Module 08:
"Bridge"

Agenda

- Introductory Example: Printing Reservations
- Challenges
- Implementing the Bridge Pattern
- Pattern: Bridge
- Overview of Bridge Pattern

Introductory Example: Printing Reservations

```
class Reservation
{
    public DateTime When { get; set; }
}
```

```
class EventTicket : Reservation
{
    ...
    public override string ToString() =>
        EventName + Environment.NewLine +
        $"Venue:\t{Venue}" + Environment.NewLine +
        When + Environment.NewLine +
        $"Seat:\t{Seat}";
}
```

Challenges

- How do we add more reservation types and more reservation formatting independently?
- What if we need to select a format for a specified reservation at runtime?
  - Compile-time binding between the two is then a bad idea!

Pattern: Bridge

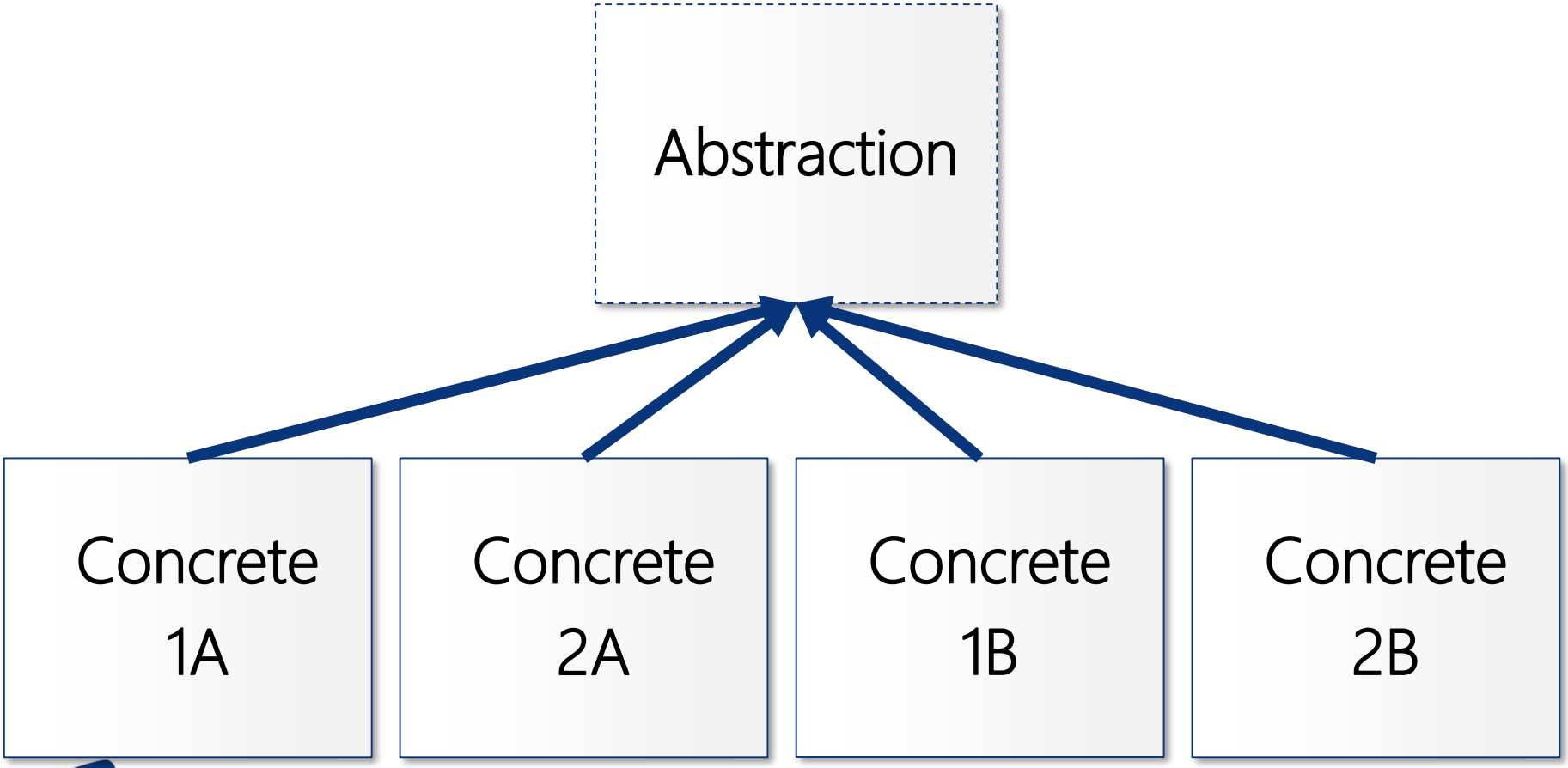
- *Decouple an abstraction from its implementation so that the two can vary independently.*

- Purpose
  - Separate abstraction and its implementation
  - Implement the abstraction by delegating to an implementor object

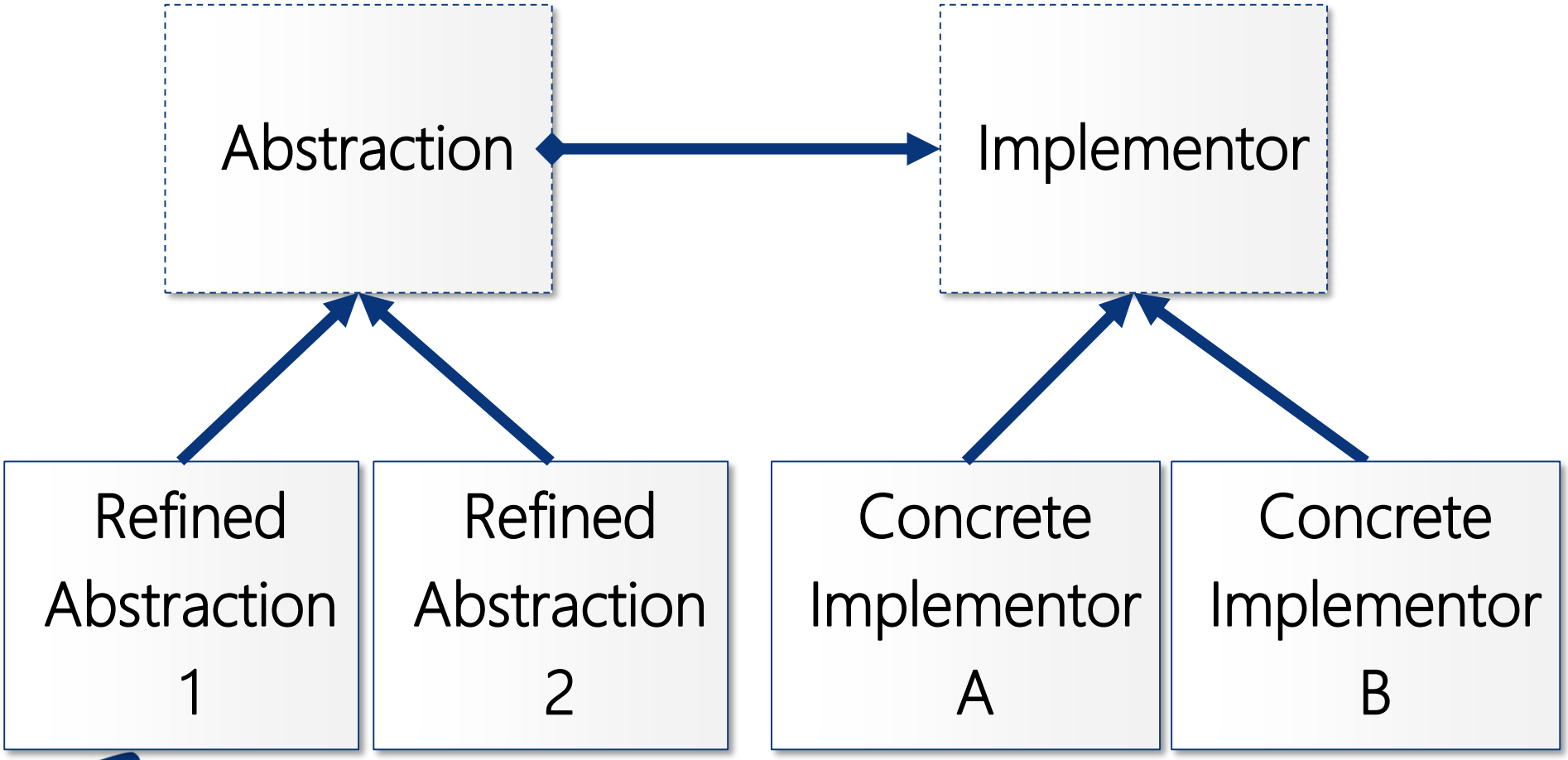
  - Prefer Composition over Inheritance…!

- Origin: Gang of Four

Without the Bridge Pattern



Overview of Bridge Pattern



Overview of the Bridge Pattern

- Abstraction
  - Interface or abstract class for primary class hierarchy
  - Holds reference to Implementor
- Refined Abstraction
  - Concrete class of primary class hierarchy
  - Provides primary state and behavior
- Implementor
  - Interface or abstract class for secondary functionality
- Concrete Implementor
  - Implements Implementor interface
  - Concrete class providing secondary functionality


WINCUBATE
Jesper Gulmann Henriksen
PhD, MCT, MCSD, MCPD
Phone : +45 22 12 36 31
Email : jgh@wincubate.net
WWW : http://www.wincubate.net
Hasselvangen 243
8355 Solbjerg
Denmark