

Structural Pattern: Composite



Kevin Dockx

Architect

@Kevindockx | www.kevindockx.com

Coming Up



Describing the composite pattern

Implementation:

- File system scenario

Structure of the composite pattern



Coming Up



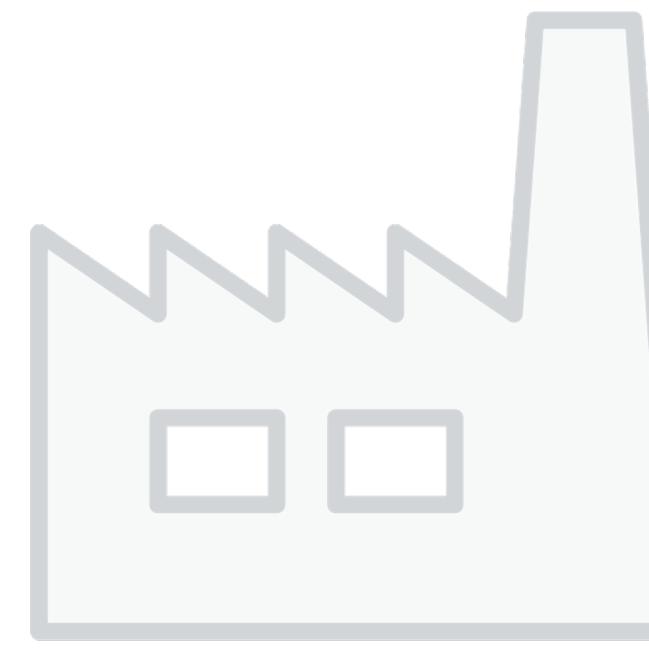
Use cases for this pattern

Pattern consequences

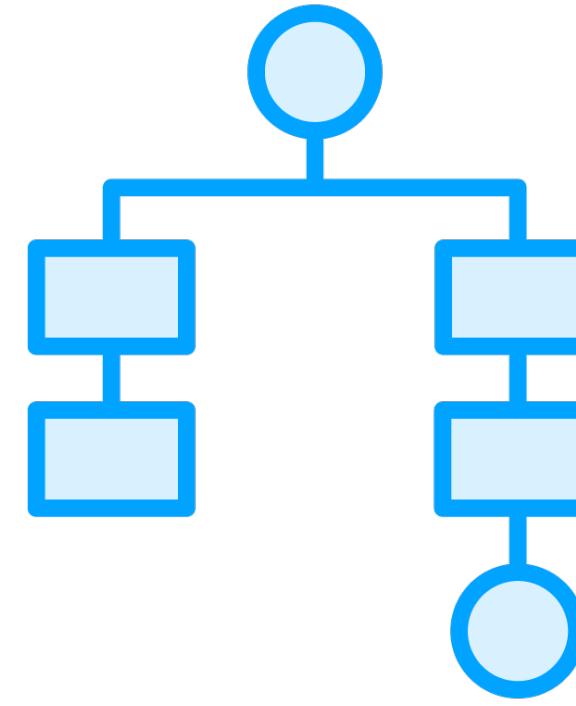
Related patterns



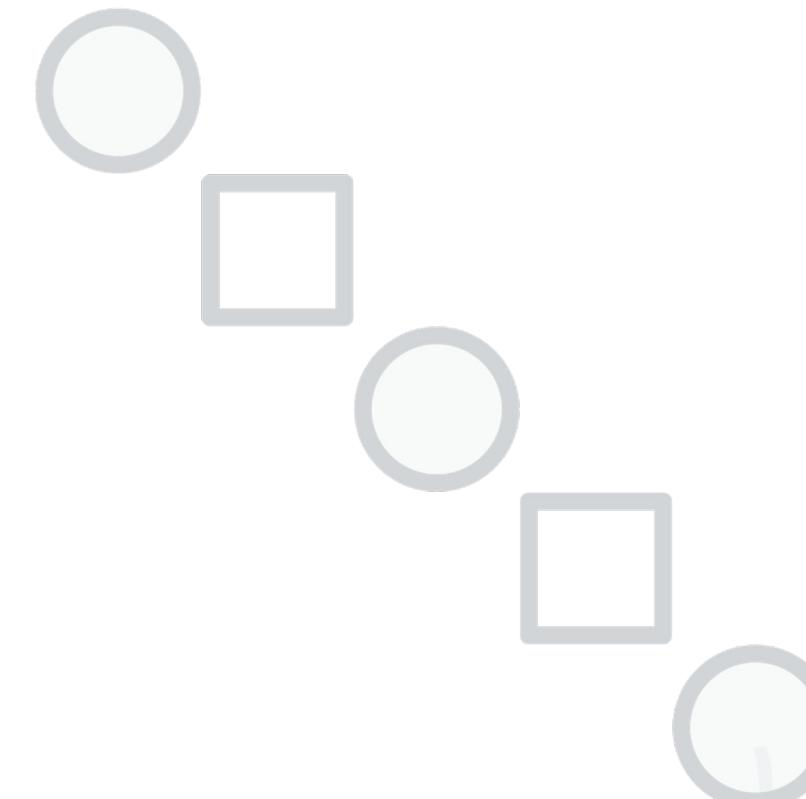
Describing the Composite Pattern



Creational



Structural



Behavioral



Composite

The intent of this pattern is to compose objects into tree structures to represent part-whole hierarchies. As such, it lets clients treat individual objects and compositions of objects uniformly: as if they all were individual objects.



Describing the Composite Pattern

Examples:

- XML document structure
- Composing a drawing



```
public class File { }  
public class Directory {  
}  
}
```

Describing the Composite Pattern



```
public class File { }  
public class Directory {  
}  
}
```

Describing the Composite Pattern



```
public class File { }

public class Directory {
    // other directories and/or files
}
```

Describing the Composite Pattern



```
public class File { }

public class Directory {
    // other directories and/or files
}

public class Client {
    // calculate the size of a tree of Directory / File
}
```

Describing the Composite Pattern

Intrinsic knowledge about the object type is required

- **Class**
- **Method to call**
- **Nesting level**



Describing the Composite Pattern

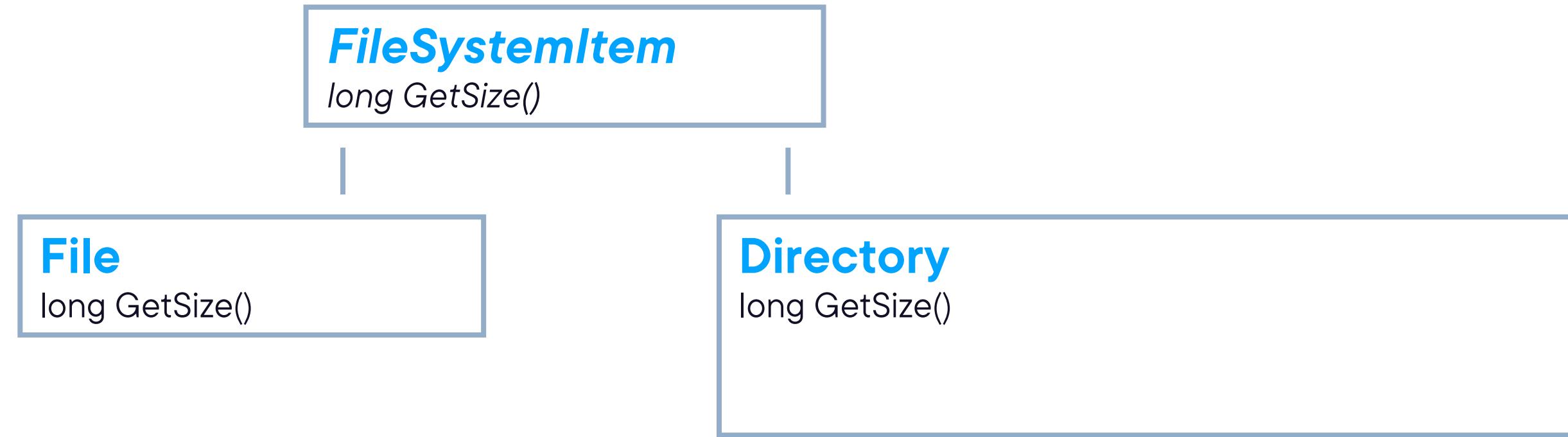


Describing the Composite Pattern

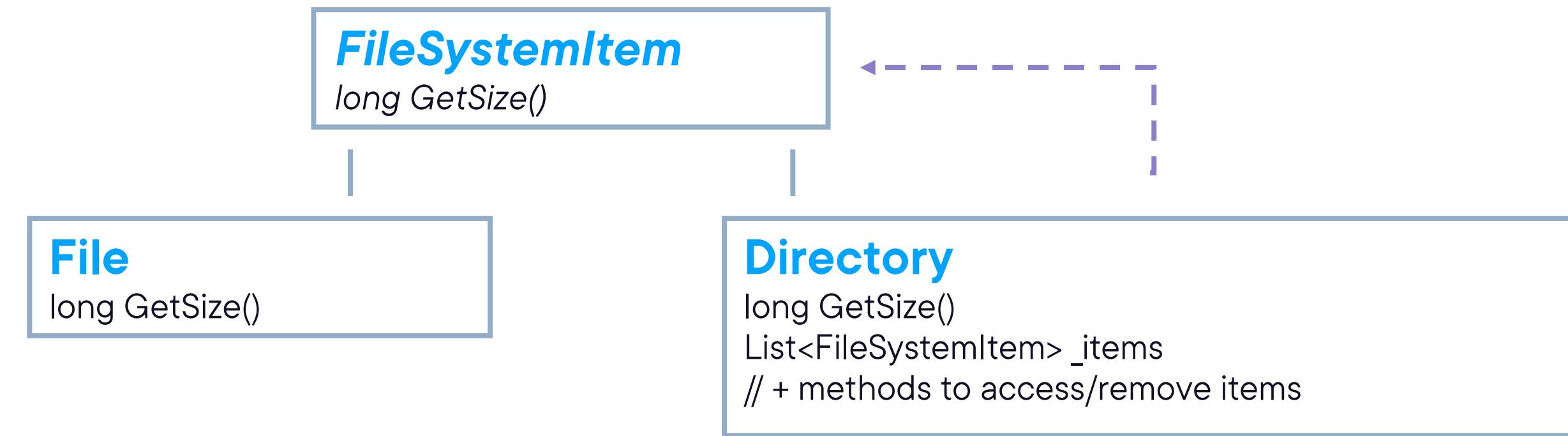
FileSystemItem
long GetSize()



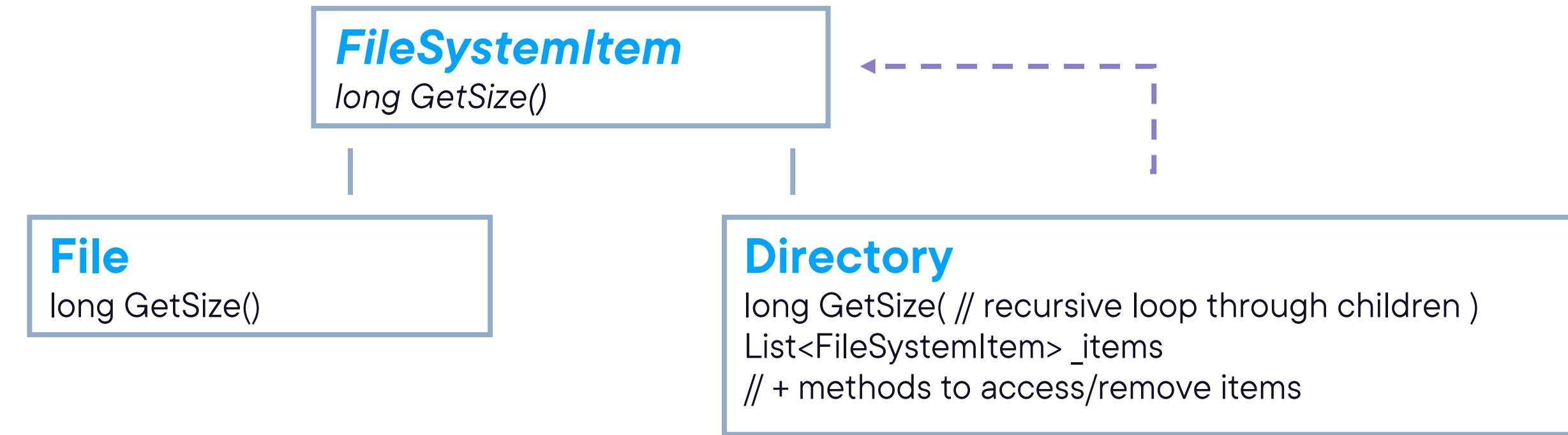
Describing the Composite Pattern



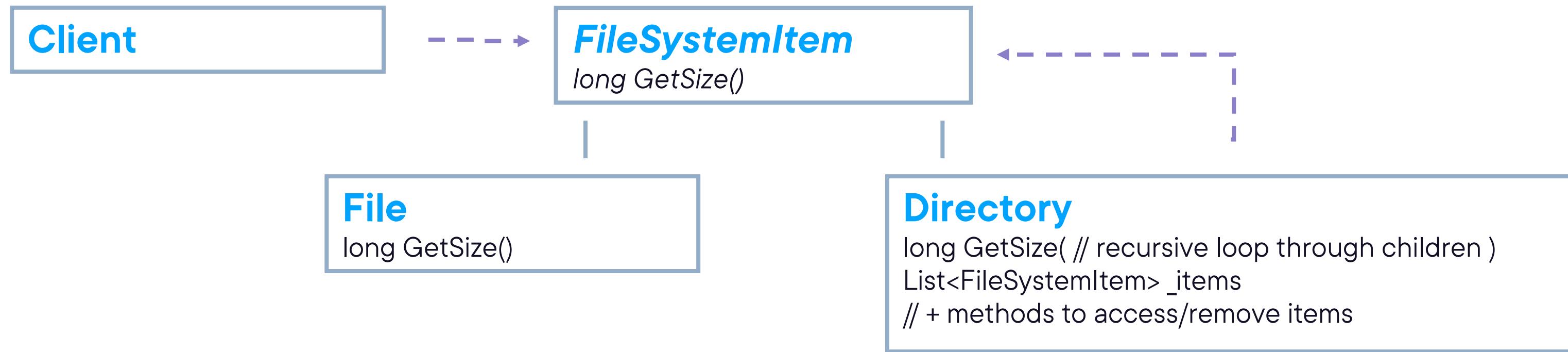
Describing the Composite Pattern



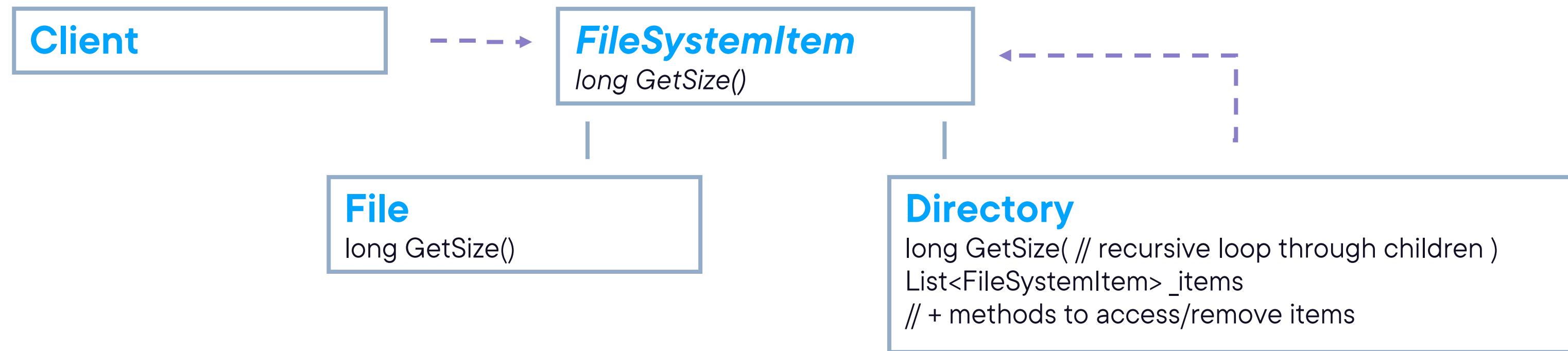
Describing the Composite Pattern



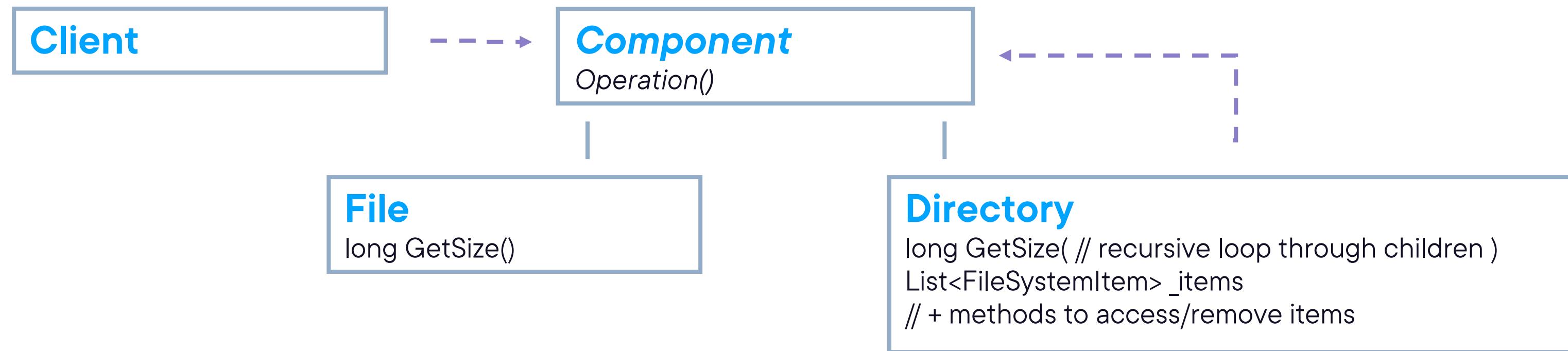
Describing the Composite Pattern



Structure of the Composite Pattern



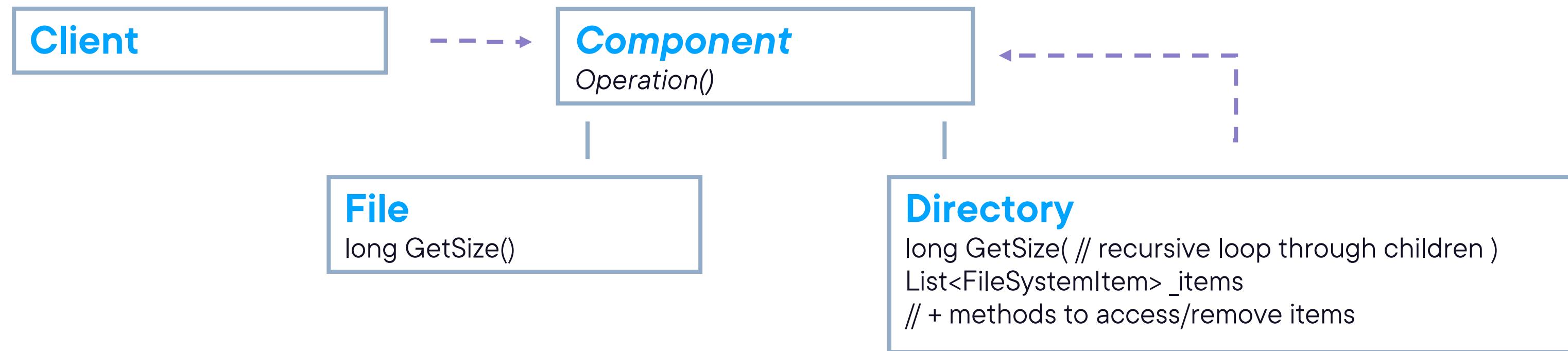
Structure of the Composite Pattern



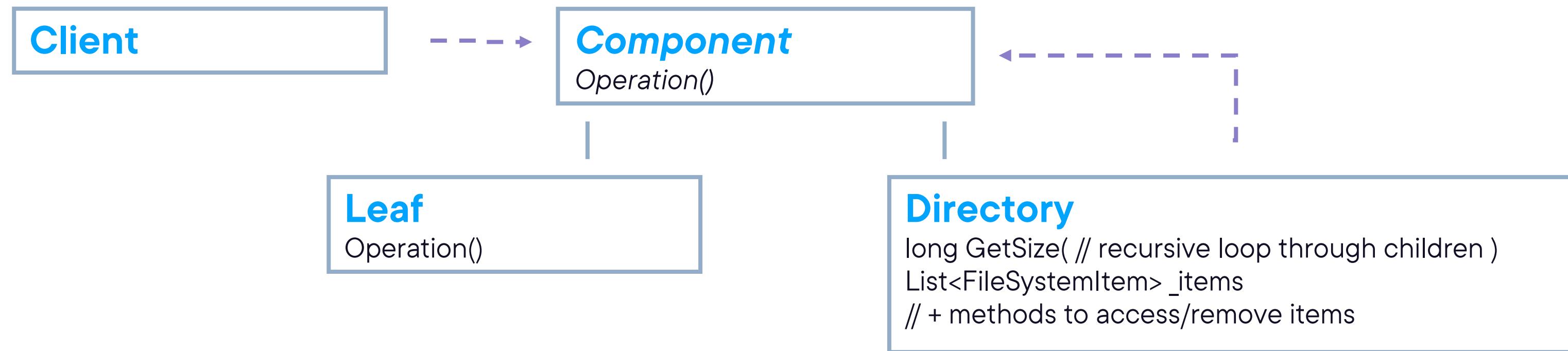
**Component declares the
interface for objects on the
compositions, and contains
a common operation**



Structure of the Composite Pattern



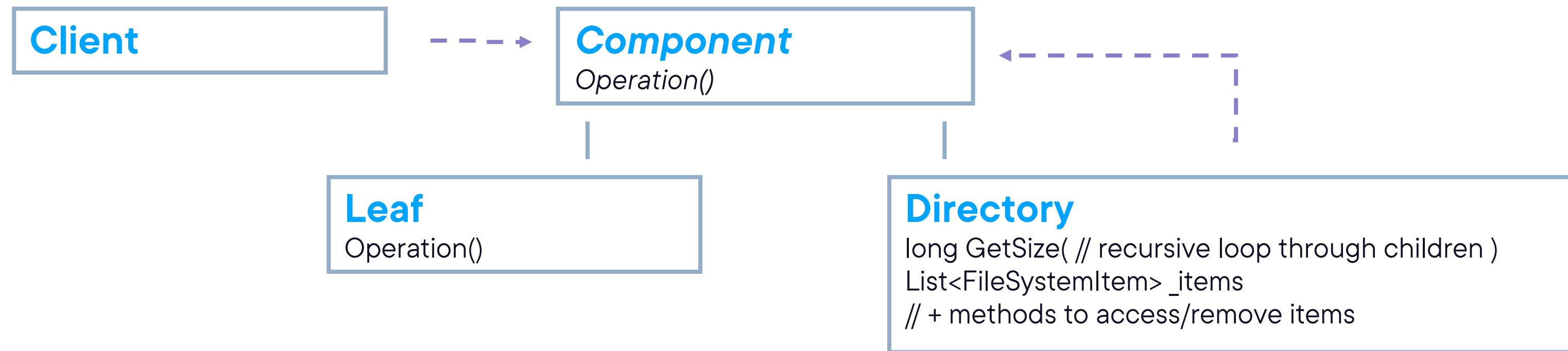
Structure of the Composite Pattern



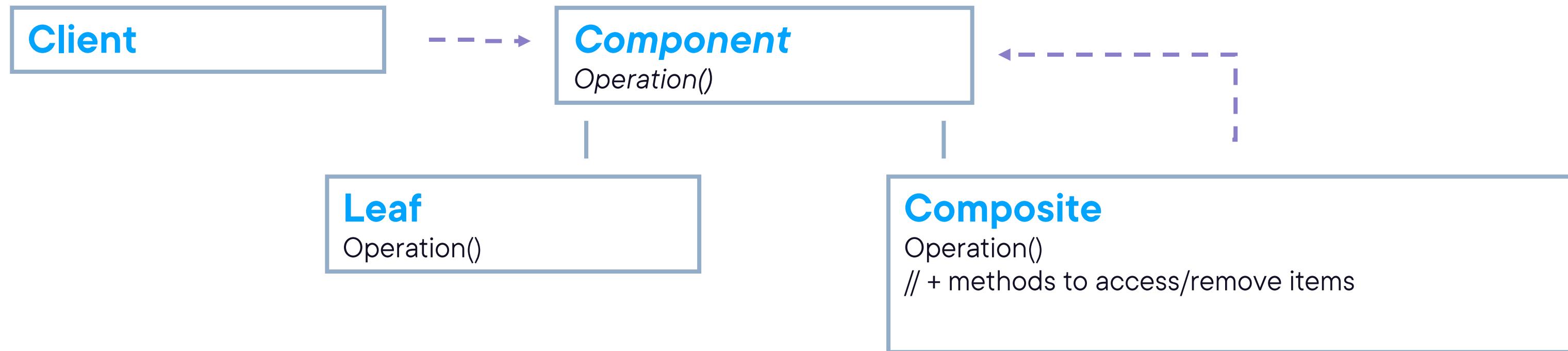
Leaf represents leaf objects in the composition, and has no children. It defines behavior for primitive objects in the composition.



Structure of the Composite Pattern



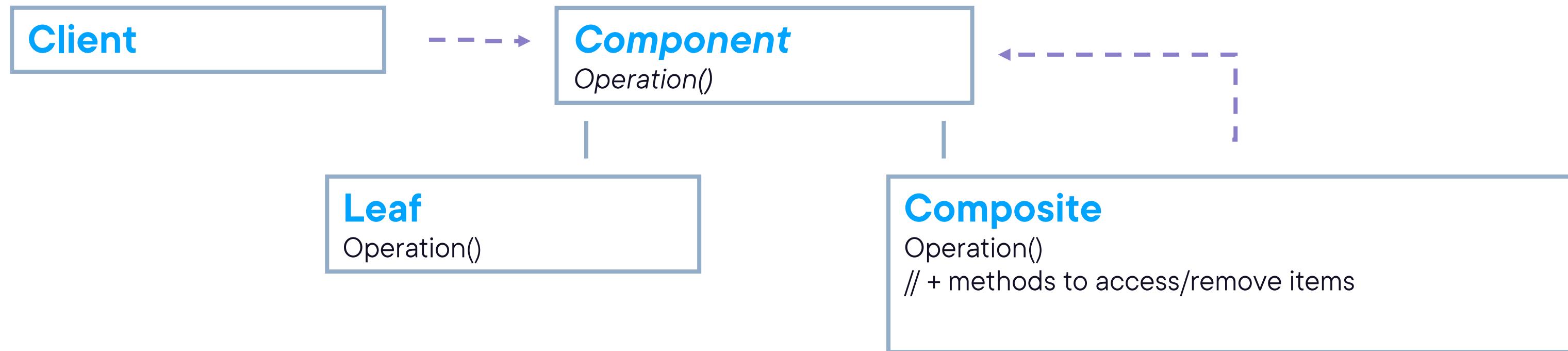
Structure of the Composite Pattern



**Composite stores child
components and defines
behavior for components
having children**



Structure of the Composite Pattern



**Client manipulates objects
in the compositions through
the Component interface**



Structure of the Composite Pattern

Variations exist

- Original template defines an operation to get a specific child
- Original template defines child management operations on the **Component** abstract base class



Demo



Implementing the composite pattern



Use Cases for the Composite Pattern



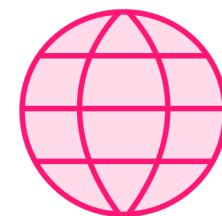
When you want to represent part-whole hierarchies of objects



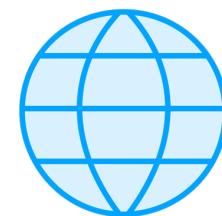
When you want to be able to ignore the difference between compositions of objects and individual objects



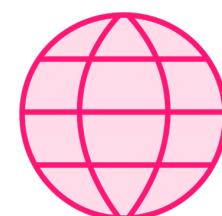
Use Cases for the Composite Pattern



An investment portfolio



Menu systems



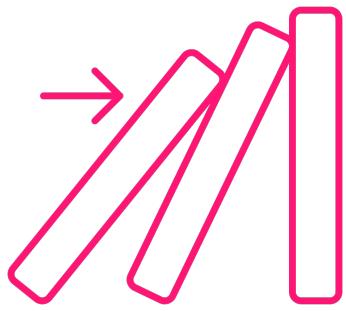
An organization chart



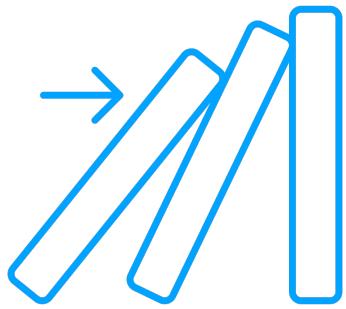
Representing a drawing in a drawing application



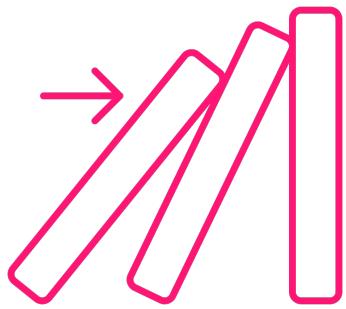
Pattern Consequences



Makes the client simple



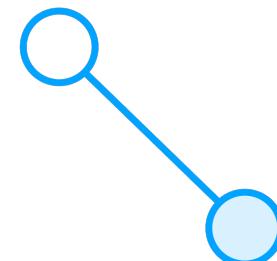
It's easy to add new kinds of components: [open/closed principle](#)



It can make the overall system too generic

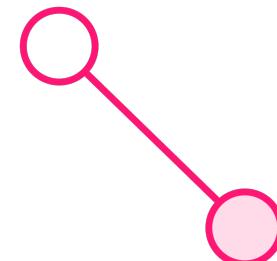


Related Patterns



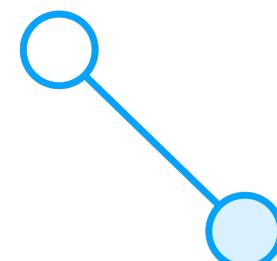
Chain of responsibility

Leaf component can pass a request through a chain of all parent components



Iterator

Can be used to traverse composites



Visitor

Can be used to execute an operation over the full composite tree



Summary



Intent of the composite pattern:

- Provide a transparent, easy way to work with tree-like structures

Example of recursiveness



Up Next:

Structural Pattern: Facade

