Design Patterns, Decorator, Proxy

Sébastien Jean

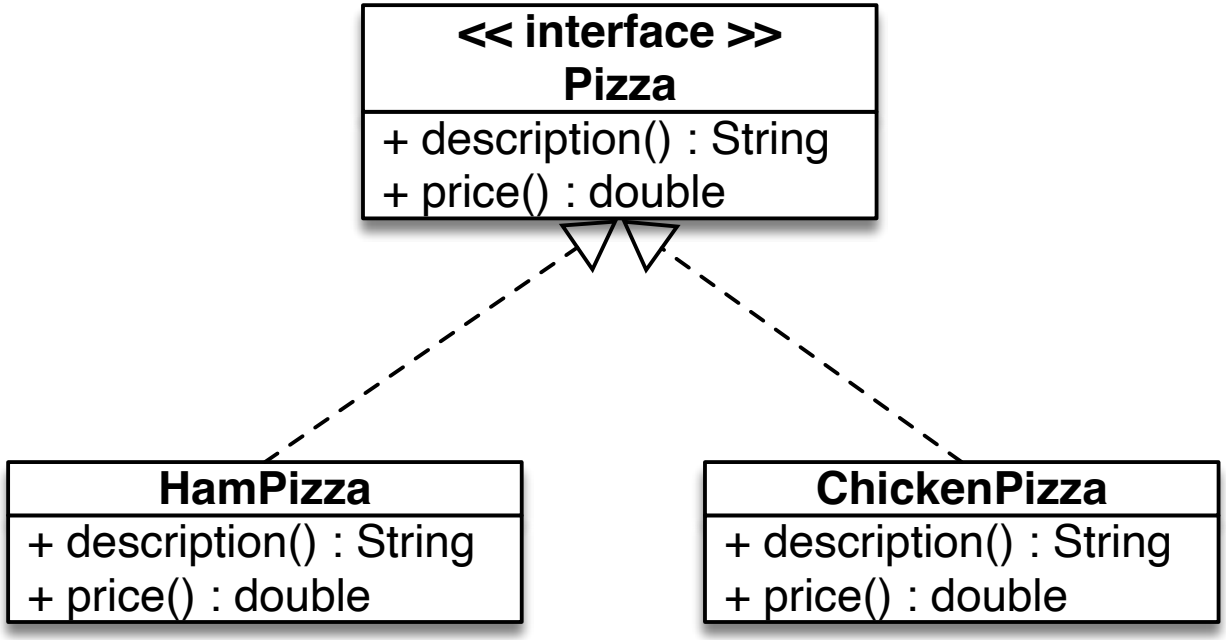
IUT de Valence
Département Informatique

v4.1, 27 novembre 2022

IUT A Valence
Université Grenoble Alpes

Decorator, problème

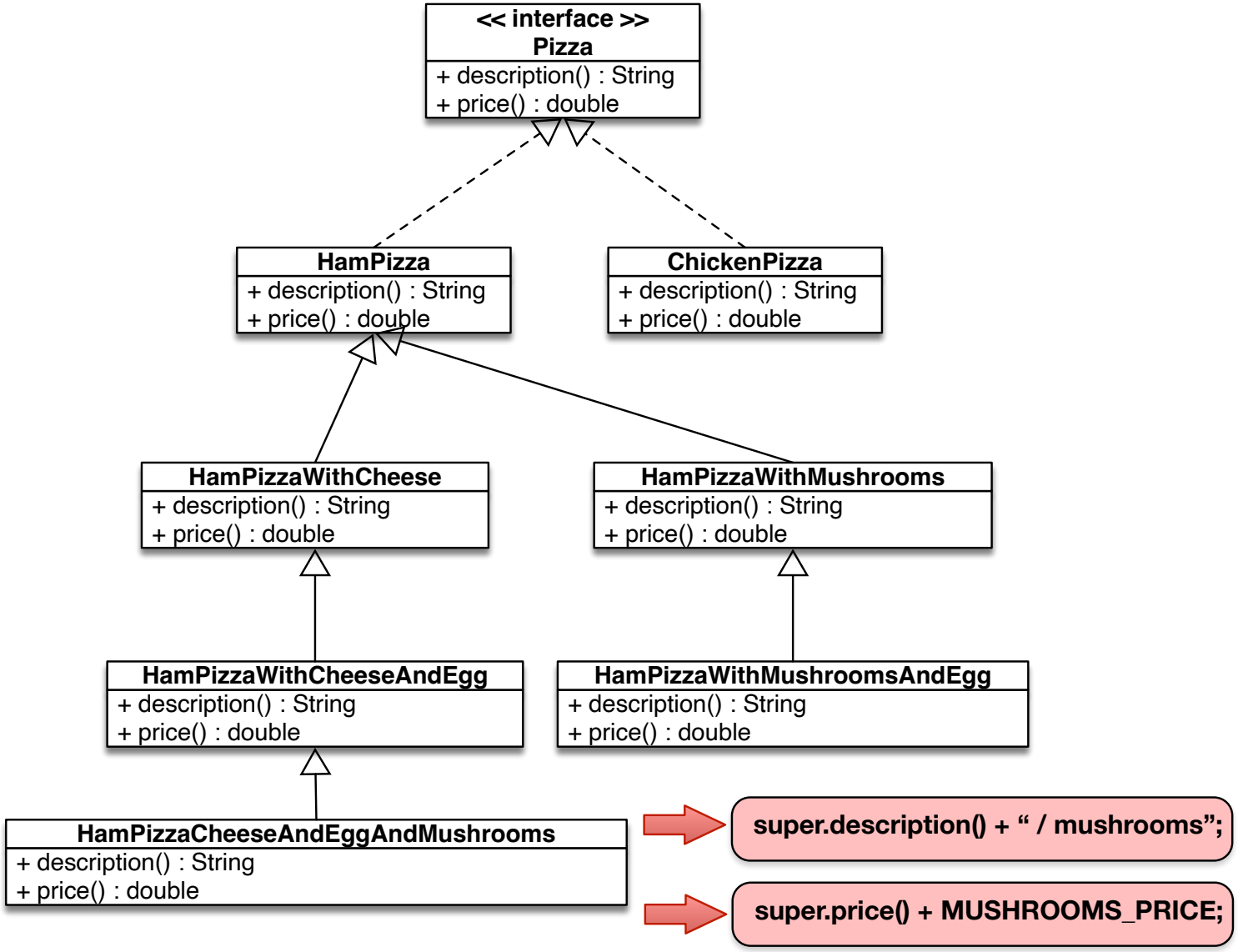
- Une pizzeria propose initialement deux sortes de pizza
  - au **jambon**
  - au **poulet**



IUT Valence
Université Grenoble Alpes

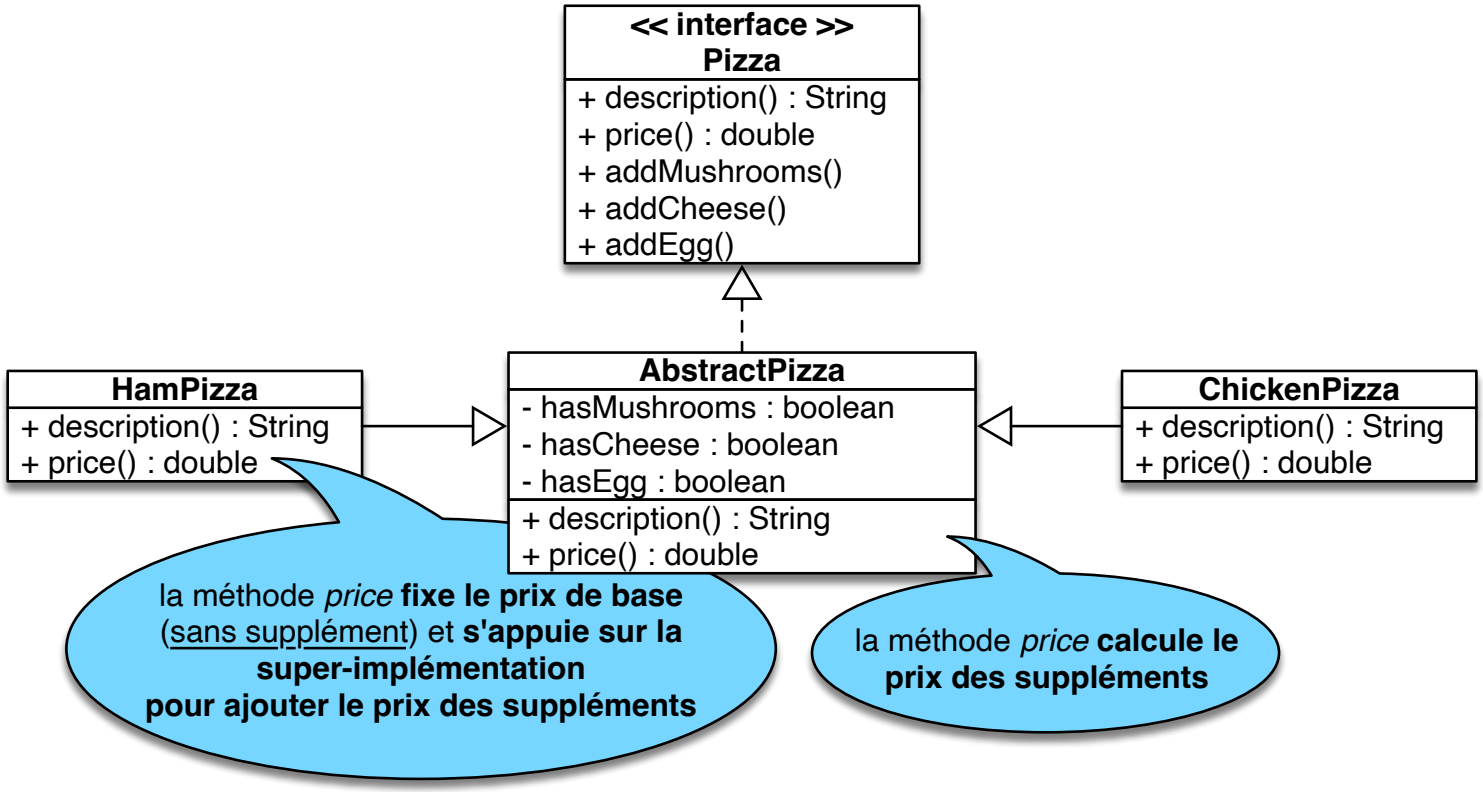
Decorator, problème (suite)

- Les clients veulent pouvoir **rajouter des suppléments**



Decorator, problème (suite)

- Comment **calculer le prix facilement** ?
- Idée :
  - intégrer l'ajout de suppléments à l'interface `Pizza`
  - définir une classe abstraite factorisant la gestion des suppléments (description / prix)



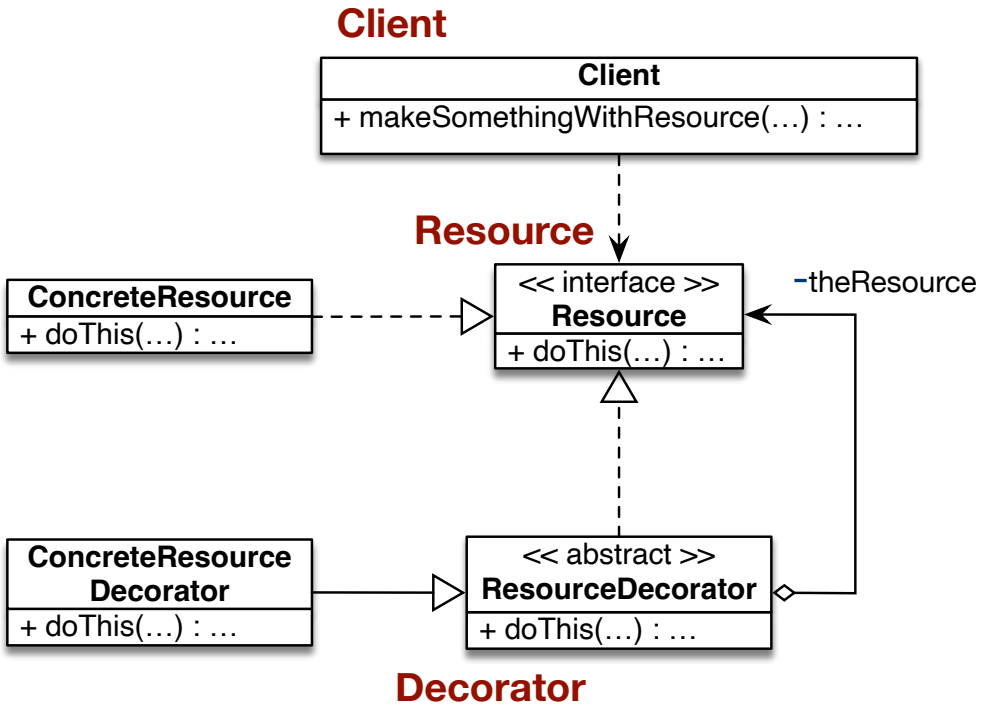
Decorator, problème (fin)

- L'interface `Pizza` et la classe `AbstractPizza` ne **respectent pas le principe *Ouvert-Fermé***
  - L'ajout d'un nouvel supplément oblige à modifier le code

IUT Valence
Université Grenoble Alpes

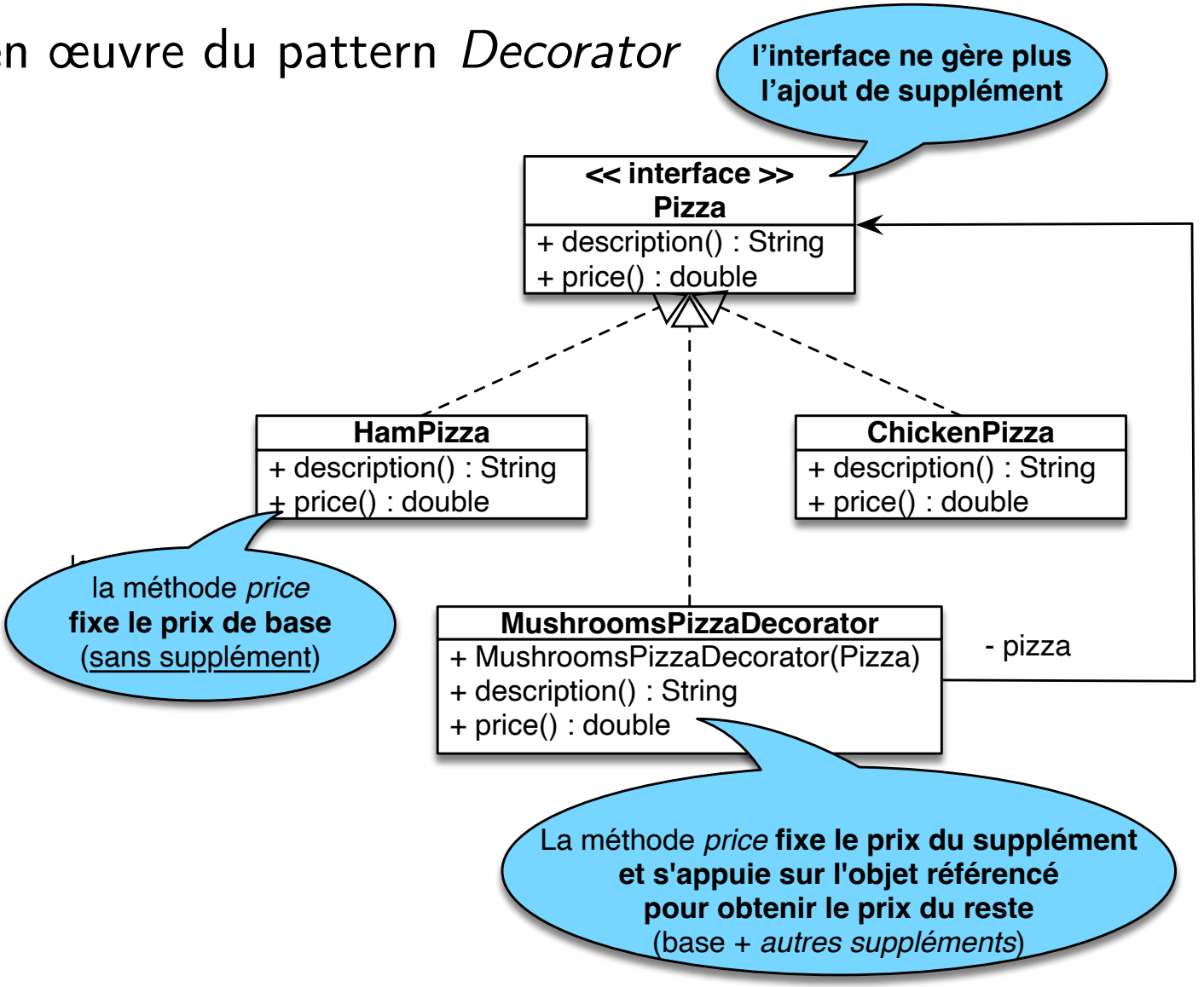
Decorator, définition

- Ajouter dynamiquement des fonctionnalités à un objet **en utilisant la composition et l'interposition plutôt que l'héritage**
  - Le **client** collabore avec une **ressource** (objet *cible* à l'exécution) à travers une interface
  - Le pattern **Decorator** consiste à interposer un objet fournissant l'interface attendue par le client, en enrichissant les fonctionnalités de l'objet cible.



Decorator, solution

- Mise en œuvre du pattern *Decorator*



- Le décorateur `MushroomsPizzaDecorator` décore une pizza avec l'ajout d'un supplément champignons

IUT Valence Université Grenoble Alpes

Decorator, solution

- Ecrire la classe `MushRoomsPizzaDecorator`

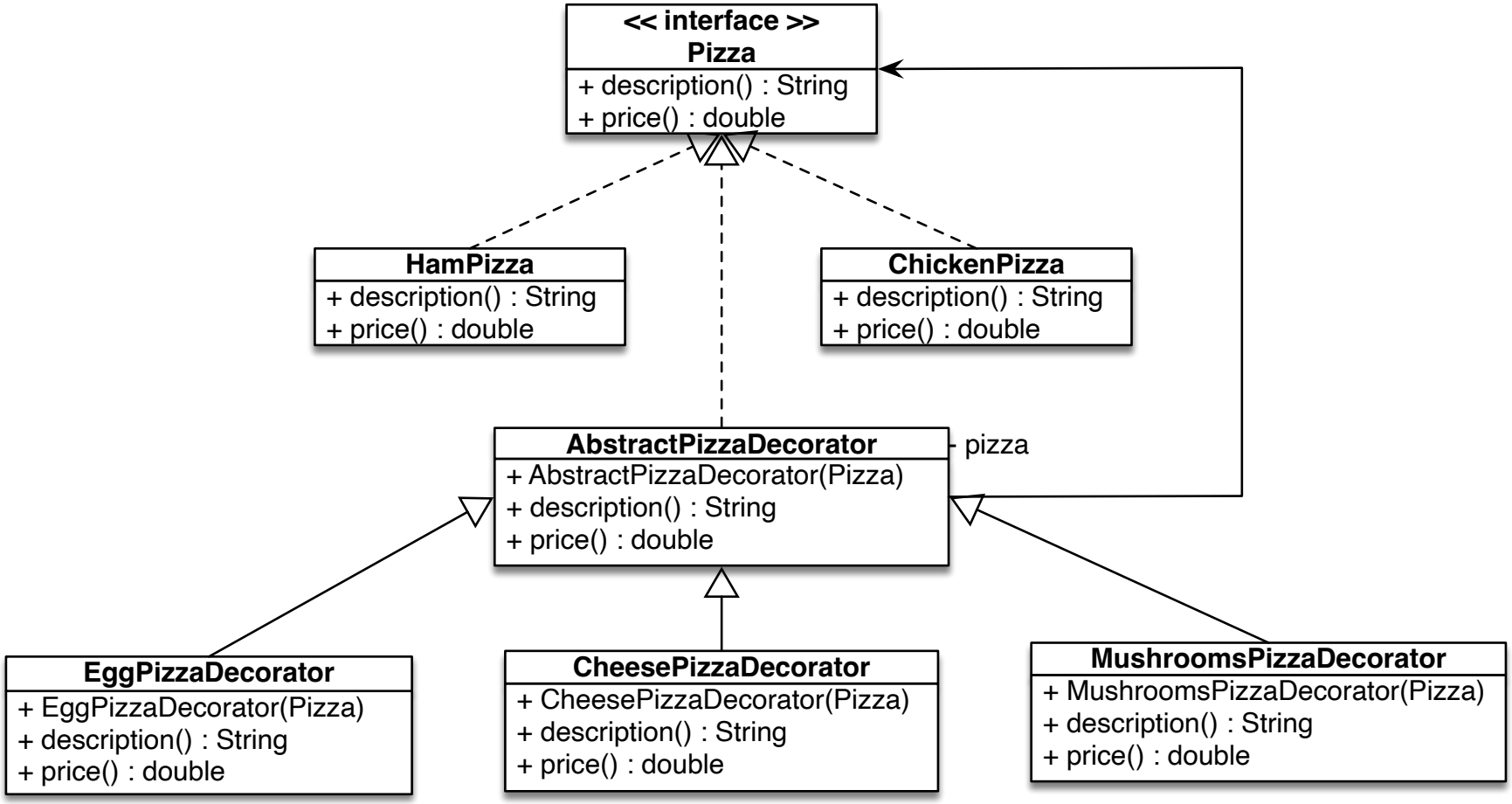
IUT A Valence
Université Grenoble Alpes

Decorator, solution

IUT Valence
Université Grenoble Alpes

Decorator, solution

- Référence vers la pizza décorée factorisable à un niveau abstrait (`AbstractPizzaDecorator`)



IUT Valence
Université Grenoble Alpes

Decorator, solution

- Ecrire une application qui *assemble* une pizza au jambon avec suppléments champignons/oeuf/fromage

Decorator, solution

IUT Valence
Université Grenoble Alpes

Decorator, solution (fin)

- Une pizza au jambon, avec supplément champignons, fromage et oeuf utilise **3 décorateurs**
  - un décorateur `MushroomsPizzaDecorator` **qui décore** ...
  - un décorateur `CheesePizzaDecorator` **qui décore** ...
  - un décorateur `EggPizzaDecorator` **qui décore** ...
  - une `HamPizza`

- Question subsidiaire : quel pattern pour construire la pizza ?

IUT Valence Université Grenoble Alpes

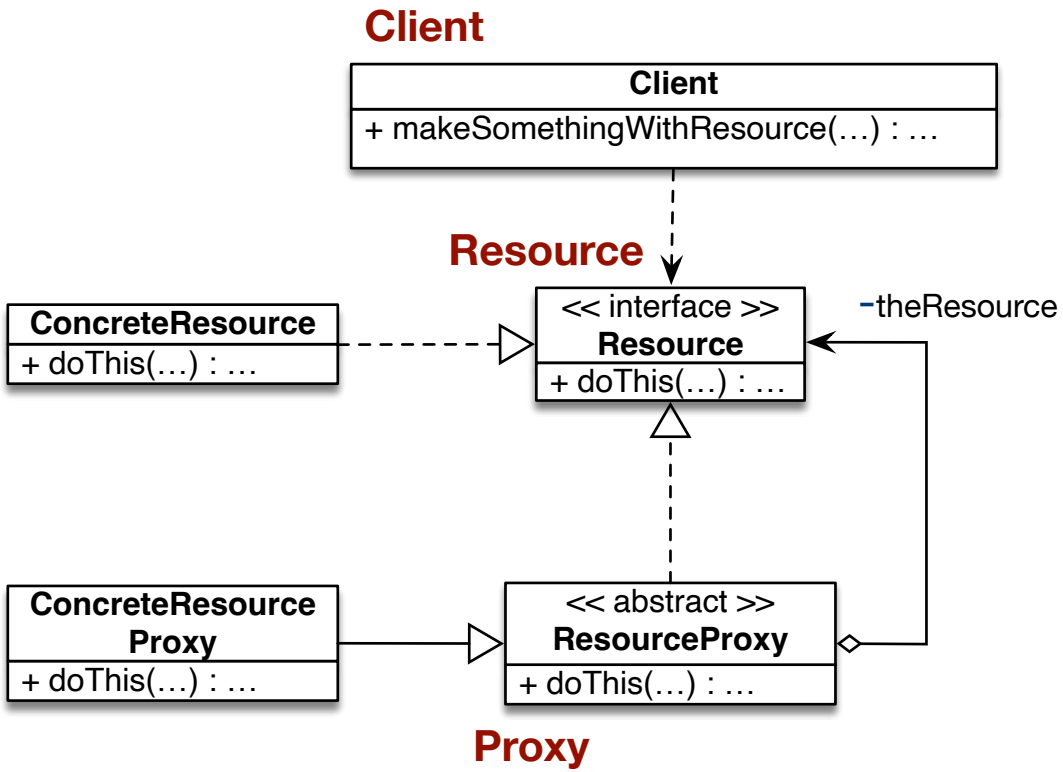
Proxy, problème

- Comment
  - rendre transparente la **distribution** d'une l'application ?
    - i.e. faire croire que le comportement d'un objet distant est local
  - **retarder la création** d'objets lorsqu'elle est coûteuse ?
    - i.e. faire croire qu'un objet non utilisé est créé alors qu'il ne l'est pas
  - **contrôler l'accès** à une ressource ?
    - i.e. bloquer des appels de méthodes

IUT A Valence
Université Grenoble Alpes

Proxy, définition

- Représenter un objet pour en **détourner l'accès**
  - Le **client** collabore avec une **ressource** (objet *cible* à l'exécution) à travers une interface
  - Le pattern **Proxy** consiste à interposer un objet fournissant l'interface attendue par le client , en détournant l'accès à l'objet cible



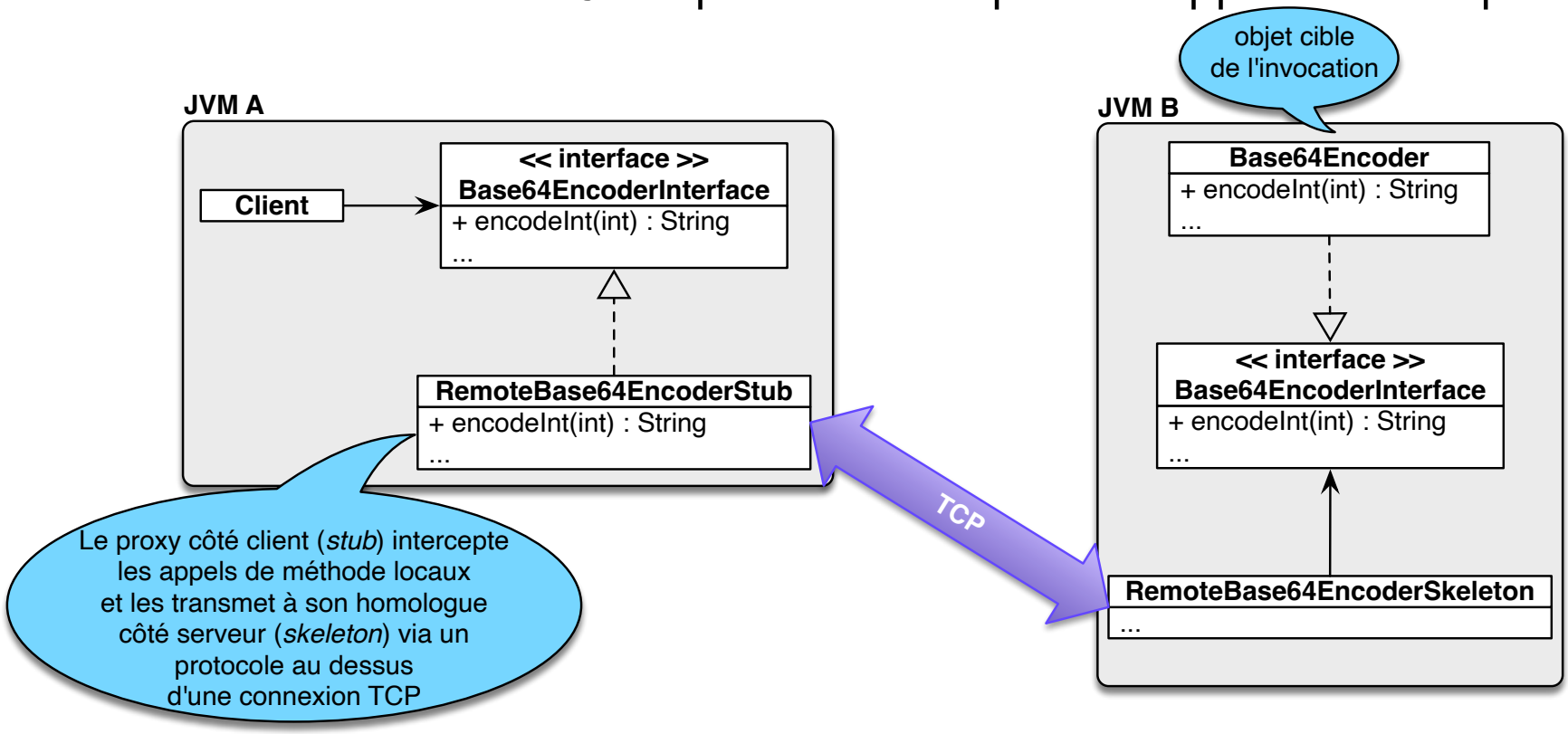
Types de proxys

- *Remote proxy*
  - Rendre transparente la **distribution** de l'application
- *Virtual proxy*
  - **Retarder la création** d'objets lorsqu'elle est coûteuse
- *Protection proxy*
  - **Contrôler l'accès** à une ressource
- API standard pour créer des proxys (*Dynamic Proxies*)

IUT Valence Université Grenoble Alpes

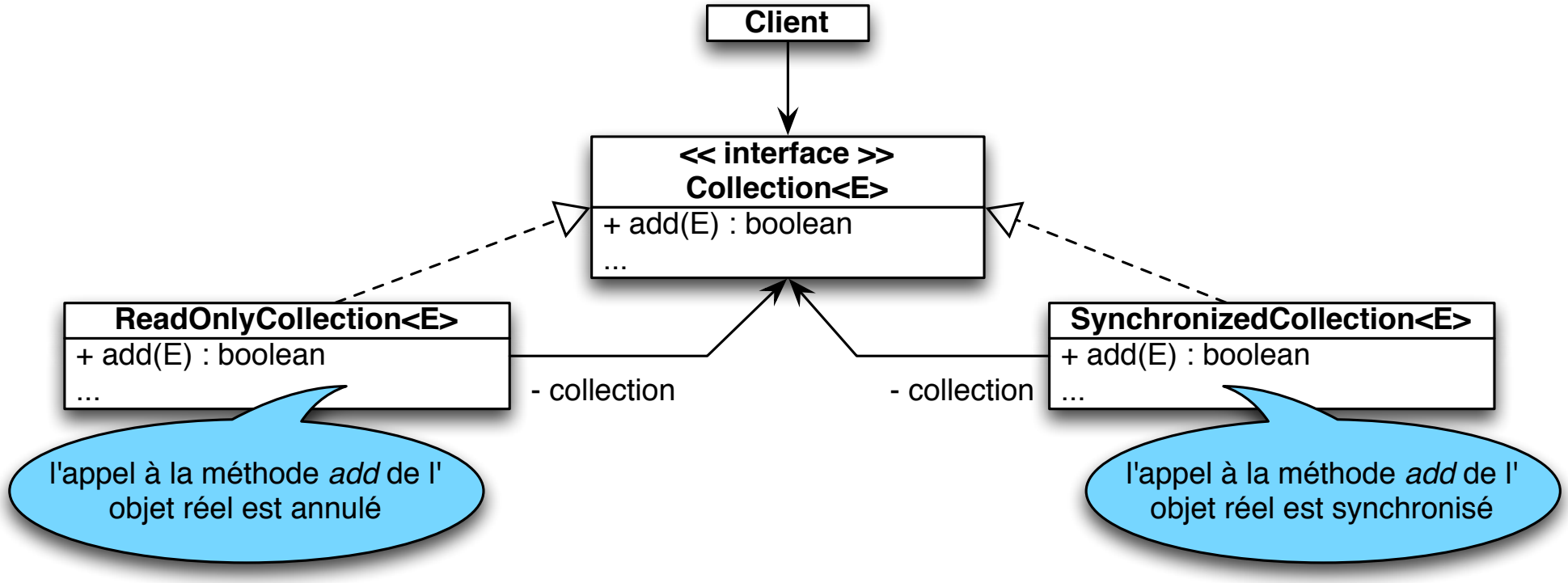
Remote Proxy

- Exemple de RMI (*Remote Method Invocation*)
  - Framework standard Java pour la conception d'applications réparties



Protection Proxy

- La classe `java.util.Collections` fournit des méthodes permettant d'obtenir des versions synchronisées ou lecture seule de collections



Fin !

IUT Valence
Université Grenoble Alpes