101    public void TestClock_10hrsIncrement()
102    {
103        for (i = 0; i < 36000; i++)
104        {
```



```
105         clockOBJ.ClockStart();
106     }
107     var expected = clockOBJ.Time;
108     Assert.That(expected, Is.EqualTo("10:00:00"));
109     Console.WriteLine("Expected Result:10:00:00");
110     Console.WriteLine($"Given Result:{clockOBJ.Time}");
111 }
112
113 [Test]
114 public void TestClock_ResetOn24hrs()
115 {
116     for (i = 0; i < 86400; i++)
117     {
118         clockOBJ.ClockStart();
119         if (i == 86400)
120         {
121             clockOBJ.ClockReset();
122         }
123     }
124     var expected = clockOBJ.Time;
125     Assert.That(expected, Is.EqualTo("00:00:00"));
126     Console.WriteLine("Expected Result:00:00:00");
127     Console.WriteLine($"Given Result:{clockOBJ.Time}");
128 }
129
130 [Test]
131 public void TestClock_MannualClockReset()
132 {
133     for (i = 0; i < 500; i++)
134     {
135         clockOBJ.ClockStart();
136     }
137     Console.WriteLine($"Before Reset:{clockOBJ.Time}");
138     clockOBJ.ClockReset();
139     var expected = clockOBJ.Time;
140     Assert.That(expected, Is.EqualTo("00:00:00"));
141     Console.WriteLine($"After Reset:{clockOBJ.Time}");
142 }
143
144 }
145 }
```

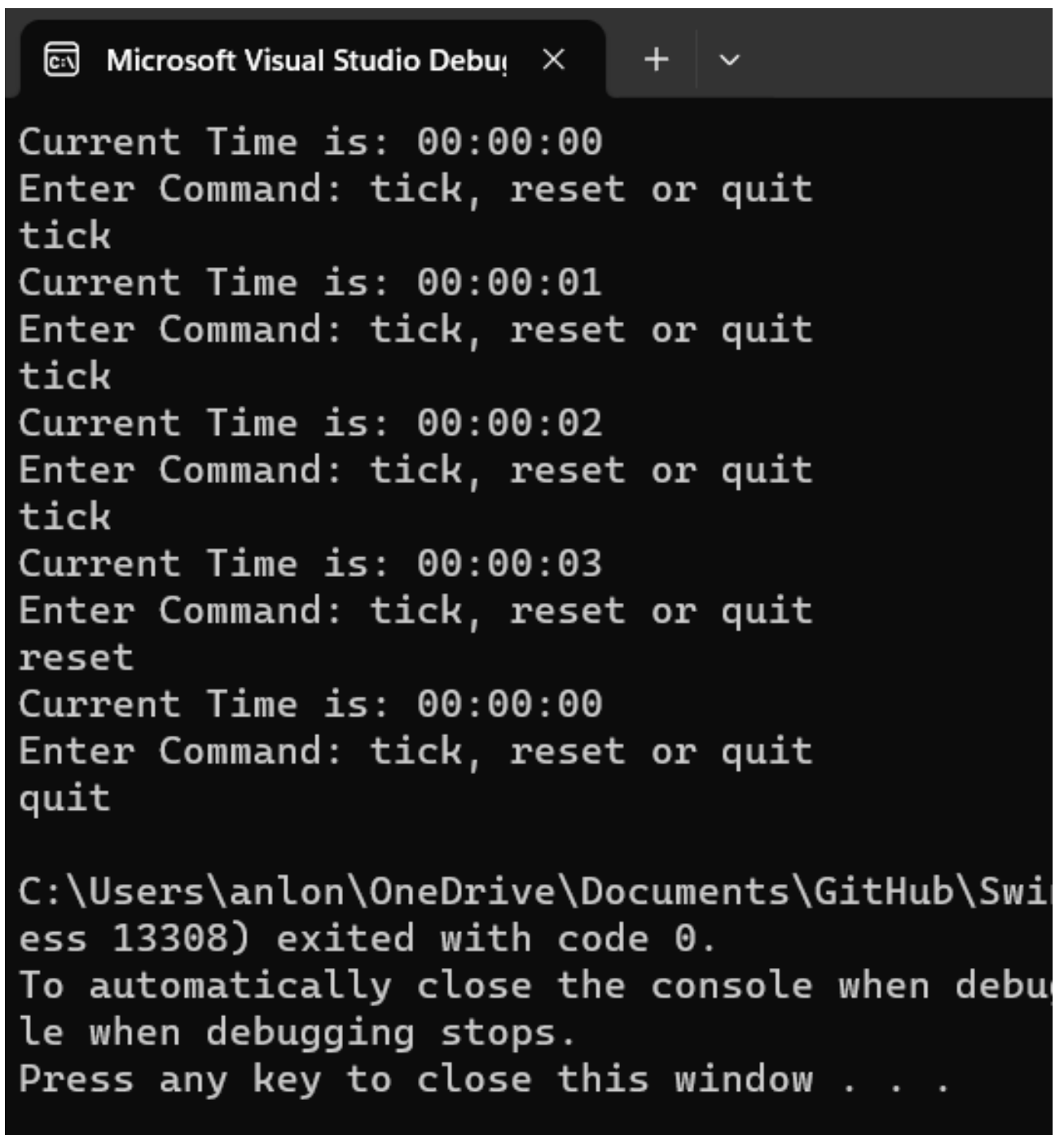
```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ClockTask
8  {
9      public class Counter
10     {
11         private int _count;
12         private string _name;
13
14         public Counter(string name)
15         {
16             _name = name;
17             _count = 0;
18         }
19
20         public void Increment()
21         {
22             _count++;
23         }
24
25         public void Reset()
26         {
27             _count = 0;
28         }
29
30         public string Name
31         {
32             set { _name = value; }
33             get { return _name; }
34         }
35
36         public int Ticks
37         {
38             get { return _count; }
39         }
40     }
41 }
```

```
1 namespace ClockTask.nUnitTests;
2
3 public class CounterTests
4 {
5     private Counter counterOBJ { get; set; } = null!;
6
7     [SetUp]
8     public void Setup()
9     {
10         const string test = "counterTEST";
11         counterOBJ = new Counter(test);
12     }
13
14     //-----Counter Class Test-----//
15     [Test]
16     public void TestCounter_StartsAt0()
17     {
18         var sut = counterOBJ.Ticks;
19         Assert.That(sut, Is.EqualTo(0));
20     }
21
22     [Test]
23     public void TestCounter_Increment()
24     {
25         counterOBJ.Increment();
26         var sut = counterOBJ.Ticks;
27         Assert.That(sut, Is.EqualTo(1));
28     }
29
30     [Test]
31     public void TestCounter_MultipleIncrement()
32     {
33         int i = 0;
34         while (i < 5)
35         {
36             counterOBJ.Increment();
37             i++;
38         }
39         var sut = counterOBJ.Ticks;
40         Assert.That(sut, Is.EqualTo(i));
41     }
42
43     [Test]
44     public void TestCounter_Reset()
45     {
46         counterOBJ.Reset();
47         var sut = counterOBJ.Ticks;
48         Assert.That(sut, Is.EqualTo(0));
49     }
50 }
```

The screenshot shows the Visual Studio Test Explorer interface. At the top, there are tabs for 'CounterTests.cs', 'ClockTests.cs', 'Program.cs', 'Clock.cs', and 'Counter.cs'. Below the tabs, the 'Test Explorer' title bar is visible. A toolbar contains icons for running tests, a search icon, and a filter icon. A status bar shows '13' tests, '13' passed, and '0' failed. Below this, a message states 'Test run finished: 13 Tests (13 Passed, 0 Failed, 0 Skipped) run in 804 ms'. The main area displays a list of tests with columns for 'Test', 'Duration', 'Traits', and 'Error Message'. The tests are grouped into 'ClockTask.nUnitTests (13)' and 'CounterTests (4)'. The 'ClockTask.nUnitTests' group is expanded, showing 9 sub-tests, all of which passed. The 'CounterTests' group is also expanded, showing 4 sub-tests, all of which passed. On the right side, a 'Group Summary' panel for 'ClockTask.nUnitTests' shows 'Tests in group: 13' and 'Total Duration: 30 ms'. Below this, an 'Outcomes' section shows '13 Passed'.

Test	Duration	Traits	Error Message
▲ ✓ 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 Debug Console
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\Swiss 13308) exited with code 0.
To automatically close the console when debugging stops.
Press any key to close this window . . .
```