



JPS Ćw 5

Wzorec projektowy Visitor (Odwiedzający)

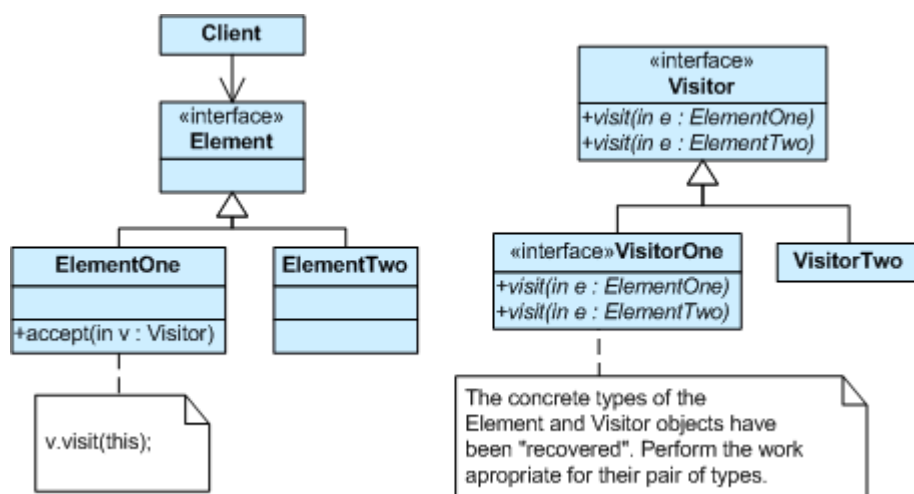
ZAŁOŻENIACH WZORCA VISITOR

Wzorec projektowy Visitor powstał z chęci oddzielenia struktury danych od operacji które mogą być na tych danych przeprowadzone. Pozwala opracować wiele algorytmów pracujących na tych samych danych i błyskawicznie przełączać który algorytm wykorzystać.

ELEMENTY WZORCA

- Struktura obiektów
 - Każda klasa musi mieć metodę `accept` pozwalającą na „wejście” obserwatora.
 - Struktura może być dowolnie złożona i mieć wiele podtypów.
- Interfejs VISITOR
 - Będzie musiał uwzględniać wszystkie typy danych jakie mogą się pojawić w strukturze. To ten interfejs będzie musiała implementować klasa z algorytmem.

IMPLEMENTACJA WZORCA



ŹRÓDŁO: [HTTP://SOURCEMAKING.COM/DESIGN_PATTERNS/VISITOR](http://sourcemaking.com/design_patterns/visitor)



IMPLEMENTACJA W C#

```
// "ObjectStructure"
class ObjectStructure{
    private ArrayList elements = new ArrayList();

    public void Attach(Element element){
        elements.Add(element);
    }

    public void Accept(Visitor visitor){
        foreach (Element e in elements){
            e.Accept(visitor);
        }
    }
}

// "Visitor"
abstract class Visitor{
    public abstract void VisitConcreteElementA(
        ConcreteElementA concreteElementA);
    public abstract void VisitConcreteElementB(
        ConcreteElementB concreteElementB);
}

// "ConcreteVisitor1"
class ConcreteVisitor1 : Visitor{
    public override void VisitConcreteElementA(
        ConcreteElementA concreteElementA){
        Console.WriteLine("{0} visited by {1}",
            concreteElementA.GetType().Name, this.GetType().Name);
    }

    public override void VisitConcreteElementB(
        ConcreteElementB concreteElementB){
        Console.WriteLine("{0} visited by {1}",
            concreteElementB.GetType().Name, this.GetType().Name);
    }
}

// "ConcreteVisitor2"
class ConcreteVisitor2 : Visitor{
    public override void VisitConcreteElementA(
        ConcreteElementA concreteElementA){
        Console.WriteLine("{0} visited by {1}",
            concreteElementA.GetType().Name, this.GetType().Name);
    }

    public override void VisitConcreteElementB(
        ConcreteElementB concreteElementB){
        Console.WriteLine("{0} visited by {1}",
            concreteElementB.GetType().Name, this.GetType().Name);
    }
}
```



```
// "Element"
abstract class Element{
    public abstract void Accept(Visitor visitor);
}

// "ConcreteElementA"
class ConcreteElementA : Element{
    public override void Accept(Visitor visitor){
        visitor.VisitConcreteElementA(this);
    }
}

// "ConcreteElementB"
class ConcreteElementB : Element
{
    public override void Accept(Visitor visitor)
    {
        visitor.VisitConcreteElementB(this);
    }
}

//USAGE:
// Setup structure
ObjectStructure o = new ObjectStructure();
o.Attach(new ConcreteElementA());
o.Attach(new ConcreteElementB());

// Create visitor objects
ConcreteVisitor1 v1 = new ConcreteVisitor1();
ConcreteVisitor2 v2 = new ConcreteVisitor2();

// Structure accepting visitors
o.Accept(v1);
o.Accept(v2);
```

Źródło: SourceMaking.com

ZALICZENIE

Zaliczenie mini-projektu 3 polega na zaimplementowaniu (najlepiej w nowym projekcie, bez łączenia z poprzednimi mini-projektami) wzorca Odwiedzający na własnym przykładzie. Przykład musi być nie trywialny i zawierać:

3 różne klasy które będą przechowywać różne dane i różnie się zachowywać

1 klasę przechowującą kolekcję obiektów z klas powyżej

2 obserwatorów którzy w różny sposób potraktują obiekty odwiedzone.

Problematyka implementacji może być dowolna ale powinna odzwierciedlać realne problemy w informatyce lub „biznesie”