Module 05:
"Prototype"

Agenda

- Introductory Example: Deck of Playing Cards
- Challenges
- Implementing Prototype Pattern with `ICloneable`
- Pattern: Prototype
- Overview of Prototype Pattern

Introductory Example: Deck of Playing Cards

```
class Card
{
    public Suit Suit { get; }
    public Rank Rank { get; }
}
```

```
class Deck : IEnumerable<Card>
{
    private List<Card> _cards;
    ...
}
```

```
Deck deck = new Deck();
deck.SwapCards(10, 20);
deck.TakeCard();
deck.SwapCards(0, 30);
deck.SwapCards(1, 29);
deck.TakeCard();
deck.Shuffle();

Deck copy = ...; // ???
```

Challenges

- Might be expensively, difficult, or even downright impossible to recreate copy of object with the same state
- Internal state might not be exposed externally to clients
- Might not want client to create new objects with constructor

ICloneable

- .NET has **ICloneable** interface built-in for implementing Prototype Pattern

```
public interface ICloneable
{
    object Clone();
}
```

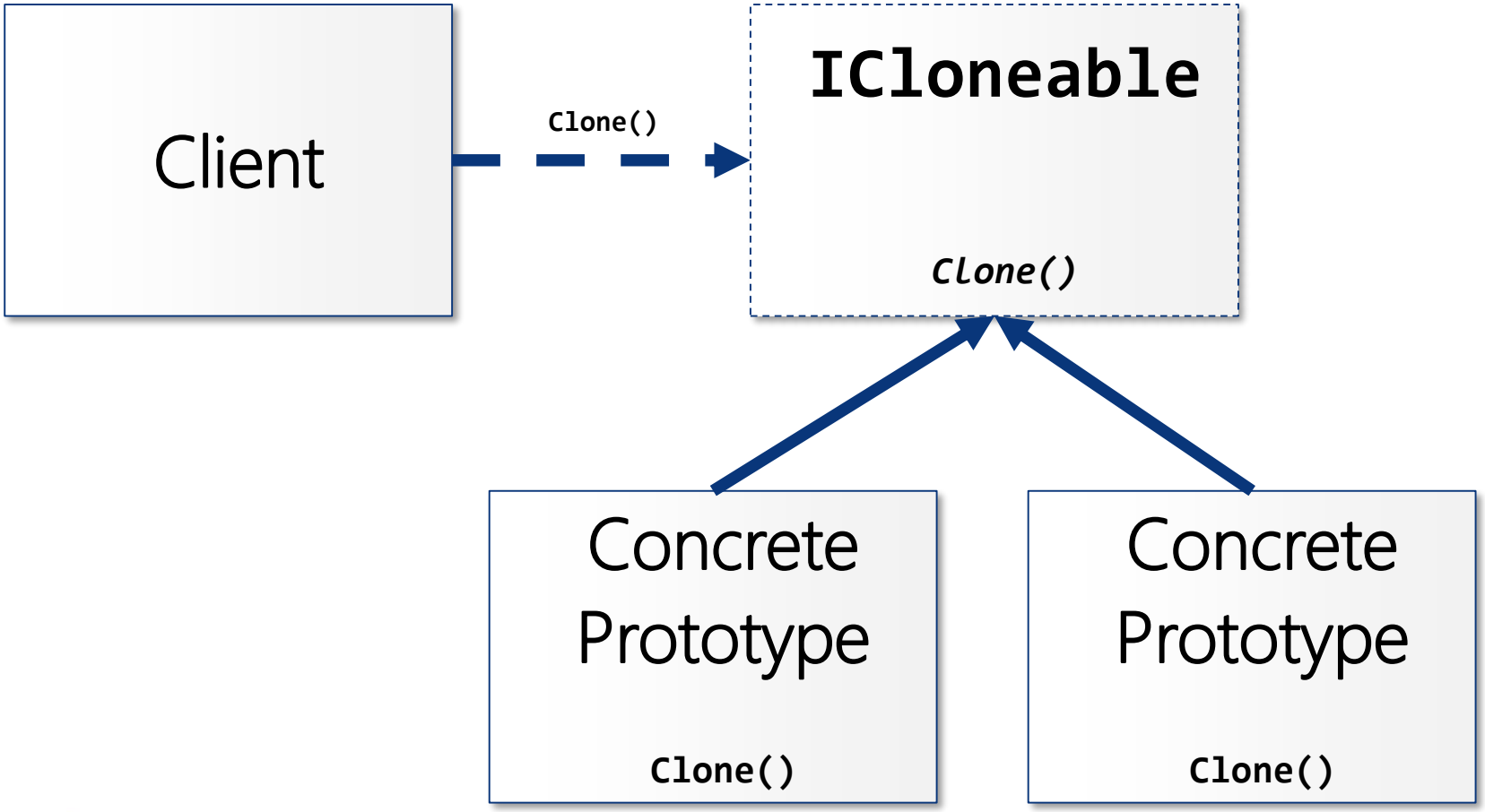
Pattern: Prototype

- *Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.*

- Outline
  - Allow client to create copies of existing objects
  - Use `ICloneable` interface to let master object clone itself

- Origin: Gang of Four

Overview of Prototype Pattern



Overview of Prototype Pattern

- `ICloneable`
  - Interface pre-built into .NET
  - Provides a `Clone()` method to be overridden in concrete prototypes
- Concrete Prototype
  - Implements `ICloneable` interface and supplies appropriate cloning logic
  - Might employ the `object.MemberwiseClone()` method built into .NET
    - Note: Beware of value types vs. reference types of elements
- Client
  - Invokes `Clone()` on Concrete Prototype to create a copy

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