School of Science, Computing and Engineering and Technologies

Object Oriented Programming

Pass Task 3.1: Clock Class with your own hour format **Overview**

In this task, you experiment with *collaboration*, a mechanism in which objects work together to achieve desired outcomes. In particular, you will develop a simple 24-hour clock application that will reuse class *Counter* that you developed in task *2.2P Counter Class*.

Purpose: Practice with the object-oriented programming techniques *collaboration*, *encapsulation*, and *reuse*.

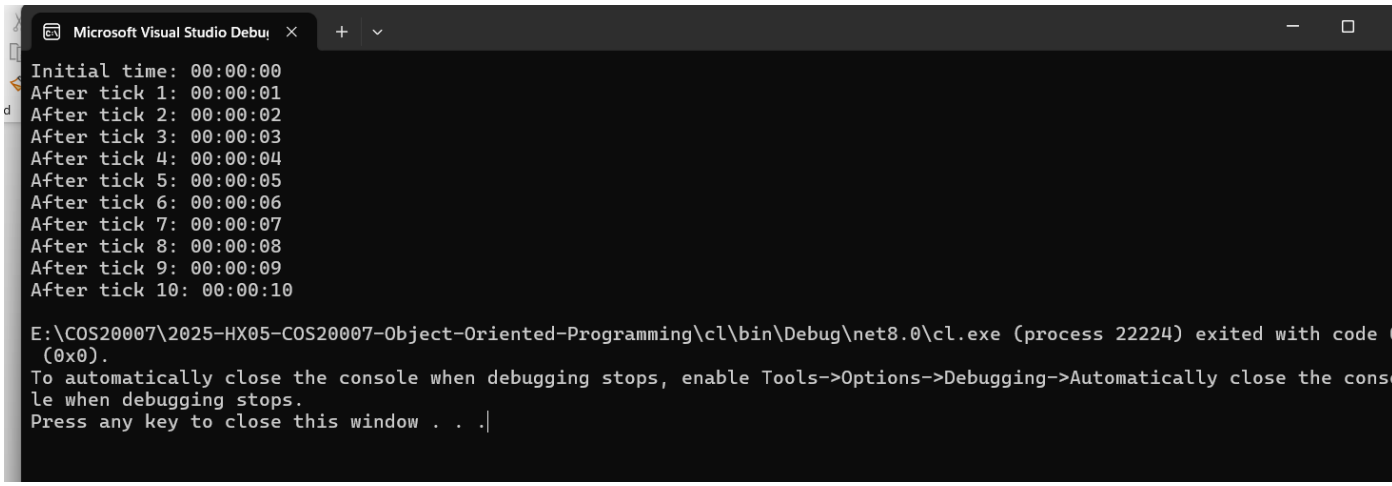
Task: **Design and develop a 24-hour clock application with personalized requirement**.

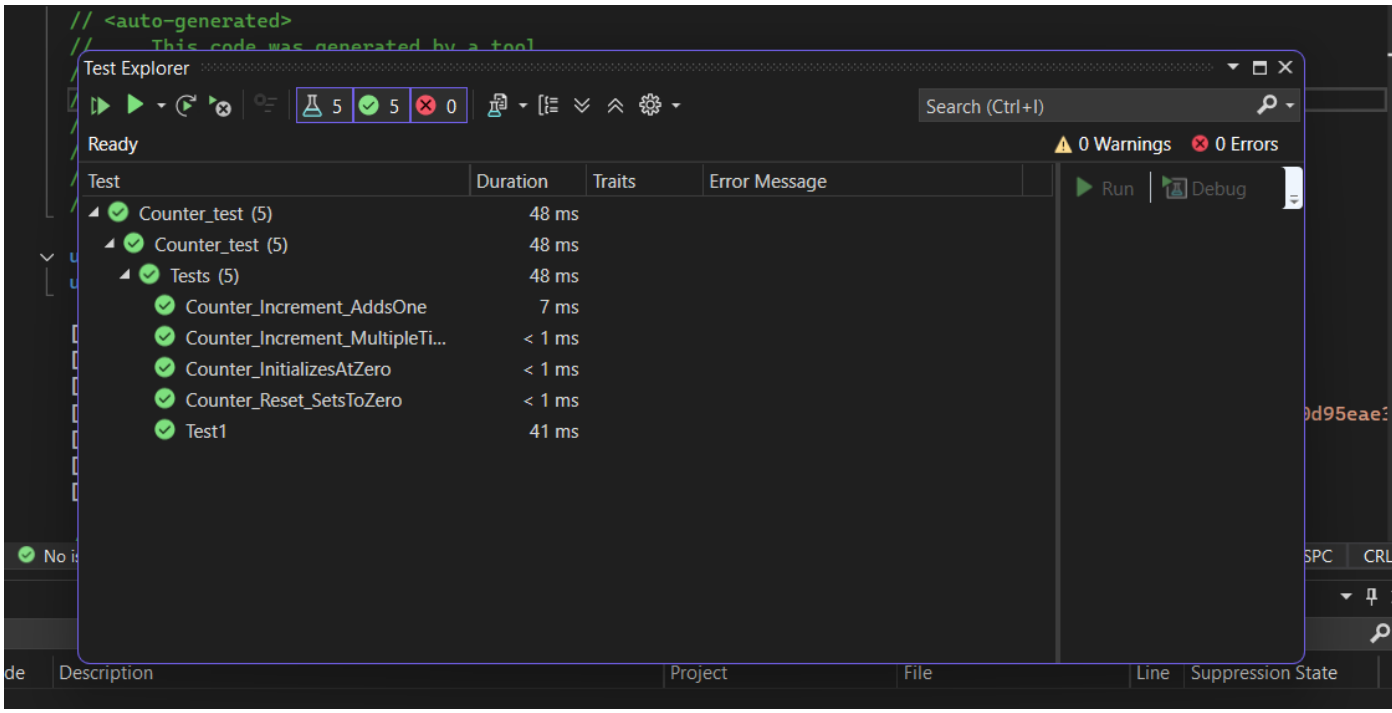
Deadline: Due by the end of week four, Friday, 30 May 2025, 23:59 Hanoi Time **(Firmed)**.

Submission Details

All students have access to the Adobe Acrobat tools. Please print your solution to PDF and combine it with the screenshots taken for this task.

- Program source code
- Test source code
- Image of class UML diagram
- Screenshots of unit test results
- Screenshot of program execution





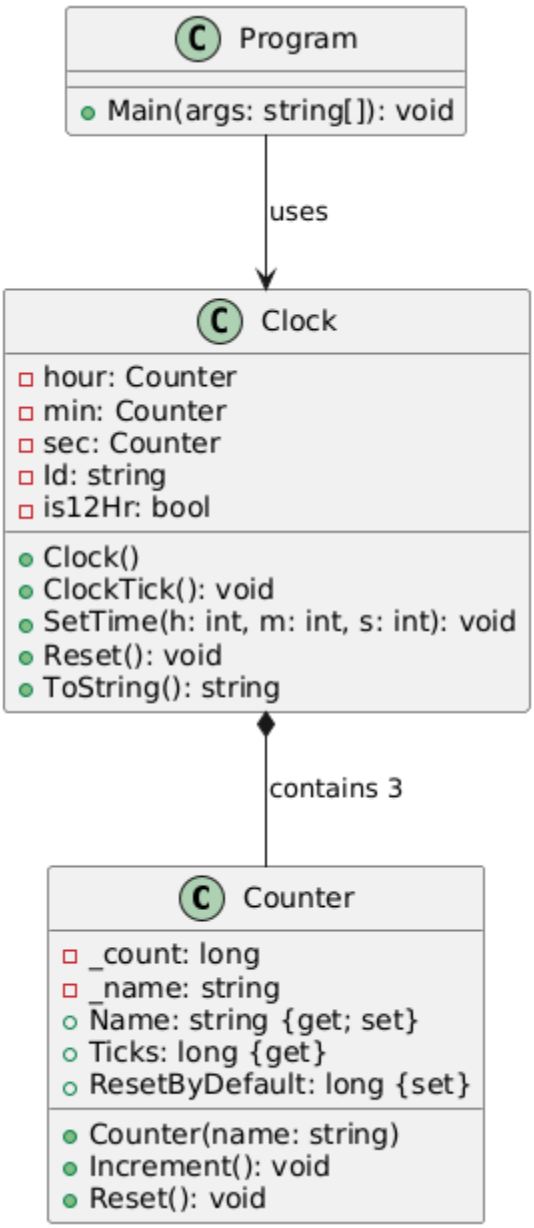

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace cl
{
    public class Clock
```

```
{
    Counter hour = new Counter("Hour");
    Counter min = new Counter("Min");
    Counter sec = new Counter("Sec");
    string Id = "SWS01358";
    bool is12Hr;

    public Clock()
    {
        // Determine 12-hour or 24-hour format based on last digit of Id
        char lastChar = Id[Id.Length - 1];
        if (char.IsDigit(lastChar) && (lastChar - '0') <= 5)
        {
            is12Hr = true;
        }
        else
        {
            is12Hr = false;
        }
        hour = new Counter("Hour");
        min = new Counter("Minute");
        sec = new Counter("Second");
    }

    public void ClockTick()
    {
        sec.Increment();
        if (sec.Ticks >= 60)
        {
            sec.Reset();
            min.Increment();
```

```
            if (min.Ticks >= 60)
            {
                min.Reset();
                hour.Increment();
                if (hour.Ticks >= (is12Hr ? 12 : 24))
                {
                    hour.Reset();
                }
            }
        }
    }

    public void SetTime(int h, int m, int s)
    {
        for (long i = 0; i < h; i++) hour.Increment();
        for (long i = 0; i < m; i++) min.Increment();
        for (long i = 0; i < s; i++) sec.Increment();
    }

    public void Reset()
    {
        hour.Reset();
        min.Reset();
        sec.Reset();
    }

    public override string ToString()
    {
        return $"{hour.Ticks:D2}:{min.Ticks:D2}:{sec.Ticks:D2}";
    }
}
```

```
}


using NUnit.Framework;
using cl;

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

        [Test]
        public void Test1()
        {
            Assert.Pass();
        }

        [Test]
        public void Counter_InitializesAtZero()
        {
            var counter = new Counter("Test");
            Assert.That(counter.Ticks, Is.EqualTo(0));
        }

        [Test]
        public void Counter_Increment_AddsOne()
        {
```

```
            var counter = new Counter("Test");
            counter.Increment();
            Assert.That(counter.Ticks, Is.EqualTo(1));
        }

        [Test]
        public void Counter_Increment_MultipleTimes()
        {
            var counter = new Counter("Test");
            for (int i = 0; i < 5; i++) counter.Increment();
            Assert.That(counter.Ticks, Is.EqualTo(5));
        }

        [Test]
        public void Counter_Reset_SetsToZero()
        {
            var counter = new Counter("Test");
            counter.Increment();
            counter.Increment();
            counter.Reset();
            Assert.That(counter.Ticks, Is.EqualTo(0));
        }
    }
}
```


Program
Main(args: string[]): void
uses
Clock
hour: Counter
min: Counter
sec: Counter
Id: string
is12Hr: bool
Clock()
ClockTick(): void
SetTime(h: int, m: int, s: int): void
Reset(): void
ToString(): string
contains 3
Counter
_count: long
_name: string
Name: string {get; set}
Ticks: long {get}
ResetByDefault: long {set}
Counter(name: string)
Increment(): void
Reset(): void