# Test Doubles

## Inversión de Dependencia

public interface IDataAccess  
{  
    decimal GetShippingCosts(Order order);  
  
    void SaveOrder(Order order);  
  
    Order GetOrder(int id);  
}

private IDataAccess dataAccess;  
public OrderServices()  
{  
    this.dataAccess = new DataAccess();  
}

## Inyección de Dependencias

[TestClass]  
public class OrderServicesTests  
{  
    [TestMethod]  
    public void CalculateTotal\_OrderFromUS\_FreeShipping()  
    {  
 Order order = new Order { Country = "US", ItemTotal = 100 };  
        OrderServices orderServices = new OrderServices(new DataAccess());  
  
        var total = orderServices.CalculateTotal(order);  
  
        Assert.AreEqual(100, total);  
    }  
  
    [TestMethod]  
    public void CalculateTotal\_OrderOutsideUS\_ShippingCostIsAdded()  
    {  
 Order order = new Order { Country = "PER", ItemTotal = 100 };  
        OrderServices orderServices = new OrderServices(new DataAccess());  
  
        var total = orderServices.CalculateTotal(order);  
  
        Assert.AreEqual(110, total);  
    }  
  
    [TestMethod]  
    public void Save\_ValidOrder\_TheOrderIsPersisted()  
    {  
 Order order = new Order { Id = 1, Country = "PER", ItemTotal = 100, Total = 110 };  
        OrderServices orderProcessor = new OrderServices(new DataAccess());  
  
        orderProcessor.Save(order);  
  
        Order orderFromDb = orderProcessor.GetOrder(order.Id);  
        Assert.IsNotNull(orderFromDb);  
    }  
}

## Test Doubles Manuales

    [TestClass]  
    public class OrderServicesTests\_ManualTestDoubles  
    {  
        [TestMethod]  
        public void CalculateTotal\_OrderFromUS\_FreeShipping()  
        {  
            OrderServices orderServices = new OrderServices(null);  
            Order order = new Order { Country = "US", ItemTotal = 100 };  
   
            var total = orderServices.CalculateTotal(order);  
   
            Assert.AreEqual(100, total);  
        }  
   
        [TestMethod]  
        public void CalculateTotal\_OrderOutsideUS\_ShippingCostIsAdded()  
        {  
            Order order = new Order { Country = "PER", ItemTotal = 100 };  
            var dataAccess = new SimpleDataAccess();  
            dataAccess.ShippingCosts = 10;  
            OrderServices orderServices = new OrderServices(dataAccess);  
   
            var total = orderServices.CalculateTotal(order);  
   
            Assert.AreEqual(110, total);  
        }  
   
        [TestMethod]  
        public void Save\_ValidOrder\_TheOrderIsPersisted()  
        {  
            Order order = new Order { Id = 1, Country = "PER", ItemTotal = 100, Total = 110 };  
            var dataAccess = new SimpleDataAccess();  
            OrderServices orderProcessor = new OrderServices(dataAccess);  
   
            orderProcessor.Save(order);  
   
            Assert.AreEqual(order,dataAccess.OrderSaved);  
        }  
   
        private class SimpleDataAccess : IDataAccess  
        {  
            public decimal ShippingCosts;  
            public decimal GetShippingCosts(Order order)  
            {  
                return this.ShippingCosts;  
            }  
   
            public Order GetOrder(int id)  
            {  
                throw new **NotImplementedException**();  
            }  
   
            public Order OrderSaved;  
            public void SaveOrder(Order order)  
            {  
                OrderSaved = order;  
            }  
        }  
    }

## Test Double Framework

[TestClass]  
public class OrderServicesTests  
{  
    [TestMethod]  
    public void CalculateTotal\_OrderFromUS\_FreeShipping()  
    {  
        Order order = new Order { Country = "US", ItemTotal = 100 };  
        OrderServices orderServices = new OrderServices(null);  
  
        var total = orderServices.CalculateTotal(order);  
  
        Assert.AreEqual(100, total);  
    }  
  
    [TestMethod]  
    public void CalculateTotal\_OrderOutsideUS\_ShippingCostIsAdded()  
    {  
        Order order = new Order { Country = "PER", ItemTotal = 100 };  
        var dataAccess = new Mock<IDataAccess>();  
        dataAccess.Setup(x => x.GetShippingCosts(order)).Returns(10);  
        OrderServices orderServices = new OrderServices(dataAccess.Object);  
  
        var total = orderServices.CalculateTotal(order);  
  
        Assert.AreEqual(110, total);  
    }  
  
    [TestMethod]  
    public void Save\_ValidOrder\_TheOrderIsPersisted()  
    {  
        Order order = new Order { Id = 1, Country = "PER", ItemTotal = 100, Total = 110 };  
        var dataAccess = new Mock<IDataAccess>();  
        OrderServices orderProcessor = new OrderServices(dataAccess.Object);  
  
        orderProcessor.Save(order);  
  
        dataAccess.Verify(x => x.SaveOrder(order));  
    }  
}

## Mocking Frameworks

Si se encuentra antes o igual en el índice de la lista, es válido

    [TestClass]  
    public class LogManagerTests  
    {  
        private Mock<IConfiguration> configuration;  
        private Mock<IEmailSender> emailSender;  
        private Mock<IAppender> appender;  
        private LogManager logManager;  
   
        [TestInitialize]  
        public void Setup()  
        {  
            this.configuration = new Mock<IConfiguration>();  
            this.emailSender = new Mock<IEmailSender>();  
            this.appender = new Mock<IAppender>();  
            this.logManager = new LogManager(configuration.Object, emailSender.Object, appender.Object);  
        }  
   
        [TestMethod]  
        public void IsEnabled\_MessageLevelBeforeLoggerLevel\_ReturnTrue()  
        {  
            configuration.Setup(x => x.LoggerLevel()).Returns(Level.Info);  
   
            bool isEnabled = this.logManager.IsEnabled(Level.Error);  
   
            Assert.IsTrue(isEnabled);  
        }  
   
        [TestMethod]  
        public void IsEnabled\_MessageLevelAfterLoggerLevel\_ReturnFalse()  
        {  
            configuration.Setup(x => x.LoggerLevel()).Returns(Level.Info);              
   
            bool isEnabled = logManager.IsEnabled(Level.Debug);  
   
            Assert.IsFalse(isEnabled);  
        }  
   
        [TestMethod]  
        public void Write\_LevelError\_SendMailtoAdmin()  
        {  
            logManager.Write("message", Level.Error);  
   
            emailSender.Verify(x => x.SendToAdmin("message"));  
        }  
   
        [TestMethod]  
        public void Write\_IsEnabled\_WriteToAppender()  
        {  
            configuration.Setup(x => x.LoggerLevel()).Returns(Level.Info);  
   
            logManager.Write("message", Level.Info);  
   
            appender.Verify(x => x.Write("message"));  
        }  
    }