Day 7

Dharunya S

1. Create a dependency injection sample for Painter class

1.1 inject tools (or paintbrush) to painter object via Dependency injection

2. Create at least 10 NUnit tests

2.1 run the tests using both visual studio and command line

Program.cs

```
using Day8daily;
using System;

namespace PainterApp
{
   class Program
   {
      public static void Main(string[] args)
      {
         ITool brush = new PaintBrush();
         ITool brush2 = new PaintBrush2();

         Painter alice = new Painter("Alice", brush);
         Painter bob = new Painter("Bob", brush2);

         alice.GoesToWork();
         bob.GoesToWork();
      }
   }
}
```

Painter.cs

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

namespace Day8daily
{
   public class Painter
   {
      public string Name { get; set; }
      public ITool Tool { get; set; }
```

```
        public Painter(string name, ITool tool)
        {
            Name = name;
            Tool = tool;
        }

        public void GoesToWork()
        {
            Console.WriteLine($"{Name}  {Tool.Use()}");
        }

    }
}
```

PaintBrush.cs

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

namespace Day8daily
{

        public class PaintBrush : ITool
        {
            public string Use()
            {
                return "Painting with a brush.";
            }
        }
    }
```

PaintBrush2.cs

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

namespace Day8daily
{
    public class PaintBrush2 : ITool
    {
```

```
        public string Use()
        {
            return "Painting with a paintbrush2.";
        }
    }
}
```

Itool.cs

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

namespace Day8daily
{
    public interface ITool
    {
        string Use();
    }
}
```

UnitTest1.cs

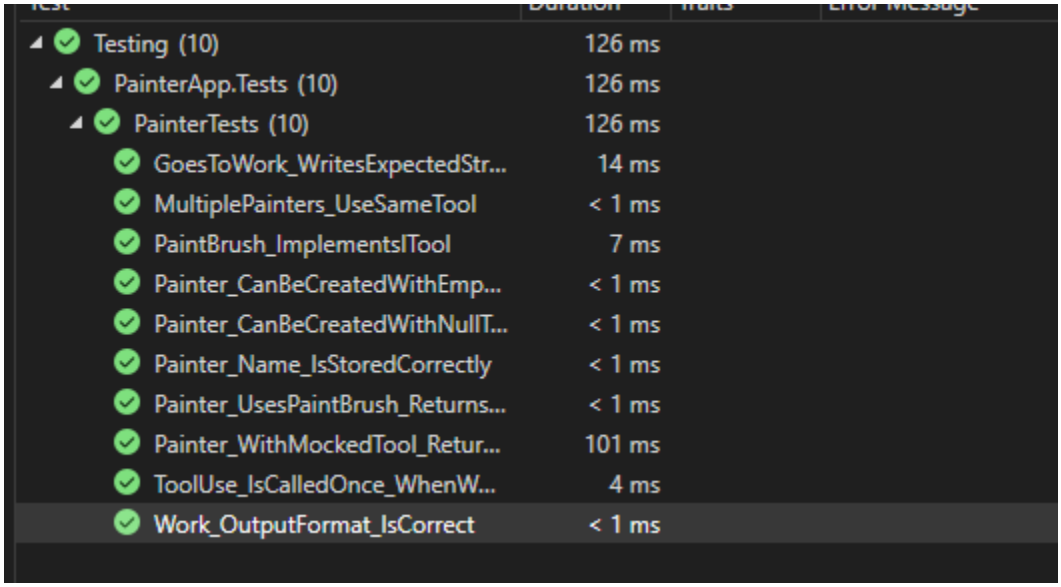
```
using Day8daily;
using Moq;
using NUnit.Framework;

using System;
using System.Reflection;

namespace PainterApp.Tests
{
    [TestFixture]
    public class PainterTests
    {
        [Test]
        public void Painter_UsesPaintBrush_ReturnsCorrectOutput()
        {
            var brush = new PaintBrush();
            var painter = new Painter("Alice", brush);

            var result = painter.Tool.Use();

            Assert.AreEqual("Painting with a brush.", result);
```

```
}

[Test]
public void PaintBrush_ImplementsITool()
{
    ITool brush = new PaintBrush();
    Assert.IsInstanceOf<ITool>(brush);
}

[Test]
public void Painter_Name_IsStoredCorrectly()
{
    var brush = new PaintBrush();
    var painter = new Painter("Bob", brush);

    Assert.AreEqual("Bob", painter.Name);
}

[Test]
public void GoesToWork_WritesExpectedString()
{
    var brush = new PaintBrush();
    var painter = new Painter("Carl", brush);

    using var sw = new System.IO.StringWriter();
    Console.SetOut(sw);

    painter.GoesToWork();

    var expected = $"Carl  Painting with a brush.{Environment.NewLine}";
    Assert.AreEqual(expected, sw.ToString());
}

[Test]
public void Painter_WithMockedTool_ReturnsMockedOutput()
{
    var mockTool = new Mock<ITool>();
    mockTool.Setup(t => t.Use()).Returns("mock painting");

    var painter = new Painter("Frank", mockTool.Object);

    Assert.AreEqual("mock painting", painter.Tool.Use());
}
```

```
[Test]
public void MultiplePainters_UseSameTool()
{
    var brush = new PaintBrush();

    var painter1 = new Painter("Grace", brush);
    var painter2 = new Painter("Harry", brush);

    Assert.AreEqual("Painting with a brush.", painter1.Tool.Use());
    Assert.AreEqual("Painting with a brush.", painter2.Tool.Use());
}

[Test]
public void ToolUse_IsCalledOnce_WhenWorkIsCalled()
{
    var mockTool = new Mock<ITool>();
    mockTool.Setup(t => t.Use()).Returns("Test use").Verifiable();

    var painter = new Painter("Ian", mockTool.Object);

    var result = painter.Tool.Use();

    mockTool.Verify(t => t.Use(), Times.Once);
    Assert.AreEqual("Test use", result);
}

[Test]
public void Work_OutputFormat_IsCorrect()
{
    var brush = new PaintBrush();
    var painter = new Painter("Jane", brush);

    var workOutput = $"{painter.Name}  {painter.Tool.Use()}";

    Assert.AreEqual("Jane  Painting with a brush.", workOutput);
}

[Test]
public void Painter_CanBeCreatedWithNullTool()
{
    var painter = new Painter("NullTool", null);
    Assert.IsNull(painter.Tool);
}
[Test]
```

```
        public void Painter_CanBeCreatedWithEmptyName()
        {
            var brush = new PaintBrush();
            var painter = new Painter(string.Empty, brush);
            Assert.AreEqual(string.Empty, painter.Name);
        }


    }
}
```

Output:
Alice Painting with a brush.
Bob Painting with a paintbrush2.



Output.cmd
C:\Users\dharunya.s\source\repos\Day8daily\Testing>dotnet test Testing.csproj
Restore complete (0.8s)
** Day8daily succeeded (0.3s) → C:\Users\dharunya.s\source\repos\Day8daily\Day8daily\bin\Debug\net8.0\Day8daily.dll**
** Testing succeeded with 10 warning(s) (0.8s) → bin\Debug\net8.0\Testing.dll**
**   C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(106,51): warning CS8625: Cannot convert null literal to non-nullable reference type.**

C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(21,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(37,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(52,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(63,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(74,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(75,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(89,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(100,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
C:\Users\dharunya.s\source\repos\Day8daily\Testing\UnitTest1.cs(114,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md)
NUnit Adapter 4.5.0.0: Test execution started
Running all tests in C:\Users\dharunya.s\source\repos\Day8daily\Testing\bin\Debug\net8.0\Testing.dll
NUnit3TestExecutor discovered 10 of 10 NUnit test cases using Current Discovery mode, Non-Explicit run
NUnit Adapter 4.5.0.0: Test execution complete
Testing test succeeded (3.0s)

Test summary: total: 10, failed: 0, succeeded: 10, skipped: 0, duration: 3.0s
Build succeeded with 10 warning(s) in 6.0s