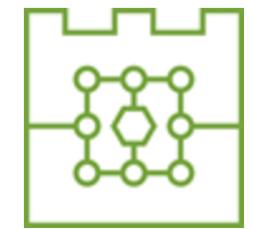


# **Wzorzec projektowy Strategy**

Podmiana algorytmu w trakcie  
wykonywania programu

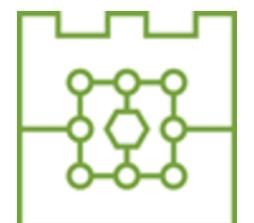
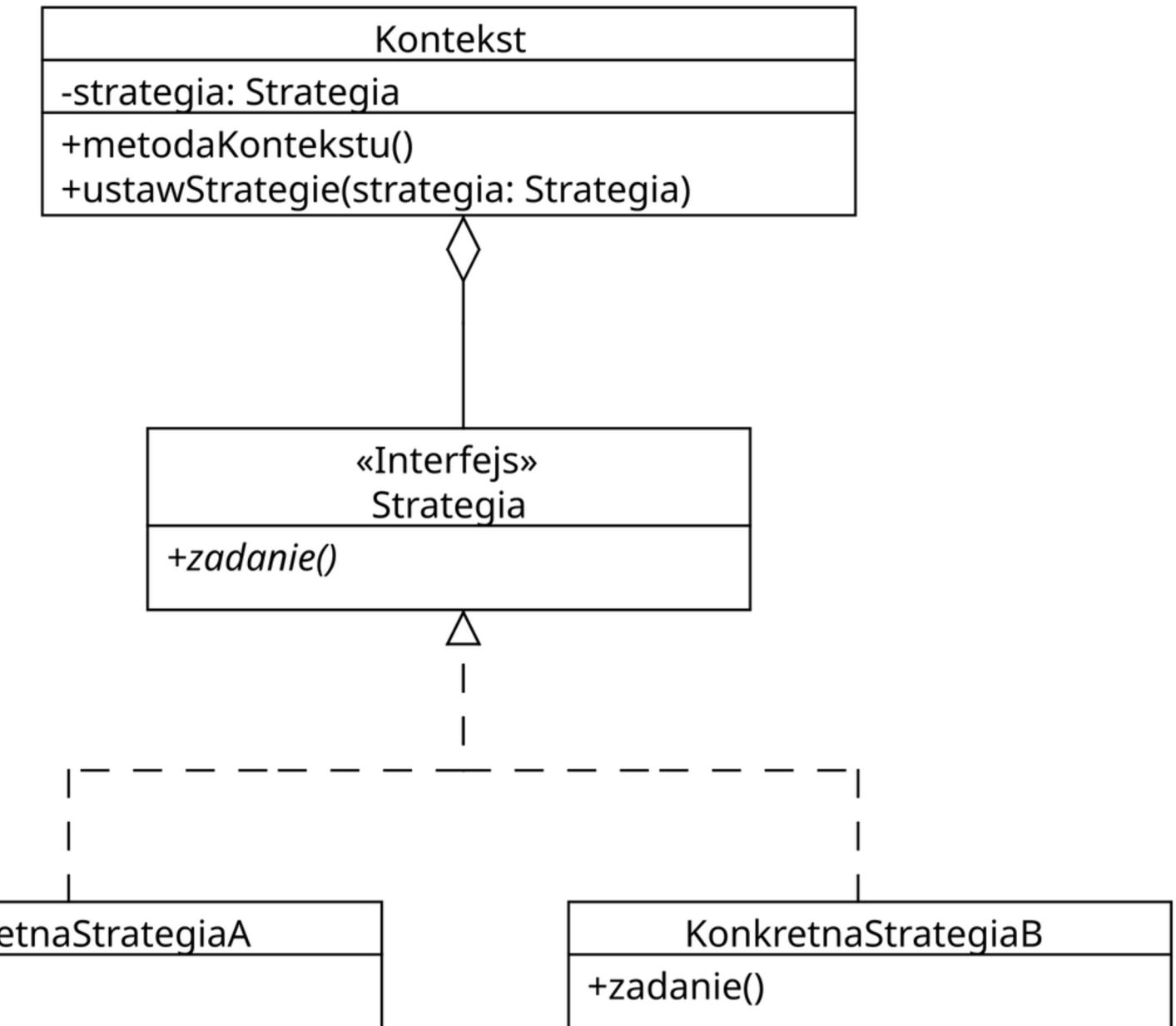


Szymon Ptasznik



# Zasady działania

- istnieje wspólny interfejs dla każdej ze strategii.
- istnieje klient, który jest użytkownikiem strategii i posiada do nich referencje.
- konkretne strategie są ustawiane przez klienta używając klasy kontekstu.
- klient komunikuje się i nadzoruje działanie strategii.



# Przykład - program do formatowania tekstu

```
public class JsonFormatter : ITextFormatter
{
    public void Format(string text)
    {
        Console.WriteLine("formatting in JSON");
    }
}

public class XmlFormatter : ITextFormatter
{
    public void Format(string text)
    {
        Console.WriteLine("formatting in XML");
    }
}

public class CsvFormatter : ITextFormatter
{
    public void Format(string text)
    {
        Console.WriteLine("formatting in CSV");
    }
}
```

implementacje konkretnych strategii



```
public interface ITextFormatter
{
    void Format(string text);
}
```

interfejs strategii

```
public class TextEditor
{
    private ITextFormatter _formatter;

    public TextEditor(ITextFormatter formatter)
    {
        _formatter = formatter;
    }

    public void SetFormatter(ITextFormatter formatter)
    {
        _formatter = formatter;
    }

    public void PublishText(string text)
    {
        _formatter.Format(text);
    }
}
```

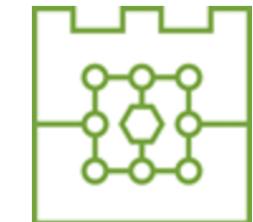
klasa kontekstu

```
public class Program
{
    public static void Main(string[] args)
    {
        var editor = new TextEditor(new JsonFormatter());
        editor.PublishText("Hello!");

        editor.SetFormatter(new XmlFormatter());
        editor.PublishText("Hello!");

        editor.SetFormatter(new CsvFormatter());
        editor.PublishText("Hello!");
    }
}
```

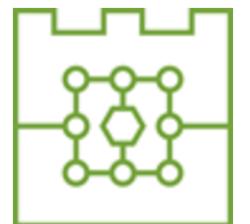
przykład użycia



# Potencjalne zastosowania

- Systemy płatności w aplikacjach internetowych.
- Program kompresji plików.
- Programowanie AI postaci w grach.
- Logowanie w aplikacjach.

I wiele innych, tam gdzie chcemy zaimplementować pewną funkcjonalność na kilka różnych sposobów, możemy użyć strategy. Pozwoli nam to na podmienianie tych implementacji w zależności od wymogów użytkownika.

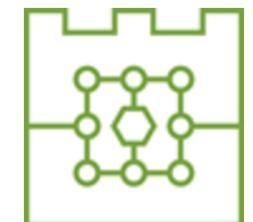


# **zalety**

- Łatwość dodania nowych algorytmów.
- Łatwa zmiana wykonywanych algorytmów.
- Brak zmiany kontekstu przy zmianie algorytmu.

# **gdzie wady?**

- Przy prostym programie nie zmieniającym algorytmów dodawanie nowych klas i interfejsów może być niepotrzebne.
- Klient musi wiedzieć o istniejących strategiach.
- Jeśli strategii jest bardzo dużo, zarządzanie nimi może być trudne.



# Dziękuję za uwagę

