Listing 4.11. ENCJA używająca WSTRZYKIWANIA METODĄ

```
public class Customer
{
    public Guid Id { get; private set; }
    public string Name { get; private set; }

    public Customer(Guid id, string name)
    {
        ...
    }
    public void RedeemVoucher(
        Voucher voucher,
        IVoucherRedemptionService service)
    {
        if (voucher == null)
            throw new ArgumentNullException("voucher");
        if (service == null)
            throw new ArgumentNullException("service");

        service.ApplyRedemptionForCustomer(
            voucher,
            this.Id);
    }

    public void MakePreferred(IEventHandler handler)
    {
        if (handler == null)
            throw new ArgumentNullException("handler");

        handler.Publish(new CustomerMadePreferred(this.Id));
    }
}
```

Używając WSTRZYKIWANIA METODĄ, obydwie metody domeny ENCJI, RedeemVoucher i MakePreferred, akceptują wymagane ZALEŻNOŚCI – IVoucherRedemptionService i IEventHandler. Weryfikują parametry i używają dostarczonej ZALEŻNOŚCI

Wewnątrz komponentu `CustomerServices` metoda `RedeemVoucher` należąca do `Customer` może być wywołana przez przekazywanie ZALEŻNOŚCI `IVoucherRedemtionService` wraz z wywołaniem, tak jak pokazano to poniżej.

Listing 4.12. Komponent używający WSTRZYKIWANIA METODĄ do przekazywania ZALEŻNOŚCI

```
public class CustomerServices : ICustomerServices
{
    private readonly ICustomerRepository repository;
    private readonly IVoucherRedemptionService service;

    public CustomerServices(
        ICustomerRepository repository,
        IVoucherRedemptionService service)
    {
```

Klasa CustomerServices używa WSTRZYKIWANIA KONSTRUKTOREM, żeby statystycznie zdefiniować jej wymagane ZALEŻNOŚCI. IVoucherRedemptionService jest jedną z tych ZALEŻNOŚCI

```
        this.repository = repository;
        this.service = service;
    }
    public void RedeemVoucher(
        Guid customerId, Voucher voucher)
    {
        var customer =
            this.repository.GetById(customerId);

        customer.RedeemVoucher(voucher, this.service);

        this.repository.Save(customer);
    }
}
```

ZALEŻNOŚĆ IVoucherRedemptionService jest przekazana do już skonstruowanej ENCJI Customer przy użyciu WSTRZYKIWANIA METODĄ. Customer jest stworzona wewnątrz implementacji ICustomerRepository

Na listingu 4.12 tylko pojedyncza instancja `Customer` jest pobierana z `ICustomerRepository`. Ale pojedyncza instancja `CustomerServices` może być wywoływana w kółko dla wielu różnych `consumer` i `voucher`, powodując, że ten sam `IVoucherRedepmtionService` będzie dostarczany do wielu różnych instancji `Customer`. `Customer` jest klasą konsumującą ZALEŻNOŚĆ `IVoucherRedemptionService`, która nie zmienia się dla różnych instancji klasy `Consumer`.

Jest to podobne do pierwszego przykładu WSTRZYKIWANIA METODĄ pokazanego na listingu 4.9 i metody `ApplyDiscountFor` omówionej na listingu 3.8. Przypadek odwrotny zdarza się wtedy, gdy różnicuje się ZALEŻNOŚCI przy niezmieniających się konsumentach.

PRZYKŁAD: RÓŻNICOWANIE WSTRZYKNIĘTEJ ZALEŻNOŚCI PRZY KAŻDYM WYWOŁANIU METODY

Wyobraźmy sobie system rozszerzeń dla graficznej aplikacji do rysowania, gdzie każdy mógłby podłączyć własne efekty przetwarzania obrazu. Zewnętrzne efekty przetwarzania obrazu mogą wymagać informacji na temat kontekstu czasu wykonania, które mogą być przekazane przez aplikację do rozszerzenia odpowiadającego za efekt. Jest to typowy przypadek użycia (dla zastosowania) WSTRZYKIWANIA METODĄ. Możemy zdefiniować następujący interfejs w celu zastosowania tych efektów:

ABSTRAKCJA rozszerzenia reprezentuje efekty przetwarzania obrazu. Efekty obrazu mogą być podłączone do aplikacji przez implementację tej ABSTRAKCJI

```
public interface IImageEffectAddIn
{
   Bitmap Apply(
       Bitmap source,
       IApplicationContext context);
}
```

Pozwala rozszerzeniu na zaaplikowanie jego efektu do źródła i następnie zwraca nowy obiekt Bitmap z zastosowanym efektem

Zapewnia informację kontekstową do efektu przetwarzania obrazu przez aplikację graficzną przy użyciu WSTRZYKIWANIA METODĄ

ZALEŻNOŚĆ `IApplicationContext` z `IImageEffectAddIn` może różnić się w każdym wywołaniu metody `Apply`, dostarczając rozszerzeniu informacji na temat kontekstu wywołania efektu. Jakakolwiek klasa implementująca ten interfejs może być użyta jako dodatek. Niektóre implementacje będą się przejmować zmienną `context`, a inne w ogóle.

Klient może użyć listy rozszerzeń przez wywołanie każdego z nich ze źródłem `Bitmap` i kontekstem do wracającego zagregowanego wyniku, tak jak pokazano to poniżej.

Listing 4.13. Przykładowe rozszerzenia klienta

```
public Bitmap ApplyEffects(Bitmap source)
{
    if (source == null) throw new ArgumentNullException("source");

    Bitmap result = source;
    foreach (IImageEffectAddIn effect in this.effects)
{
        result = effect.Apply(result, this.context);
    }

    return result;
}
```

Prywatne pole `effects` jest listą instancji `IImageEffectAddIn`, które pozwala klientowi przejść przez listę w celu wywołania metody `Apply` na każdym z rozszerzeń. Za każdym razem, gdy metoda `Apply` jest wywołana na rozszerzeniu, kontekst działania, reprezentowany przez pole `context`, jest przekazywany jako parametr metody:

```
result = effect.Apply(result, this.context);
```

Czasami wartość i kontekst działania są zawarte w pojedynczej ABSTRAKCJI, która działa jako kombinacja tych dwóch. Ważną kwestią, o której warto pamiętać, jest to, że tak jak widoczne to było na obu przykładach, ZALEŻNOŚĆ wstrzykiwana przez WSTRZYKIWANIE METODĄ staje się częścią definicji ABSTRAKCJI. Jest to zazwyczaj pożądane, w przypadku kiedy ZALEŻNOŚĆ zawiera informację czasu wykonania dostarczaną przez jej bezpośrednie wywołania.

W przypadkach gdy ZALEŻNOŚĆ jest szczegółem implementacji dla wywołania, powinno się chronić ABSTRAKCJĘ przed byciem „zanieczyszczoną"; dlatego też WSTRZYKIWANIE KONSTRUKTOREM jest lepszym wyborem. W innych przypadkach łatwo można skończyć na przekazywaniu ZALEŻNOŚCI z góry grafu obiektów aplikacji na sam dół, powodując przy tym szeroko zakrojone zmiany.

Wcześniejsze przykłady pokazały użycie WSTRZYKIWANIA METODĄ spoza PODSTAWY KOMPOZYCJI. Było to celowe. WSTRZYKIWANIE METODĄ nie jest odpowiednie, kiedy używa się go wewnątrz PODSTAWY KOMPOZYCJI. Wewnątrz KOMPOZYCJI WSTRZYKIWANIE METODĄ może inicjować wcześniej skonstruowaną klasę wraz z jej ZALEŻNOŚCIAMI. Jednak zrobienie tego w taki sposób prowadzić może do CZASOWEGO WIĄZANIA i z tego powodu jest to silnie odradzane.