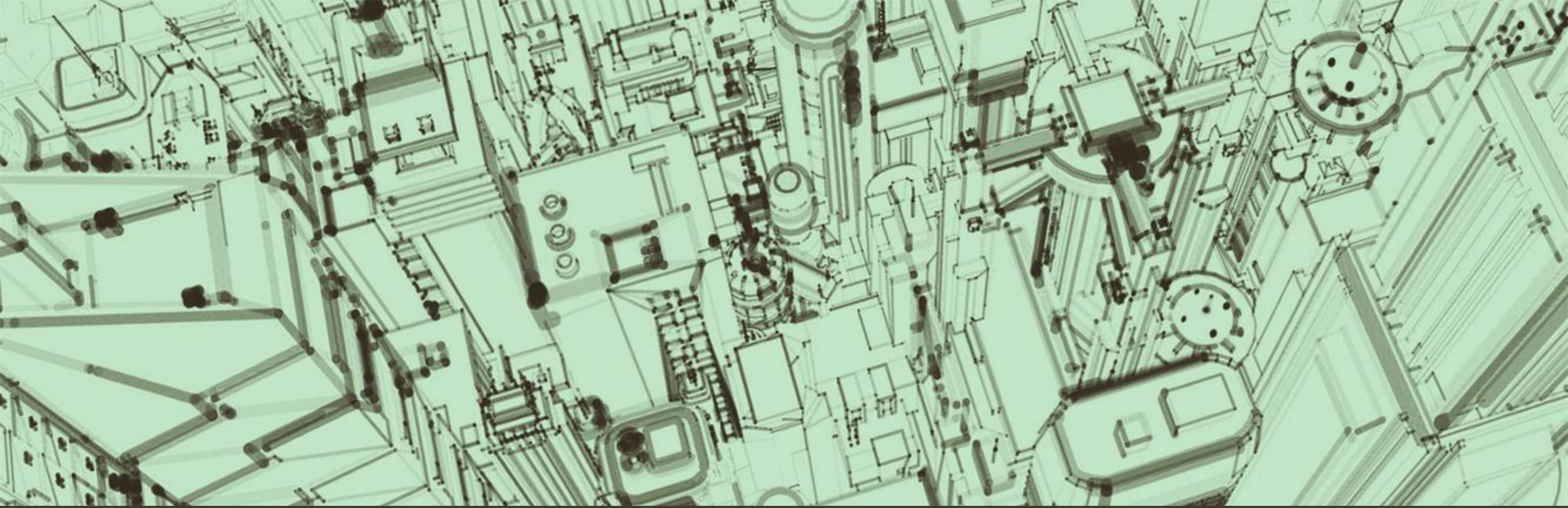


Java Design Patterns

Efthimios Alepis

Why use Design Patterns?

- ✓ Design Patterns are already defined and provide industry standard approaches to solve recurring problems, so they can save time
- ✓ There are many java design patterns that we can use in our java based projects.
- ✓ Using design patterns promotes reusability that leads to more robust and highly maintainable code
- ✓ It helps in reducing total cost of ownership (TCO) of the software product
- ✓ Since design patterns are already defined, this makes our code easy to understand and debug
- ✓ Lead to faster development since new members of software teams understand them more easily

More Advantages

- They are reusable in multiple projects
- They provide the solutions that help to define the system architecture
- They capture the software engineering experiences
- They provide transparency to the design of an application
- They are well-proved and tested solutions since they have been built upon the knowledge and experience of expert software developers
- Design patterns don’t guarantee an absolute solution to a problem. They provide clarity to the system architecture and increase the possibility of building a better system

When should we use the design patterns?

- We must use the design patterns during the analysis and requirement phase of SDLC(Software Development Life Cycle)
- Design patterns ease the analysis and requirement phase of SDLC by providing information based on prior hands-on experiences

Core Java Design Patterns

- Creational Design Patterns
- Structural Design Patterns
- Behavioral Design Patterns

Creational Design Patterns

- Factory Pattern
- Abstract Factory Pattern
- Singleton Pattern
- Prototype Pattern
- Builder Pattern

Creational design patterns provide solution to instantiate an object in the best possible way for specific situations

Structural Design Patterns

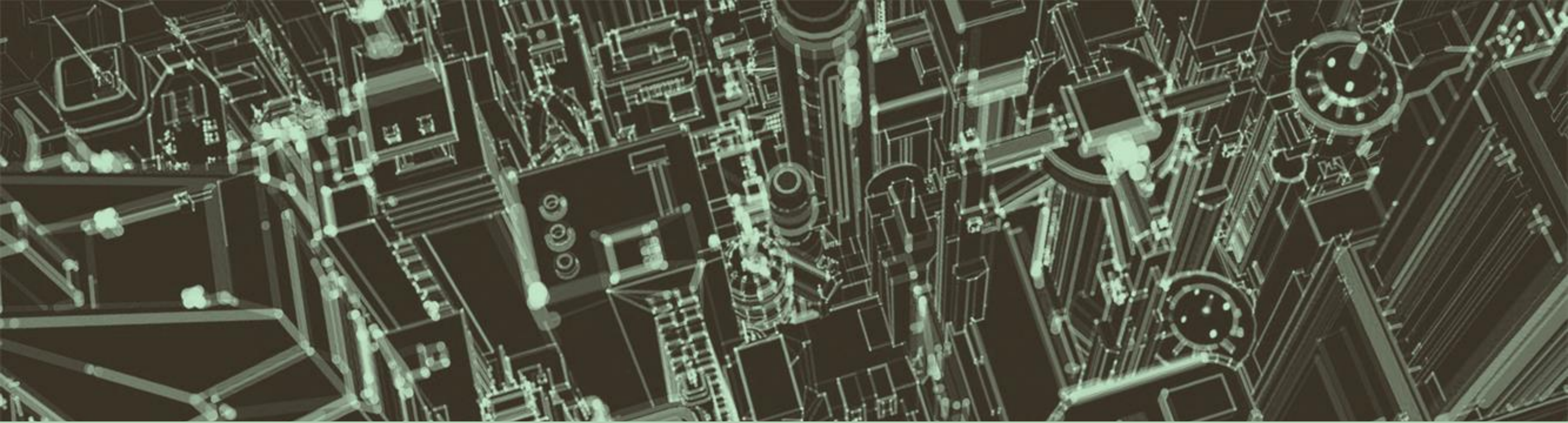
- Adapter Pattern
- Bridge Pattern
- Composite Pattern
- Decorator Pattern
- Facade Pattern
- Flyweight Pattern
- Proxy Pattern

  Structural patterns provide different ways to create a class structure, for example using inheritance and composition to create a large object from small objects

Behavioral Design Patterns

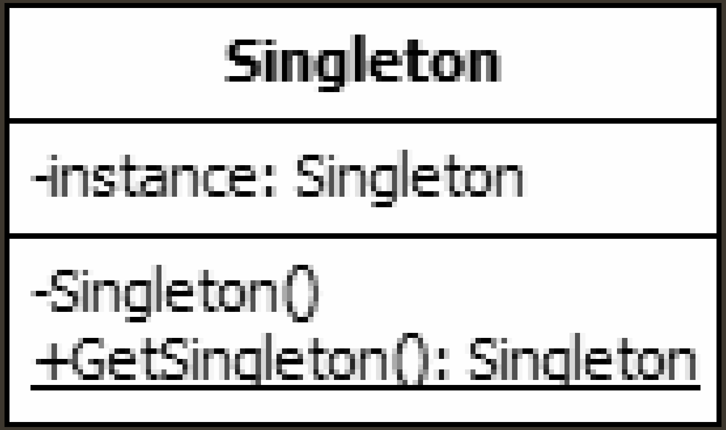
- Chain Of Responsibility Pattern
- Command Pattern
- Interpreter Pattern
- Iterator Pattern
- Mediator Pattern
- Memento Pattern
- Observer Pattern
- State Pattern
- Strategy Pattern
- Template Pattern
- Visitor Pattern

Behavioral patterns provide solutions for the better interaction between objects and also provide lose coupling and flexibility to extend easily



Some Examples of Common Design Patterns

Singleton design pattern

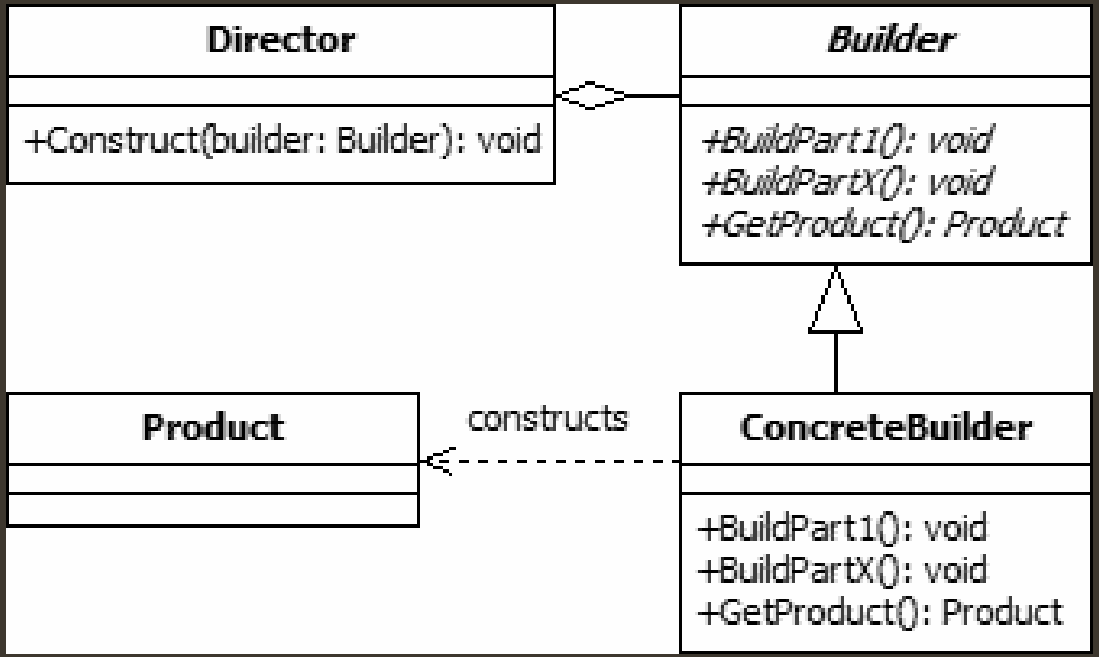


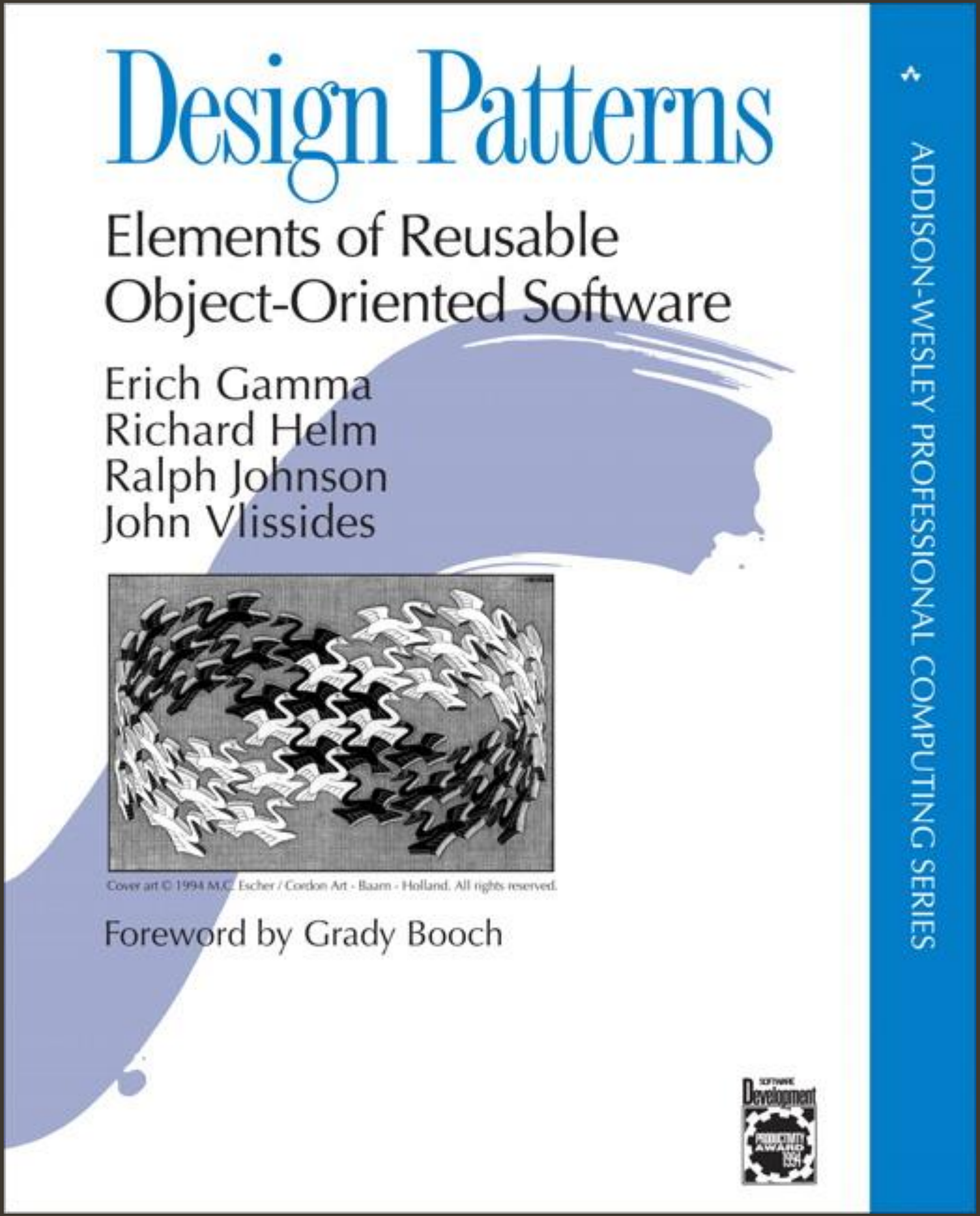
Singleton

-instance: Singleton

-Singleton()
+GetSingleton(): Singleton

Builder Design Pattern





Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides

Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch

ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES