Implementacja DDD .NET - Zadanie 3

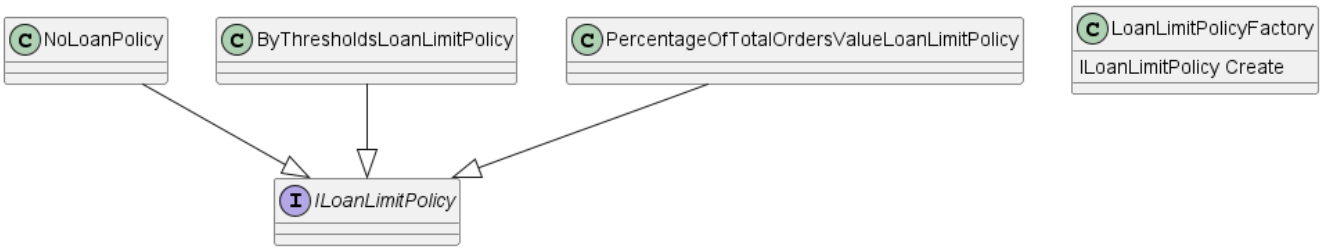
Bottega IT Minds

1. Zadanie 3 - Fabryki i Polityki

1.1. Wprowadzenie

W przypadku kiedy chcemy odseparować aspekty stabilne od niestabilnych i mieć możliwość "dostrajania" modelu domenowego na podstawie konifguracji możemy skorzystać z tzw. *Polityk* (wzorzec *Strategii*).

W module *Scoring* zostały zaimplementowane różne *Polityki* wyliczania limitu kredytowego (*loan limit*) oraz odpowiednia *Fabryka*, która decyduje, którą *Politykę* należy zastosować.



Przykład polityki, która liczy limit kredytowy jako procent od wartości wszystkich zamówień:

```
public class PercentageOfTotalOrdersValueLoanLimitPolicy : ILoanLimitPolicy
{
    private readonly Percentage _percentage;

    private readonly List<OrderData> _orders;

    public PercentageOfTotalOrdersValueLoanLimitPolicy(Percentage percentage,
List<OrderData> orders)
    {
        _percentage = percentage;
        _orders = orders;
    }

    public Money Calculate()
    {
        var allOrdersValue = _orders.Select(x => x.Value).Sum();

        return _percentage * allOrdersValue;
    }
}
```

Każda z tych polityk oraz fabryka została przetestowana jednostkowo.

Okazuje się, że w module *Sales* podczas składania zamówień też dobrze by było mieć większą kontrolę nad opcjami rabatowania.

1.2. Treść Zadania

Na bazie implementacji polityk wyznaczania limitu kredytu w module *Scoring* zaimplementuj polityki wyznaczania rabatu podczas składania zamówienia w module *Sales*.

Zamówienie składa się z listy Pozycji na Zamówieniu. Pozycja na Zamówieniu reprezentowana jest przez rekord `OrderLine`:

```
public record OrderLine(Guid ProductId, int Quantity, Money UnitPrice);
```

Zaimplementuj następujące polityki:

1. Brak rabatu
2. Procent od całej wartości zamówienia (procent jako parametr)
3. Rabat na kolejną sztukę tego samego produktu np. 50% na drugi ten sam produkt (na który ten sam produkt i ile rabatu jako 2 parametry).

Przetestuj każdą z tych polityk jednostkowo.

Zaimplementuj fabrykę, która wyznaczy odpowiednią politykę na podstawie konfiguracji.

Zaimplementuj metodę `Create` encji `Order`, wykorzystując wzorzec strategii wyznaczając wartość przed rabatem, wartość rabatu i wartość po rabacie.

```
public class Order : Entity
{
    private Order(Guid customerId, Money discount, Money beforeDiscountValue, Money
afterDiscountValue)
    {
        CustomerId = customerId;
        _discount = discount;
        _beforeDiscountValue = beforeDiscountValue;
        _afterDiscountValue = afterDiscountValue;

        this.AddDomainEvent(new OrderCreatedDomainEvent(
            CustomerId,
            _beforeDiscountValue.Amount,
            _discount.Amount,
            _afterDiscountValue.Amount));
    }

    public Guid CustomerId { get; }

    private Money _discount;

    private Money _beforeDiscountValue;

    private Money _afterDiscountValue;
```

```
    // public static Order Create(...)
    // {
    //     ...
    // }
}
```