



Wzorce projektowe i testowanie automatyczne w C#

Szymon Szyllhaber

Copyright 2021 © Szymon Szyllhaber

Kim jestem ?

Programista (praktyk, pragmatyk) od .Net 1.0 beta (20 lat)

W tym momencie konsultant, trener, architekt
MVC, DDD, CQRS, Testowanie (TDD, BDD), Craftsmanship, REST

Szkolenia:

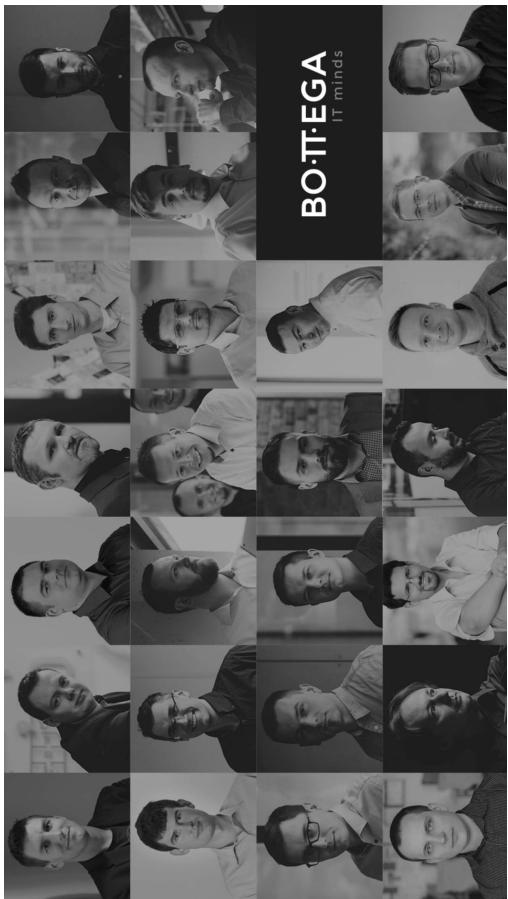
Wzorce projektowe, Wzorce aplikacyjne, Architektura aplikacji i systemów, Microservices, SOA, WCF, ASP

Prywatnie:

Rowerzysta, Kajakarz, Mąż i ojciec 2 szkółek

Copyright 2021 © Szymon Szylhaber

Kim my jesteśmy - Bottega IT Minds



Copyright 2021 © Szymon Szyłhabel

Daj się poznać?

- Imię ...
- Pracuję na stanowisku ...
- Moje doświadczenie ...
- Moja znajomość agendy ...
- Mój cel, dlaczego tu jestem ... ?

Agenda (Dzień 1)

Przypomnienie

- Podstawy programowania obiektowego
- Paradymat
- Drogowskazy
 - SOLID
 - YAGNI
 - KISS
 - DRY

Copyright 2021 © Szymon Szylhaber

Agenda (Dzień 1/2)

- Testowanie
 - Piramida Testów
 - Rodzaje testów
 - TDD
 - Testing Katta
 - Testy jednostkowe
 - Zaślepki (mocki)
 - Wykonywane specyfikacje

Copyright 2021 © Szymon Szyliabel

Agenda (Dzień 2/3)

- Wzorce architektoniczne
 - Architektura Hexagonalna
- Wzorce biznesowe
 - Strategia
 - Singleton
 - Fabryka
 - Dekorator
 - Łańcuch odpowiedzialności
 - Budowniczy
 - Stan (State)
 - Metoda szablonowa
 - Metoda Fabrykująca
 - Adapter
 - Value Object
 - Kompozyt
 - Specyfikacja

Copyright 2021 © Szymon Szylhaber

Agenda (Dzień 3)

- Wzorce infrastrukturalne
 - Repozytorium
 - Memento
 - Command
 - MVC / MVP / MVVM
 - Proxy
 - Observer
 - Service Locator
 - Interpreter
 - Iterator

Copyright 2021 © Szymon Szylhaber

Podstawy programowania obiektowego

Copyright 2021 © Szymon Szylhaber

Czy to przykład programowania obiektowego?

```
public class Order
{
    public Guid Id { get; set; }
    public string Number { get; set; }
    public string Title { get; set; }
    public List<OrderItem> OrderItems { get; set; }
}

public class OrderItem
{
    public Guid ProductId { get; set; }
    public int Count { get; set; }
    public decimal UnitPrice { get; set; }
}

public class OrderProcessor
{
    public decimal CalculateOrderPrice(Order order)
    {
        decimal result = 0;
        order.OrderItems.ForEach(f => result += f.Count * f.UnitPrice);
        return result;
    }

    public void AddProduct(Guid orderId, Guid productId)
    {
        Order o = _orderRepo.Load(orderId);
        o.OrderItems.Add(new OrderItem(productId, 1, 100));
    }
}
```

Copyright 2021 © Szymon Szylhaber

Programowanie OO != Programowanie z użyciem klas



Copyright 2021 © Szymon Szyliabel

Copyright 2021 © Szymon Szylhaber

PARADYGMAT

Abstrakcja



Copyright 2021 © Szymon Szyłhabel

Abstrakcja



Copyright 2021 © Szymon Szyłhabel



Enkapsulacija



```
human  
.DigestionSystem  
.Peritoneum  
.Stomach  
.Add(new Sausage(2));
```

```
human.Eat(new Sausage(2));  
public void Eat(Food f)  
{  
    if (! iLike(f))  
    {  

```

Copyright 2021 © Szymon Szylhaber

Enkapsulacja - Zadanie

Problem:

Pobieramy opłaty od klientów za wykonanie usług

- Sprawdzamy ilość środków na koncie
- Jeżeli ilość środków nie jest wystarczająca
- Udzielamy kredytu klientom VIP
- O ile nie przekroczyli limitu



Enkapsulacja - techniki

- Getery / Setery !!!
- Hollywood Principle

Copyright 2021 © Szymon Szyliabel

Enkapsulacja - techniki

- Getery / Setery !!!
- Hollywood Principle

Pytanie

Jak ktoś przychodzi do projektu i robi encje i od razu robi do pół getery i settery to kiedy może to być dramat?

Dziedziczenie - Polimorfizm



Copyright 2021 © Szymon Szylhaber

Drogowskazy

SOLID

YAGNI

KISS

DRY

SOLID

Copyright 2021 © Szymon Szylhaber

Ćwiczenie - co jest nie tak z tym kodem ?

```

public static void Main()
{
    string connectionstring = ConfigurationManager.ConnectionStrings["database"].ConnectionString;
    InitDB.RunConnectionString();
    using (var cli = new WebClient())
    {
        string data = cli.DownloadString("http://www.nbp.pl/kursy/xml/lastA.xml");
        using (IDbConnection connection = new SqlConnection(connectionString))
        {
            InEnumerable<Currency> oldcurrencies = connection.Query<Currency>("select name,price from Currency");
            XElement element = XElement.Load(new StringReader(data));
            InEnumerable<Currency> currencies = from nm in element.Elements("pozycja")
                                                    select new Currency(nm.Element("kod_waluty").Value, decimal.Parse(nm.Element("kurs_sredni").Value));
            using (TransactionScope ts = new TransactionScope(TransactionScopeOption.Required))
            {
                connection.Execute("delete from Currency");
                connection.Execute("insert into Currency(name,price) values(@Name,@Price)", currencies);
                ts.Complete();
            }
            foreach (var currency in oldcurrencies)
            {
                Currency newCurrency = currencies.FirstOrDefault(f => f.Name == currency.Name);
                if (newCurrency.Price != currency.Price)
                    Console.WriteLine($"{newCurrency.Name} - {newCurrency.Price - currency.Price}");
            }
        }
    }
}

```

Copyright 2021 © Szymon Szylhaber

Solid

Single Responsibility Principle

Open Closed Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

Single Responsibility Principle



Single Responsibility Principle

*Just because you *can* doesn't mean you *should*.*

Copyright 2021 © Szymon Szyliabel

SRP - Dyskusja

Przykład

Modem powinien wysyłać komunikat pod wskazany adres i zwracać odpowiedź

Założenia

- Mamy dane biblioteki, które umożliwiają:
 - Nawiązanie połączenia – różne sposoby komunikacji (HTTP, FTP, UDP)
 - Szyfrowanie komunikacji lub bez szyfrowania
 - Zerwanie połączenia
 - Wysłanie wiadomości – różne sposoby kompresji lub bez

Copyright 2021 © Szymon Szylhaber

SRP - Dyskusja

```
// źródło  
public class Model  
{  
    public void Connect();  
    public string Send(string message);  
    public void Disconnect();  
}
```

Copyright 2021 © Szymon Szylhaber

SRP - Dyskusja

```
// Źle
public class Model
{
    public void Connect();
    public string Send(string address, string message);
    public void Disconnect();
}
```

```
// Dobrze
public class Model
{
    public string Send(string address, string message);
}
```

Copyright 2021 © Szymon Szylhaber

Open Closed Principle



Open-Closed Principle

Open-chest surgery isn't needed when putting on a coat.

Copyright 2021 © Szymon Szylhaber

OCP - Przykłady

Graphic Editor

Copyright 2021 © Szymon Szylhaber

OCP - Przykłady

ActionFilters

```
[Authorize]
public class ClientAddWizardController : Controller
{
    private readonly IClientCommandService _clientCommandService;
    public ClientAddWizardController(IClientCommandService clientCommandService)
    {
        _clientCommandService = clientCommandService;
    }

    [Transaction]
    [Log]
    public ActionResult Save(ClientViewModel client)
    {
        _clientCommandService.Save(client);
    }
    ...
}
```

Copyright 2021 © Szymon Szylhaber

OCP a enum'y

```
enum SerializationType
{
    Json,
    Xml,
    Yaml
}

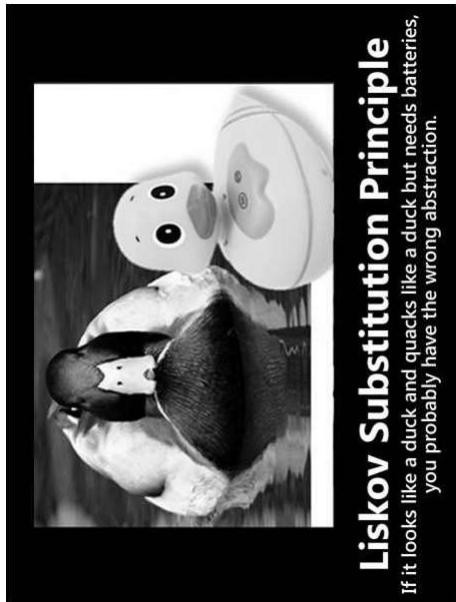
class Serializer
{
    public Serializer(SerializationType type)
    {
        ...
    }

    public string Serialize(byte[] data)
    {
        ...
    }
}
```

```
Serializer s = new Serializer(SerializationType.Json);
var result = s.Serialize(data);
```

Copyright 2021 © Szymon Szyliabel

Liskov Substitution Principle

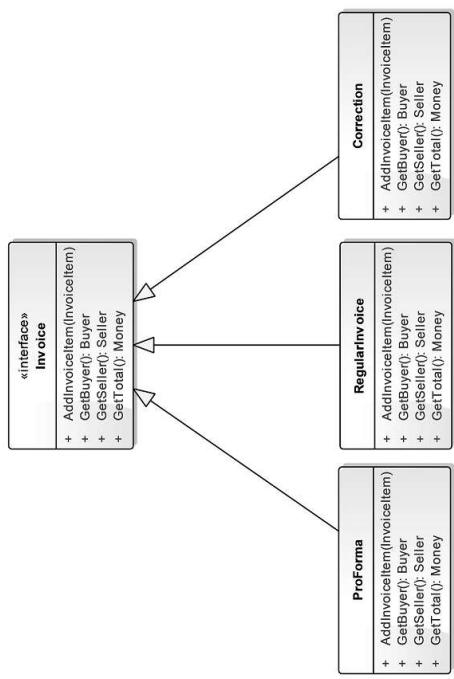


Liskov Substitution Principle

If it looks like a duck and quacks like a duck but needs batteries,
you probably have the wrong abstraction.

Copyright 2021 © Szymon Szylhaber

LSP - Przykład 1 - faktury



Copyright 2021 © Szymon Szylhaber

LSP - Przykład 2 - Net. Framework

Gdzie LSP jest tamane w Net Framework?

Copyright 2021 © Szymon Szylhaber

LSP - Przykład 2 - Net. Framework

Gdzie LSP jest tamane w Net Framework?

Array

- implementuje ICollection a przy Add() rzeka NotSupportedException
- IsReadOnly zwraca false

ReadOnly collections

Stream

- Seek/Read/Write
- daje chociaż możliwość sprawdzenia co dana klasa obsługuje
- CanRead, CanSeek, CanWrite...

Copyright 2021 © Szymon Szylhaber

LSP - Przykład 3 - PersistentSet

DEMO

Interface Segregation Principle



Copyright 2021 © Szymon Szylhaber

ISP - przykład 1 - Invoice

```
public interface IInvoice
{
    public Buyer Buyer { get; }
    public Seller Seller { get; }
    public void AddInvoiceItem(InvoiceItem item);
    public Money Total { get; }
}

public interface IBookableInvoice : IInvoice
{
    public void Book();
}
```

```
public class Proforma : IInvoice
{
    ...
}

public class RegularInvoice : IBookableInvoice
{
    ...
}

public class Correction : IBookableInvoice
{
    ...
}
```

Copyright 2021 © Szymon Szylhaber

ISP - przykład 2 - Repository

```
public interface IRepository
{
    void Insert(Document doc);
    void Remove(int docId);
    void Update(Document doc);
    void DeleteDatabase();
    void AddUserToDatabase(string userName);
}
```

Copyright 2021 © Szymon Szylhaber

ISP - przykład 2 - Repository

```
// Źródło
public interface IRepository
{
    void Insert(Document doc);
    void Remove(int docId);
    void Update(Document doc);
}
public interface IDatabaseAdministration
{
    void DeleteDatabase();
    void AddUserToDatabase(string userName);
}

// Dobrze
public interface IRepository
{
    void Insert(Document doc);
    void Remove(int docId);
    void Update(Document doc);
}
public interface IDatabaseAdministration
{
    void DeleteDatabase();
    void AddUserToDatabase(string userName);
}
```

Copyright 2021 © Szymon Szylhaber

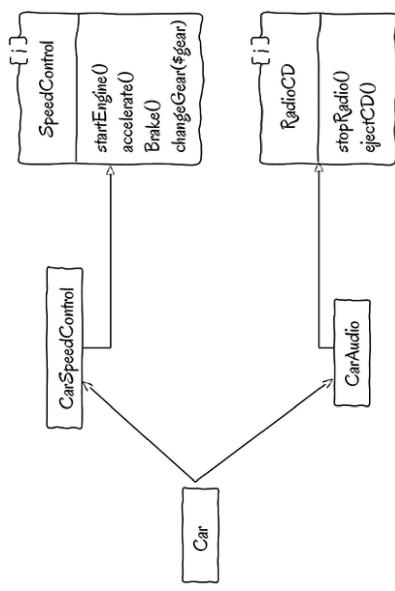
ISP - przykład 3 - Auto



„The interface-segregation principle (ISP) states that no client should be forced to depend on methods it does not use.”

Copyright 2021 © Szymon Szylhaber

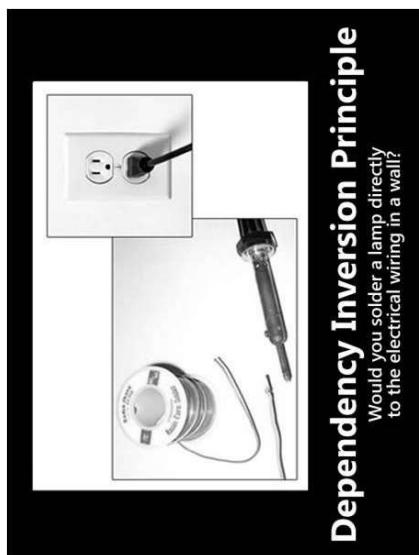
ISP - przykład 3 - Auto



„The interface-segregation principle (ISP) states that no client should be forced to depend on methods it does not use.”

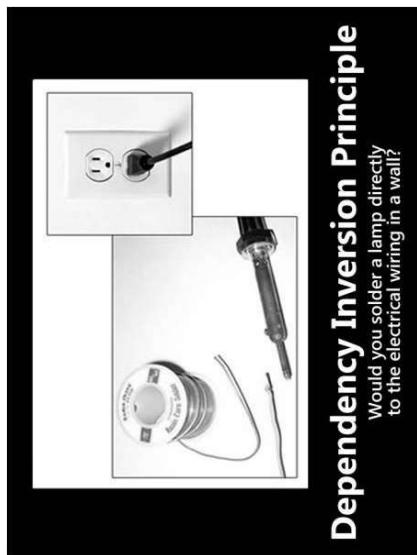
Copyright 2021 © Szymon Szylhaber

Dependency Inversion



Copyright 2021 © Szymon Szylhaber

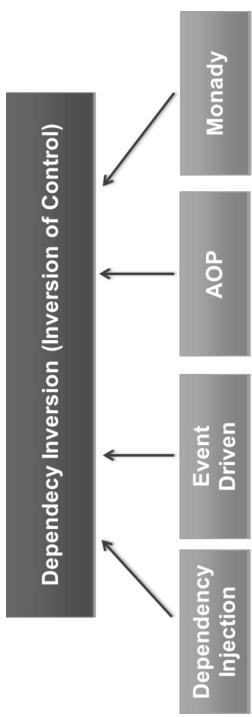
Dependency Inversion



Jakie są rodzaje "Dependency Inversion"?

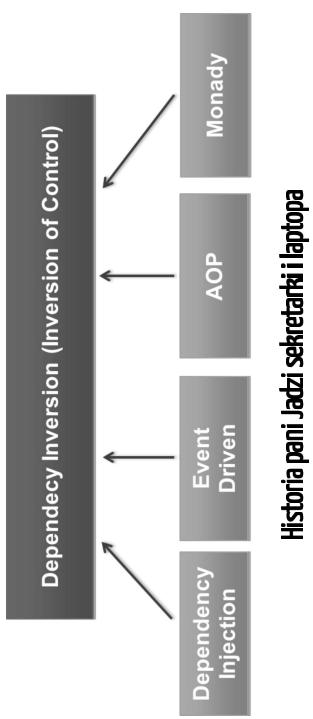
Copyright 2021 © Szymon Szylhaber

DIP - Rodzaje



Copyright 2021 © Szymon Szylhaber

DIP - Rodzaje



Historia pani Jadzi sekretarki i laptopa

Copyright 2021 © Szymon Szylhaber

DIP - dyskusja

Po co robimy dependency injection?

Copyright 2021 © Szymon Szylhaber

DIP - dyskusja

Po co robimy dependency injection?

Ćwiczenie

- Projektujemy moduł systemu ERP który oblicza wartość zamówienia biorąc pod uwagę promocje i status klienta.
- Chcemy dodać rabaty na wybrane produkty
- Lista produktów objętych rabatami znajduje się w pliku na zewnętrznym FTP
- Posiadamy dostęp do FTP
 - Plik zawiera kolejne pozycje w formacie EAN, rabat
 - Plik zawiera zdublowane EAN, obowiązuje ostatni wpis

Copyright 2021 © Szymon Szylhaber

DIP - przykład 1 - Dependency injection

Constructor injection

Method injection

Field/Property injection

```
public class CurrencyService
{
    private readonly IRepository<CurrencyCalculator> _repository;
    public ICurrencyCalculator Calculator { get; set; }
    public CurrencyService(IRepository<ICurrencyCalculator> repository)
    {
        _repository = repository;
    }

    public void SetExchangeRate(string code, decimal amount)
    {
        ...
        _repository.Save(code, amount);
        ...
    }

    public void ExportCurrencyRates(IExporter exporter)
    {
        IEnumerable<ExchangeRate> exchangeRates = _repository.LoadAll();
        foreach (var exchangeRate in exchangeRates)
        {
            exporter.Export(exchangeRate);
        }
    }
}
```

Copyright 2021 © Szymon Szylhaber

DIP - przykład 2 - zdarzenia

```
public class BadObservableList<T> : List<T>
{
    public new void Add(T obj)
    {
        base.Add(obj);
        Console.WriteLine("Added: " + obj);
    }

    ...
}

BadObservableList<int> list = new BadObservableList<int>();
list.Add(11);
```

Copyright 2021 © Szymon Szylhaber

DIP - przykład 2 - zdarzenia

```

public class BadObservableList<T> : List<T>
{
    public new void Add(T obj)
    {
        base.Add(obj);
        Console.WriteLine("Added: "+obj);
    }
}

BadObservableList<int> list = new BadObservableList<int>();
list.Add(11);
...

```

```

public class GoodObservableList<T> : List<T>
{
    public event CollectionChangedDelegate Changed =
        (sender, obj) => { };
    public new void Add(T obj)
    {
        base.Add(obj);
        Changed(this, obj);
    }
}

GoodObservableList<int> list = new GoodObservableList<int>();
list.Changed += (sender, obj) =>
{
    Console.WriteLine("Added: "+obj);
};

```

Copyright 2021 © Szymon Szylhaber

Aspekty (AOP)

Copyright 2021 © Szymon Szylhaber

Czym są zagadnienia przecinające (cross cutting concerns) ?

Czym są zagadnienia przecinające (cross cutting concerns) ?

- Logging
- Security (Authentication / Authorization)
- Auditing
- Validation
- Transaction

Przykład

```
class AuthorizationProvider
{
    public void LogIn()
    {
        Stopwatch stopwatch = Stopwatch.Start();
        // Logika biznesowa
        Debug.WriteLine(stopwatch.ElapsedMilliseconds);
    }
}
```

Copyright 2021 © Szymon Szylhaber

Jaki jest cel stosowania AOP?

Copyright 2021 © Szymon Szylhaber

Pojęcia

- Concern

Pojęcia

- Concern
- Jointpoint

Pojęcia

- Concern
- Jointpoint
- Weaving

Pojęcia

- Concern
- Jointpoint
- Weaving
- Aspect = jointpoint + concern

Copyright 2021 © Szymon Szyliabel

Style AOP

- 1. Wbudowane w Framework**
- 2. Interception**
- 3. IL Weaving**

Copyright 2021 © Szymon Szylhaber

1. Wbudowane w framework

ASP.NET MVC

```
[Authorize]
public class ClientAddWizardController : Controller
{
    private readonly IClientCommandService _clientCommandService;
    public ClientAddWizardController(IClientCommandService clientCommandService)
    {
        _clientCommandService = clientCommandService;
    }

    [Transaction]
    [Log]
    public ActionResult Save(ClientViewModel client)
    {
        _clientCommandService.Save(client);
    }
    ...
}
```

Copyright 2021 © Szymon Szylhaber

1. Wbudowane w framework

WCF

(Behaviors)

```
public class Service1 : IService1
{
    [UserAccess("Residents")]
    public string GetData(int value)
    {
        return string.Format("You entered: {0}", value);
    }

    [UserAccess("Admin")]
    public string GetDataUsingDataContract(int value)
    {
        return string.Format("You entered: {0}", value);
    }
}
```

Copyright 2021 © Szymon Szylhaber

2. Interception

```
public class Repository : IRepository
{
    public void Add(Item item)
    {
        ...
    }

    public class LoggingDecorator : IRepository
    {
        private readonly IRepository _repository;
        public LoggingDecorator(IRepository repository)
        {
            _repository = repository;
        }

        public void Add(Item item)
        {
            Stopwatch sw = Stopwatch.StartNew();
            _repository.Add(item);
            _logger.Debug($"Duration: {sw.Elapsed}s");
        }
    }

    IRepository repo = new LoggingDecorator(new Repository());
    repo.Add(item);
}
```

Copyright 2021 © Szymon Szylhaber

2. Interception

```

public class Repository : IRepository
{
    public void Add(Item item)
    {
        ...
    }
}

public class LoggingDecorator
{
    private readonly object _decorated;
    public LoggingDecorator(object decorated)
    {
        _decorated = decorated;
    }

    public void Invoke(string methodName, obj[] args)
    {
        Stopwatch sw = Stopwatch.StartNew();
        _decorated.Invoke(methodName, args);
        _logger.Debug($"Duration: {sw.Elapsed}s");
    }
}

IRepository repo = new LoggingDecorator(new Repository());
repo.Add(item);

```

- Istnieje już w implementacjach kontenerów IoC (CastleWindsor)
- Silnie polega na wzorcu dekoratora
- Brak silnego typowania
- Operacja na poziomie uruchomienia programu (Run-Time)
- Łatwe do zaimplementowania w skali całego programu
- Łatwe do zaimplementowania jeżeli już używasz kontenerów IoC

Copyright 2021 © Szymon Szyłhabel

2. Interception

```

public class Repository : IRepository
{
    public void Add(Item item)
    {
        ...
    }
}

public class LoggingDecorator
{
    private readonly object _decorated;
    public LoggingDecorator(object decorated)
    {
        _decorated = decorated;
    }

    public void Invoke(string methodName, obj[] args)
    {
        Stopwatch sw = Stopwatch.StartNew();
        _decorated.Invoke(methodName, args);
        _logger.Debug($"Duration: {sw.Elapsed}s");
    }
}

IRepository repo = new LoggingDecorator(new Repository());
repo.Add(item);

```

- Istnieje już w implementacjach kontenerów IoC (CastleWindsor)
- Silnie polega na wzorcu dekoratora
- Brak silnego typowania
- Operacja na poziomie uruchomienia programu (Run-Time)
- Łatwe do zaimplementowania w skali całego programu
- Łatwe do zaimplementowania jeżeli już używasz kontenerów IoC

DEMO

Copyright 2021 © Szymon Szyłhabel

3. LL Waving

- Aplikowane po komplikacji
- Straszne dla ludzi :)
- Może być zaaplikowane w dowolne miejsce
- Nie wymusza zmian w istniejącym kodzie
- Łatwe do dodania do istniejącego kodu

Copyright 2021 © Szymon Szylhaber

3. LL Waving

- Aplikowane po komplikacji
- Straszne dla ludzi :)
- Może być zaaplikowane w dowolne miejsce
- Nie wymusza zmian w istniejącym kodzie
- Łatwe do dodania do istniejącego kodu

Przykład

Copyright 2021 © Szymon Szylhaber

W jakiej sytuacji jesteś?

Nowy projekt



Istniejący projekt



Copyright 2021 © Szymon Szylhaber

Narzędzia

Interception

- Castle Windsor
- StructureMap
- Ninject (Ninject.Extensions)
- AutoFac ([AutoFac.Extras](#))

IL Weaving

- PostSharp
- LinFu
- Fody
- Fody.addins

RealProxy in .Net Core

- [DispatchProxy](#).

Copyright 2021 © Szymon Szylhaber

Copyright 2021 © Szymon Szylhaber

Monada

([Short version](#))

[long version](#)

Monada

```

public class Wrapper<T>
{
    readonly T _value

    public Wrapper(T value) // unit function that converts raw value into Wrapper value
    {
        _value = value
    }

    public static implicit operator Wrapper<T>(T value)
    {
        return new Wrapper<T>(value);
    }

    public override string ToString()
    {
        return _value.ToString();
    }

    public Wrapper<U> Bind<U>(Func<T,U> func)
    {
        return (Wrapper<U>) func(_value);
    }
}

Wrapper<int> x = 3;
var y = x.Bind(n => n * 2)
      .Bind(n => n + 1)
      .Bind(n => n * Math.Sqrt(n))

```

Copyright 2021 © Szymon Szylhaber

Przykłady monad w C#?

Copyright 2021 © Szymon Szylhaber

Przykłady monad w C#?

- Nullable
- Task
- IEnumerable
- Observable
- Reactive extensions - monadyczna wersja zdarzeń

Copyright 2021 © Szymon Szylhaber

Przykłady monad w C# ?

- Nullable
- Task
- IEnumerable
- Observable
- Reactive extensions - monadyczna wersja zdarzeń

```
Task<string> futureHtml = new WebClient().DownloadStringTaskAsynch("www.google.com")
Console.WriteLine( futureHtml.ContinueWith(f=>f.Result.Length) )
```

Copyright 2021 © Szymon Szylhaber

Przykłady monad w C# ?

- Nullable
- Task
- IEnumerable
- Observable
- Reactive extensions - monadyczna wersja zdarzeń

```
Task<string> futureHtml = new WebClient().DownloadStringTaskAsynch("www.google.com")  
Console.WriteLine( futureHtml.ContinueWith(f=>f.Result.Length) )
```

```
Console.WriteLine( await futureHtml ).Length )
```

Copyright 2021 © Szymon Szylhaber

Dependency Inversion - Podsumowanie

- Dependency Injection
 - Constructor
 - Property
 - Method
- Zdarzenia
- AOP
 - Wbudowane w framework (ASP MVC, WCF)
 - Interception
 - IL Waving
- Monady

Copyright 2021 © Szymon Szylhaber



Zadanie: Refaktor zgodnie z zasadami solid

- Odpowiedzialności
- Gotowość na rozbudowę
- Odwrócenie zależności
 - Dependency injection
 - Oddzielne elementów infrastrukturalnych (nie chcemy duplikacji kodu)

Pytania kontrolne

- Czy nie ma duplikacji kodu (DRY)
- Czy ma jedną odpowiedzialność
- Czy szansa zmiany wszystkich elementów w klasie jest taka sama
- Czy klasa zależy od elementów które zmieniają się (rzadziej niż dana klasa)

Copyright 2021 © Szymon Szylhaber



SOLID - czas na zmiany

- Zmienia się ścieżka do pliku
- Zmienia sięConnectionString
- Zmienia się sposób pobierania pliku
- Zmienia się format zapisu statystyk
- Zmienia się sposób zliczania statystyk

Copyright 2021 © Szymon Szyliabel

Drogowskazy

SOLID

YAGNI

KISS

DRY