Design Patterns in Practice

Rasmus Lystrøm
External Associate Professor
ITU
rnie@itu.dk

Agenda

IoC Container
Adapter
Factory Method
Singleton
Façade
Chain of Responsibility
Strategy
Bridge

Saved for later:
Command (MVVM)
Observer (MVVM)
Proxy (Web API)

Probably not covered:
Decorator
Visitor
Composite

IoC Container

Tool to facilitate dependency injection.

Using a factory to either manually or automatically create types at runtime.

Various implementations:

- Microsoft.Extensions.DependencyInjection
- Ninject
- Unity
- AutoFac
- StructureMap

IoC Container II

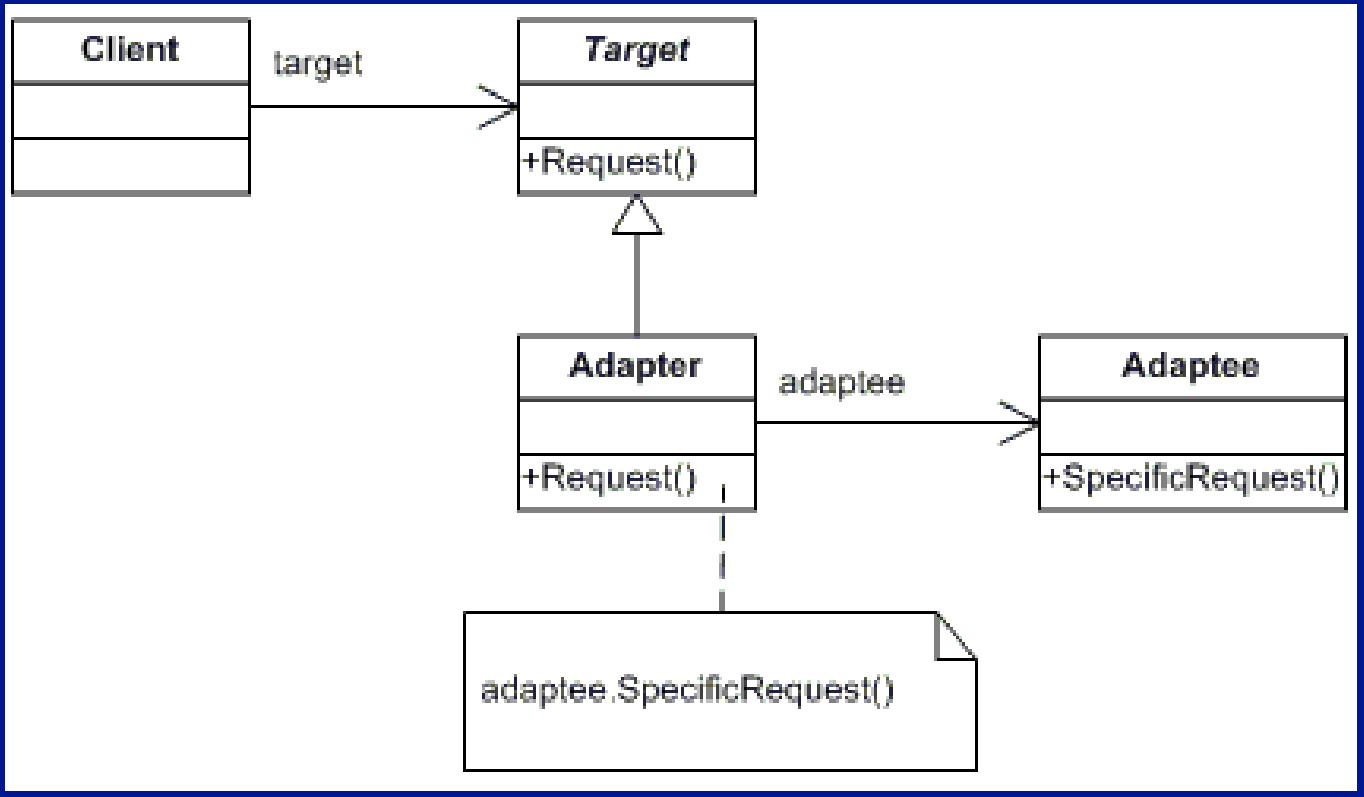
Lifetime:

- Transient (every time)
- Scoped (once per request)
- Singleton (once)

Adapter aka Wrapper

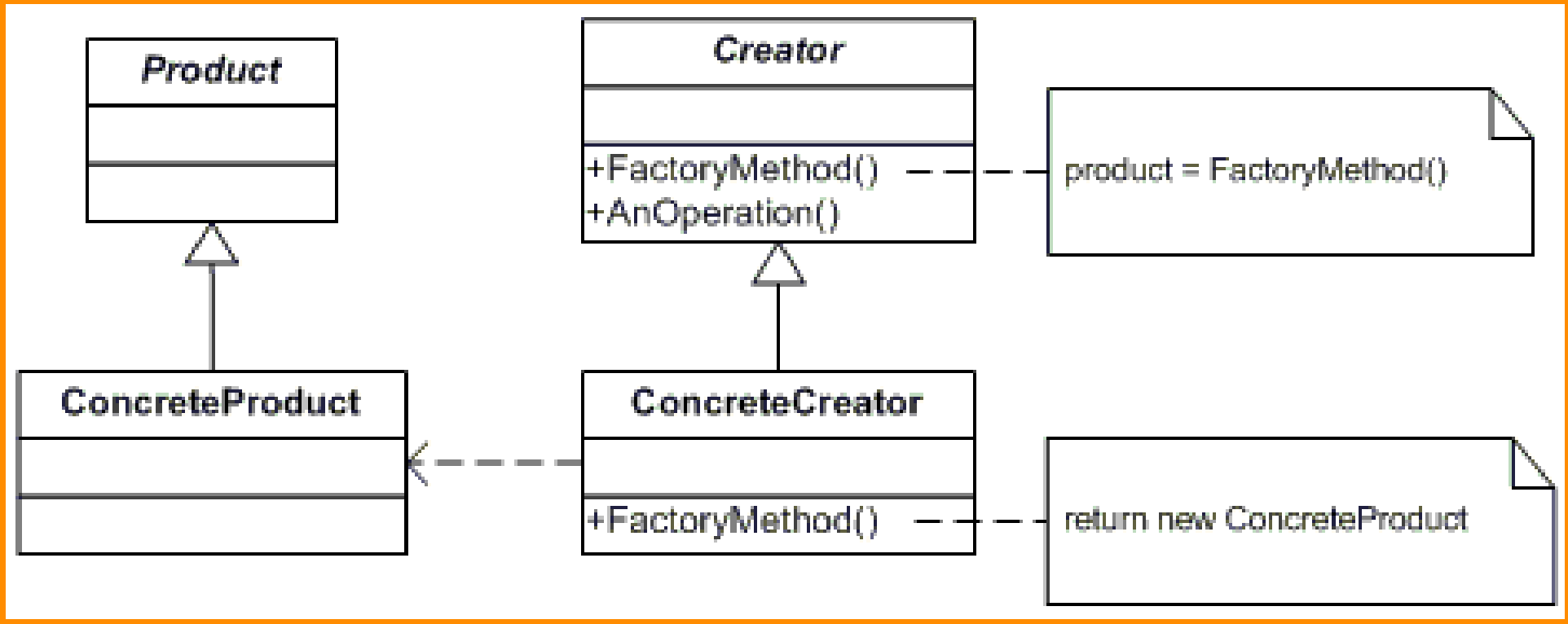
Unmodifiable implementation which does not match the interface you need.

Static or sealed class or class in another assembly.



Factory Method

A method which can creates instances of a given type.



Singleton

Only ever one single instance of a given type.

Considered an anti-pattern by many, it:

- is overused
- introduces unnecessary restrictions in situations where a sole instance of a class is not actually required
- introduces global state into an application

Singleton II

Use carefully

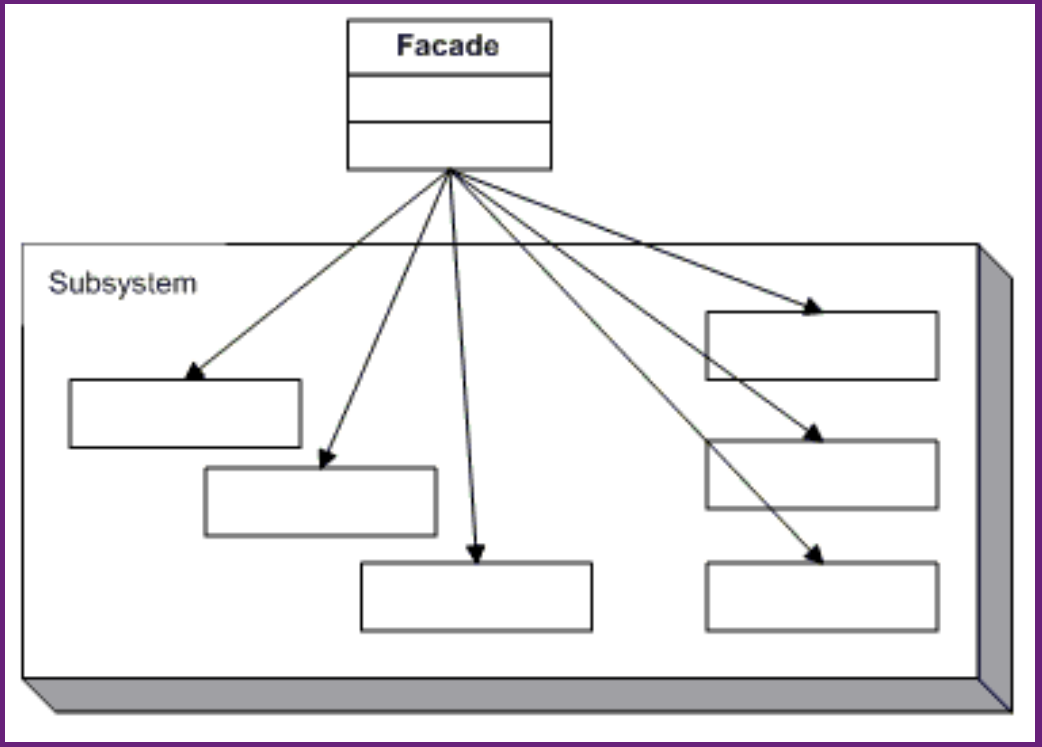
Implement using an interface

Use an IoC container

Façade

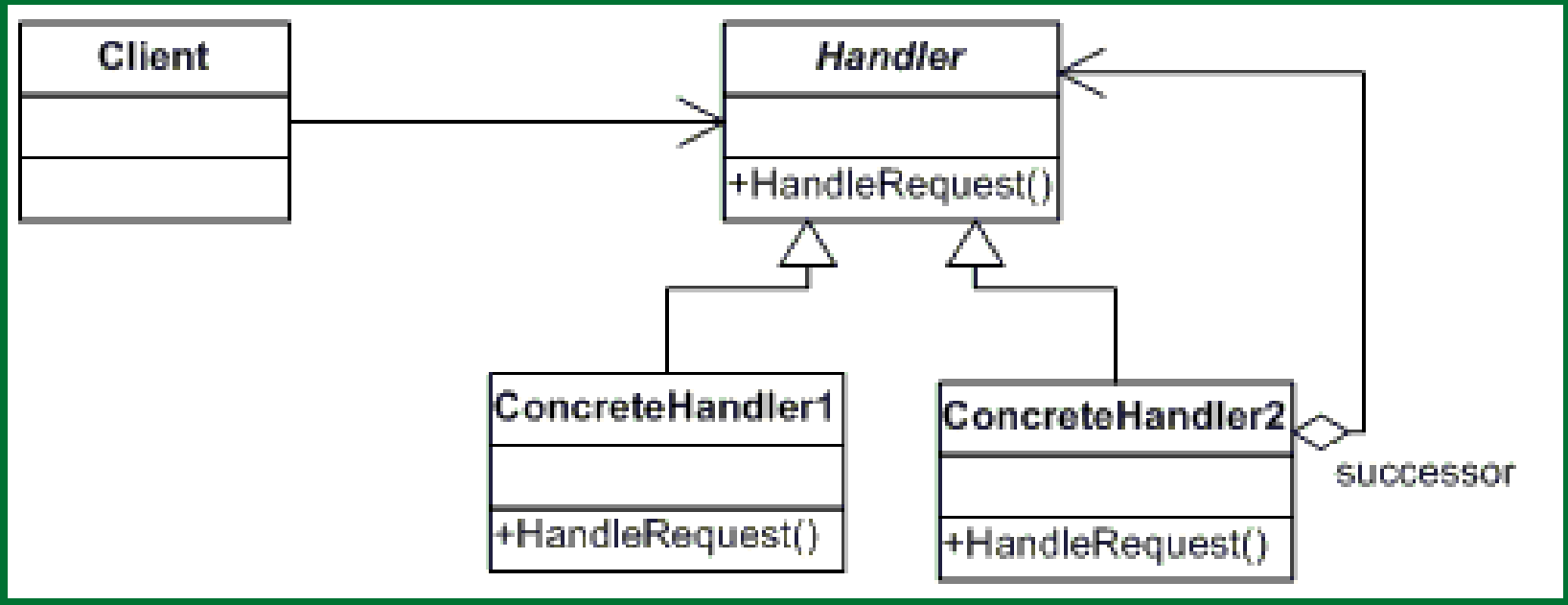
Simplify the use of a system

Provide a unified interfaces for a group of “dispersed” functionalities from a multitude of interfaces/classes



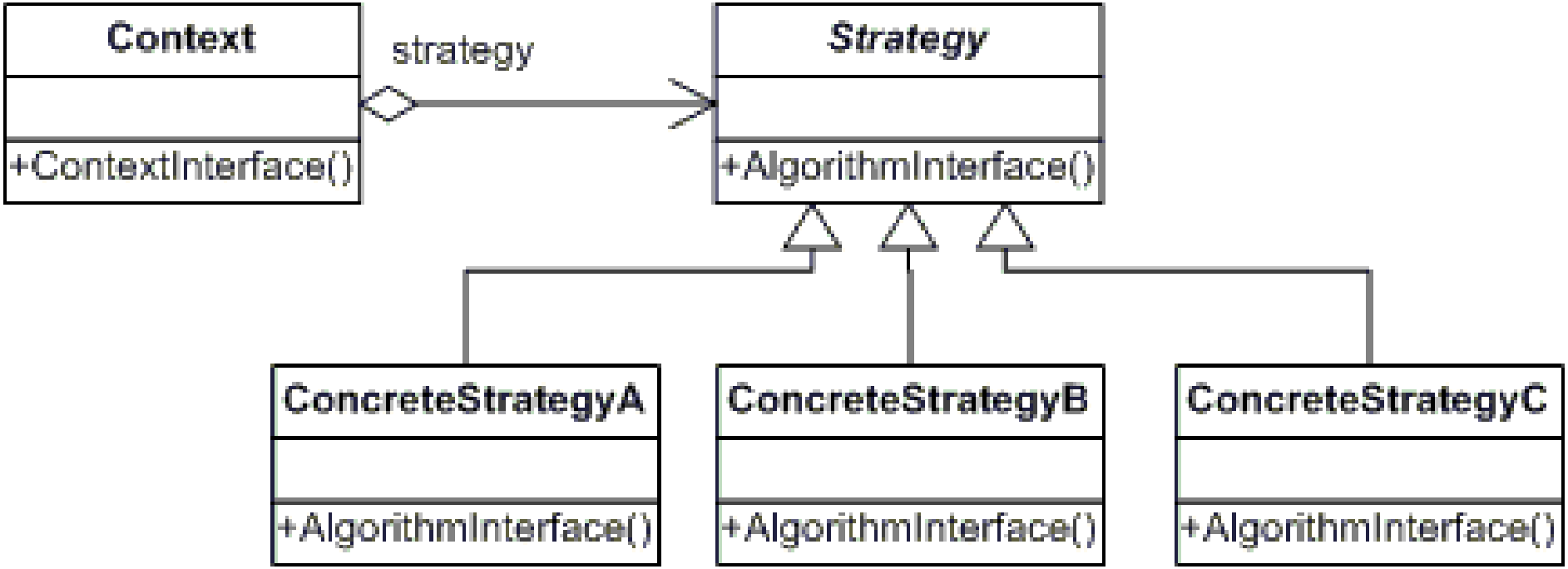
Chain of Responsibility

Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.



Strategy

Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.



Bridge

Decouple an abstraction from its implementation so that the two can vary independently.