1. 컴포지트 패턴(Composite Pattern)

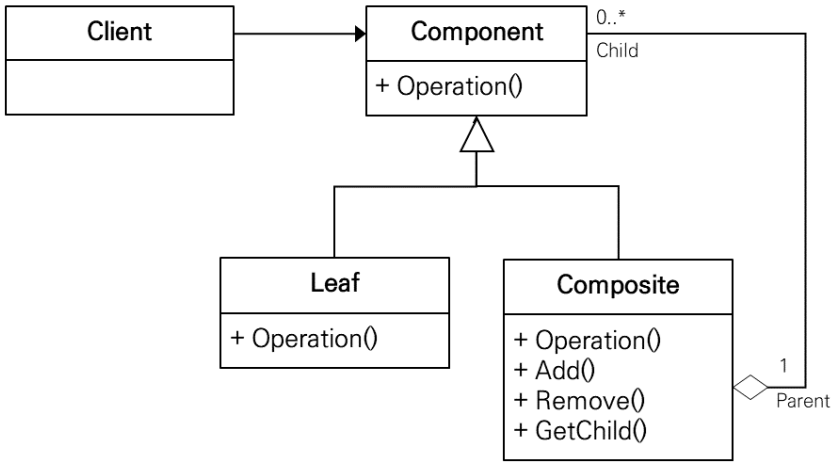
1) 컴포지트 패턴

- 객체들의 관계를 트리 구조로 구성하여 부분-전체 계층을 표현하는 패턴
- 클라이언트에서 개별 객체와 다른 객체들로 구성된 복합 객체(Composite)를 똑같은 방법으로 다룰 수 있음
- 구성 요소 : 복합 구조에 들어 있는 것, 복합 객체와 잎 노드가 존재
- 이터레이터 패턴(Iterator Pattern)을 함께 알아 두면 좋음

  이터레이터 패턴 : 컬렉션을 표현하는 방법을 노출시키지 않으면서도 집합체 내에 있는 모든 객체들에 하나씩 접근하는 방법을 제공하는 패턴

2) 클래스 다이어그램

Composite Pattern_Class



3) 예제

```
#pragma once

#include <iostream>
#include <vector>

using namespace std;

class CComponent
{
public:
	virtual void traverse() = 0;
};

class CLeaf : public CComponent
{
private:
	int value;

public:
	CLeaf(int val) { value = val; }
	void traverse() { cout << value << ' '; }
};

class CComposite : public CComponent
{
private:
	vector < CComponent * > children;

public:
	void add(CComponent *ele) { children.push_back(ele); }
	void traverse()
	{
		for (int i = 0; i < children.size(); i++)
			children[i]->traverse();
	}
};

//========================================================================================

#include "Composite.h"

int main()
{
	CComposite containers[4];

	for (int i = 0; i < 4; i++)
		for (int j = 0; j < 3; j++)
			containers[i].add(new CLeaf(i * 3 + j));

	for (int i = 1; i < 4; i++)
		containers[0].add(&(containers[i]));

	for (int i = 0; i < 4; i++)
	{
		containers[i].traverse();
		cout << endl;
	}
}
```