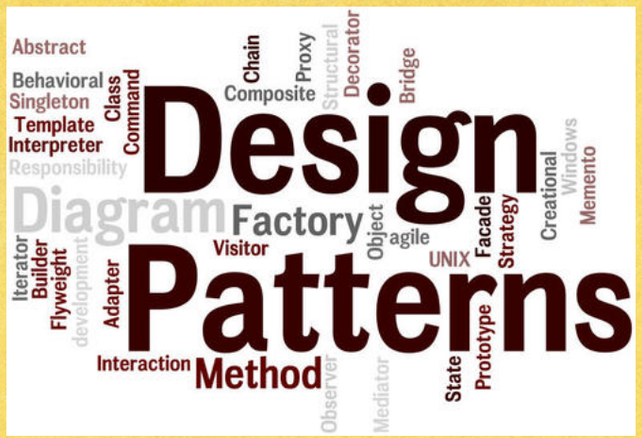
Using Design Patterns in App Development



Michael P. Redlich

@mpredli

Who's Mike?

- BS in CS from RUTGERS THE STATE UNIVERSITY OF NEW JERSEY
- "Petrochemical Research Organization"
- InfoQ Java Queue News Editor
- Garden State Java User Group
- Jakarta EE Ambassadors

@mpredli

Objectives

- What are Design Patterns?
- How Design Patterns Solve Design Problems
- Design Pattern Categories
- Review Factory, Decorator and Observer Design Patterns (with demos!)

What are Design Patterns?

What are Design Patterns? (1)

- Recurring solutions to software design problems that are repeatedly found in real-world application development
- All about the design and interaction of objects

What are Design Patterns? (2)

- Four essential elements:
  - pattern name
  - problem
  - solution
  - consequences

What are Design Patterns? (3)

- A pattern is a solution to a problem in a context
- A context is the situation to which a pattern applies

What are Design Patterns? (4)

- The problem refers to the desired goal in the context, but also refers to any constraints that may occur
- The solution is a general design that anyone can apply

How Design Patterns Solve Design Problems

Design Patterns Solve Design Problems (1)

- Help identify less obvious abstractions
- Clients should only know about abstract classes that define an interface
- Reduce implementation dependencies

Design Patterns Solve Design Problems (2)

- Avoid:
  - directly creating objects
  - dependencies on specific operations
  - algorithmic dependencies
  - tight coupling

Thinking in Design Patterns

- Keep to simple
- Design patterns are not the “magic bullet”
- Know when to apply a design pattern
- Don’t be afraid to remove a design pattern

Design Pattern

Categories

Design Pattern Categories

- Creational
  - Abstract the instantiation process
  - Dynamically create objects so they don't have to be instantiated directly
- Structural
  - Compose groups of objects into larger structures

Design Pattern Categories

- Behavioral
  - Define communication among objects in a given system
  - Provide better control of flow in a complex application

Creational Patterns

- Abstract Factory
- Builder
- Factory Method
- Prototype
- Singleton

Structural Patterns

- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

Behavioral Patterns

- Chain of Responsibility
- Command
- Interpreter
- Iterator
- Mediator
- Memento
- Observer
- State
- Strategy
- Template Method
- Visitor

What are OO Design Principles?

- A set of underlying principles for creating flexible designs that are easy to maintain and adaptable to change
- Understanding the basics of OOP isn't enough

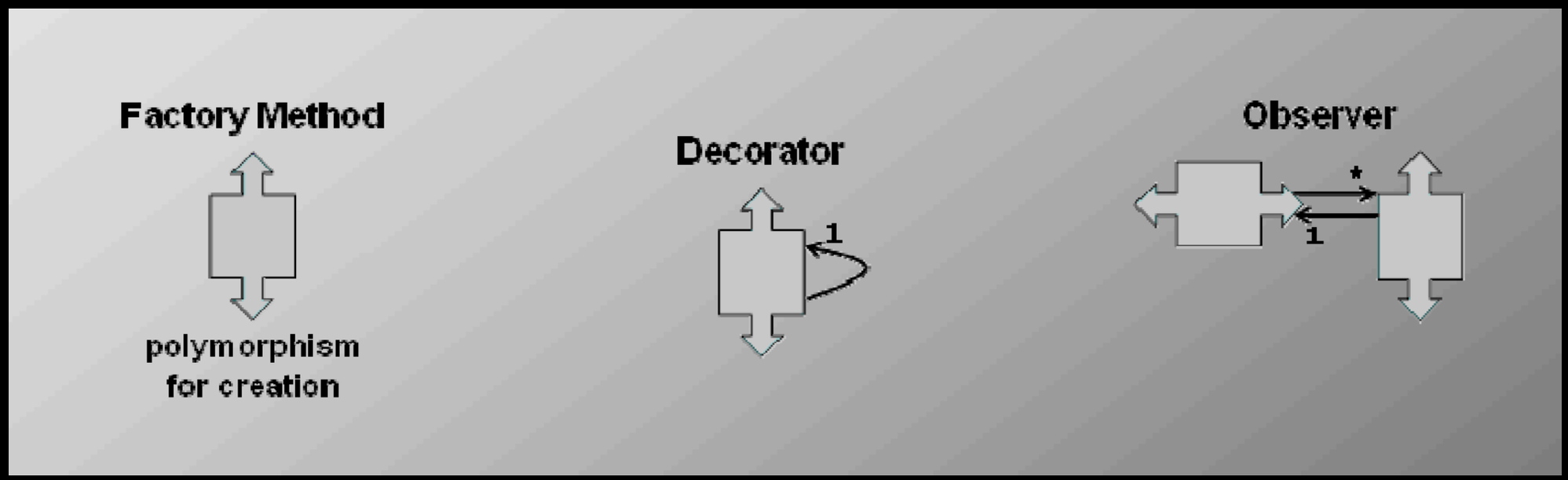
Some OO Design Principles (1)

- Encapsulate What Varies
- Program to Interfaces, Not Implementations
- Favor Composition Over Inheritance
- Classes Should Be Open for Extension, But Closed for Modification

Some OO Design Principles (2)

- Strive for Loosely Coupled Designs Between Objects That Interact
- A Class Should Have Only One Reason to Change

Let's Check Out These Patterns

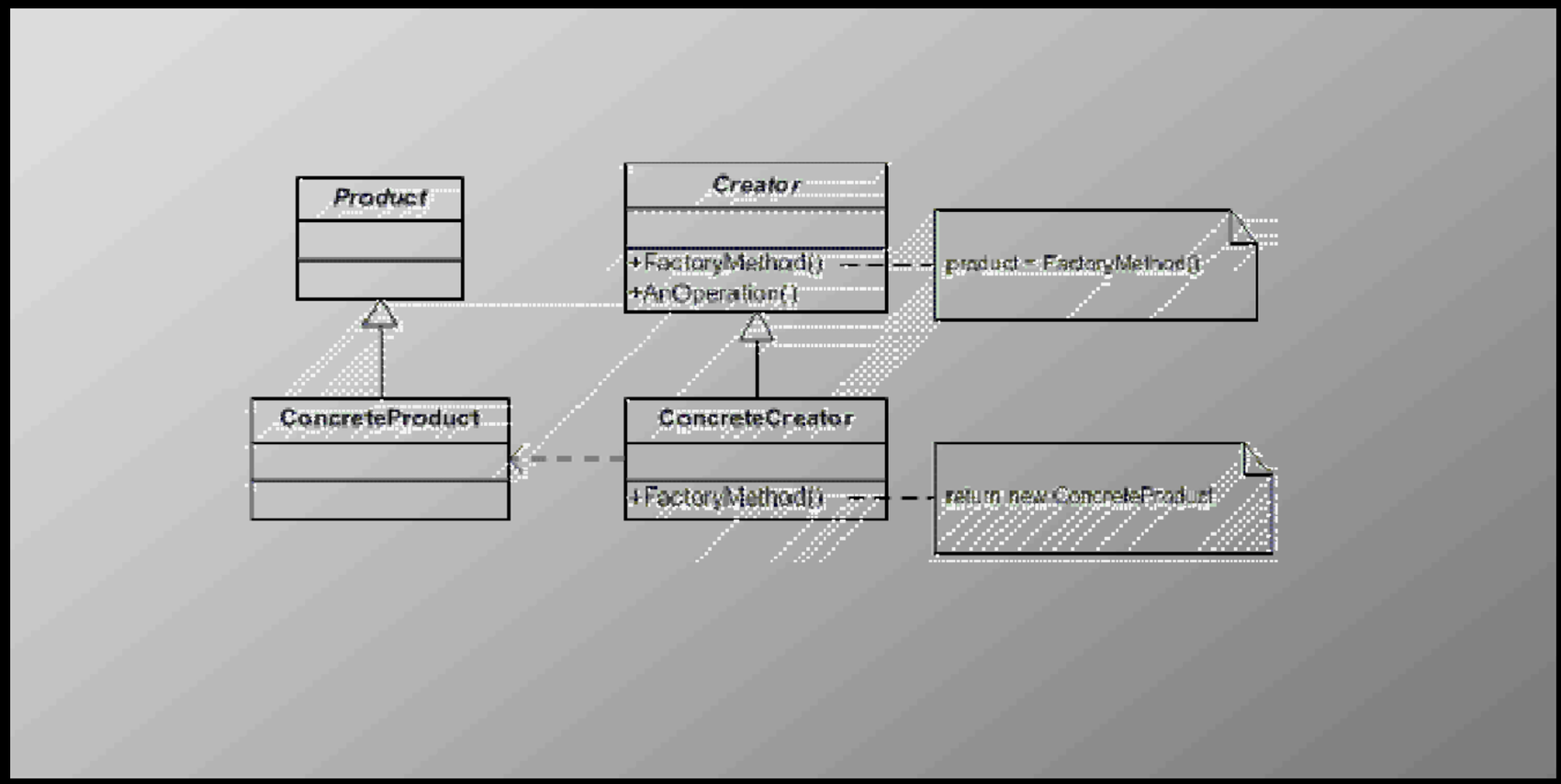


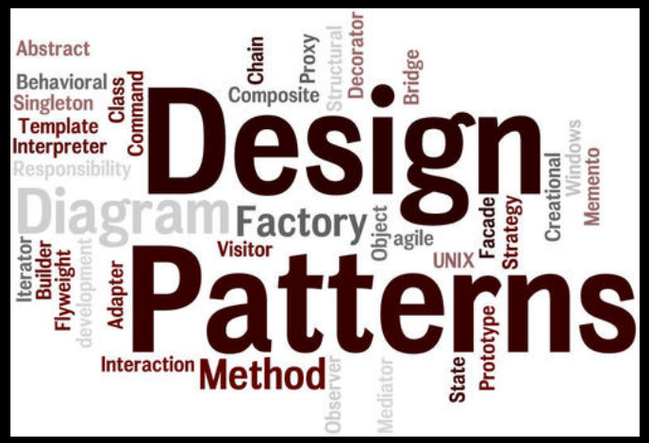
@mpredli

Factory Method

- Intent
  - Defines an interface for creating an object, but lets subclasses decide which classes to instantiate
- Design Principles
  - Depend on abstractions, not concrete classes
  - AKA The Dependency Inversion Principle

Generic Design



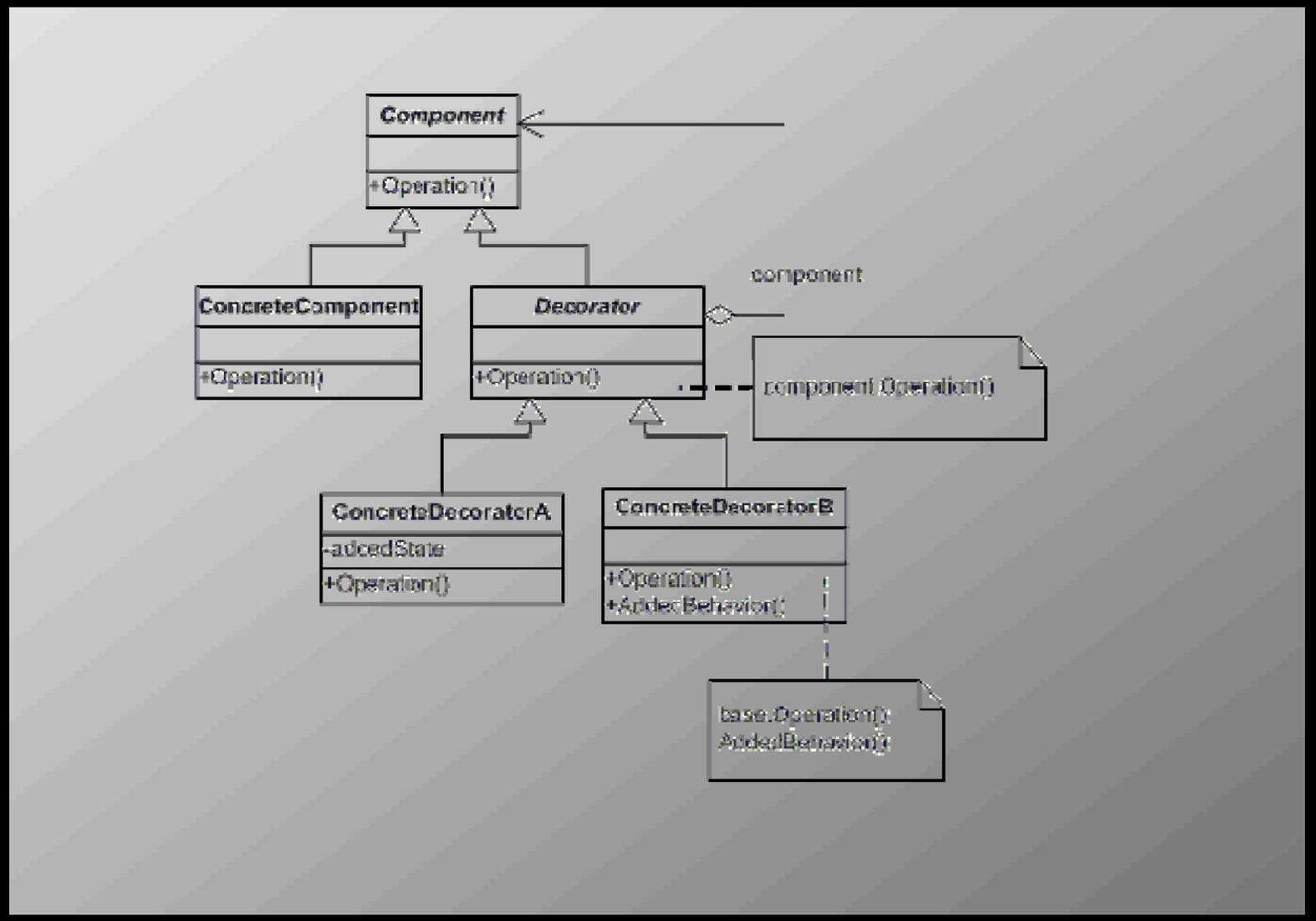

Abstract
Behavioral
Singleton
Template
Interpreter
Responsibility
Class
Command
Chain
Composite
Proxy
Structural
Decorator
Bridge
Design
Diagram
Factory
Visitor
Object
agile
UNIX
Facade
Strategy
Creational
Windows
Memento
Iterator
Builder
Flyweight
development
Adapter
Patterns
Interaction
Method
Observer
Mediator
State
Prototype

Demo

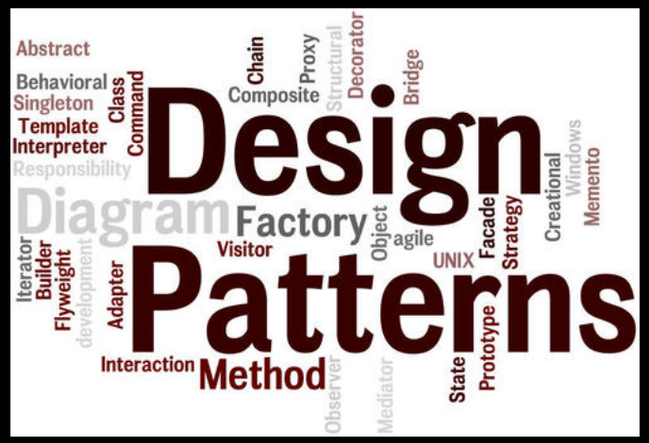
Decorator

- Intent
  - Dynamically attaches additional responsibilities to an object
  - Provides an alternative to subclassing
- Design Principle
  - Classes should be open for extension, but closed for modification

Generic Design



@mpredli

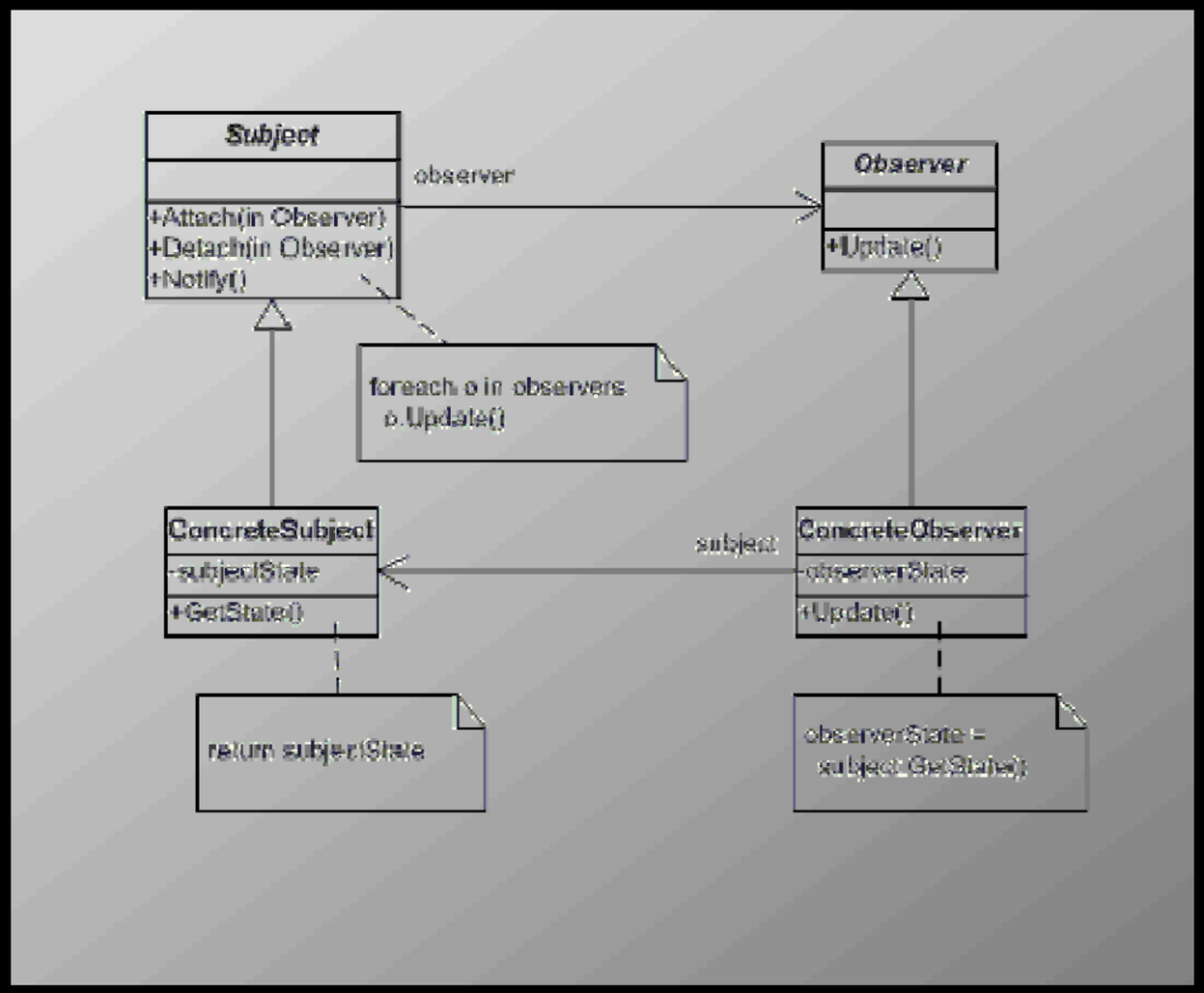

Abstract
Behavioral
Singleton
Template
Interpreter
Responsibility
Class
Command
Chain
Composite
Proxy
Structural
Decorator
Bridge
Design
Diagram
Factory
Visitor
Object
agile
UNIX
Facade
Strategy
Creational
Windows
Memento
Iterator
Builder
Flyweight
development
Adapter
Patterns
Interaction
Method
Observer
Mediator
State
Prototype

Demo

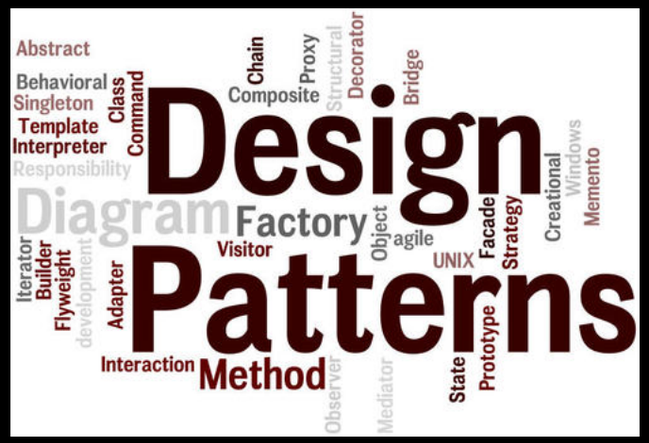
Observer

- Intent
  - Defines a one-to-many dependency among objects such that when one object changes state, all of its dependents are notified and updated
- Design Principle
  - Strive for loosely coupled designs that interact

Generic Design

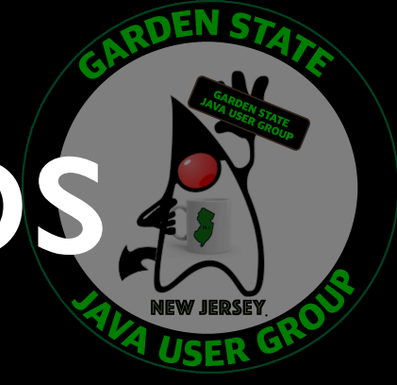


@mpredli


Abstract
Behavioral
Singleton
Template
Interpreter
Responsibility
Class
Command
Chain
Composite
Proxy
Structural
Decorator
Bridge
Design
Diagram
Factory
Visitor
Object
agile
UNIX
Facade
Strategy
Creational
Windows
Memento
Iterator
Builder
Flyweight
development
Adapter
Patterns
Interaction
Method
Observer
Mediator
State
Prototype

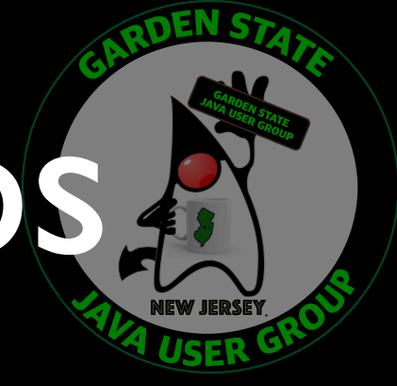
Demo

Local Java User Groups (1)



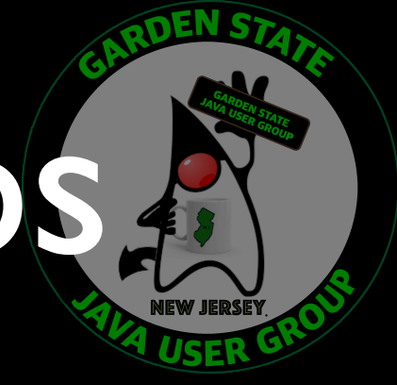
- Garden State Java Users Group (GSJUG)
  - facilitated by the GSJUG Leadership Team
  - `gsjug.org`
- NYJavaSIG
  - facilitated by Frank Greco, et.al
  - `javasig.com`

Local Java User Groups (2)



- PhillyJUG
  - facilitated by Paul Burton, et. al.
  - `meetup.com/PhillyJUG`
- Jersey City Java Users Group
  - facilitated by Amitai Schleier
  - `meetup.com/Jersey-City-Java-User-Group-JC-JUG/`

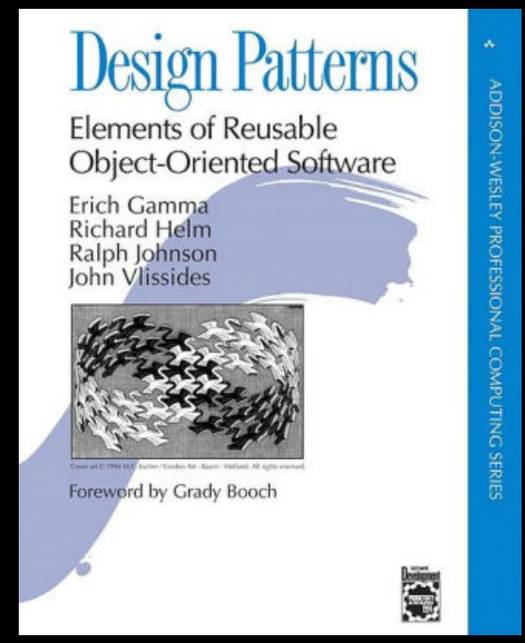
Local Java User Groups (3)



- Capital District Java Developers Network
  - facilitated by Dan Patsey
  - `cdjdn.com`
  - currently restructuring

Gang of Four

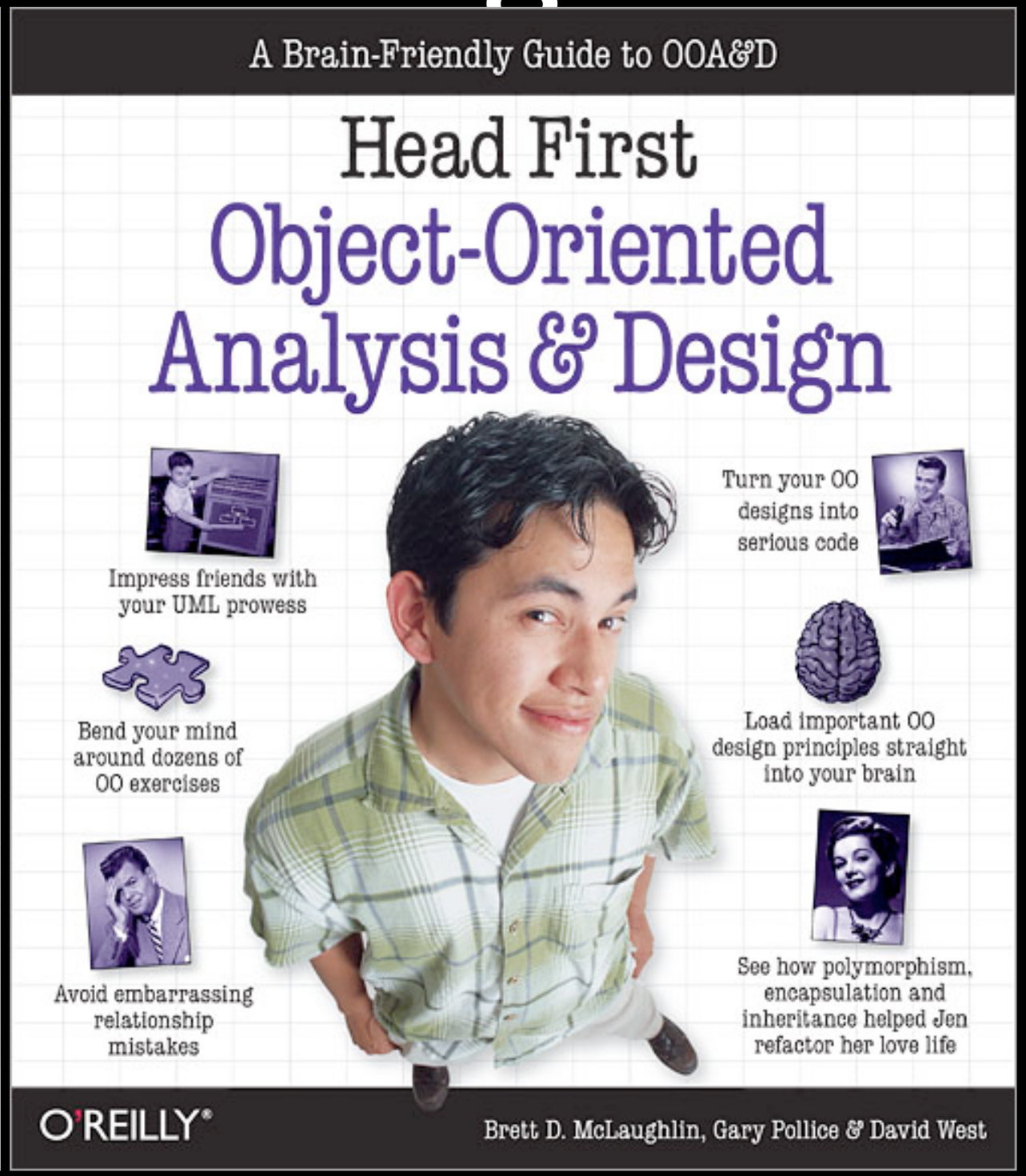
- Erich Gamma
- Richard Helm
- Ralph Johnson
- John Vlissides
- Design Patterns - Elements of Reusable Object-Oriented Software



Gang of Four Next Generation?

- Eric Freeman
- Elisabeth Freeman
- Kathy Sierra
- Bert Bates
- Head First Design Patterns

Further Reading



Resources

- `java.sun.com`
- `headfirstlabs.com`
- `themeteorbook.com`
- `eventedmind.com`
- `atmosphere.meteor.com`

Contact Info

mike@redlich.net

@mpredli

redlich.net

redlich.net/portfolio

github.com/mpredli01

Thanks!

@mpredli