# Composite Design Pattern

**What is *Composite Design pattern* ?**

The *Composite Design Pattern* *composes* objects into a *tree structure* to represent *part-whole hierarchies*. It allows clients to treat *individual objects* and *compositions* of objects *uniformly*.

***Problems* in Designing a File System**:

When we design a *file system*, we have two different entities:

- *File* → simple, indivisible objects (leaf nodes).
- *Folder* → can contain multiple files or subfolders (composite nodes).

*Now the issues:*

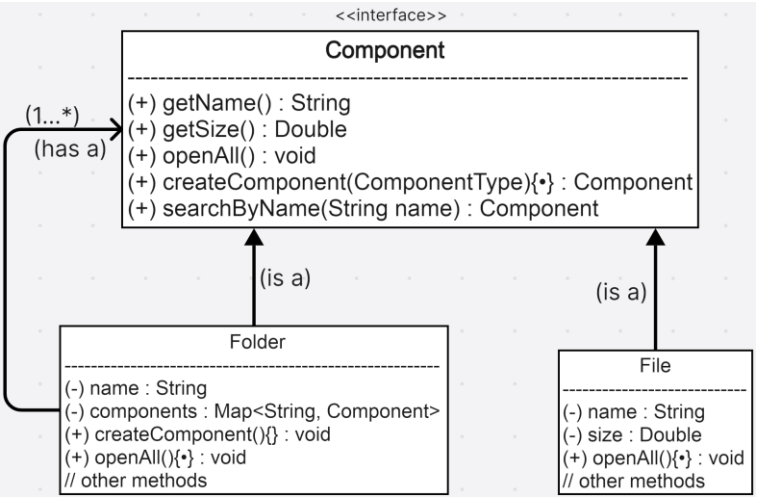
1. Different *Types* Require Different *Handling*
   - A *File* has operations like *open*(), *getSize*().
   - A *Folder* has operations like *add*(), *remove*(), *listContents*().
   - If the client has to check "*is it a file or a folder*?" before calling methods, the code becomes full of *if-else* or *instanceof* checks.
2. Code Becomes *Complex* and *Hard* to *Maintain*
   - The *client* needs to know the *exact type* (File or Folder) every time.
   - Extending the system (adding a new type like *Shortcut*) requires changes in client code.
3. *Inconsistent* Operations
   - Some operations make sense for both files and folders (like *getName()* or *delete()*), but others don't.
   - Without a unified design, you might duplicate code or *create confusing APIs*.

***Solution* Composite Design Pattern**:

Composite provides a *common interface* (Component) for both *File* and *Folder*.

- *Uniform Treatment*:
  - Both *File* and *Folder implement* the same *Component interface* (e.g., showDetails(), getSize()).
  - Client code just calls the method, without worrying if it's a file or folder.
- *Hierarchical Tree Structure*:
  - A *Folder* (composite) can hold a *collection* of *Component objects*, which could be *Files* (leaves) or other *Folders* (nested composites).
  - This naturally models the *recursive structure* of a real file system.
- *Extensibility*:
  - If later we *add new types* (like Shortcut, CompressedFile), we just make them *implement Component* — no changes needed in client code.

***UML***:




***Code link:*** *https://github.com/sibasundarj8/-System-Design-/tree/main/Codes/19___Composite%20Design%20Pattern%20code*

***Standard UML*:**