Conception Orientée Objet

Design patterns – Partie 1

Tianxiao LIU
Master IISC 1ère Année
CY Cergy Paris Université
http://depinfo.u-cergy.fr/~tliu/coo.php

Les design patterns

- Histoire
  - Description dans le livre *Gang of Four (GoF)*
  - E. Gamma, R. Helm, R. Johnson et J. Vlissides
  - *Design patterns – Elements of Reusable Object-Oriented Software*

- Pour chaque pattern
  - Un problème constamment présent
  - Une solution standard
  - Des variantes

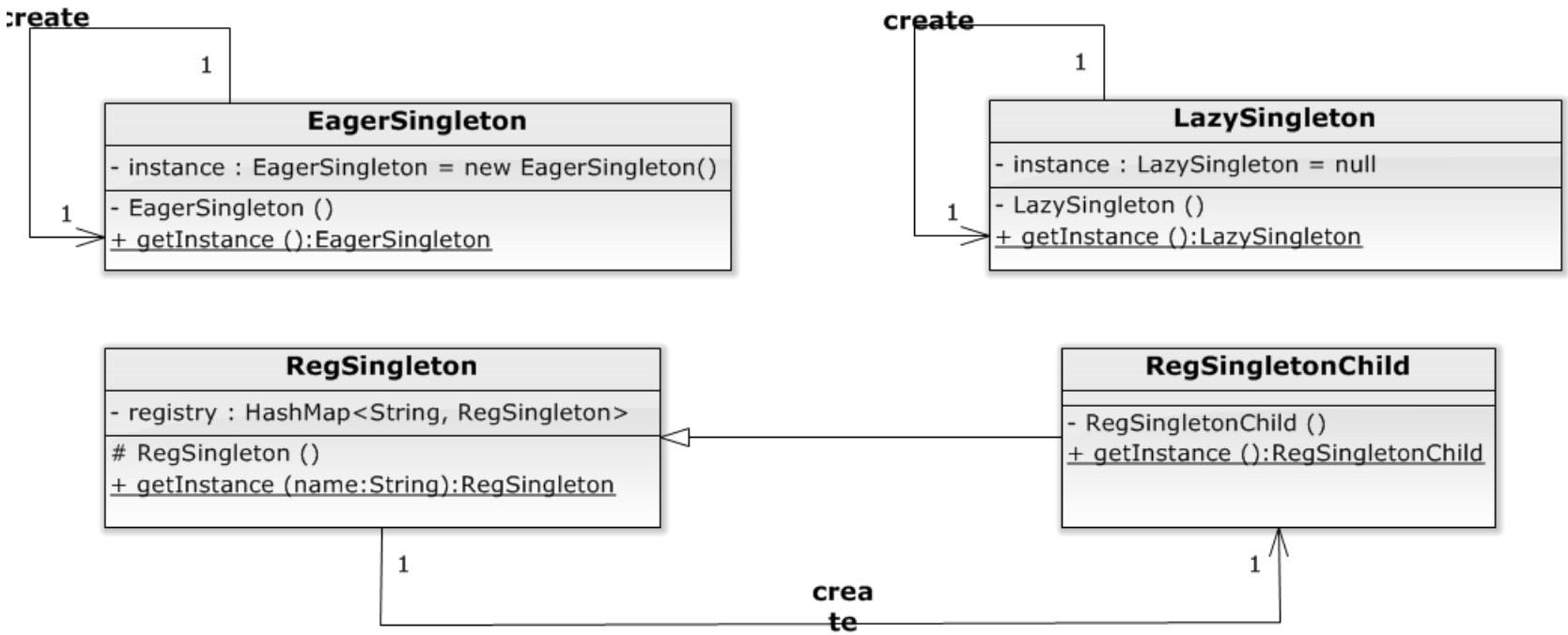
Sommaire : les 23 patterns GoF

Creational	Structural	Behavioral
Abstract Factory	Adapter	Chain of responsibility
Builder	Bridge	Command
Factory Method	Composite	Interpreter
Prototype	Decorator	**Iterator**
Singleton	Facade	Mediator
	Flyweight	Memento
	Proxy	Observer
		State
		Strategy
		Template Method
		Visitor

Présentés dans cette séance

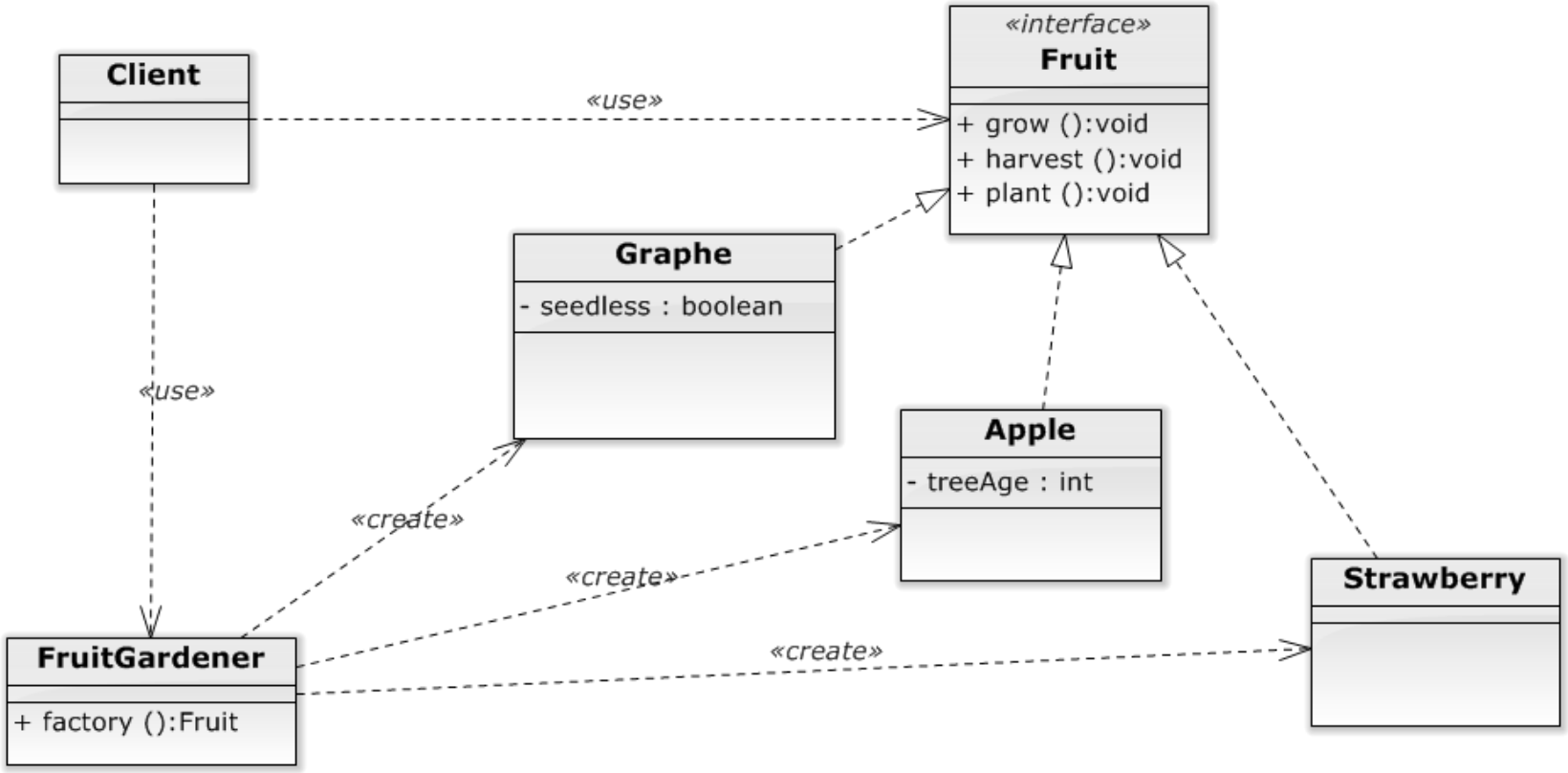
Pattern : *Singleton*

- Principe
  - Une seule instance durant la vie du système
  - Il existe trois variantes : ***Eager***, ***Lazy*** et ***Register***



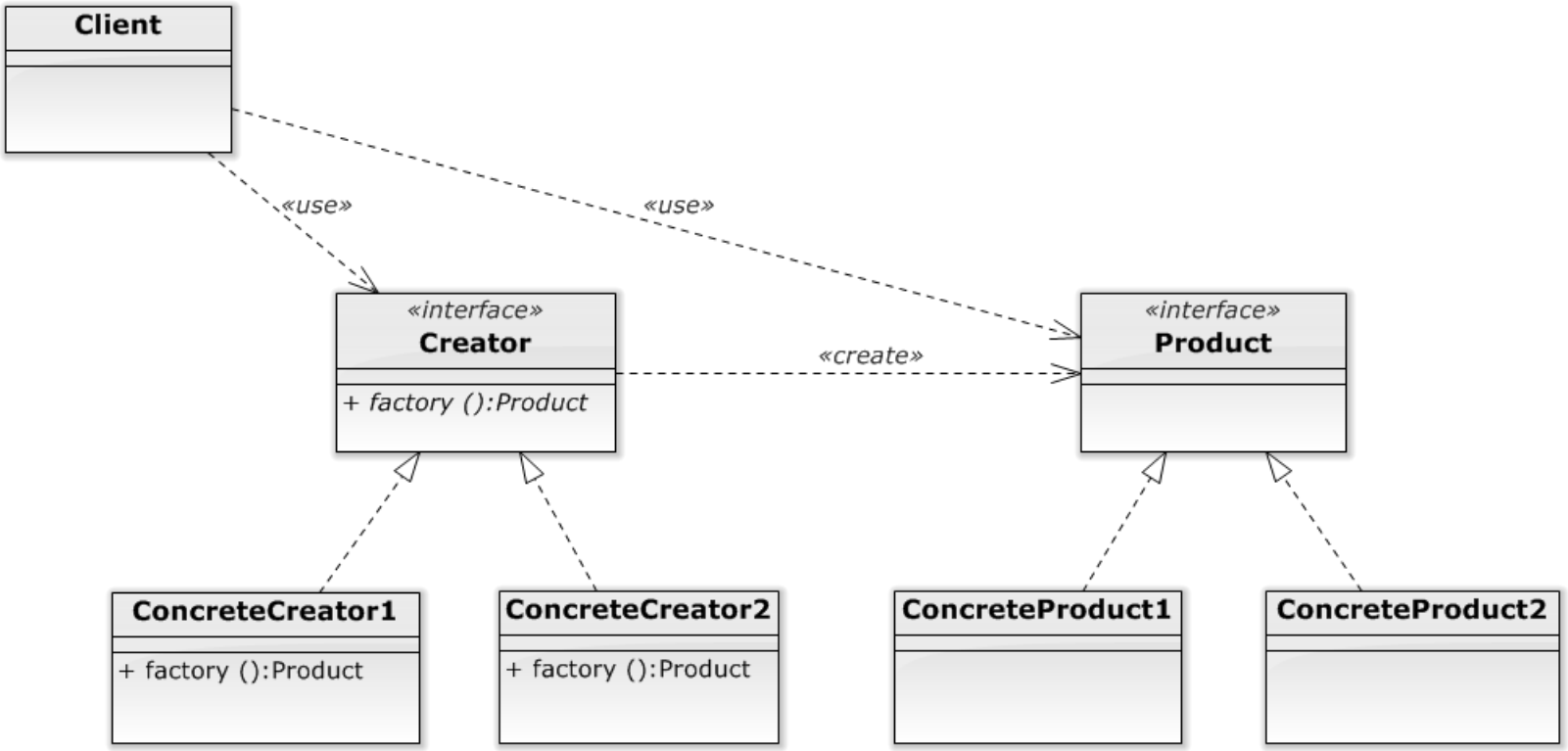
Simple factory (non GoF)

- Comprendre par un exemple : Fruit



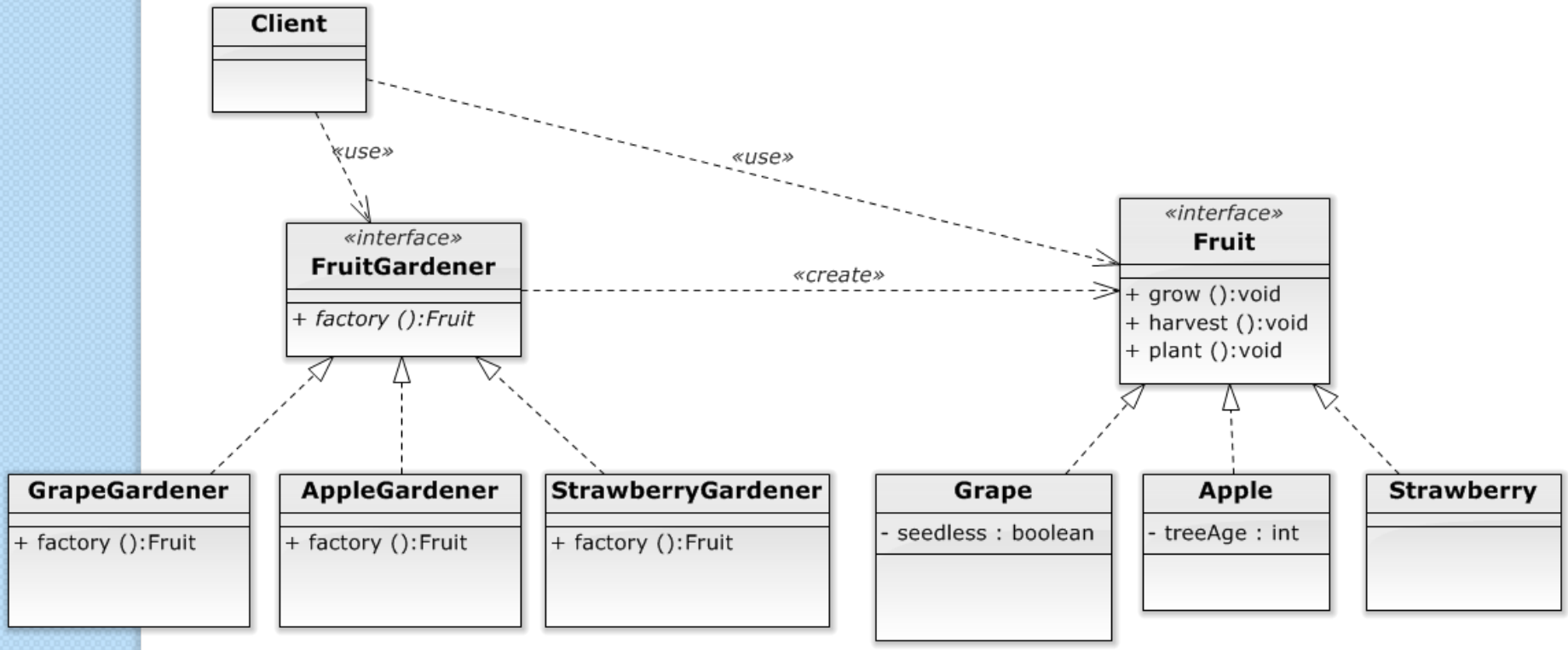
Pattern : *Factory Method*

- Principe
  - Abstraire la solution *Simple Factory*
  - Un produit → Une classe concrète pour le fabriquer



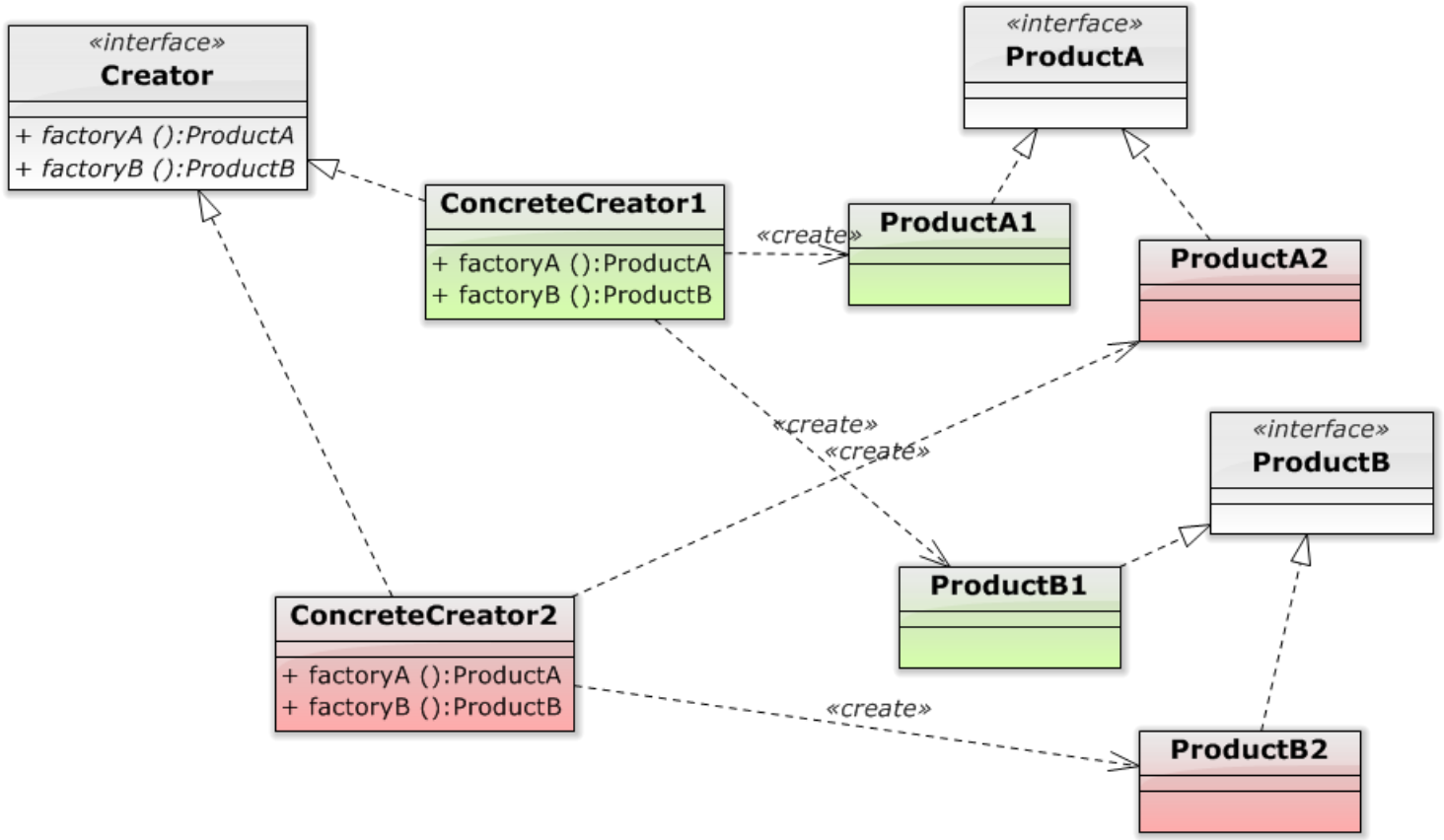
Pattern : *Factory Method*

- Adaptation de l'exemple Fruit



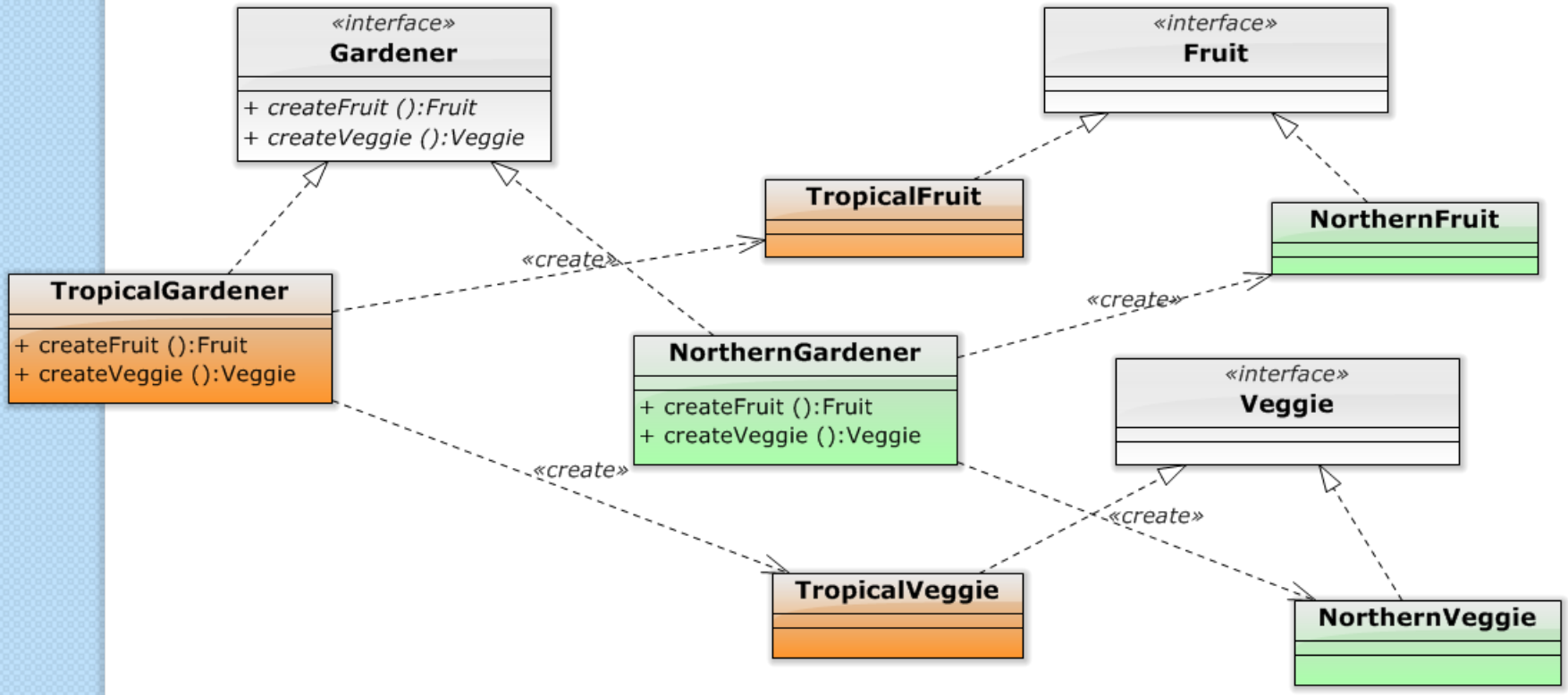
Pattern : *Abstract Factory*

- Principe
  - Modèle de *factory* complètement général
  - Support de ***deux dimensions*** de classifications



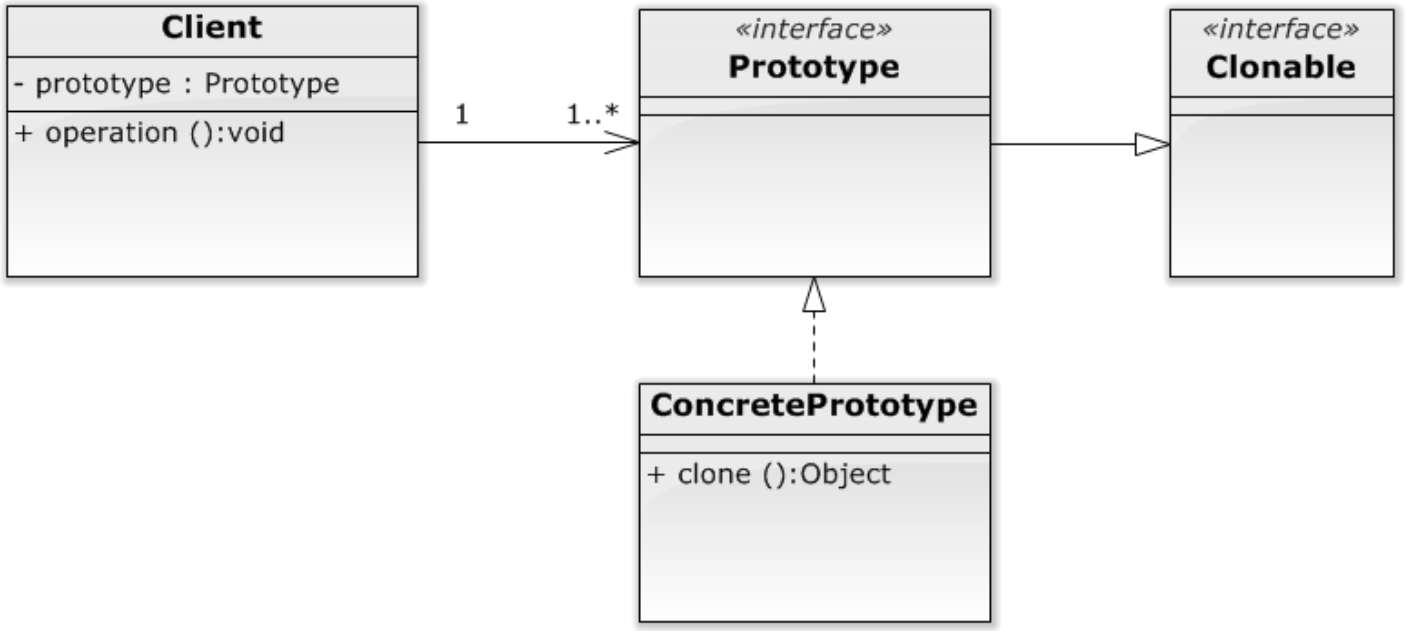
Pattern : *Abstract Factory*

- Un exemple



Pattern : *Prototype*

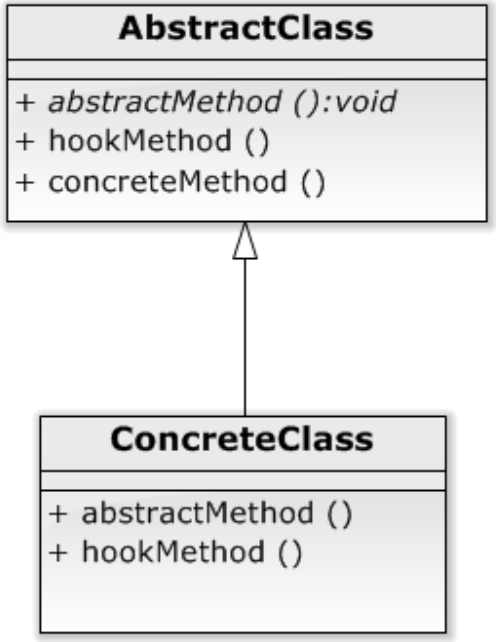
- Principe
  - Quand un objet à produire varie durant l'exécution du programme, on lui confie la tâche de copie



Pattern : *Template Method*

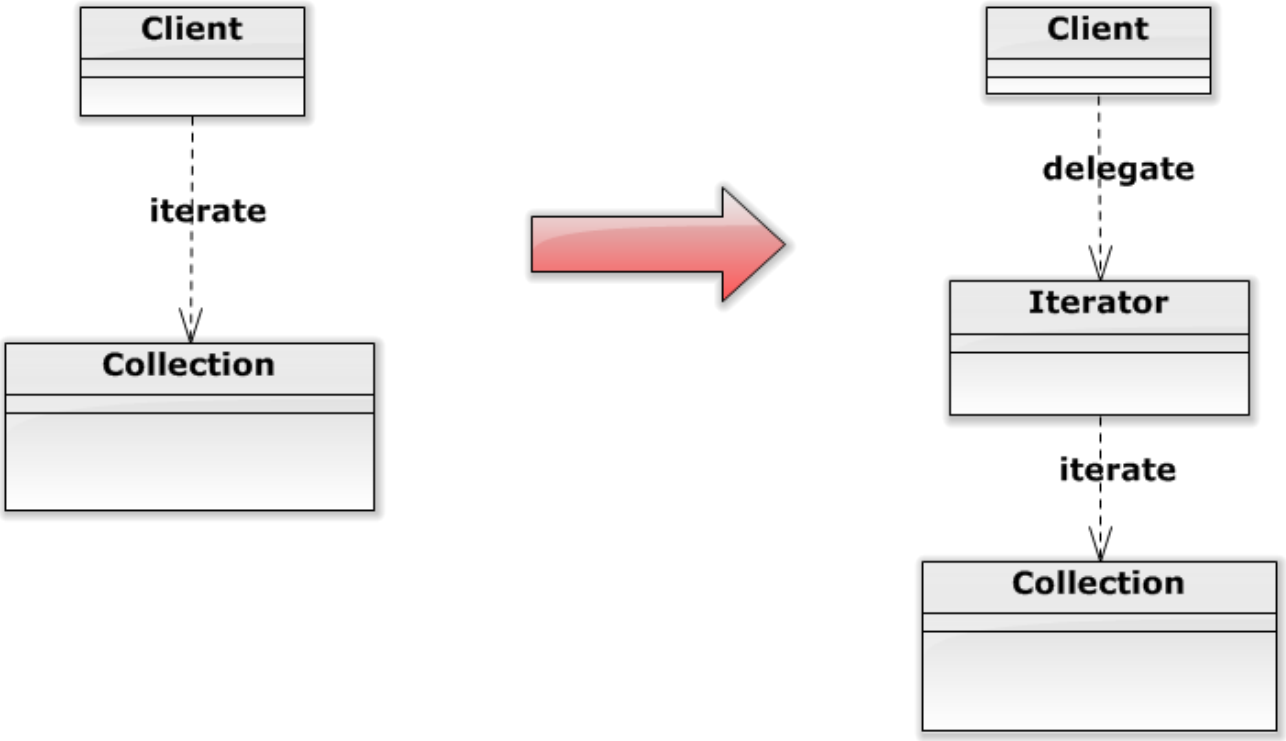
- Principe
  - Utilisation d'une classe abstraite qui définit une partie de réalisation **concrète** et une autre partie **abstraite**
  - Partie **abstraite** : implémentation dans les sous-classes

- Types de méthodes
  - ***abstract method***
    - implémentation dans les sous-classes
  - ***hook method***
    - implémentation vide ou par défaut
    - ex. toString
  - ***concrete method***
    - sans override dans les sous-classes



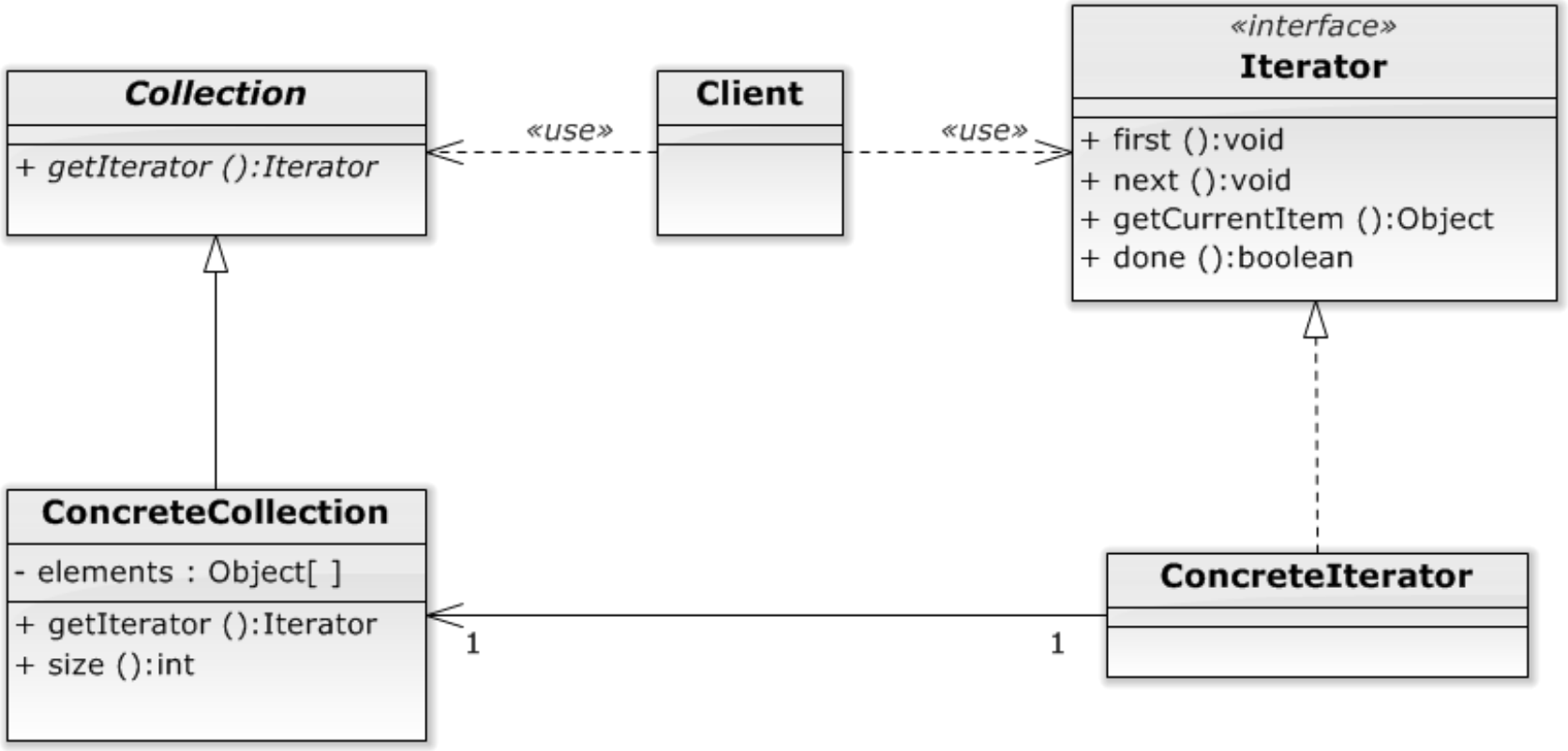
Pattern : *Iterator*

- Motivation
  - Parcourir linéairement les éléments d'une collection
  - Abstraction de haut niveau (ignorer les types concrets)



Pattern : *Iterator*

- *Extrinsic Iterator (Cursor Iterator)*



Pattern : *Iterator*

- *Intrinsic Iterator*