Design Patterns

A short introduction

Definition

- A *pattern* is a recurring solution to a standard problem, in a context

Elements of Design Patterns

- Design patterns have 4 essential elements:
  - Pattern name: increases vocabulary of designers
  - Problem: intent, context, when to apply
  - Solution: UML-like structure, abstract code
  - Consequences: results and tradeoffs

Three Types of Patterns

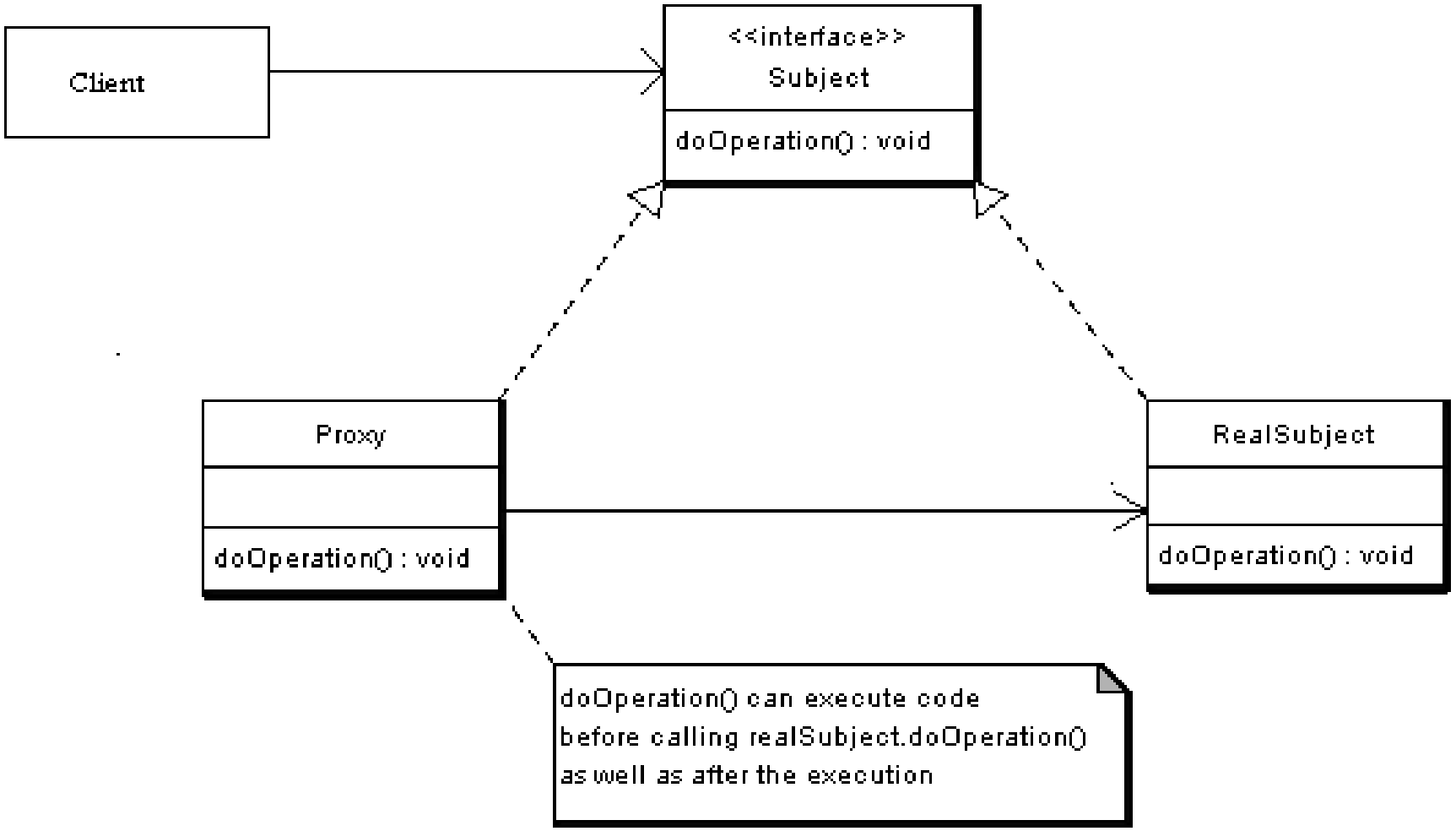
- **Creational patterns:**
  - Deal with initializing and configuring classes and objects
- **Structural patterns**:
  - Deal with decoupling interface and implementation of classes and objects
  - Composition of classes or objects
- **Behavioral patterns**:
  - Deal with dynamic interactions among societies of classes and objects
  - How they distribute responsibility

The “gang of four” (GoF) Design Pattern Classification

Creational	Structural	Behavioral
Factory Method	Adapter	Interpreter
Abstract Factory	Bridge	Template Method
Builder	Composite	Chain of Responsibility
Prototype	Decorator	Command
Singleton	Flyweight	Iterator
	Facade	Mediator
	Proxy	Memento
		Observer
		State
		Strategy
		Visitor

The Proxy Pattern

- A Structural Pattern
- Provides surrogate for another object, controls access to real target object
- Separates interface from implementation
- Real target may not always be instantiated directly due to performance, location or access restrictions
- One of the simpler patterns


Client
<<interface>>
Subject
doOperation() : void
Proxy
doOperation() : void
RealSubject
doOperation() : void
doOperation() can execute code
before calling realSubject.doOperation()
as well as after the execution

Example Application in Image Processing

- Loading image data when it is needed