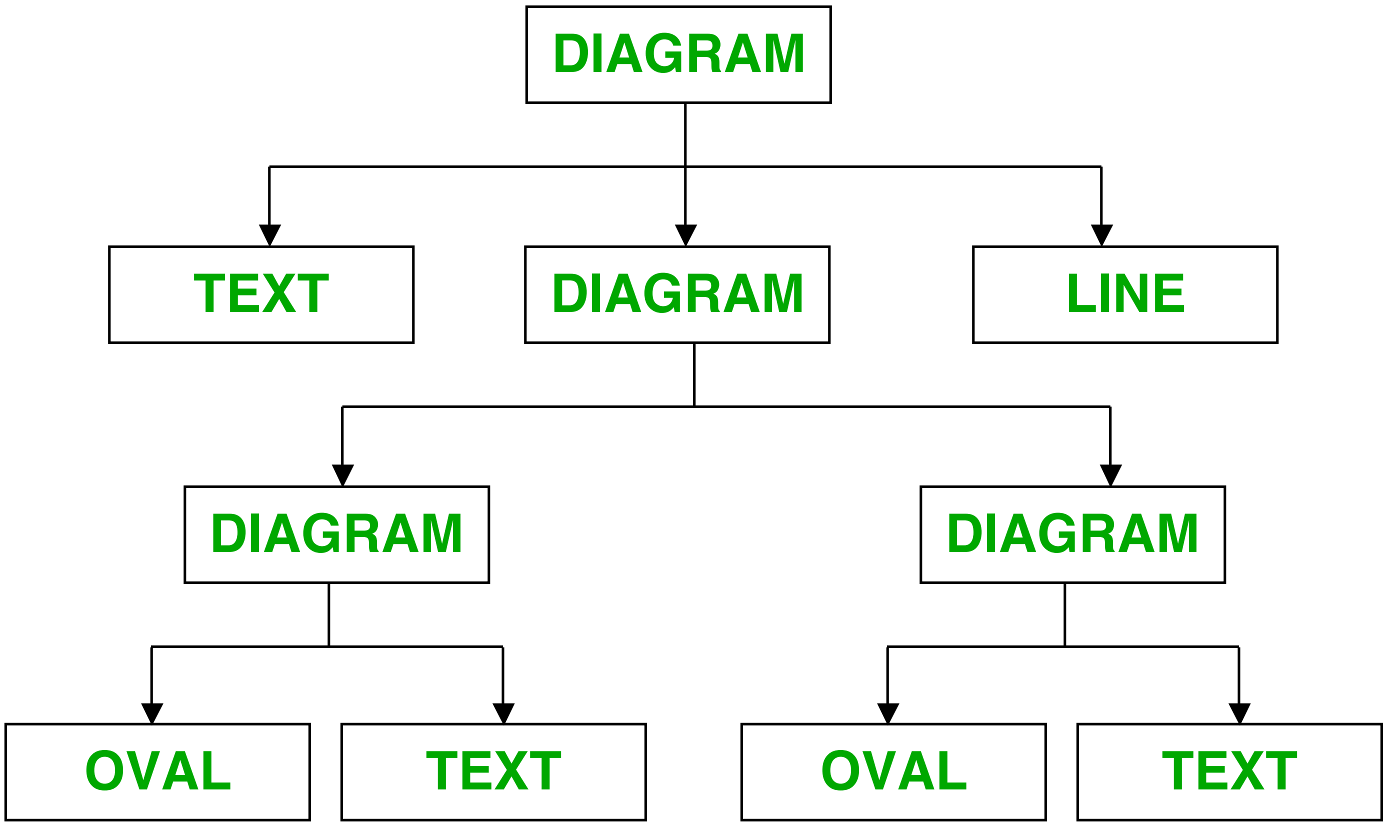
Composite Pattern – Structural

- Intent
  - » Compose objects into tree structures representing part-whole hierarchies
  - » Clients deal uniformly with individual objects and hierarchies of objects

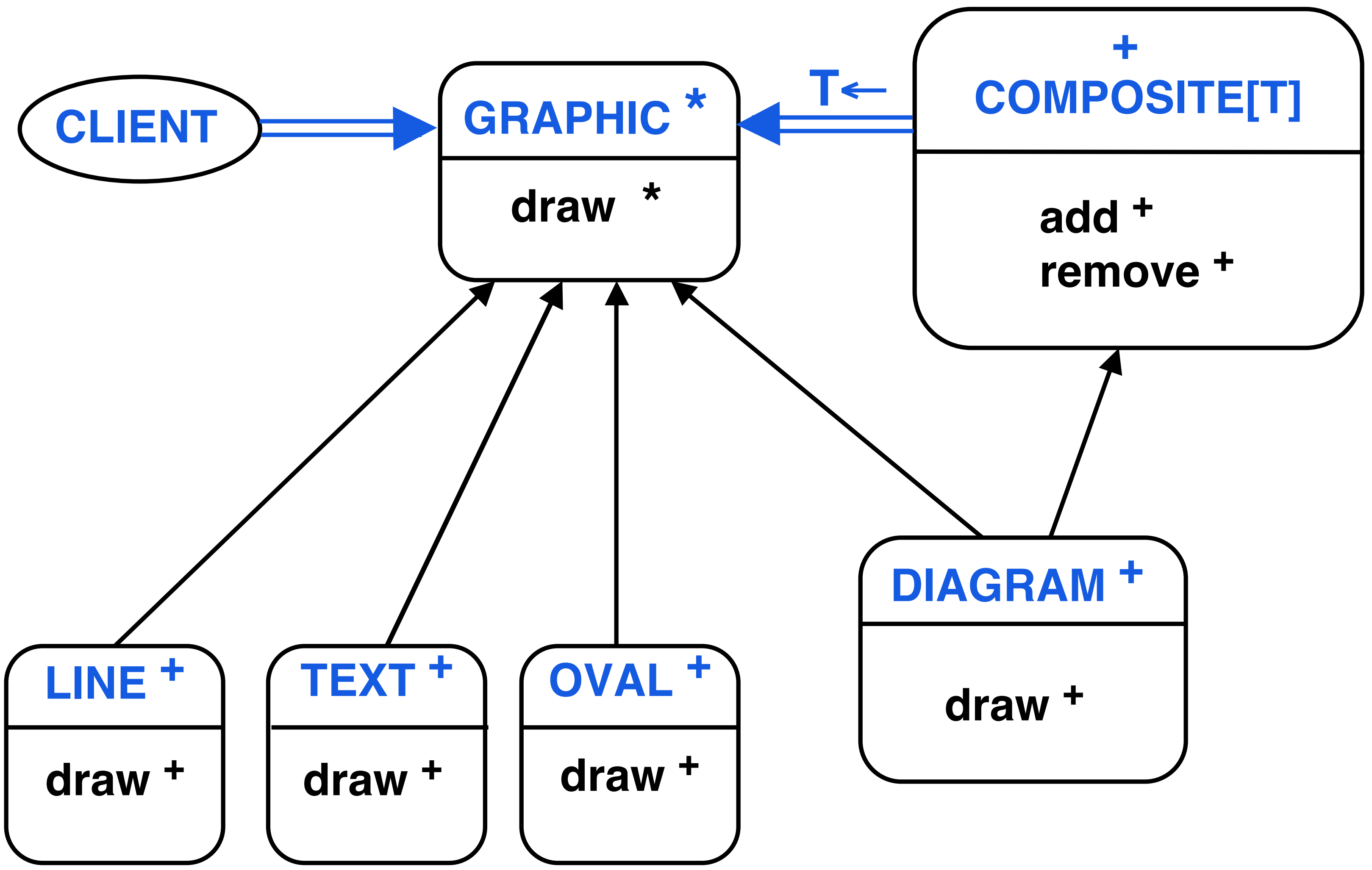
Composite – Motivation

- Applications that have recursive groupings of primitives and groups
  - » **Drawing programs**
    - **lines, text, figures and groups**
  - » **Eiffel static structure**
    - **classes and clusters**
- Operations on groups are different than primitives but users treat them in the same way

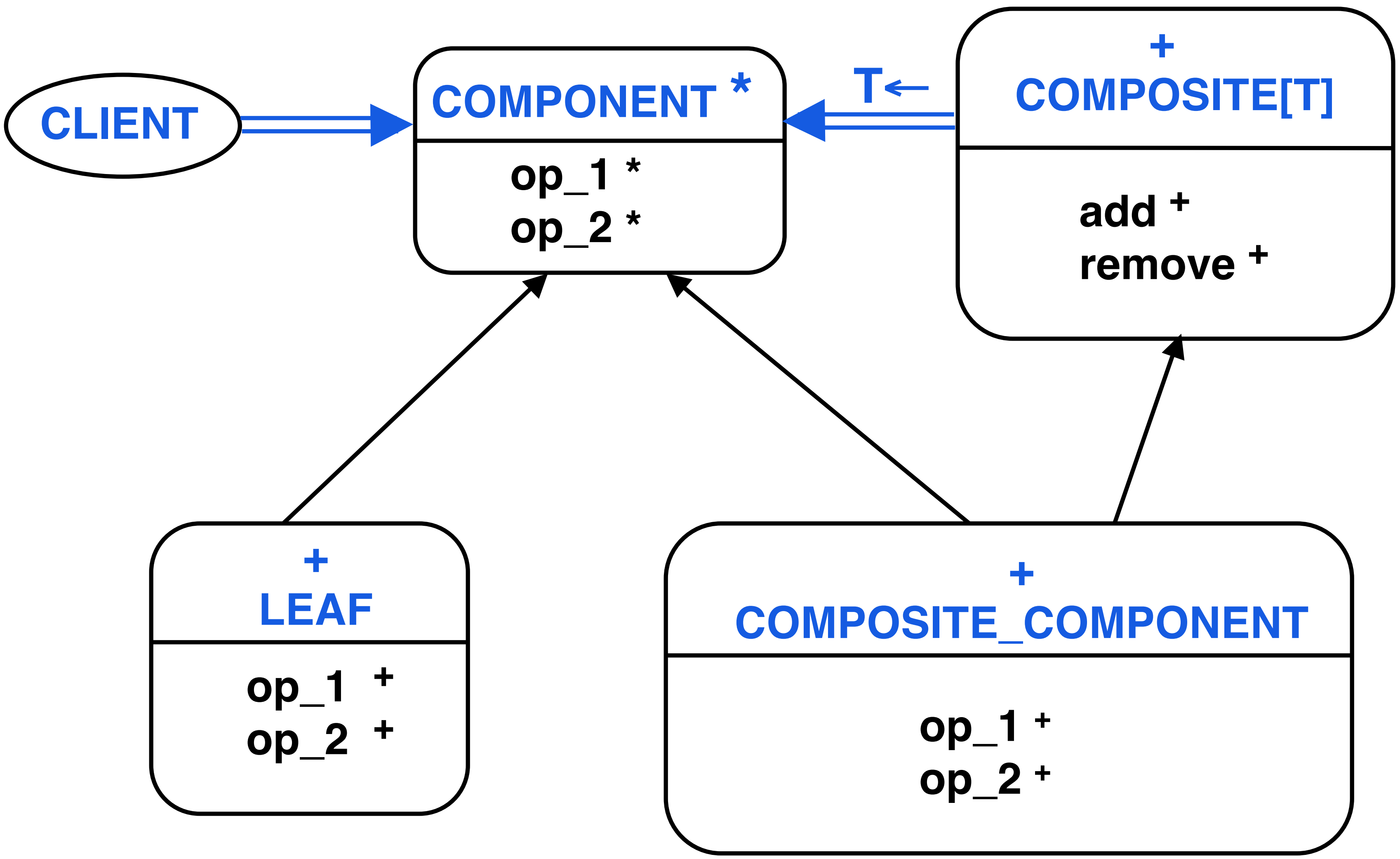
Composite – Drawing Example



Composite – Example Architecture



Composite – Abstract Architecture



Composite – Applicability

- Represent part-whole hierarchies of objects
- Clients can ignore difference between individual objects and compositions
- Clients deal with all objects in a composition in the same way

Composite – Participants

- Component

  Defines properties of an entity

- Leaf

  Defines properties of a primitive entity

- Composite

  Declares properties of a collection of entities

- Composite Component

  Combines properties of a collection of entities and properties of a primitive entity

- Client

  Uses component and composite properties

Composite – Consequences

- Whenever client expects a primitive it can accept a composite
- Client is simplified by removing tag-case statements to identify parts of the composition
- Easy to add new components by subclassing, client does not change
- If compositions are to have restricted sets of components have to rely on run-time checking

Composite – Related Patterns

- Component-parent link is a Chain of Responsibility
- Decorator is used together with composite but then decorators have to support add, remove, iterator
- Flyweight permits sharing components but cannot refer to parents
- Iterator can be used to traverse composites
- Visitor localizes operations that would be distributed across composite and leaf classes