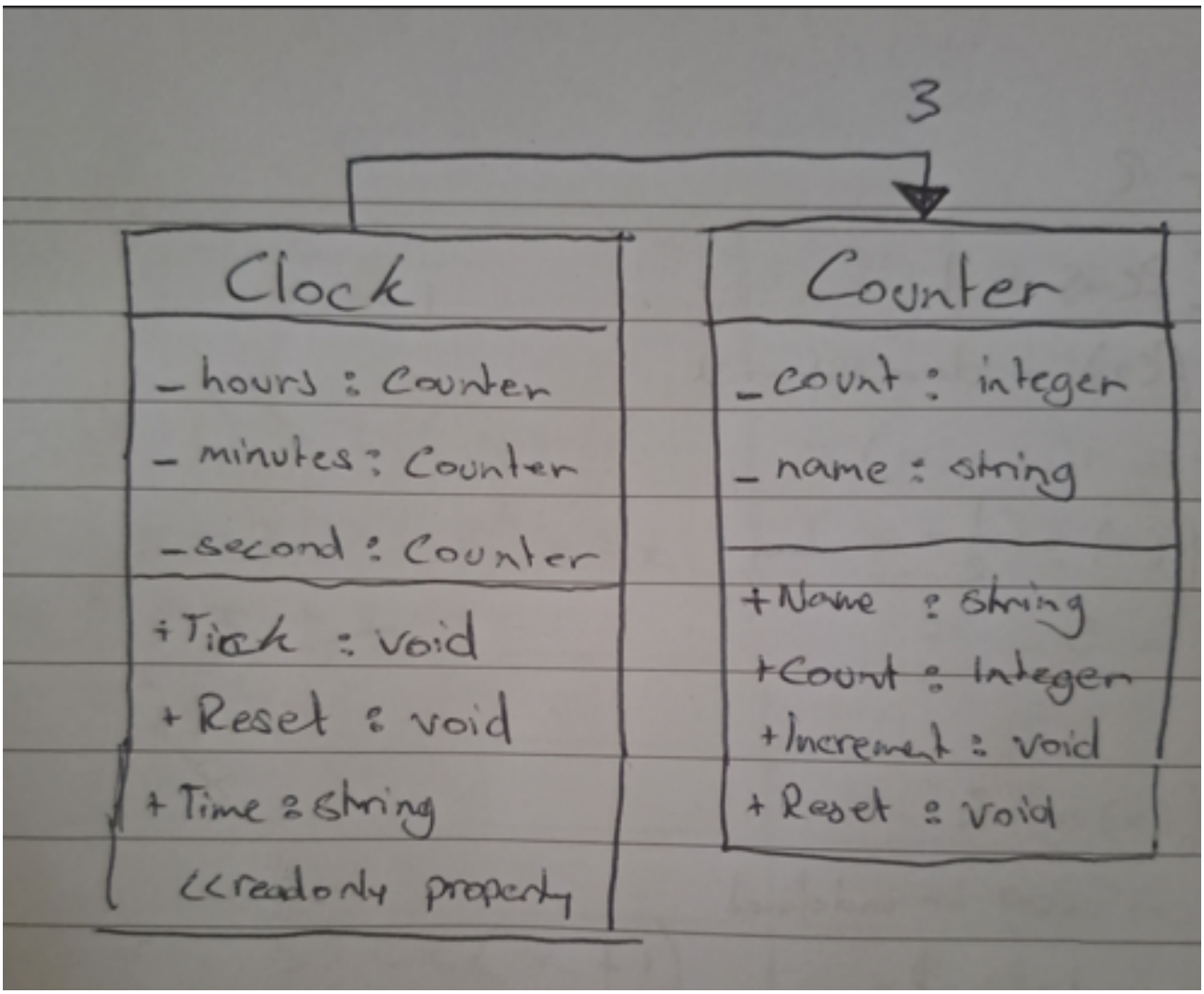
Swinburne University of Technology

COS20007 Object Oriented Programming

3.1P - Clock Class

PDF generated at 11:49 on Friday 17th March, 2023


3
Clock
- hours : Counter
- minutes : Counter
- second : Counter
+ Tick : void
+ Reset : void
+ Time : String
<<readonly property
Counter
- count : integer
- name : string
+ Name : String
+ Count : Integer
+ Increment : void
+ Reset : void

```
using System;
using CounterTask;
using Clock_Class;

namespace Clocks
{

    class MainClass
    {

        public static void Main(string[] args)
        {
            Clock main_clock = new Clock();

            for (int i = 0; i < 356; i++)
            {
                main_clock.Tick();
            }

            Console.WriteLine(main_clock.Time);
        }
    }
}
```

```
using System;
using CounterTask;

namespace Clock_Class
{

    public class Clock
    {

        private Counter _hours = new Counter("Hours");
        private Counter _minutes = new Counter("Minutes");
        private Counter _seconds = new Counter("Seconds");

        public void Tick()
        {

            _seconds.Increment();

            if (_seconds.Count == 60)
            {

                _seconds.Reset();

                _minutes.Increment();
            }

            if (_minutes.Count == 60)
            {

                _minutes.Reset();

                _hours.Increment();
            }

            if (_hours.Count == 24)
            {

                _hours.Reset();

            }
        }

        public void Reset()
        {

            _seconds.Reset();

            _minutes.Reset();

            _hours.Reset();
        }

        public string Time
```

```
54          {
55
56              get
57              {
58
59                  string seconds = "";
60                  string minutes = "";
61                  string hours = "";
62
63                  if (_seconds.Count < 10) { seconds = "0" + (_seconds.Count); } else
    ↪  { seconds = Convert.ToString(_seconds.Count); }
64                  if (_minutes.Count < 10) { minutes = "0" + (_minutes.Count); } else
    ↪  { minutes = Convert.ToString(_minutes.Count); }
65                  if (_hours.Count < 10) { hours = "0" + (_hours.Count); } else {
    ↪  hours = Convert.ToString(_hours.Count); }
66
67                  return hours + ":" + minutes + ":" + seconds;
68              }
69          }
70
71          public int Hours
72          {
73              get { return _hours.Count; }
74          }
75
76          public int Minutes
77          {
78              get { return _minutes.Count; }
79          }
80
81          public int Seconds
82          {
83              get { return _seconds.Count; }
84          }
85
86      }
87
88  }
```

```
using Clock_Class;

namespace Clock_Test
{
    public class Tests
    {
        [SetUp]
        public void Setup()
        {
        }

        [Test]
        public void clockInitializeStartsAtZero()
        {

            Clock test_clock = new Clock();

            if (Convert.ToInt32(test_clock.Hours) == 0 &&
↪   Convert.ToInt32(test_clock.Minutes) == 0 && Convert.ToInt32(test_clock.Seconds)
↪   == 0)
            {
                Assert.Pass();
            } else
            {
                Assert.Fail();
            }
        }

        [Test]

        public void clockIncrementAddsOne()
        {
            Clock test_clock = new Clock();

            test_clock.Tick();

            if (Convert.ToInt32(test_clock.Hours) == 0 &&
↪   Convert.ToInt32(test_clock.Minutes) == 0 && Convert.ToInt32(test_clock.Seconds)
↪   == 1)
            {
                Assert.Pass();
            } else
            {
                Assert.Fail();
            }
        }

        [Test]

        public void clockMultipleIncrementsWork() {

            Clock test_clock = new Clock();

```

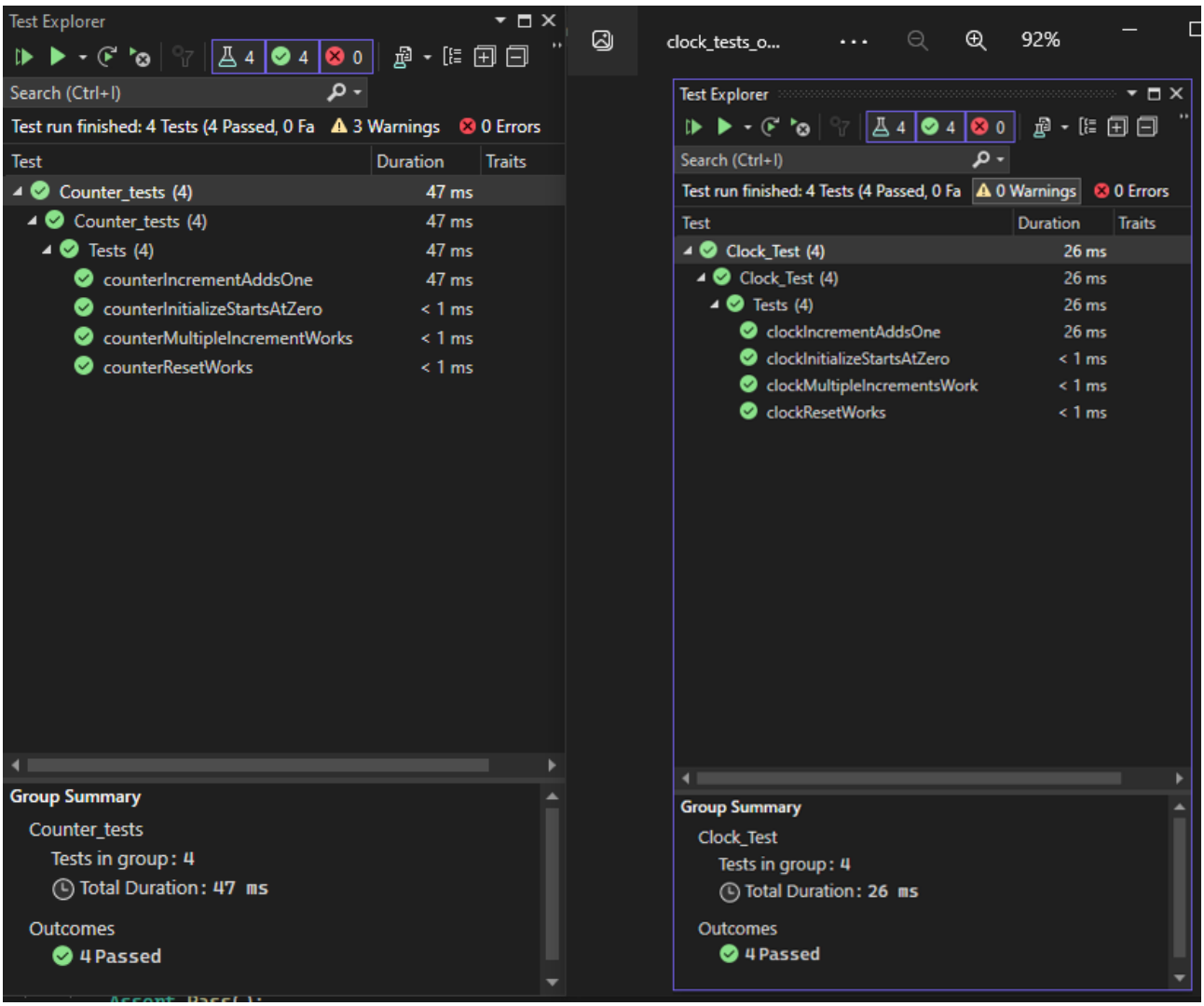
```
50              for (int i = 0; i < 115; i++)
51              {
52                  test_clock.Tick();
53              }
54
55              if (Convert.ToInt32(test_clock.Hours) == 0 &&
    ↪  Convert.ToInt32(test_clock.Minutes) == 1 && Convert.ToInt32(test_clock.Seconds)
    ↪  == 55)
56              {
57                  Assert.Pass();
58              }
59              else
60              {
61                  Assert.Fail();
62              }
63          }
64
65          [Test]
66
67          public void clockResetWorks()
68          {
69              Clock test_clock = new Clock();
70
71              for (int i = 0; i < 115; i++)
72              {
73                  test_clock.Tick();
74              }
75
76              test_clock.Reset();
77
78              if (Convert.ToInt32(test_clock.Hours) == 0 &&
    ↪  Convert.ToInt32(test_clock.Minutes) == 0 && Convert.ToInt32(test_clock.Seconds)
    ↪  == 0)
79              {
80                  Assert.Pass();
81              }
82              else
83              {
84                  Assert.Fail();
85              }
86          }
87      }
88  }
```

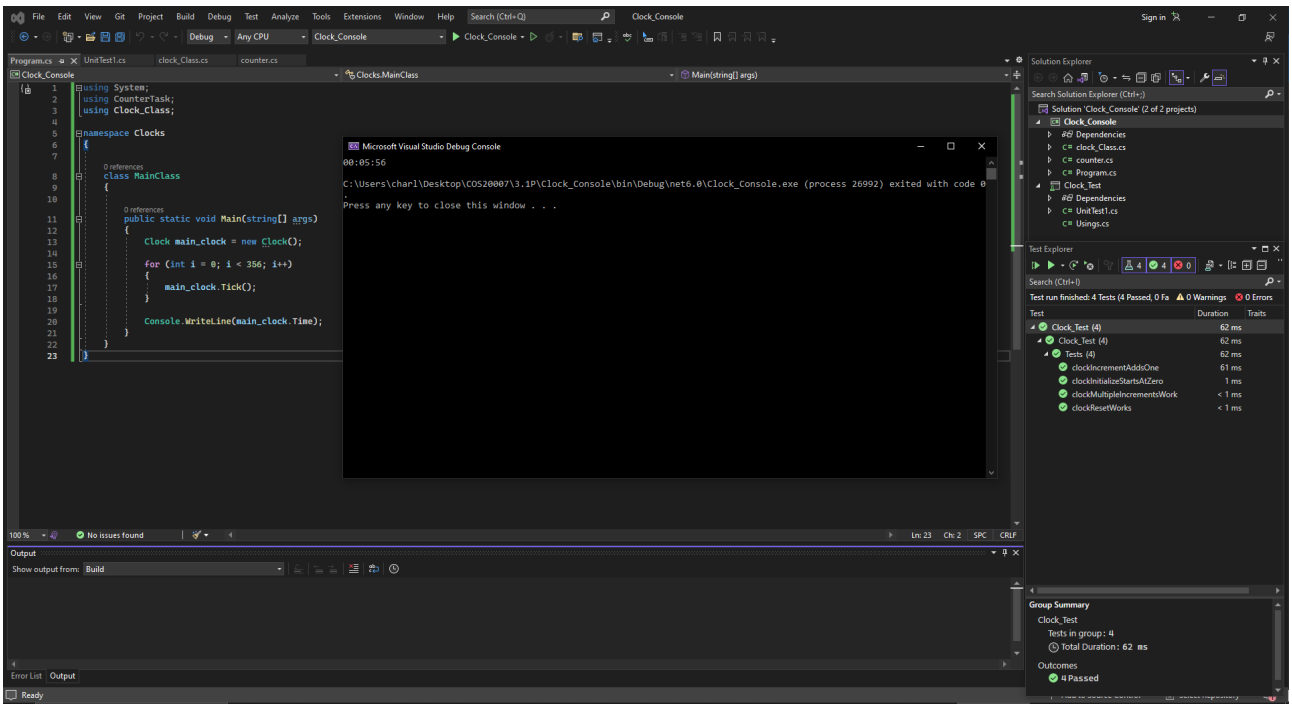
```
using System;

namespace CounterTask {

    public class Counter {

        private int _count;
        private string _name;

        public string Name {

            get {
                return _name;
            }

            set {
                _name = value;
            }

        }

        public int Count {

            get {
                return _count;
            }

            set {
                _count = value;
            }

        }

        public Counter(string name) {

            _name = name;

            _count = 0;

        }

        public void Increment() {

            _count ++;

        }

        public void Reset() {

            _count = 0;

        }

```

```
54
55      }
56  }
```

```
using CounterTask;

namespace Counter_tests
{
    public class Tests
    {
        [SetUp]
        public void Setup()
        {
        }

        [Test]
        public void counterInitializeStartsAtZero()
        {
            Counter test_counter = new Counter("test");

            if (test_counter.Count == 0)
            {
                Assert.Pass();
            } else
            {
                Assert.Fail();
            }
        }

        [Test]

        public void counterIncrementAddsOne()
        {
            Counter test_counter = new Counter("test");

            test_counter.Increment();

            if (test_counter.Count == 1)
            {
                Assert.Pass();
            } else
            {
                Assert.Fail();
            }
        }

        [Test]

        public void counterMultipleIncrementWorks()
        {
            Counter test_counter = new Counter("test");

            for (int i = 0; i < 15;  i++)
            {
                test_counter.Increment();
            }

```

```
54              if (test_counter.Count == 15)
55              {
56                  Assert.Pass();
57              } else
58              {
59                  Assert.Fail();
60              }
61          }
62
63          [Test]
64
65          public void counterResetWorks()
66          {
67              Counter test_counter = new Counter("test");
68
69              for (int i = 0;i < 15;i++)
70              {
71                  test_counter.Increment();
72              }
73
74              test_counter.Reset();
75
76              if (test_counter.Count == 0)
77              {
78                  Assert.Pass();
79              } else
80              {
81                  Assert.Fail();
82              }
83          }
84      }
85  }
```


Test Explorer
4 4 0
Search (Ctrl+I)
Test run finished: 4 Tests (4 Passed, 0 Fa
3 Warnings
0 Errors
Test
Duration
Traits
Counter_tests (4) 47 ms
Counter_tests (4) 47 ms
Tests (4) 47 ms
counterIncrementAddsOne 47 ms
counterInitializeStartsAtZero < 1 ms
counterMultipleIncrementWorks < 1 ms
counterResetWorks < 1 ms
Group Summary
Counter_tests
Tests in group: 4
Total Duration: 47 ms
Outcomes
4 Passed
clock_tests_o...
92%
Test Explorer
4 4 0
Search (Ctrl+I)
Test run finished: 4 Tests (4 Passed, 0 Fa
0 Warnings
0 Errors
Test
Duration
Traits
Clock_Test (4) 26 ms
Clock_Test (4) 26 ms
Tests (4) 26 ms
clockIncrementAddsOne 26 ms
clockInitializeStartsAtZero < 1 ms
clockMultipleIncrementsWork < 1 ms
clockResetWorks < 1 ms
Group Summary
Clock_Test
Tests in group: 4
Total Duration: 26 ms
Outcomes
4 Passed


using System;
using CounterTask;
using Clock_Class;
namespace Clocks
{
class MainClass
{
public static void Main(string[] args)
{
Clock main_clock = new Clock();
for (int i = 0; i < 356; i++)
{
main_clock.Tick();
}
Console.WriteLine(main_clock.Time);
}
}
}
Microsoft Visual Studio Debug Console
00:05:56
C:\Users\charl\Desktop\COS20007\3.1P\Clock_Console\bin\Debug\net6.0\Clock_Console.exe (process 26992) exited with code 0
.
Press any key to close this window . . .
Test run finished: 4 Tests (4 Passed, 0 Fa
0 Warnings
0 Errors
clockIncrementAddsOne
clockInitializeStartsAtZero
clockMultipleIncrementsWork
clockResetWorks
Group Summary
Clock_Test
Tests in group: 4
Total Duration: 62 ms
Outcomes
4 Passed