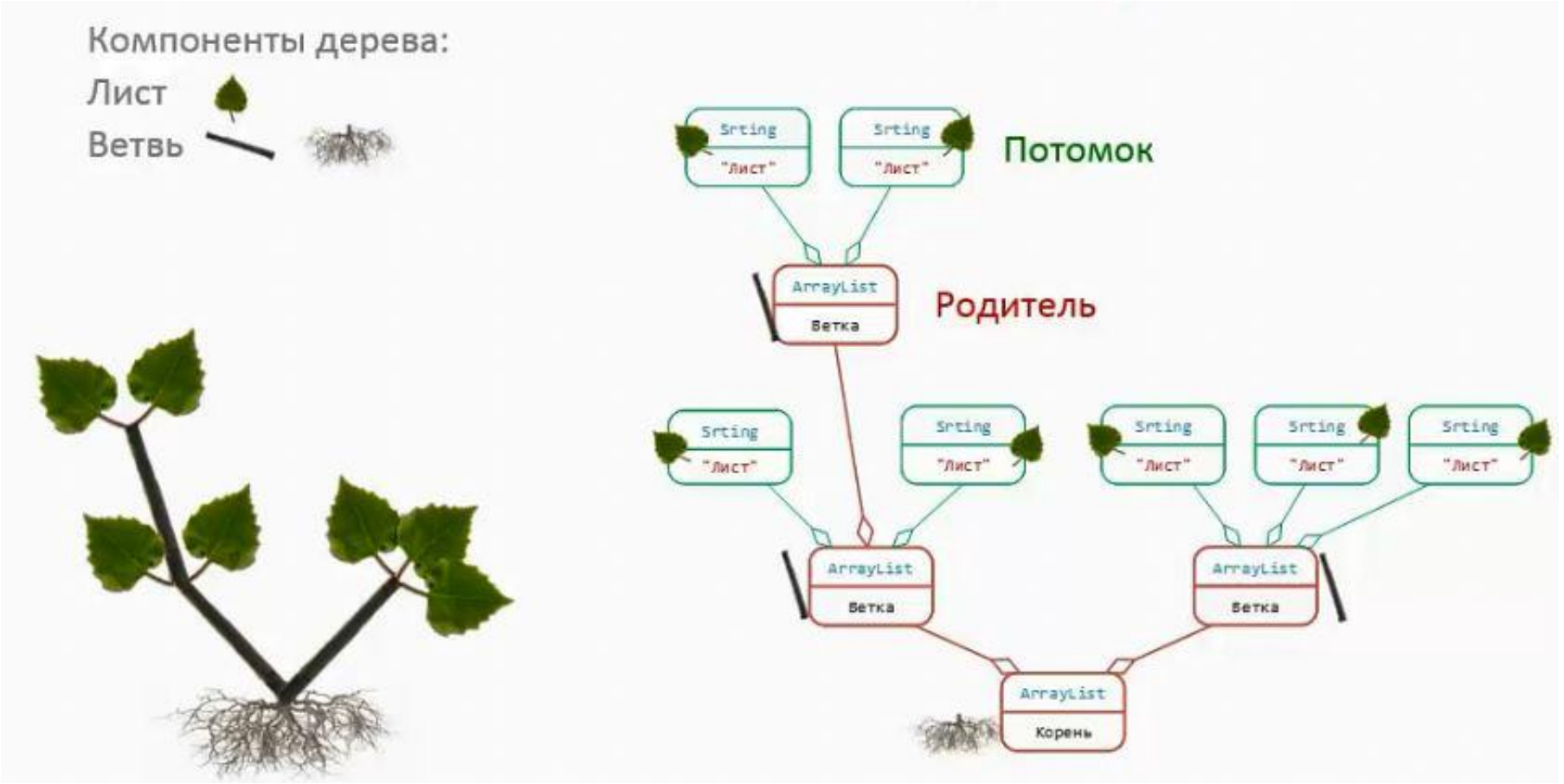
КОМПОНУВАЛЬНИК

МОДУЛЬ: ПАТЕРНИ ПРОЕКТУВАННЯ

МЕТАФОРА

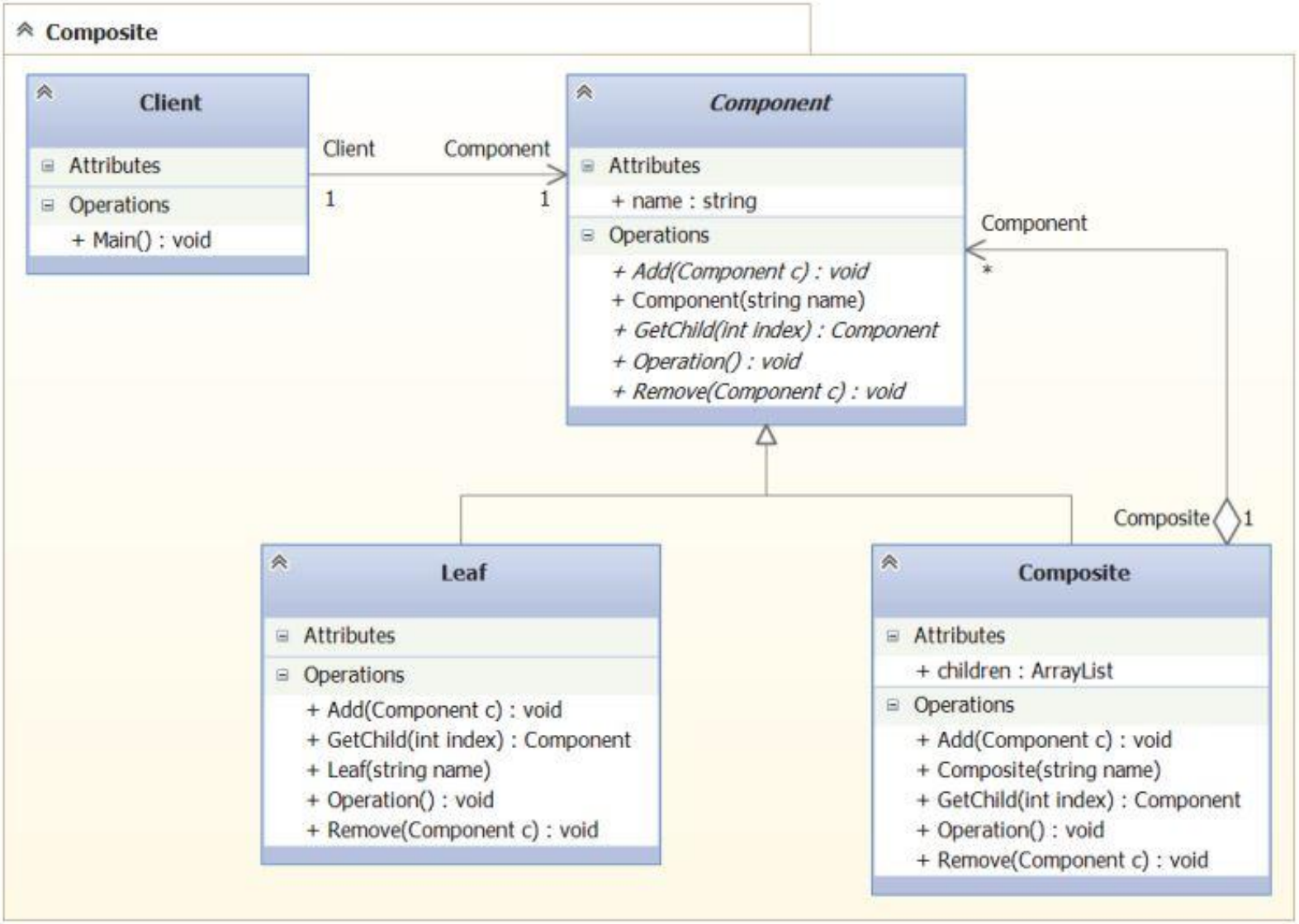
ПРИКЛАД


Компоненты дерева:
Лист
Ветвь
Srting
"Лист"
Srting
"Лист"
Потомок
ArrayList
Ветка
Родитель
Srting
"Лист"
Srting
"Лист"
Srting
"Лист"
Srting
"Лист"
Srting
"Лист"
ArrayList
Ветка
ArrayList
Ветка
ArrayList
Корень

ПРИЗНАЧЕННЯ

Побудова «дерев»

СТРУКТУРА ПАТЕРНА НА МОВІ **UML**



СТРУКТУРА ПАТЕРНА НА МОВІ C#

```
class Program
{
    static void Main()
    {
        Component root = new Composite("ROOT");
        Component branch1 = new Composite("BR1");
        Component branch2 = new Composite("BR2");
        Component leaf1 = new Leaf("L1");
        Component leaf2 = new Leaf("L2");

        root.Add(branch1);
        root.Add(branch2);
        branch1.Add(leaf1);
        branch2.Add(leaf2);
        root.Operation();
    }
}
```

```
abstract class Component
{
    protected string name;

    public Component(string name)
    {
        this.name = name;
    }

    public abstract void Operation();
    public abstract void Add(Component component);
    public abstract void Remove(Component component);
    public abstract Component GetChild(int index);
}
```

```
class Leaf : Component
{
    public Leaf(string name)
        : base(name)
    {
    }

    public override void Operation()
    {
        Console.WriteLine(name);
    }

    public override void Add(Component component)
    {
        throw new InvalidOperationException();
    }

    public override void Remove(Component component)
    {
        throw new InvalidOperationException();
    }

    public override Component GetChild(int index)
    {
        throw new InvalidOperationException();
    }
}
```

```
class Composite : Component
{
    ArrayList nodes = new ArrayList();

    public Composite(string name)
        : base(name)
    {
    }

    public override void Operation()
    {
        Console.WriteLine(name);

        foreach (Component component in nodes)
            component.Operation();
    }

    public override void Add(Component component)
    {
        nodes.Add(component);
    }

    public override void Remove(Component component)
    {
        nodes.Remove(component);
    }

    public override Component GetChild(int index)
    {
        return nodes[index] as Component;
    }
}
```

УЧАСНИКИ

- **Компонент** - Компонент:
  Надає інтерфейс для об'єктів з яких складається дерево. В окремому випадку може надавати реалізацію за замовчуванням для деяких методів.

- **Лист** - Лист:
  Є класом листових вузлів дерева і не може мати нащадків, тобто, включати в себе об'єкти, який відносяться до структури дерева (з листа не може вирости гілка або інший лист).

- **Composite** - Складений об'єкт:
  Задає поведінку об'єктів, що входять в структуру дерева, у яких є нащадки, а також сам зберігає в собі компоненти дерева (об'єкти нащадки), як вузлові, так і листові. Реалізує методи інтерфейсу компонента, пов'язані з управління нащадками.

- **Клієнт** - Клієнт:
  Маніпулює об'єктами, що входять в структуру дерева, через інтерфейс, що надається класом компонента.

ДЯКУЮ ЗА УВАГУ!