

Design Patterns 2

Composite – Decorator - Visitor

Quan Thanh Tho

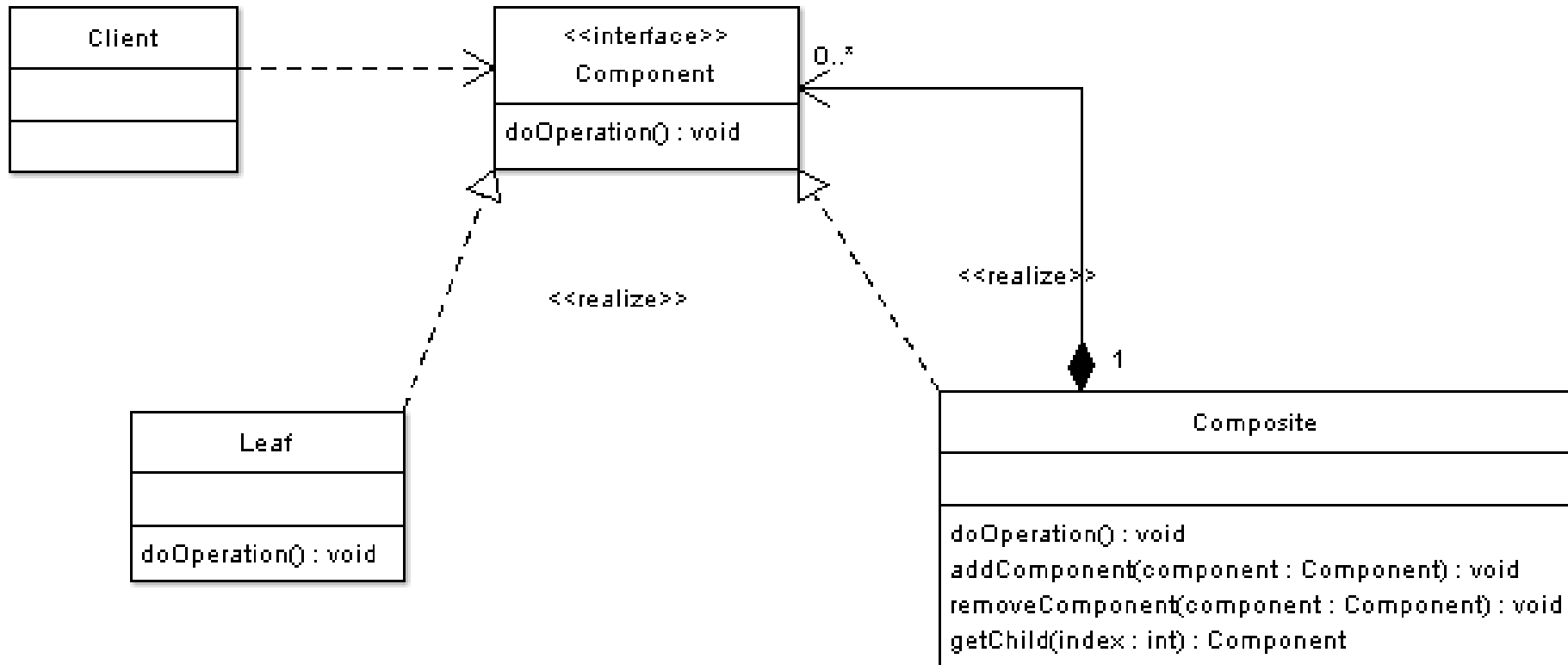
Agenda

- Composite
- Decorator
- Visitor

Composite

- Intent
 - The intent of this pattern is to compose objects into tree structures to represent part-whole hierarchies.
 - Composite lets clients treat individual objects and compositions of objects uniformly.

Class Diagram

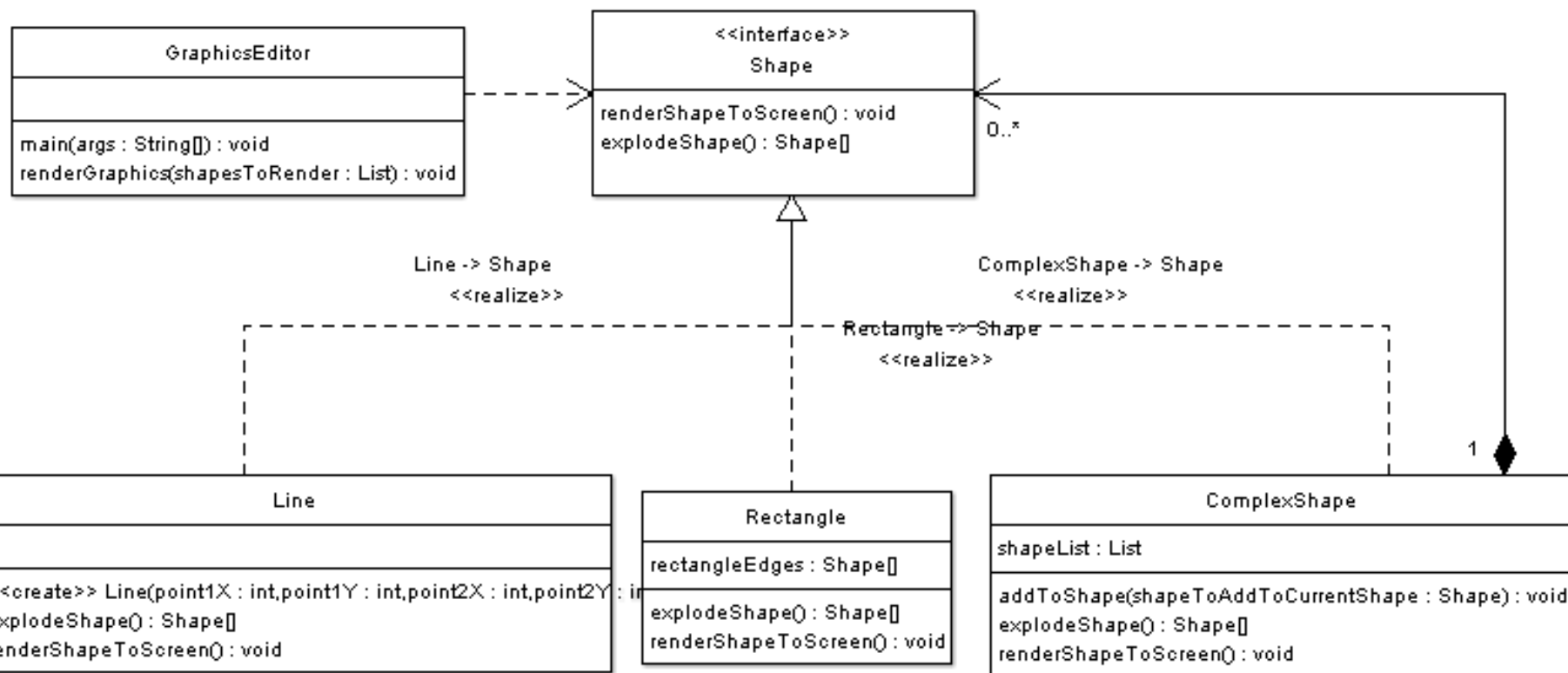


Participant Classes

- **Component** - Component is the abstraction for leafs and composites.
- **Leaf** - Leafs are objects that have no children..
- **Composite** - A Composite stores child components in addition to implementing methods defined by the component interface.
- **Client** - The client manipulates objects in the hierarchy using the component interface.

Example

- In graphics editors a shape can be basic or complex.
- A simple shape is a line, where a complex shape is a rectangle which is made of four line objects.
- Shapes have many operations in common such as rendering the shape to screen



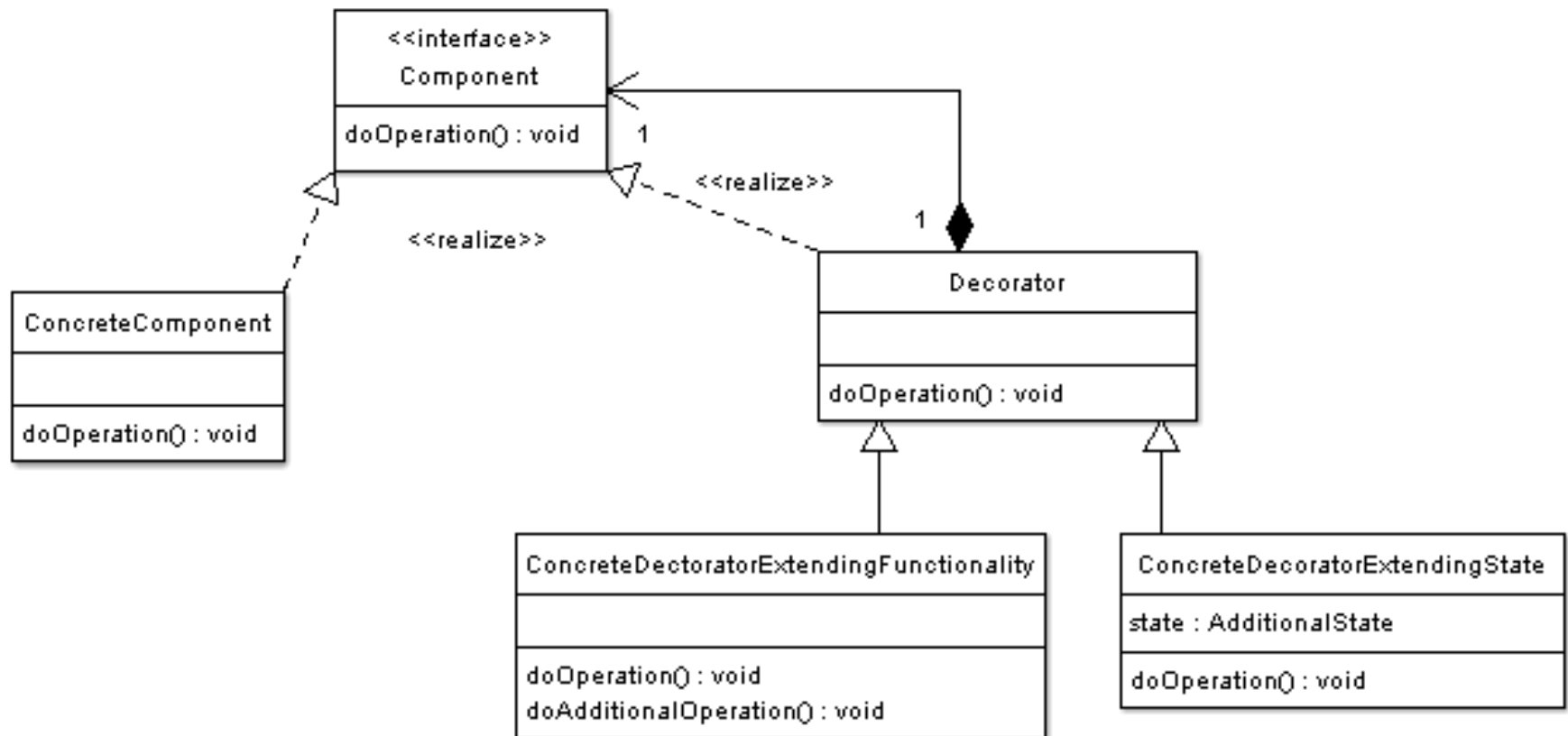
Code

- <http://www.oodesign.com/composite-pattern-shapes-example-java-sourcecode.html>

Decorator

- Intent
 - The intent of this pattern is to add additional responsibilities dynamically to an object.

Class Diagram

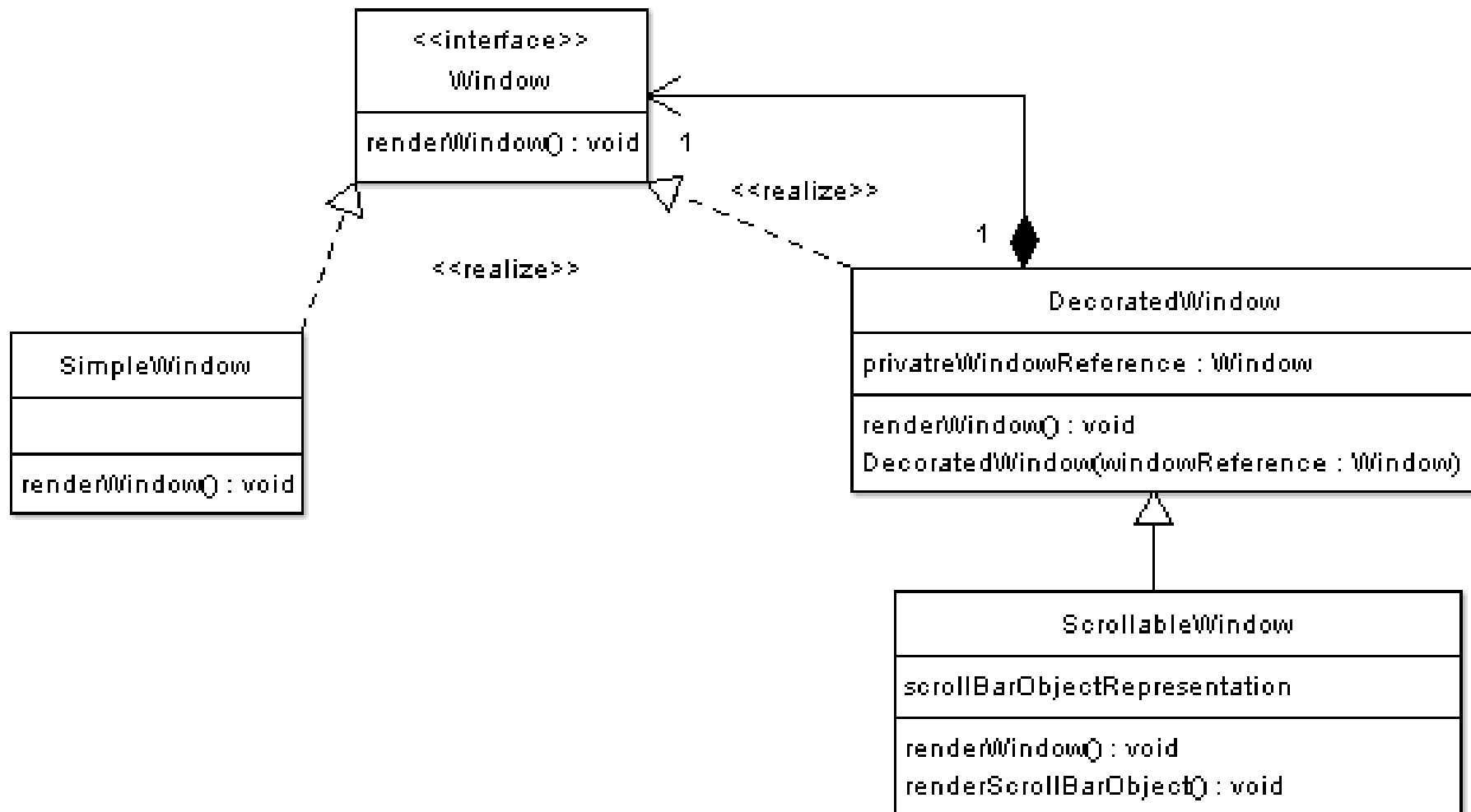


Participant Classes

- **Component** - Interface for objects that can have responsibilities added to them dynamically.
- **ConcreteComponent** - Defines an object to which additional responsibilities can be added.
- **Decorator** - Maintains a reference to a Component object and defines an interface that conforms to Component's interface.
- **Concrete Decorators** - Concrete Decorators extend the functionality of the component by adding state or adding behavior.

Example

- GUI Windows which can have scrollbar when needed

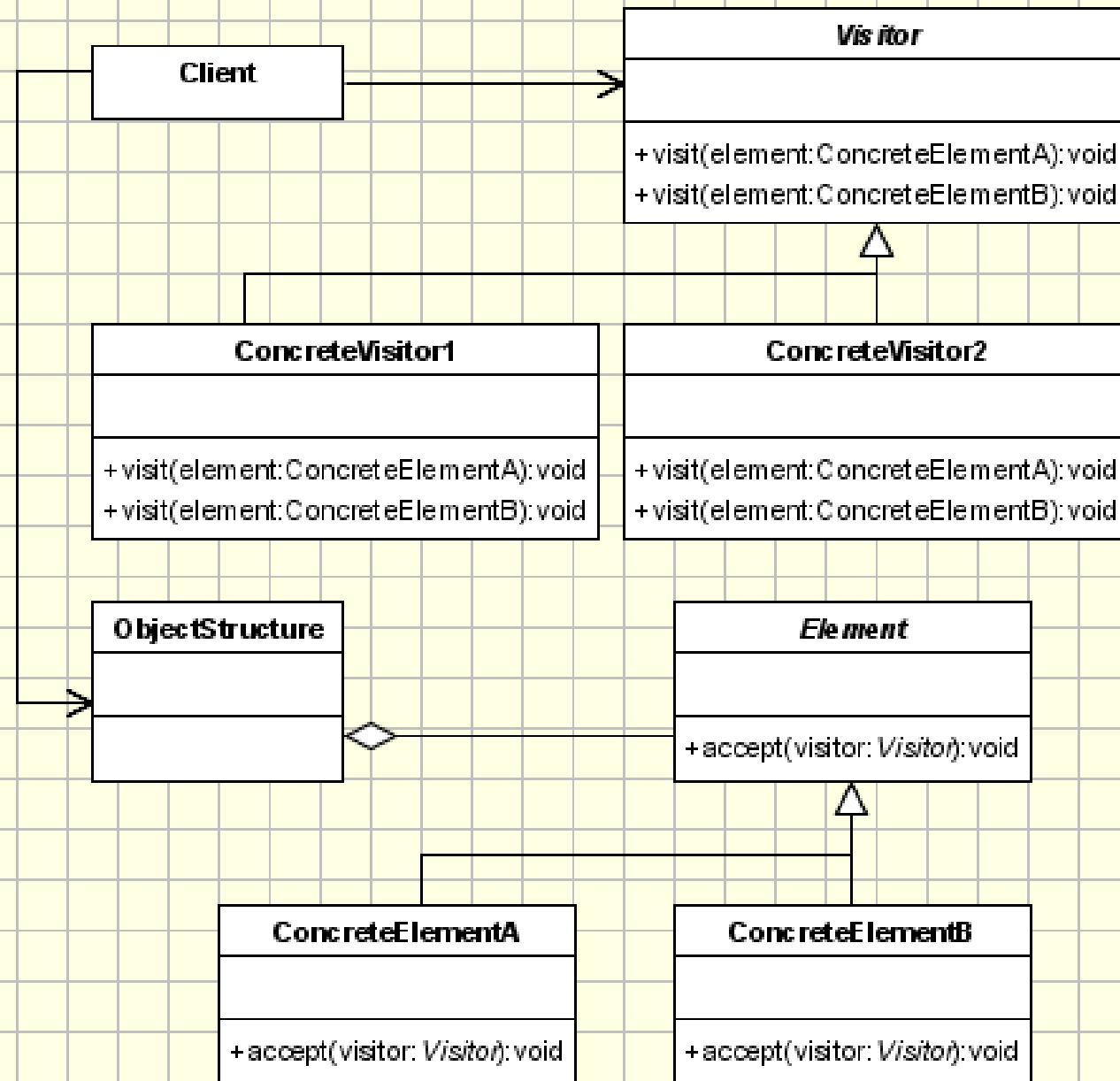


Code

<http://www.oodesign.com/decorator-pattern-gui-example-java-sourcecode.html>

Visitor

- Intent
 - Represents an operation to be performed on the elements of an object structure.
 - Visitor lets you define a new operation without changing the classes of the elements on which it operates.



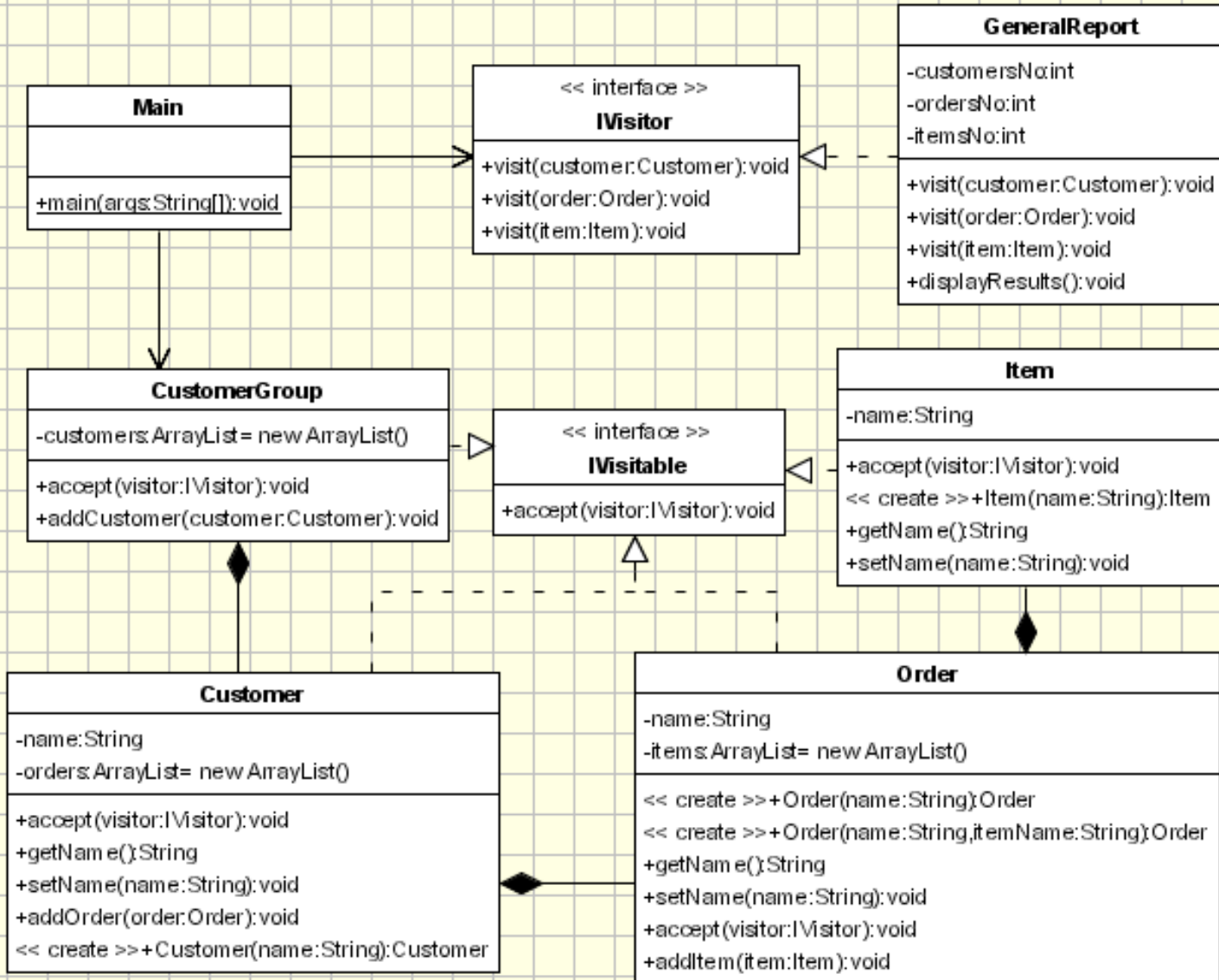
Participant Classes

- **Visitor** - This is an interface or an abstract class used to declare the visit operations for all the types of visitable classes.
- **ConcreteVisitor** - For each type of visitor all the visit methods, declared in abstract visitor, must be implemented.
- **Visitable** - is an abstraction which declares the accept operation.
- **ConcreteVisitable** - Those classes implements the Visitable interface or class and defines the accept operation.
- **ObjectStructure** - This is a class containing all the objects that can be visited. |

Example

- We want to create a reporting module in our application to make statistics about a group of customers.
- The statistics should be made very detailed so all the data related to the customer must be parsed.

cd: Visitor Customers Example -UML Class Diagram



Code

- <http://www.oodesign.com/visitor-pattern-customers-report-java-sourcecode.html>