

Outline

- Motivation for design patterns
- Overview of design patterns
- How to learn design patterns
- What is (not) a design pattern
- Sorting example
- More patterns
- Benefits and drawbacks of patterns

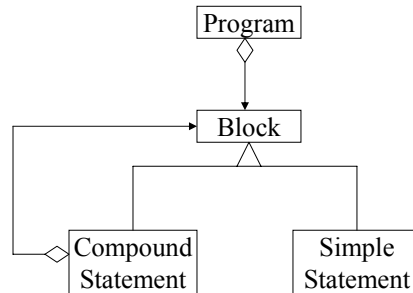
60

Composite Pattern

- **Intent:** Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.
- **Motivation:** If the composite pattern is not used, client code must treat primitive and container classes differently, making the application more complex than is necessary.

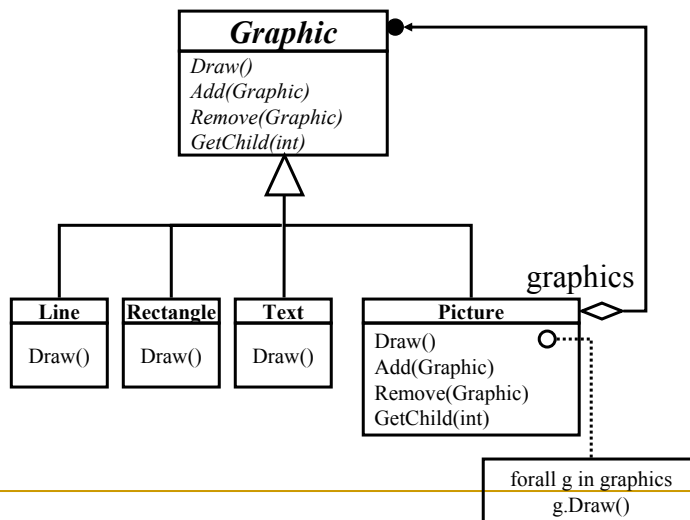
61

Example of Composite



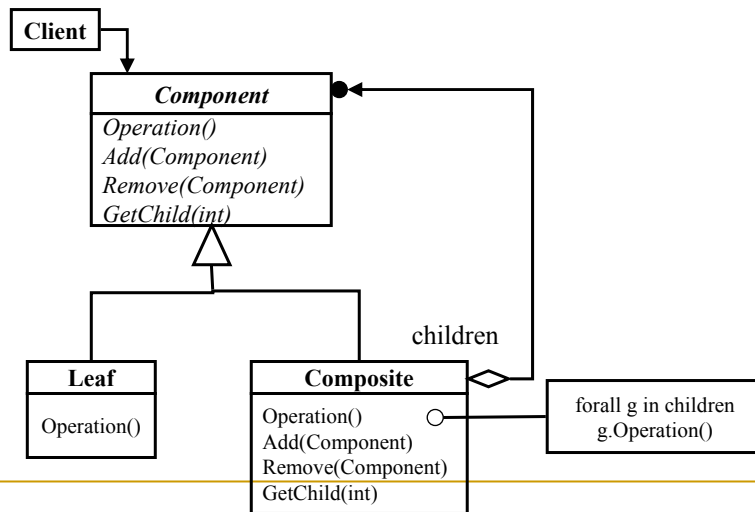
62

Example of Composite



63

Composite Pattern



64

Participants of Composite

- **Component:**
 - Declares the interface for objects in the composition.
 - Implements default behavior for the interface common to all classes.
 - Declares an interface for accessing and managing its child components.
 - Defines an interface for accessing a component's parent in the recursive structure (optional).

65

Participants of Composite

■ **Composite:**

- ❑ Defines behavior for components having children.
- ❑ Stores child components.
- ❑ Implements child-related operations in the component interface.

Participants of Composite

■ **Leaf:**

- ❑ Represents leaf objects in the composition. A leaf has no children.
- ❑ Defines behavior for primitive objects in the composition.

■ **Client:**

- ❑ Manipulates objects in the composition through the component interface.

Observer Pattern

- **Intent:** Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

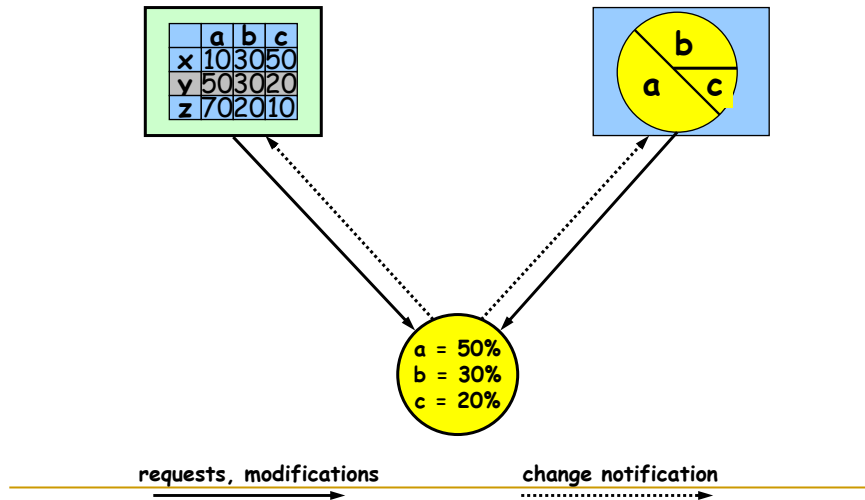
68

Observer Pattern

- **Motivation:**
 - A common side-effect of partitioning a system into a collection of cooperating classes is the need to maintain consistency between related objects.
 - You don't want to achieve consistency by making the classes tightly coupled, because that reduces their reusability.

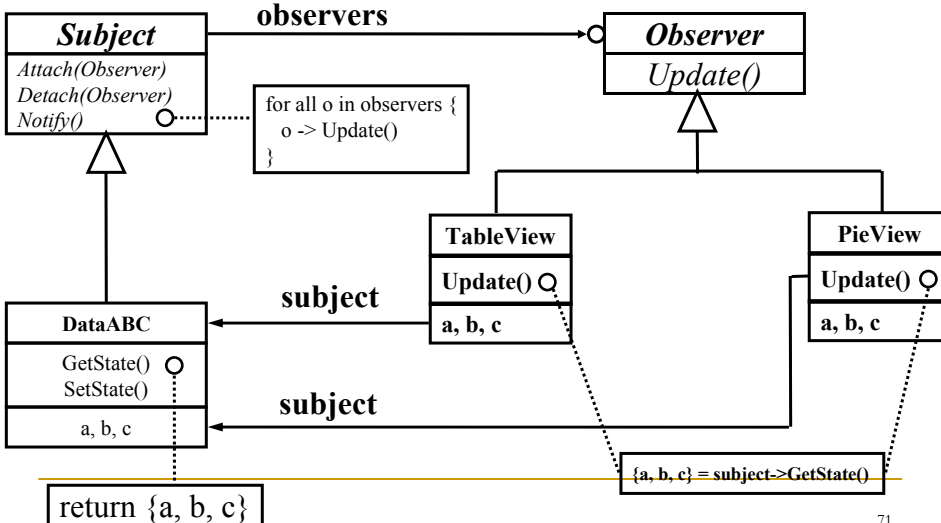
69

Example of Observer



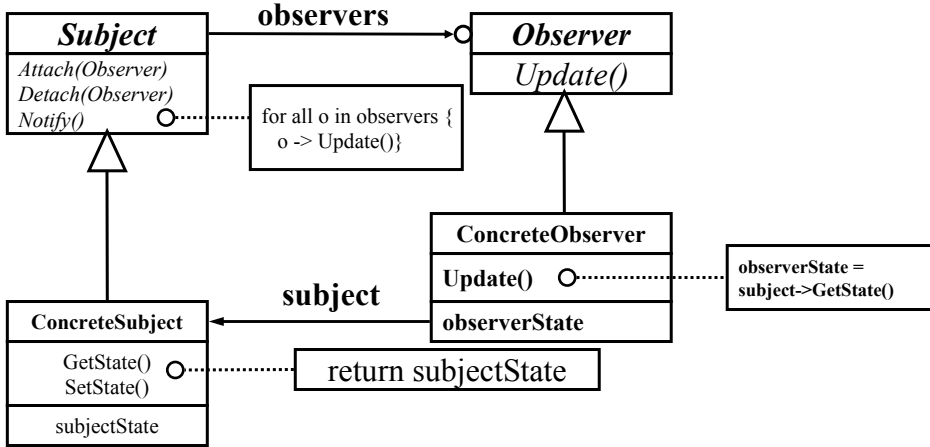
70

Observer Pattern

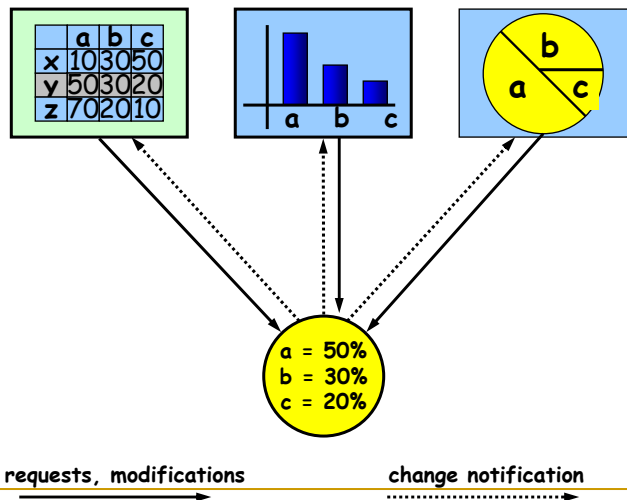


71

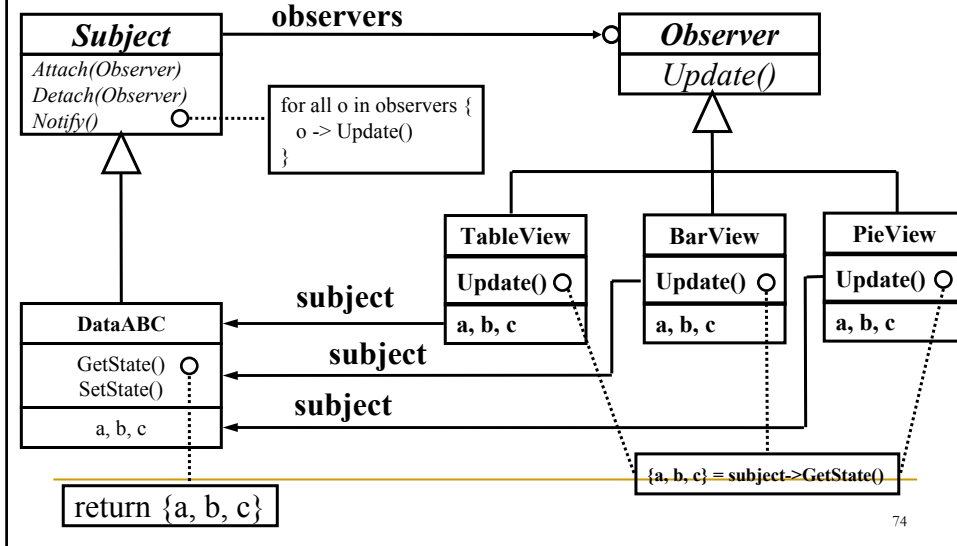
Observer Pattern



Example of Observer



Observer Pattern



74

Observer Pattern

- The key objects in this pattern are **subject** and **observer**.
 - A subject may have any number of dependent observers.
 - All observers are notified whenever the subject undergoes a change in state.

75

Participants of Observer

■ **Subject:**

- ❑ Knows its numerous observers.
- ❑ Provides an interface for attaching and detaching observer objects.
- ❑ Sends a notification to its observers when its state changes.

■ **Observer:**

- ❑ Defines an updating interface for concrete observers.

76

Participants of Observer

■ **Concrete Subject:**

- ❑ Stores state of interest to concrete observers.

■ **Concrete Observer:**

- ❑ Maintains a reference to a concrete subject object.
- ❑ Stores state that should stay consistent with the subject's.
- ❑ Implements the updating interface.

77