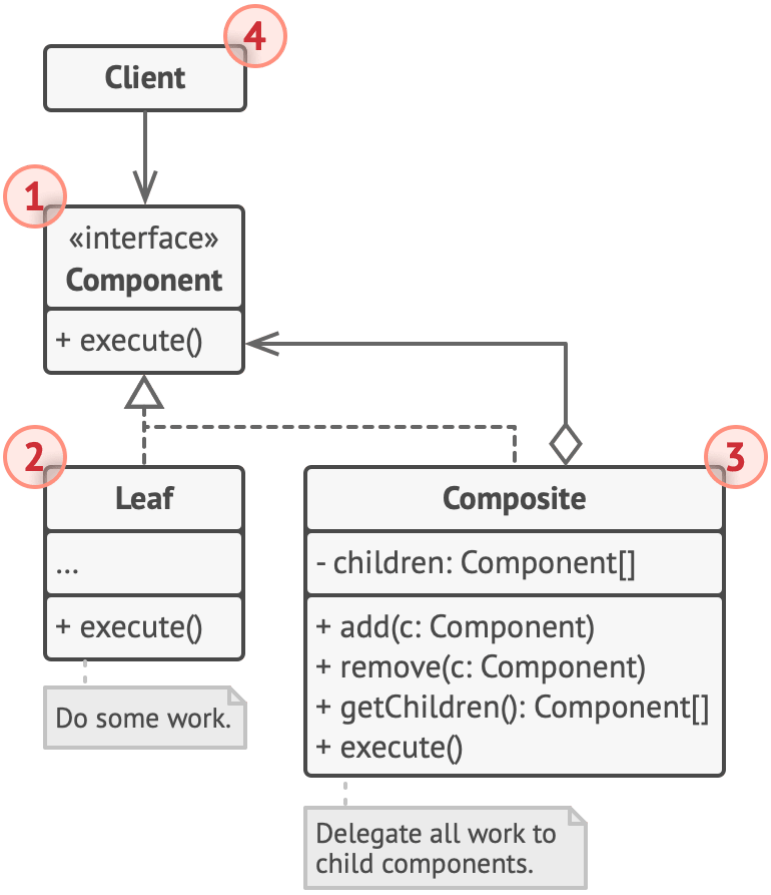
division is a set of brigades, and a brigade consists of platoons, which can be broken down into squads. Finally, a squad is a small group of real soldiers. Orders are given at the top of the hierarchy and passed down onto each level until every soldier knows what needs to be done.

Structure



1. The **Component** interface describes operations that are common to both simple and complex elements of the tree.

2. The **Leaf** is a basic element of a tree that doesn't have sub-elements.

   Usually, leaf components end up doing most of the real work, since they don't have anyone to delegate the work to.

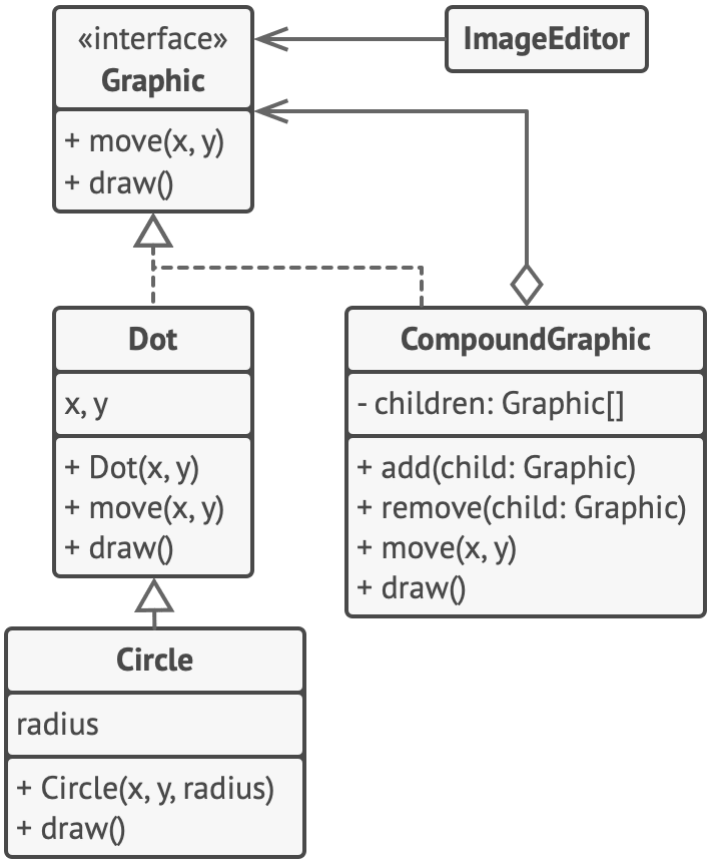
3. The **Container** (aka *composite*) is an element that has sub-elements: leaves or other containers. A container doesn't know the concrete classes of its children. It works with all sub-elements only via the component interface.

intermediate results and then returns the final result to the client.

4. The **Client** works with all elements through the component interface. As a result, the client can work in the same way with both simple or complex elements of the tree.

Pseudocode

In this example, the **Composite** pattern lets you implement stacking of geometric shapes in a graphical editor.



The geometric shapes editor example.

The `CompoundGraphic` class is a container that can comprise any number of sub-shapes, including other compound shapes. A compound shape has the same methods as a simple shape. However, instead of doing something on its own, a compound shape passes the request recursively to all its children and "sums up" the result.

The client code works with all shapes through the single interface common to all shape classes. Thus, the client doesn't know whether it's working with a simple shape or a compound one. The