

# Module 02: "Abstract Factory"



# Agenda

- ▶ Introductory Example: Devices for New Employees
- ▶ Challenges
- ▶ Implementing the Abstract Factory Pattern
- ▶ Pattern: Abstract Factory
- ▶ Overview of Abstract Factory
- ▶ Variations and Related Patterns



# Introductory Example: Devices for New Employees

```
class Iphone7
{
    public int MemoryGb { get; }
    ...
}
```

```
class Ipad
{
    public string Version { get; }
    public double ScreenSize { get; }
    ...
}
```

```
Iphone7 phone = new Iphone7(32);
phone.Call("+45 12345678");
```

```
Ipad tablet = new Ipad("Air 3", 9.7);
tablet.PowerOn();
tablet.PowerOff();
```

# Challenges

- ▶ How do we isolate clients from concrete classes and rely only on general interfaces (or general classes)?
- ▶ We don't want the New Employee Enrollment or other HR processes to require change whenever a new edition of a device emerges!
- ▶ How do we ensure that clients create object correctly?
- ▶ What if the company decides to change it's device vendor?

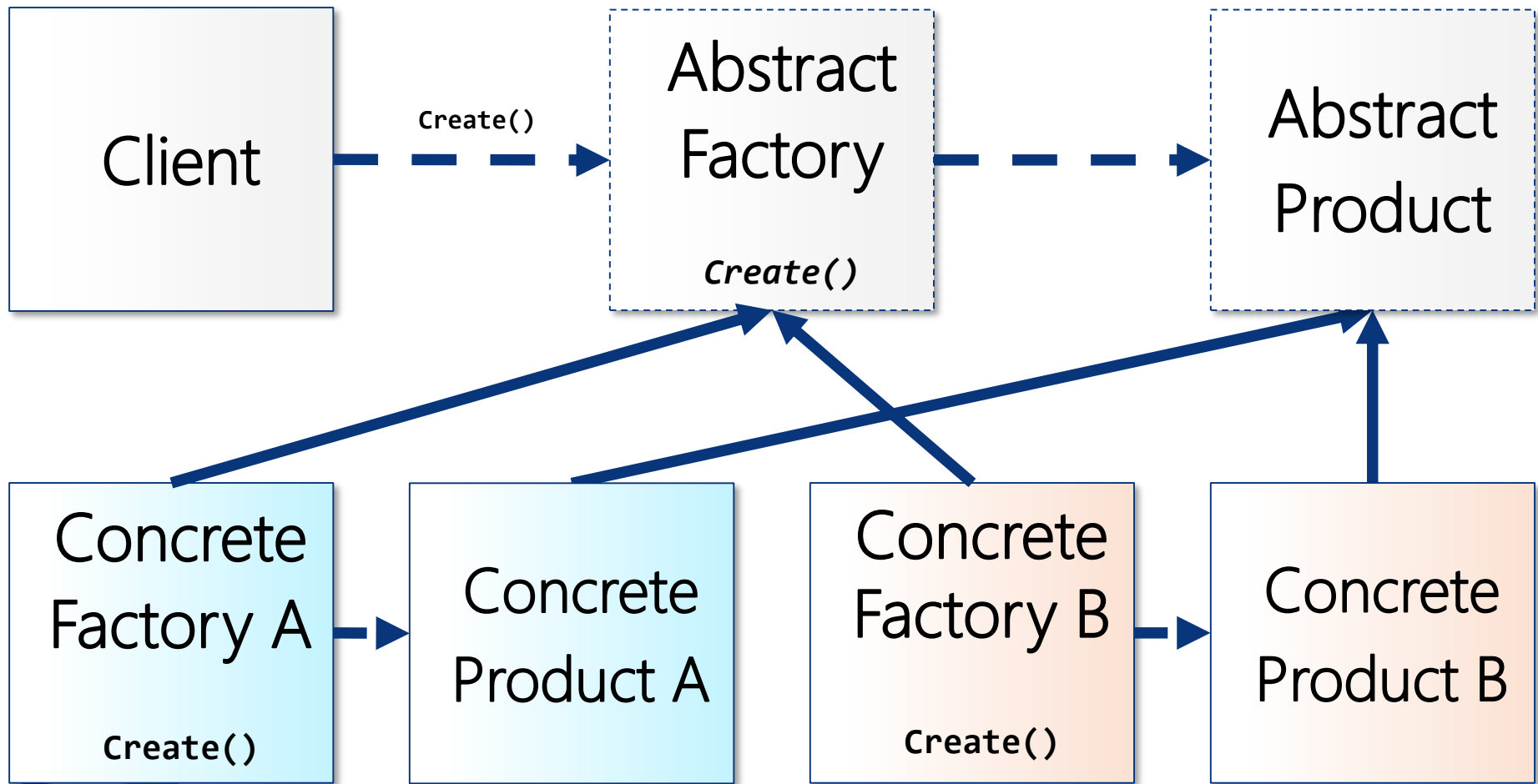


# Pattern: Abstract Factory

- ▶ *Provide an interface for creating families of related or dependent objects without specifying their concrete classes.*
- ▶ Outline
  - Encapsulate object creation in a factory object.
  - Define an interface (abstract factory) for creating objects
  - Implement that interface in concrete classes
  - Classes do not instantiate objects directly but delegate object creation to factory
- ▶ Origin: Gang of Four



# Overview of Abstract Factory Pattern



# Overview of Abstract Factory Pattern

- ▶ Client
  - Only depends upon Abstract Factory and Abstract Product
- ▶ Abstract Factory
  - Interface for object creation operations such as abstract **Create()**
- ▶ Concrete Factory
  - Concrete class implementing the Abstract Factory interface for creating Concrete Product objects in **Create()**
- ▶ Abstract Product
  - Interface or abstract class specifying a generalized product (or products)
- ▶ Concrete Product
  - Concrete class implementing the Abstract Product interface



# Variations and Related Patterns

- ▶ Each factory can create any number of products
  - `CreateXxx()`
  - `CreateYyy()`
  
- ▶ **Create()**
  - can optionally be supplied a product context parameter
  - can make use of the Builder Pattern
  - description parameter or type name might be employed
    - See Lab 02.1
  
- ▶ Abstract Factory Pattern
  - Is often confused or combined with Factory Method Pattern







WINCUBATE

***Jesper Gulmann Henriksen***

PhD, MCT, MCSD, MCPD

Phone : +45 22 12 36 31

Email : [jgh@wincubate.net](mailto:jgh@wincubate.net)

WWW : <http://www.wincubate.net>

Hasselvangel 243

8355 Solbjerg

Denmark