Information System Design

Lecture 5: Creational Design Patterns

Dr. Moustafa Alzantot

Design Patterns (revisited)

Design Pattern

- A solution for a recurring problem in a large OOP system.
- Highly optimal since it is designed and revised by experts.
- Abstract from any particular programming language.

Design Patterns (revisited)

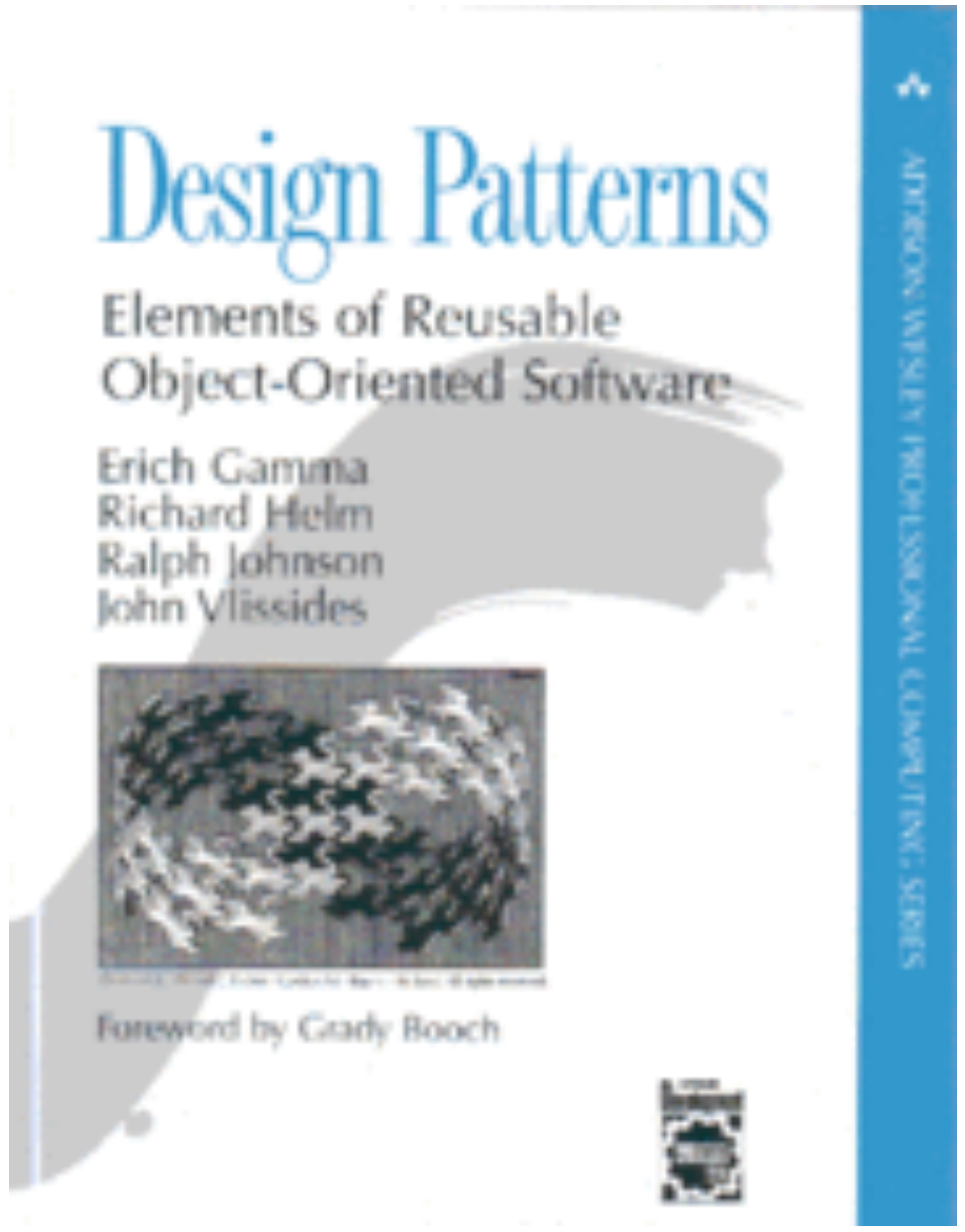
Benefits of Applying Design Patterns

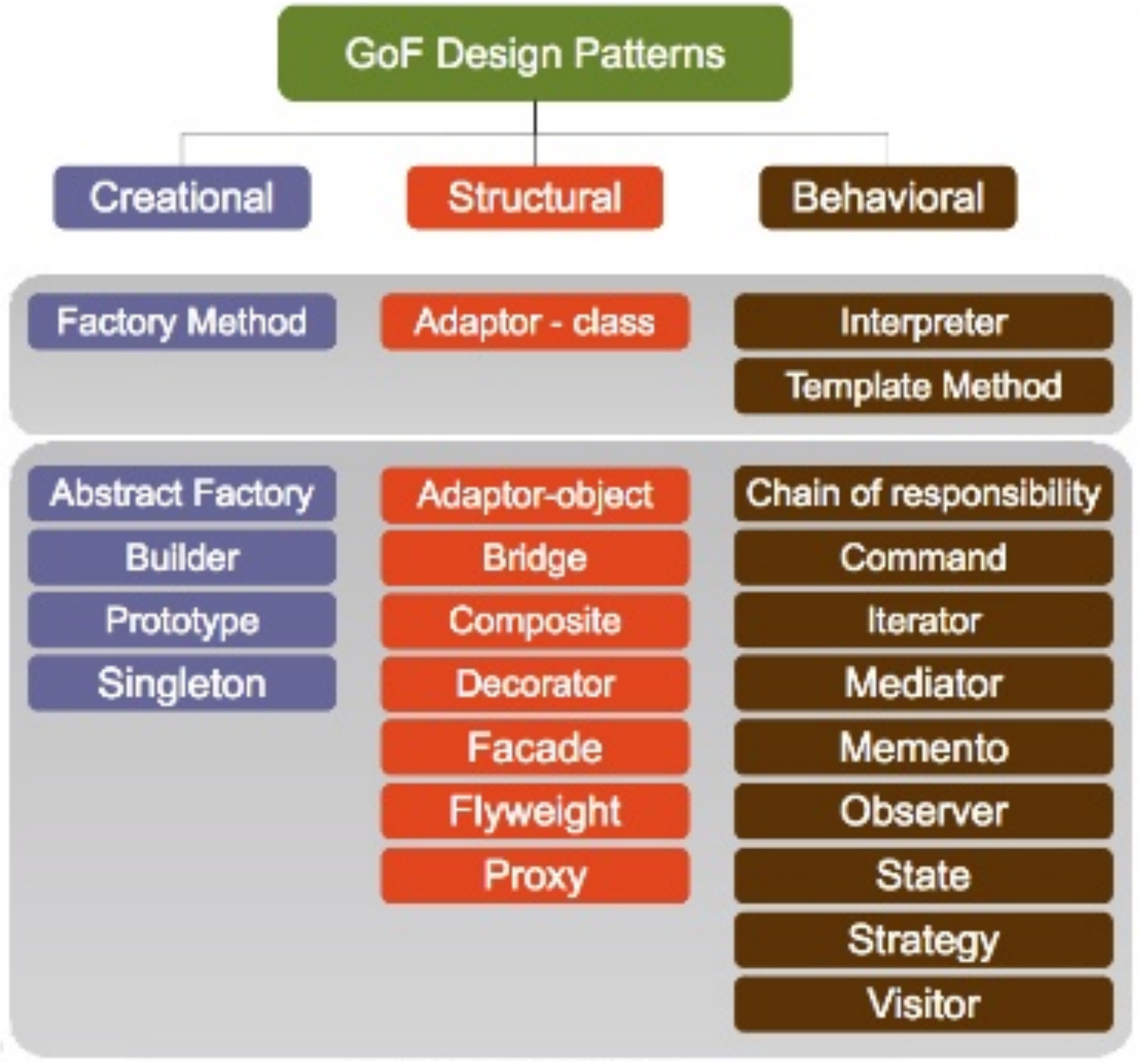
- Increases design speed and quality.
- Promotes design reuse.
- Makes it easier to other developers to understand the system design since it provides a standard vocabulary and building blocks.

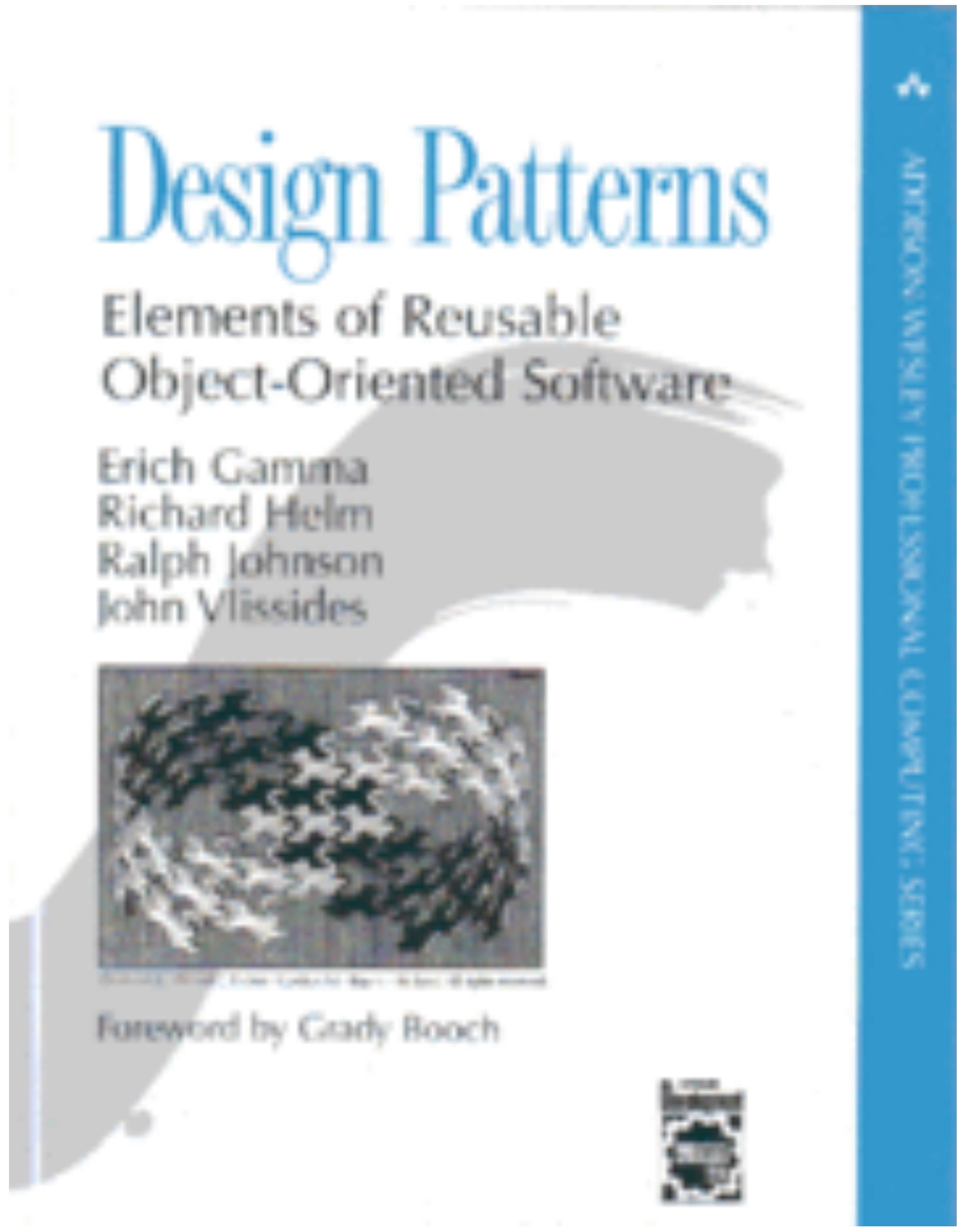
- GoF book 1995

Design Patterns: Elements of Reusable Object-Oriented Software

- Describes 23 design patterns are categorized by their purpose into 3 categories:
    - **Creational**
    - **Structural**
    - **Behavioral**


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES




Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Design Patterns (revisited)

Problem: In what situation should this pattern be used?

Solution: What should you do? What is the pattern?

- describe details of the objects/classes/structure needed
- should be somewhat language-neutral

Advantages: Why is this pattern useful?

Disadvantages: Why might someone not want this pattern?

Singleton Pattern

Singleton: An object that is the only object of its type.

(one of the most known / popular design patterns)

Problem:

- Ensures that a class has at most one instance.
- Providing a global access point to that instance.
    - e.g. providing an accessor method that allows users to get that instance anywhere in the program.

Singleton

Example use cases:

- Logger that saves program status updates to disk or somewhere else.
- Hardware interface class.
- User preferences in a mobile application.
- Database connection.

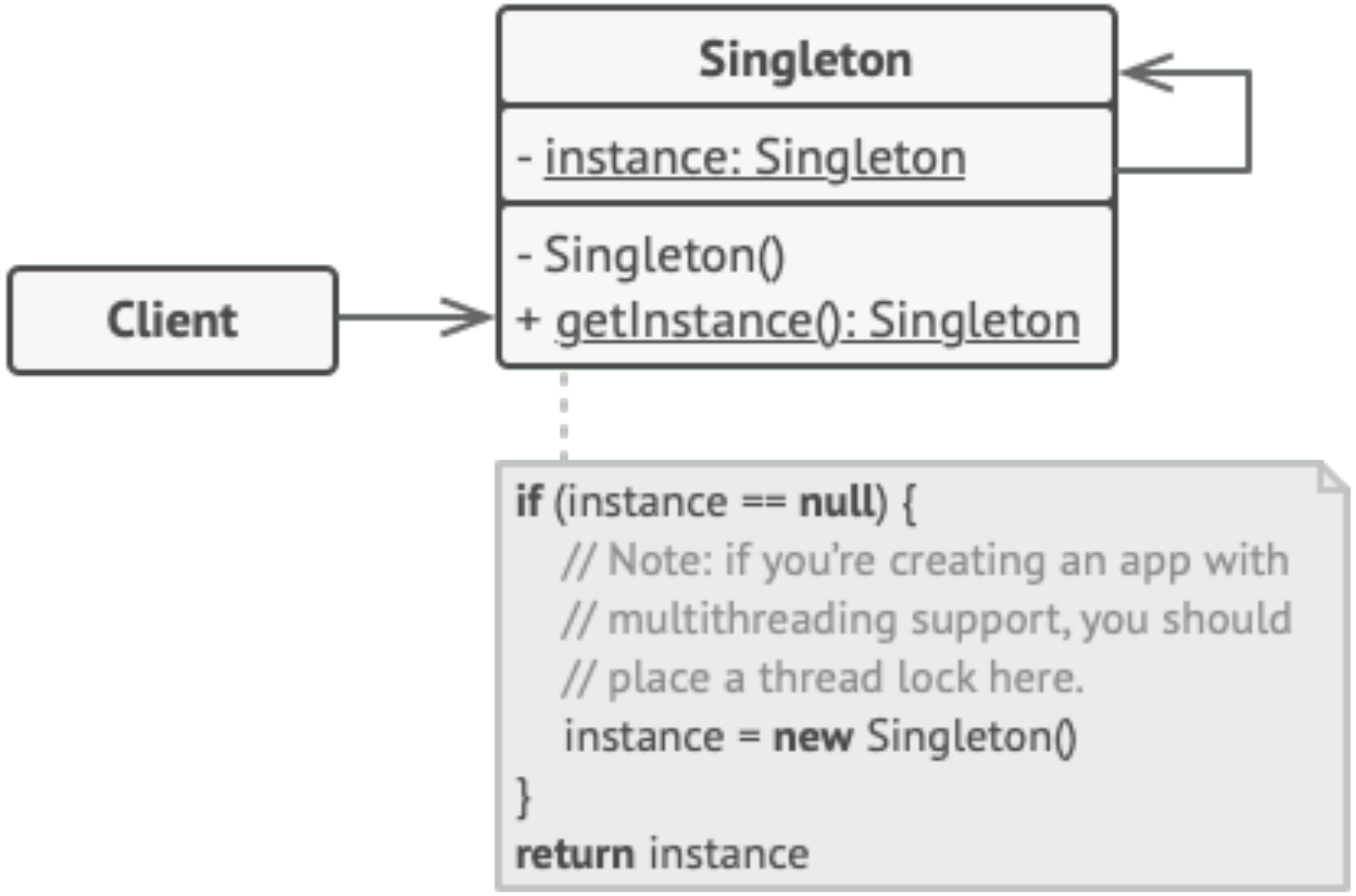
What else ??!

Singleton : Implementation

Design Recipe:

- Make constructor(s) private, so that they can not be called from outside by clients.
- Declare a single (`private static`) instance of the class.
- Write a public `GetInstance()` or similar method that allows access to that single instance.
    - It is possible to use lazy initialization to create this instance only when needed.
    - May need to protect/ synchronize this method to ensure that it will work well in a multi-threaded program.

Singleton



```
if (instance == null) {
   // Note: if you're creating an app with
   // multithreading support, you should
   // place a thread lock here.
   instance = new Singleton()
}
return instance
```

Singleton

Benefits of using Singleton:

- Takes responsibility of managing that instance away from the programmer (it becomes illegal to create more than one instance).
- Save the cost (memory and time) of creating multiple instances.
- Avoids bugs due to having multiple instances with inconsistent states.

Singleton

Disadvantages of using Singleton:

- The code becomes tightly coupled and harder to test. This could violate the dependency inversion principle.
- Could also lead to a violation of SRP; because it combines the responsibility of object creation and other business logic.