3.1 ClockClass

Source code for clock.cs

```
using counterclass;//link references to Counter.cs

using System;

using System.Collections.Generic;

using System.Diagnostics.Metrics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;


namespace clockclass

{

  public class Clock

  {

    private Counter _hours; //added constructor

    private Counter _minutes;

    private Counter _second;


    public Clock()

    {

      _second = new Counter("seconds"); //declare second object

      _minutes = new Counter("minutes");//declare minute object

      _hours = new Counter("hours");//declare hours object

    }

      public void tick() //ticks per seconds,minute and hour

      {

        _second.increment(); //increment for second till reaches 59 then reset back to 0

        if (_second.Tick > 59) //till tick 59

        {
            _minutes.increment();//increment for minute till reaches 59 then reset back to
0

            _second.Reset();//reset back value

            if (_minutes.Tick > 59)

            {

              _hours.increment();//increment for hours till reaches 59 then reset back to 0

              _minutes.Reset();//reset back value

              if (_hours.Tick > 23) //increment hour still 23:59

              {

                ResetTime();//then reset back value

              }

            }

          }

        }


        public void ResetTime()// helps reset the time back to 0

        {

          _hours.Reset(); //call back function from clock.cs which reset back count to 0

          _minutes.Reset();

          _second.Reset();


        }

        public void Settime(String s)

        {

          string[] array = s.Split(":"); //in between array set the : symbol

                            //arrange the the array into a sequenced method hh:mm:ss


          _hours = new Counter("hours", int.Parse(array[0])); //first array value
```

```
            _minutes = new Counter("minutes", int.Parse(array[1]));//second array value

            _second = new Counter("second", int.Parse(array[2]));//third array value


        }

        public string CurrentTime()

        {

            return $"{_hours.Tick:D2}:{_minutes.Tick:D2}:{_second.Tick:D2}"; //now return all
array list

        }




    }



}
```

Program.cs

```
using counterclass;

using System;

using System.Diagnostics.CodeAnalysis;

using System.Diagnostics.Metrics;

using System.Reflection;

using clockclass;


namespace counterclass

{

    public class program

    {

        static void Main(string[] args)
        {
            Clock clock = new Clock();
            int i;

            for (i = 0; i < 86400; i++)//86400 is the total seconds in a day
            {
                Thread.Sleep(100);// value inside is the millieseconds
                Console.Clear();//erase previous value and display new value
                clock.tick(); //tick with clock.cs
                Console.WriteLine(clock.CurrentTime());//write the currenttime function from clock.cs

            }
        }

    }
}
```

TestClock.cs

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

namespace Testing
{
    public class TestClock
```

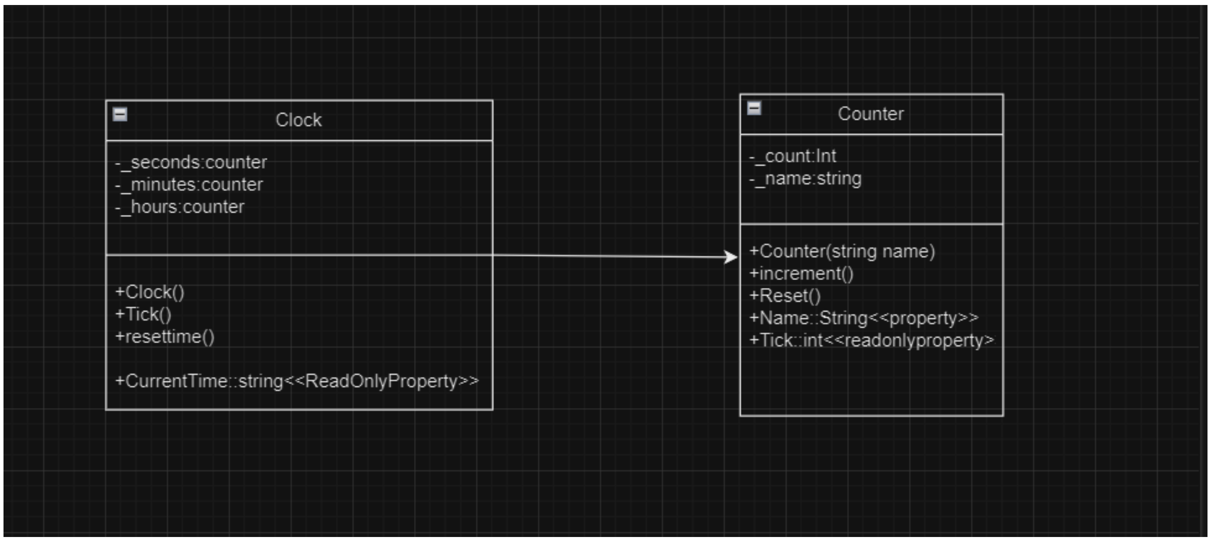
```
{
  Clock _clock;


  [SetUp]
  public void Setup()
  {
    _clock = new Clock();
  }
  [Test]
  public void TestClockStart()
  {
    Assert.That(_clock.CurrentTime(), Is.EqualTo("00:00:00"));



  }
  [Test]
  public void TestReset()
  {
    int i;
    for (i = 0; i < 86400; i++)
    {
      _clock.tick();
    }
    _clock.ResetTime();
    Assert.That(_clock.CurrentTime(), Is.EqualTo("00:00:00"));


  }
  [TestCase(0, "00:00:00")]
```

```
        [TestCase(60, "00:01:00")]

        [TestCase(120, "00:02:00")]

        [TestCase(86340, "23:59:00")]


        public void TestRunning(int tick, string currenttime)

        {

            int i;

            for (i = 0; i < tick; i++)

            {

                _clock.tick();

            }

            Assert.That(_clock.CurrentTime(), Is.EqualTo(currenttime));

        }


        [TestCase("00:01:00", "00:00:59")] //Roll to 1 min

        [TestCase("01:00:00", "00:59:59")] //Roll to 1 hr

        [TestCase("00:00:00", "23:59:59")] //Roll to 1 day


        public void TestClockRollover(string exp, string settime)

        {

            _clock.Settime(settime);

            _clock.tick();

            Assert.That(_clock.CurrentTime(), Is.EqualTo(exp));

        }

    }


}
```

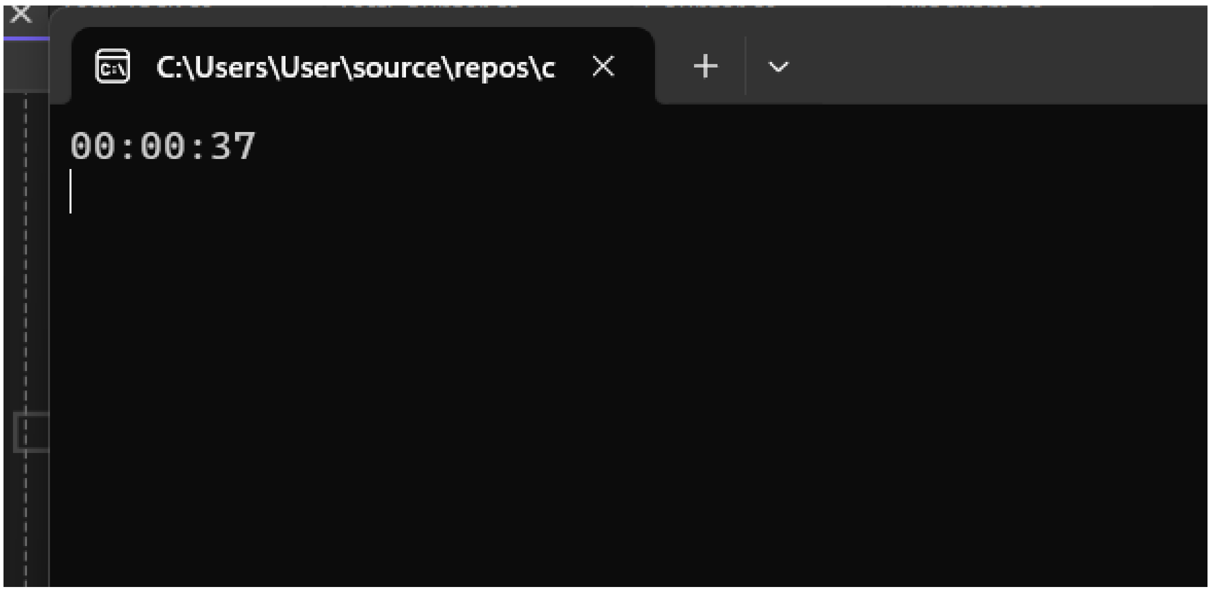
TestCounter.cs

```
using NUnit.Framework;

using counterclass;


namespace Clockclass

{

  public class TestCounter

  {

    Counter _countertest;


    [SetUp]

    public void Setup()

    {

      _countertest = new Counter("Test");

    }


    [Test]

    public void test_start9()

    {

      Assert.AreEqual(0, _countertest.Tick);

    }


    [Test]

    public void test_name()

    {

      Assert.AreEqual("Test", _countertest.Name);

    }
```

```
        [Test]

        public void test_count_reset()

        {

            _countertest.increment();

            _countertest.Reset();

            Assert.AreEqual(0, _countertest.Tick);

        }


        [TestCase(60, 60)]

        [TestCase(100, 100)]

        public void test_increment(int tick, int result)

        {

            int i;

            for (i = 0; i < tick; i++)

            {

                _countertest.increment();

            }

            Assert.AreEqual(result, _countertest.Tick);

        }

    }

}
```
UML diagram

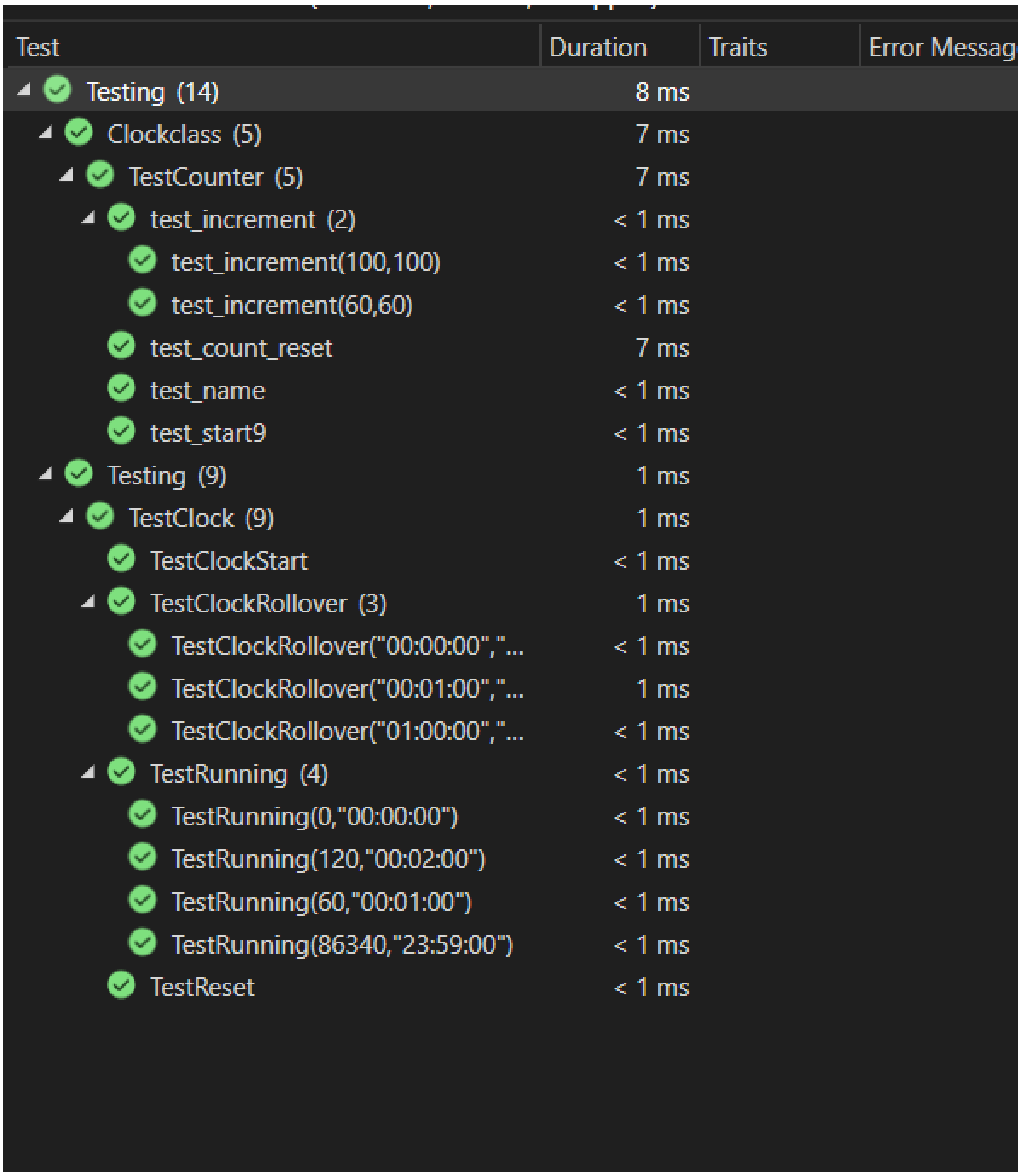


output for program



```
00:00:37
```

output for testing



```
Test                                          Duration    Traits      Error Messag
◢ ✓ Testing (14)                                  8 ms
  ◢ ✓ Clockclass (5)                              7 ms
    ◢ ✓ TestCounter (5)                           7 ms
      ◢ ✓ test_increment (2)                    < 1 ms
          ✓ test_increment(100,100)             < 1 ms
          ✓ test_increment(60,60)               < 1 ms
        ✓ test_count_reset                        7 ms
        ✓ test_name                             < 1 ms
        ✓ test_start9                           < 1 ms
  ◢ ✓ Testing (9)                                 1 ms
    ◢ ✓ TestClock (9)                             1 ms
        ✓ TestClockStart                        < 1 ms
      ◢ ✓ TestClockRollover (3)                   1 ms
          ✓ TestClockRollover("00:00:00","...   < 1 ms
          ✓ TestClockRollover("00:01:00","...     1 ms
          ✓ TestClockRollover("01:00:00","...   < 1 ms
      ◢ ✓ TestRunning (4)                       < 1 ms
          ✓ TestRunning(0,"00:00:00")           < 1 ms
          ✓ TestRunning(120,"00:02:00")         < 1 ms
          ✓ TestRunning(60,"00:01:00")          < 1 ms
          ✓ TestRunning(86340,"23:59:00")       < 1 ms
        ✓ TestReset                             < 1 ms
```