



Informatyka

3.Diagram przepływu danych & wzorce projektowe

Opracował: Maciej Penar

Spis treści

1. Zanim zaczniemy	3
2. (6 pkt) Diagramy przepływu danych	4
2. (6 pkt) Wzorce projektowe	4
Zadanie (tu jest 6 pkt)	5

1. Zanim zaczniemy

Zrelaksować się i przyswoić sobie teorię dot. diagramów klas, diagramów przepływu danych.

Materiały do DFD:

- Materiały z Lucidchart: <https://www.lucidchart.com/pages/data-flow-diagram>
- Wiki: https://pl.wikipedia.org/wiki/Diagram_przep%C5%82ywu_danych

Wzorce projektowe:

- Thinking in Patterns: <http://www.mindview.net/Books/TIPatterns/>
- Lista wzorców: <http://www.blackwasp.co.uk/gofpatterns.aspx>

Oprogramowanie:

- Lucidchart (oprogramowanie w modelu SasS do modelowania)
- Enterprise Architect (<http://www.sparxsystems.com/products/ea/>)
- Visual Paradigm (<https://www.visual-paradigm.com/download/community.jsp>)
- Visio (<https://products.office.com/pl-pl/visio/flowchart-software>)

2. (6 pkt) Diagramy przepływu danych

1. (2 pkt) Narysować diagram przepływu danych dla wypożyczalni filmów (Lista 1.1.3)
 - a. Kontekstowy
 - b. Ogólny systemu
2. (2 pkt) Narysować diagram przepływu danych modelujący rezerwacje hotelowe (Lista 1.1.2)
 - a. Kontekstowy
 - b. Ogólny systemu
3. (2 pkt) Narysować diagramy przepływu danych dla Call Center (Lista 1.1.5)
 - a. Kontekstowy
 - b. Ogólny systemu

2. (6 pkt) Wzorce projektowe

Gdy myślimy o wzorach w wytwarzaniu oprogramowania mamy na myśli dwie klasy wzorców:

- Wzorce projektowe – które dotyczą organizacji kodu aplikacji
- Wzorce architektoniczne – które organizują strukturę programu / systemu

Wzorcami architektonicznymi **nie** będziemy się zajmować, warto nadmienić najbardziej popularny/akademicki czyli wzorec MVC (Model-View-Controller) – i jego dalekich kuzynów MVVM (Model-View-ViewModel) i MVP (Model-View-Presenter).

Wzorce projektowe dzielą się na trzy grupy:

- Kreacyjne – które organizują sposób instancjonowania obiektów – co umożliwia wprowadzenie pewnych ograniczeń (np. istnienie tylko 1 obiektu, albo określonej puli)
- Behawioralne – które organizują sposób zachowania obiektów
- Strukturalne – które organizują powiązania pomiędzy obiektami

Do najważniejszych wzorców **kreacyjnych** zaliczamy:

- Singleton
- (Abstract) Factory
- Connection Pool

Do najważniejszych wzorców **strukturalnych** zaliczamy:

- Proxy
- Facade
- Decorator

Do najważniejszych wzorców **behawioralnych** zaliczamy:

- Strategy
- Observer
- Iterator
- Template Method

ZADANIE (TU JEST 6 PKT)

Napisać w C#:

1. (2 pkt) Klasę którą jest **singletonem**
2. (2 pkt) Dany jest następujący kod:

```
interface IMultiplier
{
    int Calculate(int x);
}

class Multiplier : IMultiplier
{
    public int Calculate(int x)
    {
        Thread.Sleep(1000); // Potężne, czasochłonne kalkulecje
        return x * x;
    }
}

class Program
{
    static void Main(string[] args)
    {
        // Tu można zmienić new Multiplier() na coś innego
        IMultiplier multiplier = new Multiplier();
        for (int i = 0; i < 100; ++i)
        {
            Console.WriteLine(multiplier.Calculate(i % 10));
        }
    }
}
```

W jaki sposób można przyspieszyć jego działanie za pomocą wzorca **Proxy**? Napisać potrzebny kod. Jako interfejs Proxy przyjąć IMultiplier. Pomyśleć o Proxy jak o cache'u (wykorzystać albo tablicę albo klasę Dictionary).

3. (2 pkt) Napisać dowolny kawałek kodu realizujący wzorec **Strategy**. Narysować diagram klas.