# DESIGN PATTERN

## INTRODUCTION À L'APPROCHE ORIENTÉ OBJET

- Introduction
- Composite
- Observer
- Abstract Factory

# INTRODUCTION

Ils ont été introduit la première fois en 1995 par le « Gang of Four » à travers leur livre intitulé :
'Design Patterns - Elements of Reusable Object-Oriented Software'.

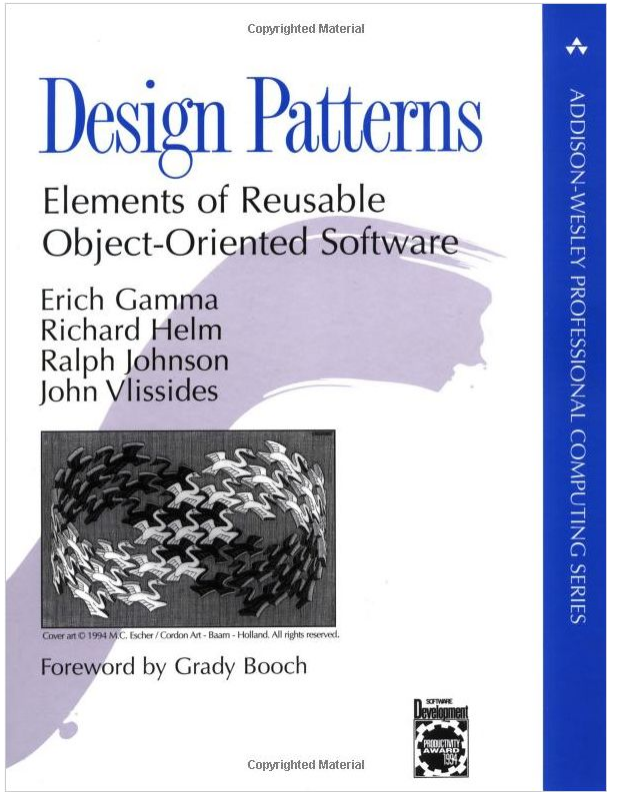
Le 'GoF' est constitué de 4 auteurs : Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides.

Un « Design Pattern » est techno agnostique et consiste à solutionner un problème clairement défini que nous rencontrons régulièrement dans notre développement.

Ceux-ci sont constitués d'objets, définis par des classes et des interfaces, reliés entre eux par les concepts fondamentaux de l'orienté objet, le tout en respectant les « bonnes » pratiques de programmation.

Il va de soit que ceux-ci devront être adaptés en fonction de différents critères comme le langage, la situation, nos besoins.

Exemple le plus parlant : C++ gère l'héritage multiple tandis que le C# non, de ce fait nous implémenterons différemment le même pattern.



# INTRODUCTION

Les modèles de conception du « Gang of Four » reprennent trois axes spécifiques.

**Orienté « Création » :**

- Fabrique abstraite (Abstract Factory)
- Monteur (Builder)
- Fabrique (Factory Method)
- Prototype (Prototype)
- Singleton (Singleton)

# INTRODUCTION

**Orienté « Structure » :**

- Adaptateur (Adapter)
- Pont (Bridge)
- Objet composite (Composite)
- Décorateur (Decorator)
- Façade (Facade)
- Poids-mouche ou poids-plume (Flyweight)
- Proxy (Proxy)

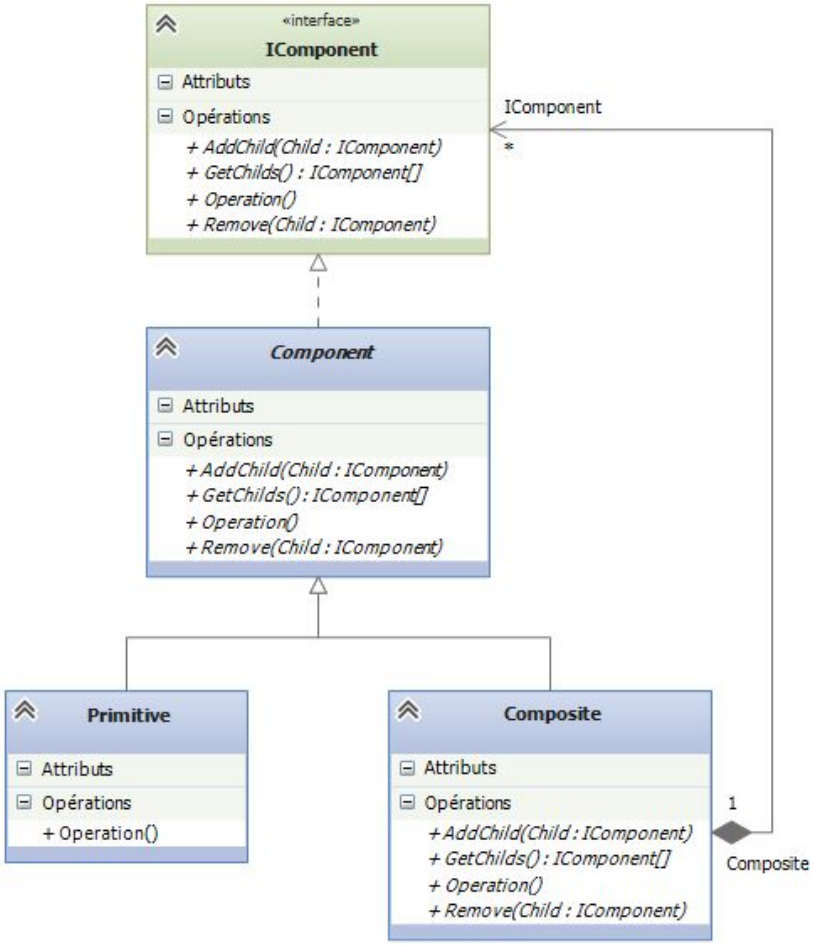
# INTRODUCTION

**Orienté « Comportement » :**

- Chaîne de responsabilité (Chain of responsibility)
- Commande (Command)
- Interpréteur (Interpreter)
- Itérateur (Iterator)
- Médiateur (Mediator)
- Mémento (Memento)
- Observateur (Observer)
- État (State)
- Stratégie (Strategy)
- Patron de méthode (Template Method)
- Visiteur (Visitor)
- Fonction de rappel (Callback)

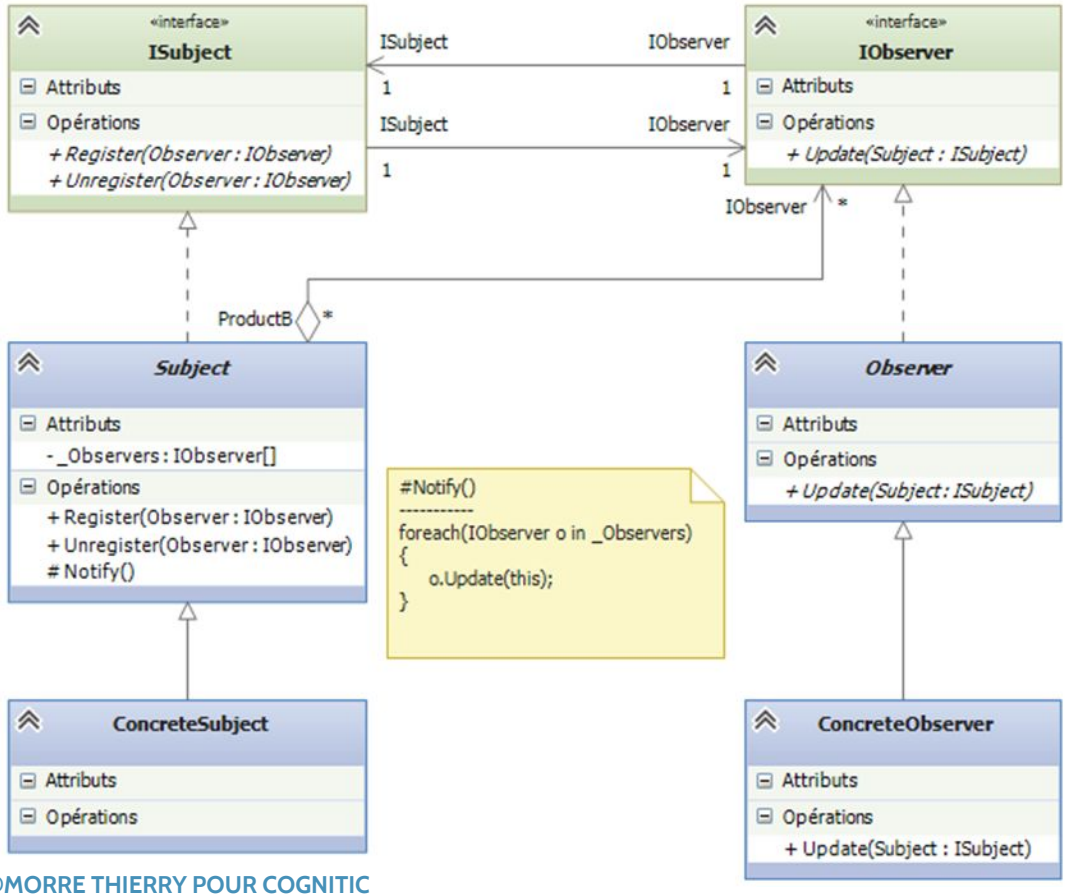
# COMPOSITE

- IComponent : est une interface qui représente les fonctionnalités nécessaire d'un composant.
- Component : est une classe abstraite qui implémente, si nécessaire, les fonctionnalités par défaut de tous les composants.
- Primitive : est une classe qui représente un composant primitif, celui-ci n'a pas d'enfant.
- Composite : est une classe qui représente un composant qui peut avoir des enfants.




Notez la notion de composition qui relie « Composite » à « Icomponent ».

# OBSERVER

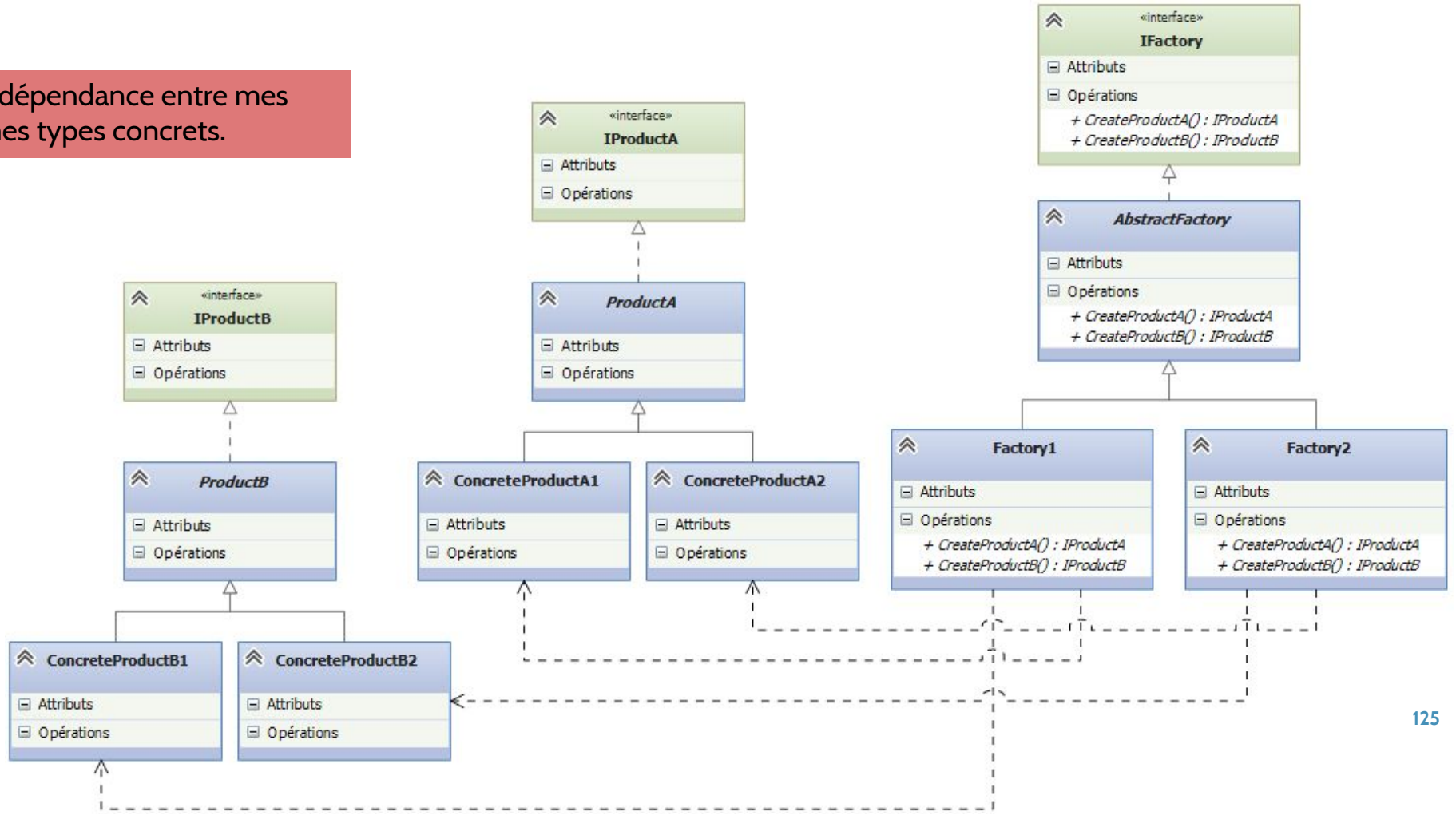



- ISubject est une interface qui représente les fonctionnalités nécessaire d'un sujet.
- Subject est une classe abstraite qui implémente les fonctionnalités par défaut de tous les sujets.
- IObserver est une interface qui représente les fonctionnalités nécessaire d'un observateur.
- Observer est une classe abstraite qui représente, si nécessaire, le fonctionnel commun à tous les observateurs.

Notez la notion d'agrégation qui relie « Subject » à « IObserver ».

# ABSTRACT FACTORY

Notez la notion de dépendance entre mes fabriques et mes types concrets.



# RESSOURCES

## INTRODUCTION À L'APPROCHE ORIENTÉ OBJET

- Design Patterns: Elements of Reusable Object-Oriented Software
  ISBN-13 : 978-0201633610
- Design Pattern :
  http://www.dofactory.com
- Images Star Wars :
  http://joewight.deviantart.com/gallery
- Cours UML 2.1 Cognitic
- UML pour les développeurs, Isabelle Mounier et Xavier Blanc

# RESSOURCES