

Interfaces



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

What is an Interface



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Agenda

- What is an Interface?
- How to declare and implement interfaces
- Interfaces and testability
- Interfaces and extensibility



What?

- A language construct that is similar to a class (in terms of syntax) but is fundamentally different.



Syntax

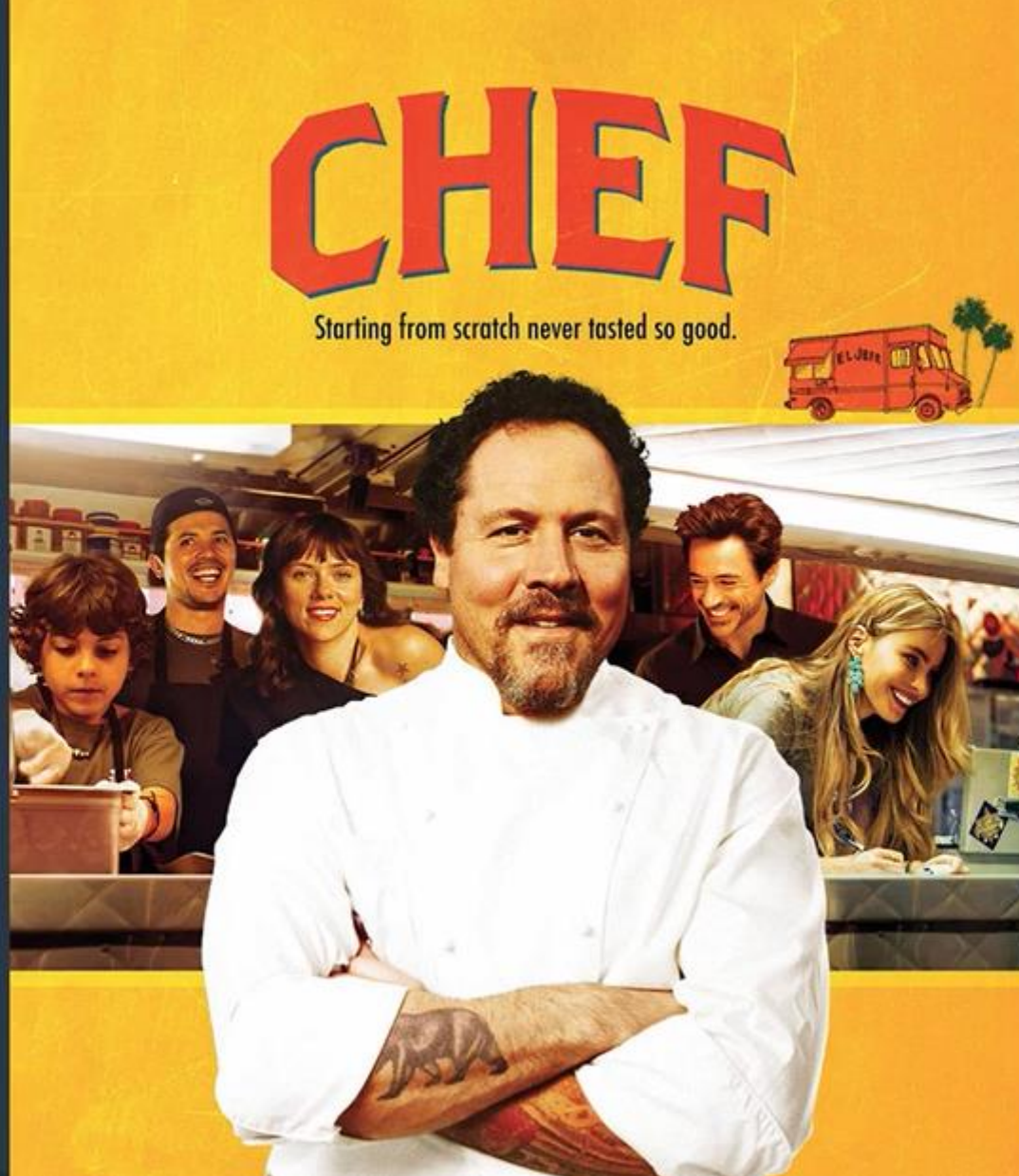
```
public interface ITaxCalculator
{
    int Calculate();
}
```



Why?

- To build loosely-coupled applications.





C# Intermediate



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

OrderProcessor



TaxCalculator





Interfaces and Testability



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
using System;

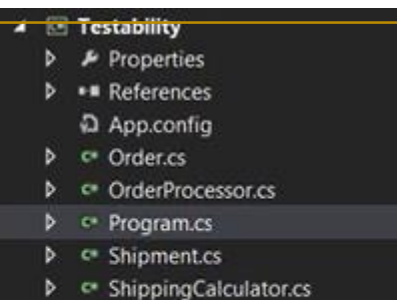
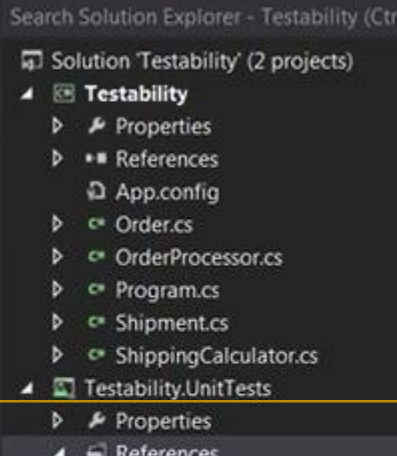
namespace Testability
{
    public class Order
    {
        public int Id { get; set; }
        public DateTime DatePlaced { get; set; }
        public Shipment Shipment { get; set; }
        public float TotalPrice { get; set; }

        public bool IsShipped
        {
            get { return Shipment != null; }
        }
    }
}
```

```
order.Shipment = new Shipment
{
    Cost = _shippingCalculator.CalculateShipping(order),
    ShippingDate = DateTime.Today.AddDays(1)
};
```

```
public class ShippingCalculator
{
    public float CalculateShipping(Order order)
    {
        if (order.TotalPrice < 30f)
            return order.TotalPrice * 0.1f;

        return 0;
    }
}
```



OrderProcessor.cs

using System;

namespace Testability

{

public class OrderProcessor

{

private readonly ShippingCalculator _shippingCalculator;

public OrderProcessor()

{

_shippingCalculator = new ShippingCalculator();

}

public void Process(Order order)

{

if (order.IsShipped)

throw new InvalidOperationException("This order is already processed.");

order.Shipment = new Shipment

{

Cost = _shippingCalculator.CalculateShipping(order),

ShippingDate = DateTime.Today.AddDays(1)

};

}

}

}

100%

Solution Explorer - Testability

Search Solution Explorer - Testability (Ctrl+)

Solution 'Testability' (1 project)

Testability

Properties

References

App.config

Order.cs

OrderProcessor.cs

Program.cs

Shipment.cs

ShippingCalculator.cs



```
Order.cs | OrderProcessor.cs | Program.cs | Shipment.cs | ShippingCalculator.cs
Testability.Program
Main(string[] args)

using System;

namespace Testability
{
    class Program
    {
        static void Main(string[] args)
        {
            var orderProcessor = new OrderProcessor();
            var order = new Order {DatePlaced = DateTime.Now, TotalPrice = 100f};
            orderProcessor.Process(order);
        }
    }
}
```

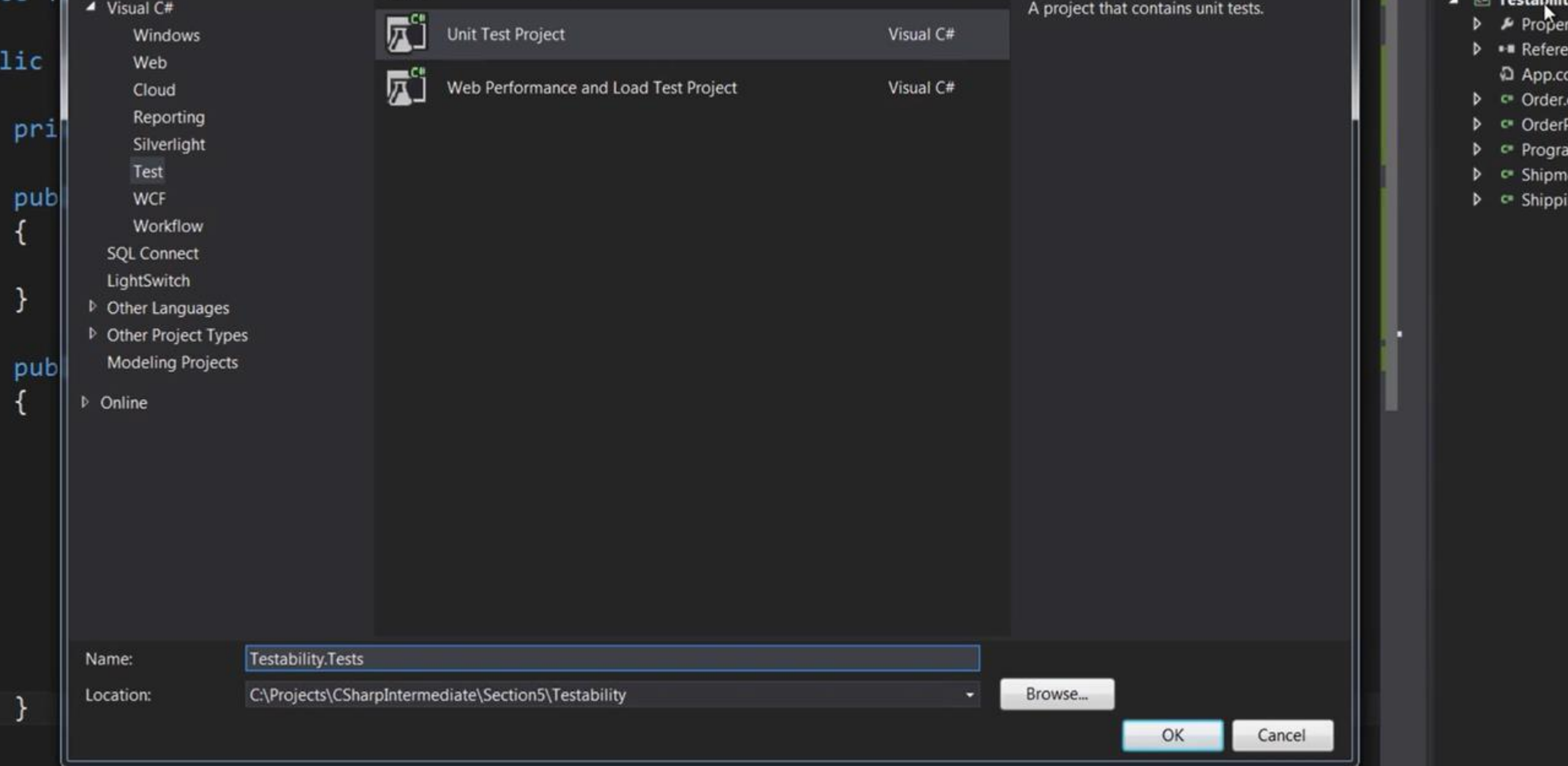
Solution Explorer - Testability

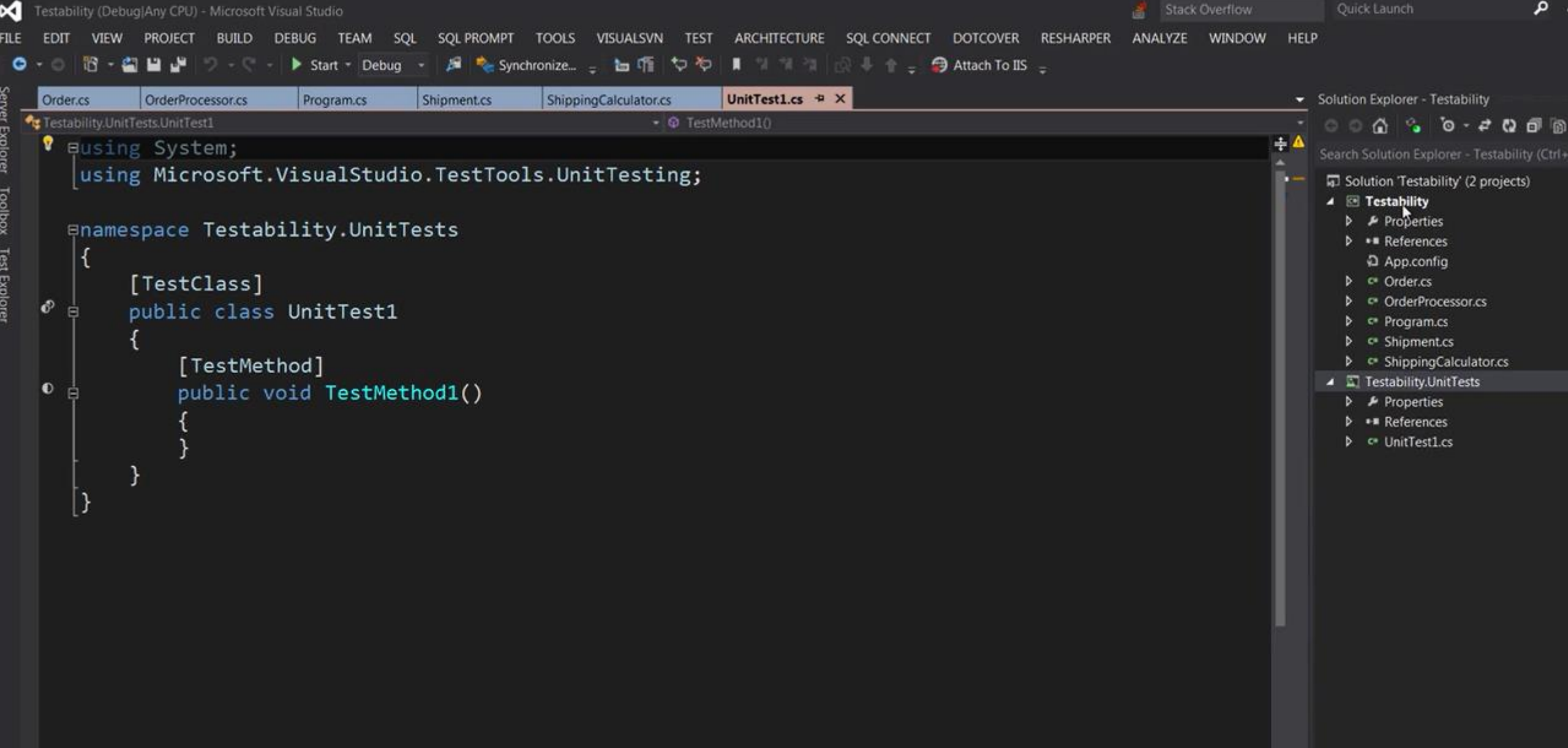
Search Solution Explorer - Testability (Ctrl+)

Solution 'Testability' (1 project)

- Testability
 - Properties
 - References
 - App.config
 - Order.cs
 - OrderProcessor.cs
 - Program.cs
 - Shipment.cs
 - ShippingCalculator.cs







C# Intermediate



Zouhair Rimale, Ph.D.
Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
namespace Testability
{
    public class OrderProcessor
    {
        private readonly ShippingCalculator _shippingCalculator;

        public OrderProcessor()
        {
            _shippingCalculator = new ShippingCalculator();
        }

        public void Process(Order order)
        {
            if (order.IsShipped)
                throw new InvalidOperationException("This order is already processed.");

            order.Shipment = new Shipment
            {
                Cost = _shippingCalculator.CalculateShipping(order),
                ShippingDate = DateTime.Today.AddDays(1)
            };
        }
    }
}
```

- Testability
 - Properties
 - References
 - App.config
 - Order.cs
 - OrderProcessor.cs
 - Program.cs
 - Shipment.cs
 - ShippingCalculator.cs
- Testability.UnitTests
 - Properties
 - References
 - UnitTest1.cs



Testability (Debug)Any CPU - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL SQL PROMPT TOOLS VISUALSVN TEST ARCHITECTURE SQL CONNECT DOTCOVER RESHARPER ANALYZE WINDOW HELP

Start Debug Synchronize Attach To IIS

Order.cs OrderProcessor.cs Program.cs Shipment.cs ShippingCalculator.cs x UnitTest1.cs

Testability.ShippingCalculator CalculateShipping(Order order)

```
namespace Testability
{
    public interface IShippingCalculator
    {
        float CalculateShipping(Order order);
    }

    public class ShippingCalculator : IShippingCalculator
    {
        public float CalculateShipping(Order order)
        {
            if (order.TotalPrice < 30f)
                return order.TotalPrice * 0.1f;

            return 0;
        }
    }
}
```

Solution Explorer - Testability

Search Solution Explorer - Testability (Ctrl+Shift+F)

Solution 'Testability' (2 projects)

- Testability
 - Properties
 - References
 - App.config
 - Order.cs
 - OrderProcessor.cs
 - Program.cs
 - Shipment.cs
 - ShippingCalculator.cs
- Testability.UnitTests
 - Properties
 - References
 - UnitTest1.cs

C# Intermediate



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
namespace Testability
{
    public class OrderProcessor
    {
        private readonly IShippingCalculator _shippingCalculator;

        public OrderProcessor(IShippingCalculator shippingCalculator)
        {
            _shippingCalculator = shippingCalculator;
        }

        public void Process(Order order)
        {
            if (order.IsShipped)
                throw new InvalidOperationException("This order is already processed.");

            order.Shipment = new Shipment
            {
                Cost = _shippingCalculator.CalculateShipping(order),
                ShippingDate = DateTime.Today.AddDays(1)
            };
        }
    }
}
```

Toolbox Test Explorer

Solution Testability (2 projects)

- Testability
 - Properties
 - References
 - App.config
 - Order.cs
 - OrderProcessor.cs
 - Program.cs
 - Shipment.cs
 - ShippingCalculator.cs
- Testability.UnitTests
 - Properties
 - References
 - UnitTest1.cs



Order.csOrderProcessor.csProgram.csShipment.csShippingCalculator.csOrderProcessorTests.cs

Testability.UnitTests.OrderProcessorTests

// METHODNAME_CONDITION_EXPECTATION

[TestMethod]

[ExpectedException(typeof(InvalidOperationException))]

public void Process_OrderIsAlreadyShipped_ThrowsAnException()

{

var orderProcessor = new OrderProcessor(new FakeShippingCalculator());

var order = new Order

{

Shipment = new Shipment()

};

orderProcessor.Process(order);

}

}

public class FakeShippingCalculator : IShippingCalculator

{

public float CalculateShipping(Order order)

{

return 1;

}

}

Solution Explorer - Testability

Search Solution Explorer - Testability (Ctrl+Shift+F)

Solution 'Testability' (2 projects)

Testability

Properties

References

App.config

Order.cs

OrderProcessor.cs

Program.cs

Shipment.cs

ShippingCalculator.cs

Testability.UnitTests

Properties

References

Microsoft.VisualStudio.QualityTools.UnitTestFramework

System

Testability

OrderProcessorTests.cs



Testability.UnitTests.OrderProcessorTests

```
var order = new Order
{
    Shipment = new Shipment()
};

orderProcessor.Process(order);
} // Process_OrderIsAlreadyShipped_ThrowsAnException

[TestMethod]
public void Process_OrderIsNotShipped_ShouldSetTheShipmentPropertyOfTheOrder()
{
    var orderProcessor = new OrderProcessor(new FakeShippingCalculator());
    var order = new Order();

    orderProcessor.Process(order);

    Assert.IsTrue(order.IsShipped);
    Assert.AreEqual(1, order.Shipment.Cost);
    Assert.AreEqual(DateTime.Today.AddDays(1), order.Shipment.ShippingDate);
}
} // OrderProcessorTests

public class FakeShippingCalculator : IShippingCalculator
{
    public float CalculateShipping(Order order)
    {
        return 1;
    }
}
```

Search Solution Explorer - Testability (Ctrl+)

- Solution 'Testability' (2 projects)
- Testability
 - Properties
 - References
 - App.config
 - Order.cs
 - OrderProcessor.cs
 - Program.cs
 - Shipment.cs
 - ShippingCalculator.cs
- Testability.UnitTests
 - Properties
 - References
 - OrderProcessorTests.cs

C# Intermediate



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Interfaces and Extensibility



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Program.cs

Extensibility.DbMigrator

Migrate()

```
using System;

namespace Extensibility
{
    public class DbMigrator
    {
        public void Migrate()
        {
            Console.WriteLine("Migrationg started at {0}", DateTime.Now);

            // Details of migrating the database

            Console.WriteLine("Migrationg finished at {0}", DateTime.Now);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Solution Explorer - Extensibility

Search Solution Explorer - Extensibility

Solution 'Extensibility' (1 project)

Extensibility

Properties

References

App.config

Program.cs

```
Extensibility ILogger
namespace Extensibility
{
    public interface ILogger
    {
        void LogError(string message);
        void LogInfo(string message);
    }
}
```

Search Solution Explorer - Extensibility (C#)

- Solution 'Extensibility' (1 project)
- Extensibility
 - Properties
 - References
 - App.config
 - ILogger.cs
 - Program.cs

```
public class DbMigrator
{
    private readonly ILogger _logger;

    public DbMigrator(ILogger logger)
    {
        _logger = logger;
    }

    public void Migrate()
    {
        _logger.LogInfo("Migrationg started at {0}" + DateTime.Now);

        // Details of migrating the database

        _logger.LogInfo("Migrationg finished at {0}" + DateTime.Now);
    }
}
```

- References
- App.config
- DbMigrator.cs
- ILogger.cs
- Program.cs

C# Intermediate



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular


```
Test Explorer
public class ConsoleLogger : ILogger
{
    public void LogError(string message)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine(message);
    }

    public void LogInfo(string message)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine(message);
    }
}
```

References

- App.config
- ConsoleLogger.cs
- DbMigrator.cs
- ILogger.cs
- Program.cs

```
Test Explorer
namespace Extensibility
{
    class Program
    {
        static void Main(string[] args)
        {
            var dbMigrator = new DbMigrator(new ConsoleLogger());
            dbMigrator.Migrate();
        }
    }
}
```

Search Solution Explorer - Extensibility (C)

Solution 'Extensibility' (1 project)

- Extensibility
 - Properties
 - References
 - App.config
 - ConsoleLogger.cs
 - DbMigrator.cs

C:\Windows\system32\cmd.exe

```
Migration started at (0)9/02/2015 6:56:26 PM
Migration finished at (0)9/02/2015 6:56:26 PM
Press any key to continue . . .
```




```
namespace Extensibility
{
    public class FileLogger : ILogger
    {
        private readonly string _path;

        public FileLogger(string path)
        {
            _path = path;
        }

        public void LogError(string message)
        {
            using (var streamWriter = new StreamWriter(_path, true))
            {
                streamWriter.WriteLine("ERROR: " + message);
            }
        }

        public void LogInfo(string message)
        {
            using (var streamWriter = new StreamWriter(_path, true))
            {
                streamWriter.WriteLine("ERROR: " + message);
            }
        }
    }
}
```

Search Solution Explorer - Extensibility (Ctrl+)

Solution 'Extensibility' (1 project)

- Extensibility
 - Properties
 - References
 - App.config
 - ConsoleLogger.cs
 - DbMigrator.cs
 - ILogger.cs
 - Program.cs



```
using System.IO;
```

```
namespace Extensibility
```

```
{
```

```
    public class FileLogger : ILogger
```

```
    {
```

```
        private readonly string _path;
```

```
        public FileLogger(string path)
```

```
        {
```

```
            _path = path;
```

```
        }
```

```
        public void LogError(string message)
```

```
        {
```

```
            Log(message, "ERROR");
```

```
        }
```

```
        public void LogInfo(string message)
```

```
        {
```

```
            Log(message, "INFO");
```

```
        }
```

```
        private void Log(string message, string messageType)
```

```
        {
```

```
            using (var streamWriter = new StreamWriter(_path, true))
```

```
            {
```

```
                streamWriter.WriteLine(messageType + ": " + message);
```

```
            }
```

```
        }
```

```
    } FileLogger
```

Solution 'Extensibility' (1 project)

Extensibility

Properties

References

App.config

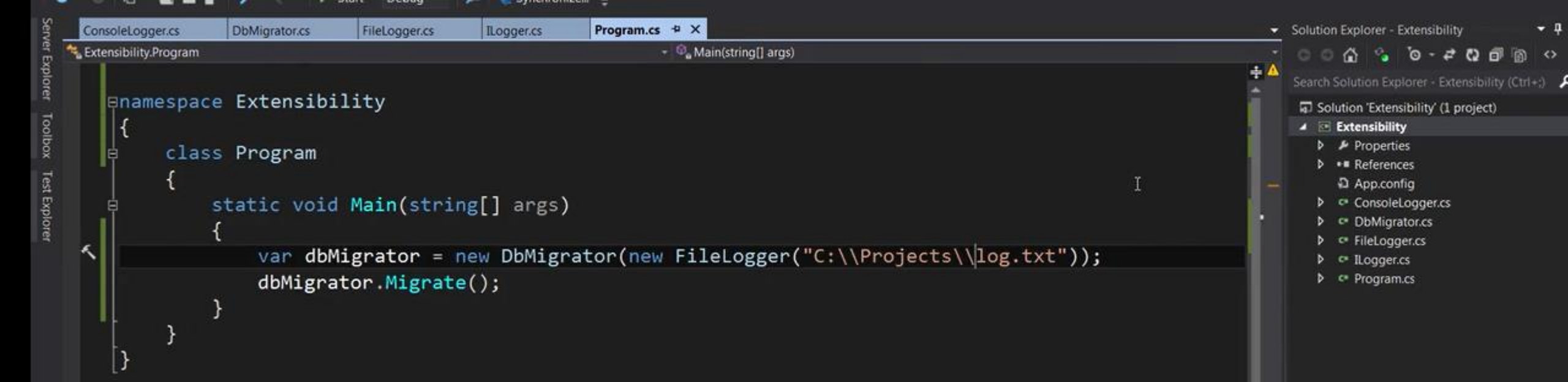
ConsoleLogger.cs

DbMigrator.cs

FileLogger.cs

ILogger.cs

Program.cs



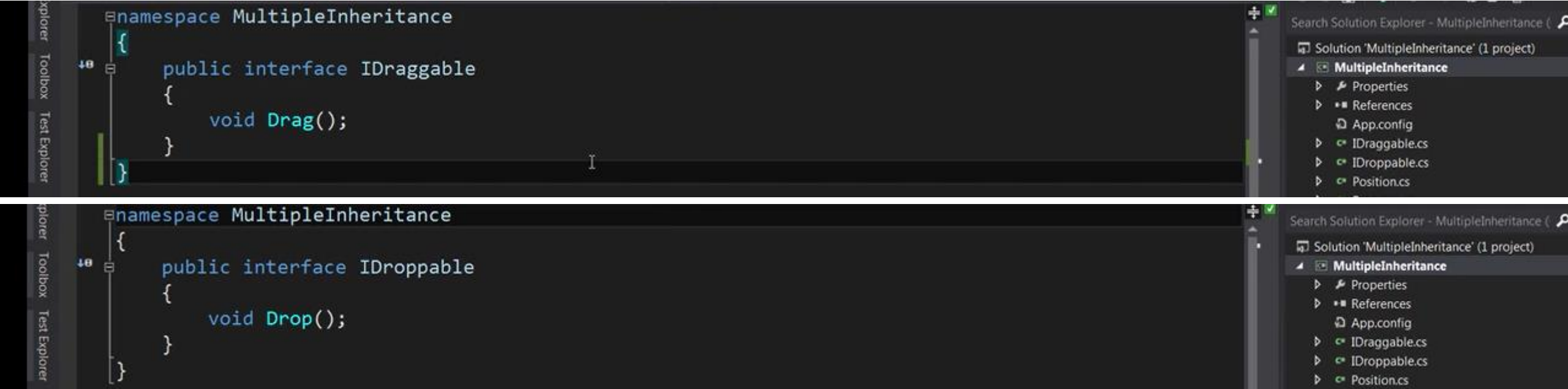
Interfaces and Inheritance

Interfaces are NOT for Multiple Inheritance.



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular




```
est Explorer

public class UIControl
{
    public string Id { get; set; }
    public Size Size { get; set; }
    public Position TopLeft { get; set; }

    public virtual void Draw()
    {
    }

    public void Focus()
    {
        Console.WriteLine("Received focus.");
    }
}
```

App.config
IDraggable.cs
IDroppable.cs
Position.cs
Program.cs
Size.cs
UiControl.cs

```
er Toolbox Test Explorer

public class TextBox : UIControl, IDraggable, IDroppable
{
    public void Drag()
    {
        throw new NotImplementedException();
    }

    public void Drop()
    {
        throw new NotImplementedException();
    }
}
```

Search Solution Explorer - MultipleInheritance (1 project)
MultipleInheritance
Properties
References
App.config
IDraggable.cs
IDroppable.cs
Position.cs
Program.cs
Size.cs
UiControl.cs



Interfaces and Polymorphism



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

```
namespace Polymorphism
```

```
{
```

```
    public class VideoEncoder
```

```
    {
```

```
        private readonly MailService _mailService;
```

```
        public VideoEncoder()
```

```
        {
```

```
            _mailService = new MailService();
```

```
        }
```

```
        public void Encode(Video video)
```

```
        {
```

```
            (method) void Polymorphism.VideoEncoder.Encode(Video video)
```

```
            // Video encoding logic
```

```
            // ...
```

```
            _mailService.Send(new Mail());
```

```
        }
```

```
    }
```

```
}
```

```
public class MailService
```

```
{
```

```
    public void Send(Mail mail)
```

```
    {
```

```
        Console.WriteLine("Sending email...");
```

```
    }
```

```
}
```

Search Solution Explorer - Polymorphism (

Solution 'Polymorphism' (1 project)

Polymorphism

Properties

References

App.config

Mail.cs

MailService.cs

Program.cs

Video.cs

VideoEncoder.cs

References

App.config

Mail.cs

MailService.cs

Program.cs

Video.cs

VideoEncoder.cs



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular


```
namespace Polymorphism
{
    public interface INotificationChannel
    {
        void Send(Message message);
    }
}

public class MailNotificationChannel : INotificationChannel
{
    public void Send(Message message)
    {
        Console.WriteLine("Sending mail...");
    }
}

public class SmsNotificationChannel : INotificationChannel
{
    public void Send(Message message)
    {
        Console.WriteLine("Sending SMS...");
    }
}
```

Search Solution Explorer - Polymorphism

- Solution 'Polymorphism' (1 project)
- Polymorphism
 - Properties
 - References
 - App.config
 - INotificationChannel.cs
 - Mail.cs
 - MailService.cs
 - Message.cs
 - Program.cs
 - Video.cs
 - VideoEncoder.cs



Explorer
Toolbox
Test Explorer

Polymorphism.VideoEncoder

RegisterNotificationChannel(INotificationChannel channel)

```
namespace Polymorphism
{
    public class VideoEncoder
    {
        private readonly IList<INotificationChannel> _notificationChannels;

        public VideoEncoder()
        {
            _notificationChannels = new List<INotificationChannel>();
        }

        public void Encode(Video video)
        {
            // Video encoding logic
            // ...

            foreach (var channel in _notificationChannels)
                channel.Send(new Message());
        }

        public void RegisterNotificationChannel(INotificationChannel channel)
        {
            _notificationChannels.Add(channel);
        }
    }
}
```

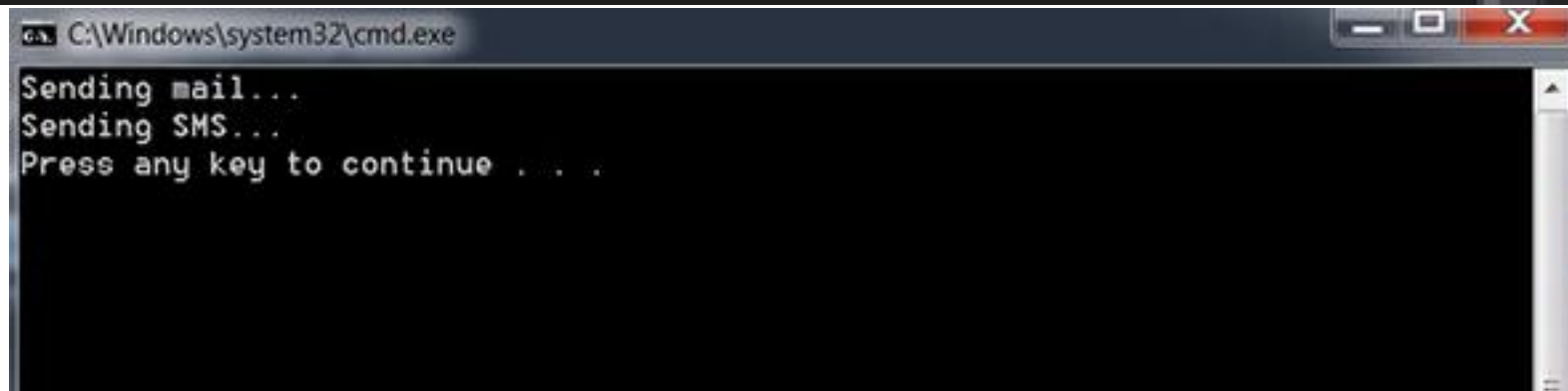
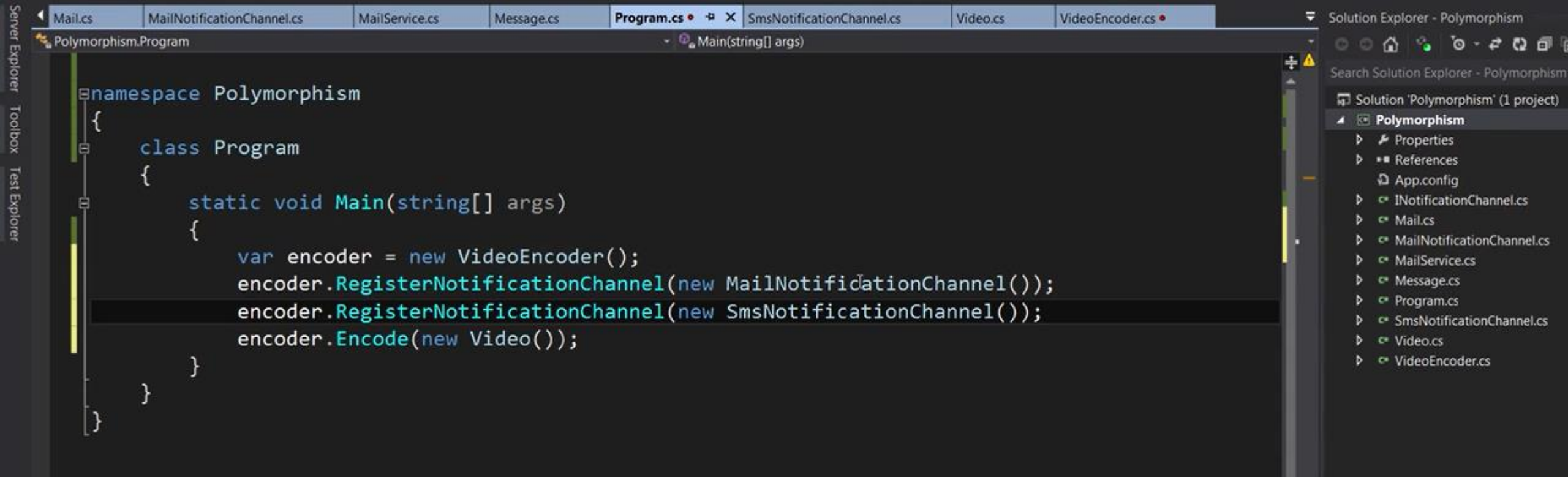
Search Solution Explorer - Polymorphism

Solution 'Polymorphism' (1 project)

Polymorphism

- Properties
- References
- App.config
- INotificationChannel.cs
- Mail.cs
- MailNotificationChannel.cs
- MailService.cs
- Message.cs
- Program.cs
- SmsNotificationChannel.cs
- Video.cs
- VideoEncoder.cs





Exercises



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular

Exercise: Design a workflow engine

Design a workflow engine that takes a workflow object and runs it. A workflow is a series of steps or activities. The workflow engine class should have one method called `Run()` that takes a workflow, and then iterates over each activity in the workflow and runs it.

We want our workflows to be extensible, so we can create new activities without impacting the existing activities.

Educational tip: we should represent the concept of an activity using an interface. Each activity should have a method called `Execute()`. The workflow engine does not care about the concrete implementation of activities. All it cares about is that these activities have a common interface: they provide a method called `Execute()`. The engine simply calls this method and this way it executes a series of activities in sequence.

The aim of this exercise is to help you understand how you can use interfaces to design extensible applications. You change the behaviour of your application by creating new classes, rather than changing the existing classes. You'll also see polymorphic behaviour of interfaces.

Real-world use case: in a real-world application you may use a workflow in a scenario like the following:

- 1- Upload a video to a cloud storage.
- 2- Call a web service provided by a third-party video encoding service to tell them you have a video ready for encoding.
- 3- Send an email to the owner of the video notifying them that the video started processing.
- 4- Change the status of the video record in the database to "Processing".

Each of these steps can be represented by an activity. For the purpose of this exercise, do not worry about these complexities. Simply use `Console.WriteLine()` in each of your activity classes. Your focus should be on sending a workflow to the workflow engine and having it run the workflow and all the activities inside it. We don't care about the actual activities.

Solution



Zouhair Rimale, Ph.D.

Expert technique .NET | SharePoint | ASP.NET MVC | WEB API | Angular


```
interface IWorkflow
{
    void Execute();
}

class VideoUploader : IWorkflow
{
    public void Execute()
    {
        Console.WriteLine("Uploading a video!");
    }
}

class CallWebService : IWorkflow
{
    public void Execute()
    {
        Console.WriteLine("Calling web service...");
    }
}

class SendEmail : IWorkflow
{
    public void Execute()
    {
        Console.WriteLine("Sending an email...");
    }
}

class ChangeStatus : IWorkflow
{
    public void Execute()
    {
        Console.WriteLine("Status changed...");
    }
}
```

```

class WorkflowEngine
{
    private List<IWorkflow> T;

    public WorkflowEngine()
    {
        T = new List<IWorkflow>();
    }

    public void AddWorkflowObject(IWorkflow iObject)
    {
        T.Add(iObject);
    }

    public void RemoveWorkflowObject(IWorkflow iObject)
    {
        T.Remove(iObject);
    }

    public void Run()
    {
        foreach (IWorkflow I in T)
        {
            I.Execute();
        }
    }
}

```




```
static void Main(string[] args)
{
    WorkflowEngine workFlow = new WorkflowEngine();
    workFlow.AddWorkflowObject(new VideoUploader());
    workFlow.AddWorkflowObject(new CallWebService());
    workFlow.AddWorkflowObject(new SendEmail());
    workFlow.AddWorkflowObject(new ChangeStatus());

    workFlow.Run();

    Console.ReadLine();
}
```

