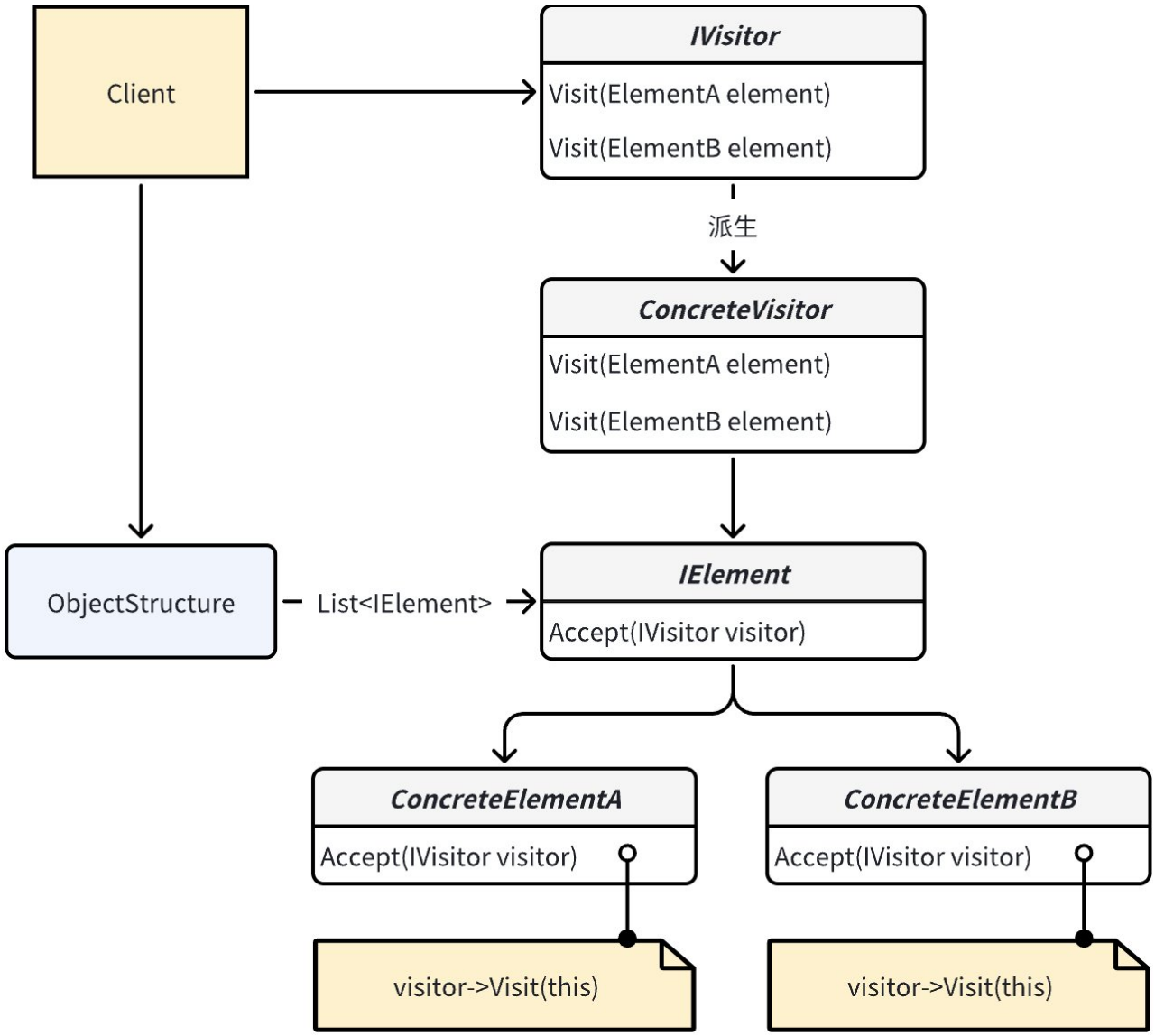
访问者模式

概念

- 定义：访问者模式（Visitor Pattern）是一种行为设计模式，它能将算法与对象结构分离，使得新的操作可以在不改变对象结构的情况下被添加。
- 设计意图：将数据结构与作用于数据结构的操作分离。这样可以在不改变数据结构的前提下，增加新的操作。
- 访问者模式适用于以下场景：
  - 对象结构相对稳定，但经常需要在此结构上定义新的操作。
  - 需要对一个对象结构中的对象进行很多不同且不相关的操作，但不希望这些操作“污染”对象的类。
  - 对象结构包含许多类，这些类有不同的接口，需要对这些类执行一些依赖于具体类的操作。
- 访问者模式主要由以下几个部分构成：
  - Visitor（访问者）：为对象结构中的每一个具体元素类声明一个访问操作接口。
  - ConcreteVisitor（具体访问者）：实现每个访问操作，定义对每个具体元素的操作。
  - Element（元素）：定义一个接受访问者的方法，该方法通常称为 Accept。
  - ConcreteElement（具体元素）：实现 Accept 方法，该方法调用访问者的相应方法。
  - ObjectStructure（对象结构）：能够枚举它的元素，可以提供一个高层的接口以允许访问者访问它的元素。
- 优点：
  - 增加新的操作很容易：可以在不改变对象结构的情况下增加新的操作。
  - 符合单一职责原则：将不相关的行为分离到不同的类中。
  - 符合开闭原则：可以在不修改现有代码的情况下增加新的功能。
- 缺点：
  - 难以增加新的元素：如果需要在对象结构中增加新的元素，所有的访问者都需要修改。
  - 破坏封装：访问者模式要求访问者对象访问并操作对象结构中的数据，这可能会破坏对象的封装性。
  - 复杂性：对于简单的对象结构，使用访问者模式可能会增加不必要的复杂性。



实例

- 访问者接口，使用函数重载实现访问不同的元素

```
interface IVisitor
{
    void Visit(ElementA element);
    void Visit(ElementB element);
}
```

- 具体访问者，实现接口方法

```
class ConcreteVisitor : IVisitor
{
    public void Visit(ElementA element)
```

```
    {
        Console.WriteLine("Visited ElementA");
    }

    public void Visit(ElementB element)
    {
        Console.WriteLine("Visited ElementB");
    }
}
```

- 元素接口

```
interface IElement
{
    void Accept(IVisitor visitor);
}
```

- 具体元素

```
class ElementA : IElement
{
    public void Accept(IVisitor visitor)
    {
        visitor.Visit(this);
    }
}

class ElementB : IElement
{
    public void Accept(IVisitor visitor)
    {
        visitor.Visit(this);
    }
}
```

- 对象结构

```
class ObjectStructure
{
    private List<IElement> elements = new List<IElement>();

```

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

    public void Detach(IElement element)
    {
        elements.Remove(element);
    }

    public void Accept(IVisitor visitor)
    {
        foreach (var element in elements)
        {
            element.Accept(visitor);
        }
    }
}
```

- 客户端代码

```
static void Main(string[] args)
{
    ObjectStructure structure = new ObjectStructure();
    structure.Attach(new ElementA());
    structure.Attach(new ElementB());

    ConcreteVisitor visitor = new ConcreteVisitor();
    structure.Accept(visitor);
}
```

- 打印结果

```
Visited ElementA
Visited ElementB
```