

# Відвідувач

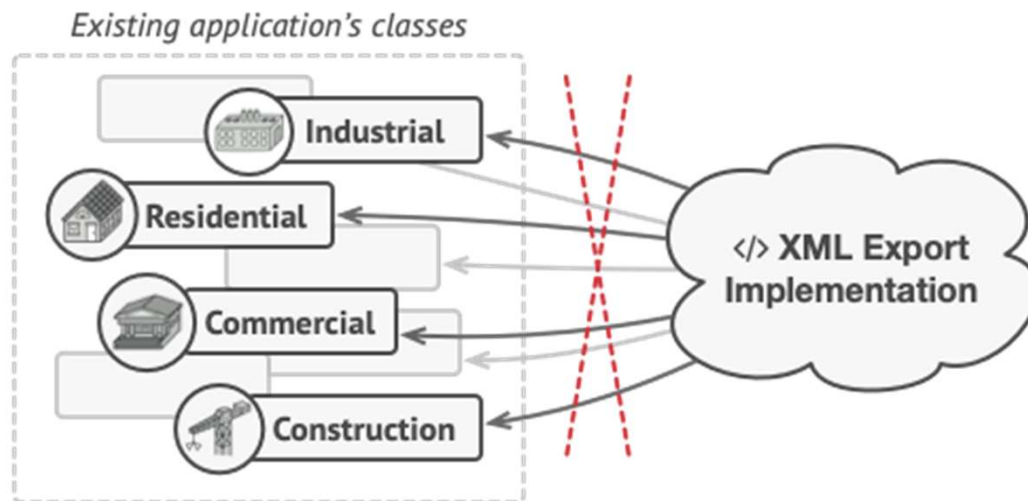
Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.

Gang of Four



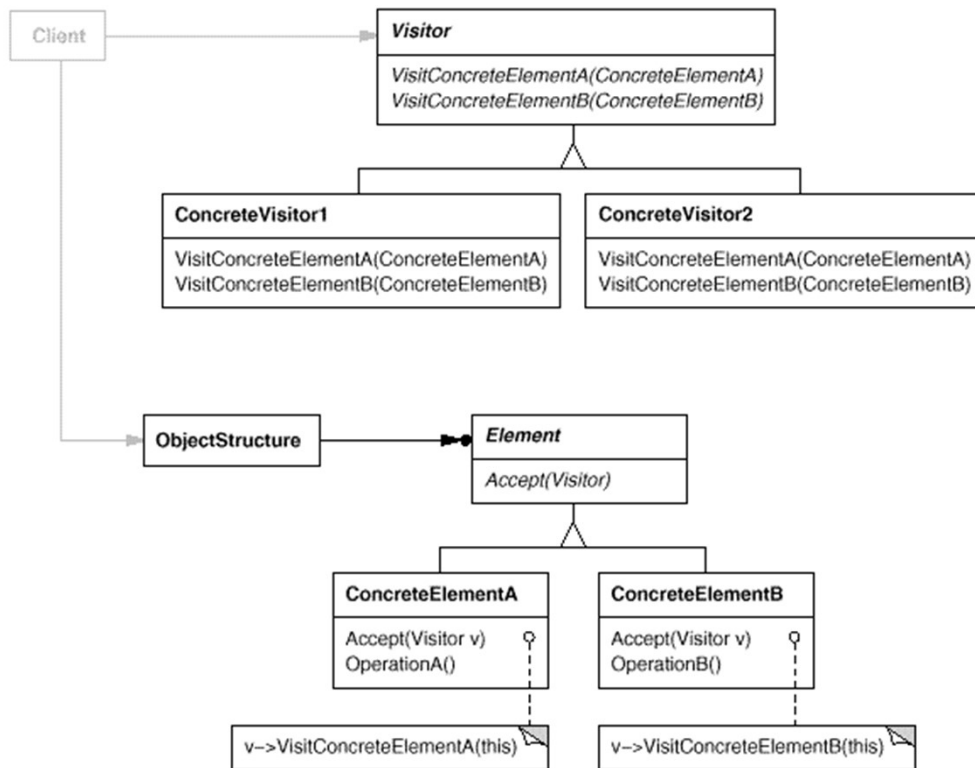
Щоб прочитати книгу у  
бібліотеці не обов'язково  
бути її працівником

# Мотивація



Як реалізувати перетворення складного графу на XML документ не міняючи структури його класів?

# Реалізація



1. Створити абстрактний клас **Visitor** із методами для відвідування кожного із конкретних елементів
2. Створити інтерфейс **Element** із методом `Accept(Visitor)`, який буде приймати відвідувача і виконувати необхідну операцію
3. Реалізувати цей інтерфейс в усіх класах, де потрібно додати новий функціонал. За потреби реалізувати додаткову логіку разом із викликом Відвідувача.
4. Для того щоб додати нову логіку до потрібного класу потрібно створити реалізацію класу **Visitor**. Для виклику методів використовуємо метод `Accept`

# Загальні переваги та недоліки цього шаблону проектування

## Переваги:

- Забезпечення принципу єдиної відповідальності
- Забезпечення принципу відкритості і закритості
- Відвідувач може зберігати корисну інформацію при роботі з об'єктами

## Недоліки:

- Треба оновлювати усіх відвідувачів при додаванні нового класу, який можна відвідати
- Відвідувач може не мати потрібного доступу до приватних полів
- Потенційно порушує інкапсуляцію

# Подвійна диспетчеризація

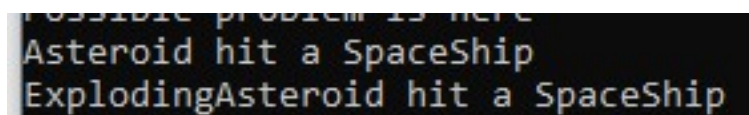
```
class SpaceShip {};  
class ApolloSpacecraft : public SpaceShip {};  
  
class Asteroid {  
public:  
    virtual void CollideWith(SpaceShip&) {  
        std::cout << "Asteroid hit a SpaceShip\n";  
    }  
    virtual void CollideWith(ApolloSpacecraft&) {  
        std::cout << "Asteroid hit an ApolloSpacecraft\n";  
    }  
};  
  
class ExplodingAsteroid : public Asteroid {  
public:  
    void CollideWith(SpaceShip&) override {  
        std::cout << "ExplodingAsteroid hit a SpaceShip\n";  
    }  
    void CollideWith(ApolloSpacecraft&) override {  
        std::cout << "ExplodingAsteroid hit an ApolloSpacecraft\n";  
    }  
};
```

## Маємо проблему

Код

```
Asteroid theAsteroid;  
SpaceShip theSpaceShip;  
ApolloSpacecraft theApolloSpacecraft;  
ExplodingAsteroid theExplodingAsteroid;  
Asteroid& theAsteroidReference = theExplodingAsteroid;  
SpaceShip& theSpaceShipReference = theApolloSpacecraft;  
theAsteroid.CollideWith(theSpaceShipReference);  
theAsteroidReference.CollideWith(theSpaceShipReference);
```

Вивід



```
possible problem is here  
Asteroid hit a SpaceShip  
ExplodingAsteroid hit a SpaceShip
```

Рішення - Відвідувач

# Відвідувач і Композит

- Відвідувач дозволяє додавати деякі операції над Композитом і його елементами
- Ефективна навігація по структурі Композиту
- Відділення структури Композиту від операцій над ним



Військовий стіл – приклад Відвідувача від армії у Композиті Могилянки

# Відвідувач і Інтерпретатор

- Відвідувач може бути використаний для реалізації Інтерпретатора
- Дозволяє зробити реалізації для декількох різних мов
- Дозволяє застосовувати потрібну реалізацію залежно від мови
- Це поєднання паттернів широко використовується при розробці компіляторів



О́тче на́шъ, ѿже ѣси на небѣхъ, да свѣтитсѧ  
ѿма твоѣ, да прїидетъ царствїе твоѣ: да бѣдетъ  
во́ла твоѧ, ѿко на небѣхъ, ѿ на зѣмли. хлѣбъ  
на́шъ на́сѣщныи даждь на́мъ днѣсь, ѿ ѿста́ви  
на́мъ до́лги на́ша, ѿкоже ѿ мы ѿста́вляемъ  
должніи́къмъ на́шимъ: ѿ не вѣди на́съ во  
иску́шенїе, но ѿзбави на́съ ѿ лѧкаваго, ѿмны.

Церковнослов'янська і старослов'янська мови дуже схожі, але мають багато відмінностей. Якщо ми хочемо мати спільний інтерфейс, то він повинен враховувати граматику кожної з них у своїй реалізації



# Коли використовувати?

- Коли потрібно виконати операцію на усіх елементах складної структури(наприклад дерева)
- Коли потрібно усунути частину допоміжних методів із класу
- Коли деякий метод потрібний лише у деяких елементах ієрархії класів і не потрібний у інших
- Коли потрібно мати можливість додавати новий функціонал не міняючи структуру класу
- Щоб забезпечити подвійну диспетчеризацію

# Застосування шаблону проектування у бібліотеках

- FileVisitor у Java
- AnnotationValueVisitor для AnnotationValue у javaх
- Декілька інших класів у javaх
- std::visit – вбудована реалізація Відвідувача у C++