Polytech Paris-Sud
Formation initiale 4e année
Spécialité informatique
Année 2016-2017

Vérification et validation

Cours 3
Spécification des opérations

Delphine Longuet
delphine.longuet@lri.fr

http://www.lri.fr/~longuet/Enseignements/16-17/Et4-VV

Opérations et méthodes

Opération : Service qui peut être demandé au système

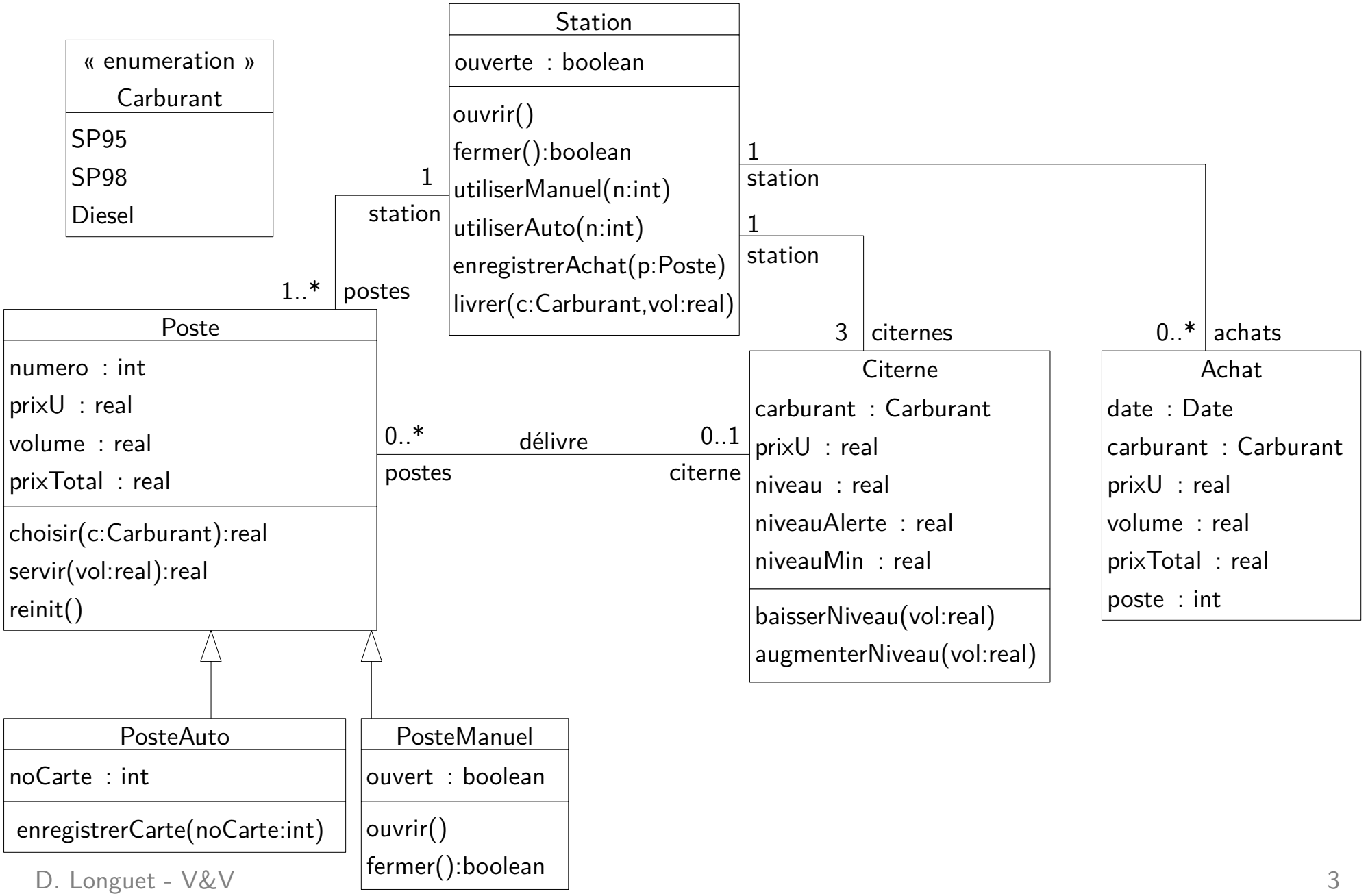
Méthode : Implémentation d'une opération, rattachée à une classe

- Rédéfinition possible d'une méthode dans une sous-classe, implantant la même opération
- Surcharge possible d'une méthode dans une classe (paramètres différents), implantant la même opération

Effets possibles d'une opération (non exclusifs) :

- Renvoyer un résultat
- Modifier l'état du système (modification de la valeur des attributs, ajout ou suppression de liens entre objets, création ou destruction d'objets)

Diagramme de classes de Tatol (conception)



Spécification d'opérations

Spécification d'une opération :

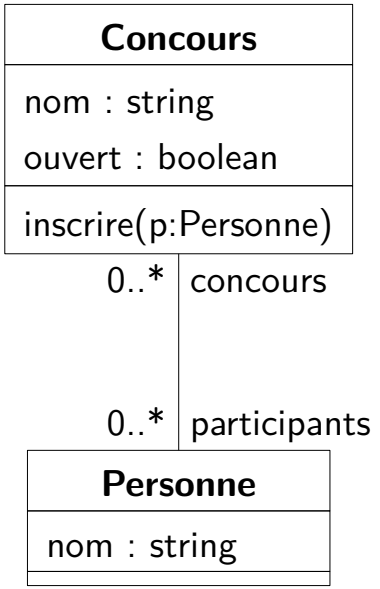
- Signature complète : (classe), nom, arguments, type du résultat
- Pré-condition : condition à l'entrée garantie par l'appelant, condition nécessaire pour exécuter l'opération
- Post-condition : condition à la sortie garantie par l'opération, propriété qui doit être vérifiée à la fin de l'exécution

Inscription d'une personne à un concours

Signature : Concours :: inscrire(p : Personne)

Pré-condition : le concours est ouvert aux inscriptions

Post-condition : la personne p est inscrite au concours

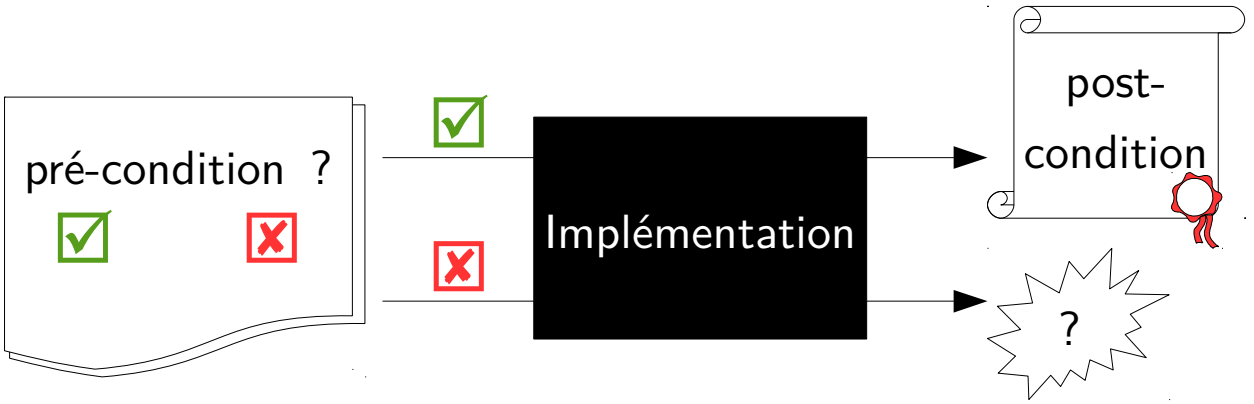


Conception par contrat

Principe : contrat entre l'opération appelée et son appelant

- Appelant responsable d'assurer que la pré-condition est vraie
- Implémentation de l'opération appelée responsable d'assurer la terminaison et la post-condition à la sortie, si la pré-condition est vérifiée à l'entrée

Si la pré-condition n'est pas vérifiée, aucune garantie sur l'exécution de l'opération



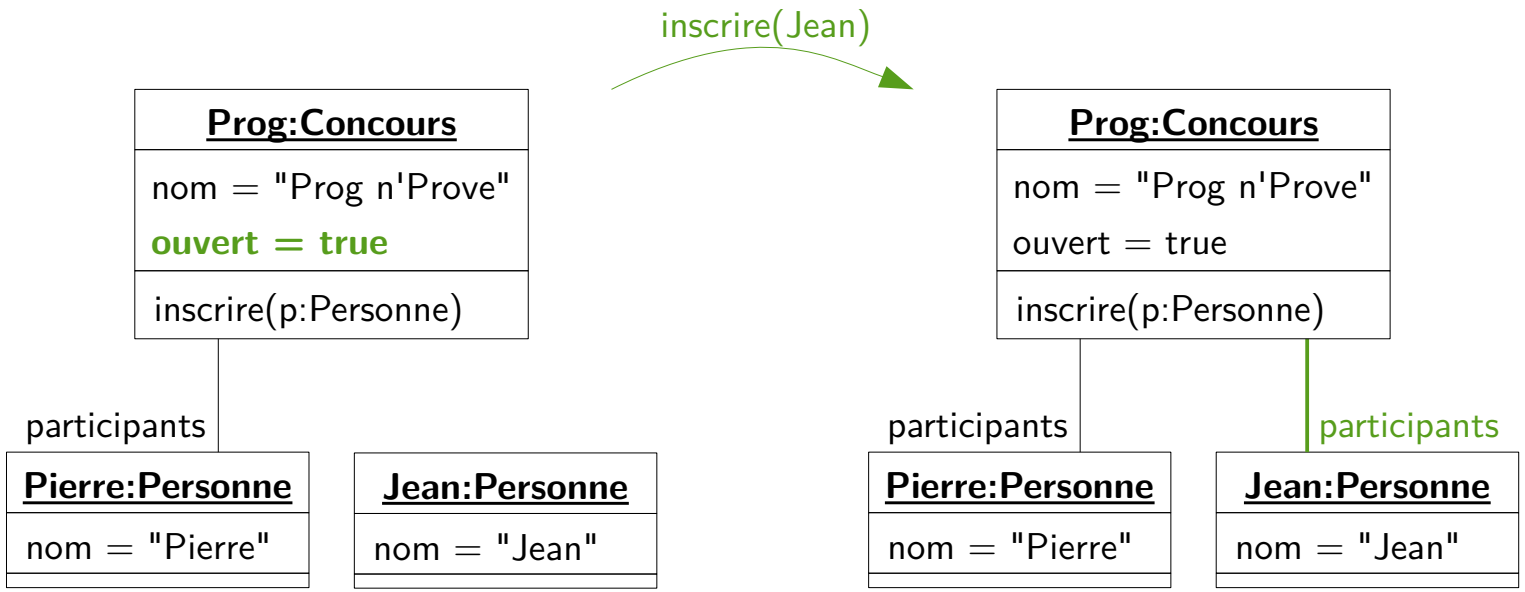
Spécification d'une opération

Concours :: inscrire(p : Personne)

pre : le concours est ouvert aux inscriptions

post : la personne est inscrite au concours

Comportement possible de l'implantation



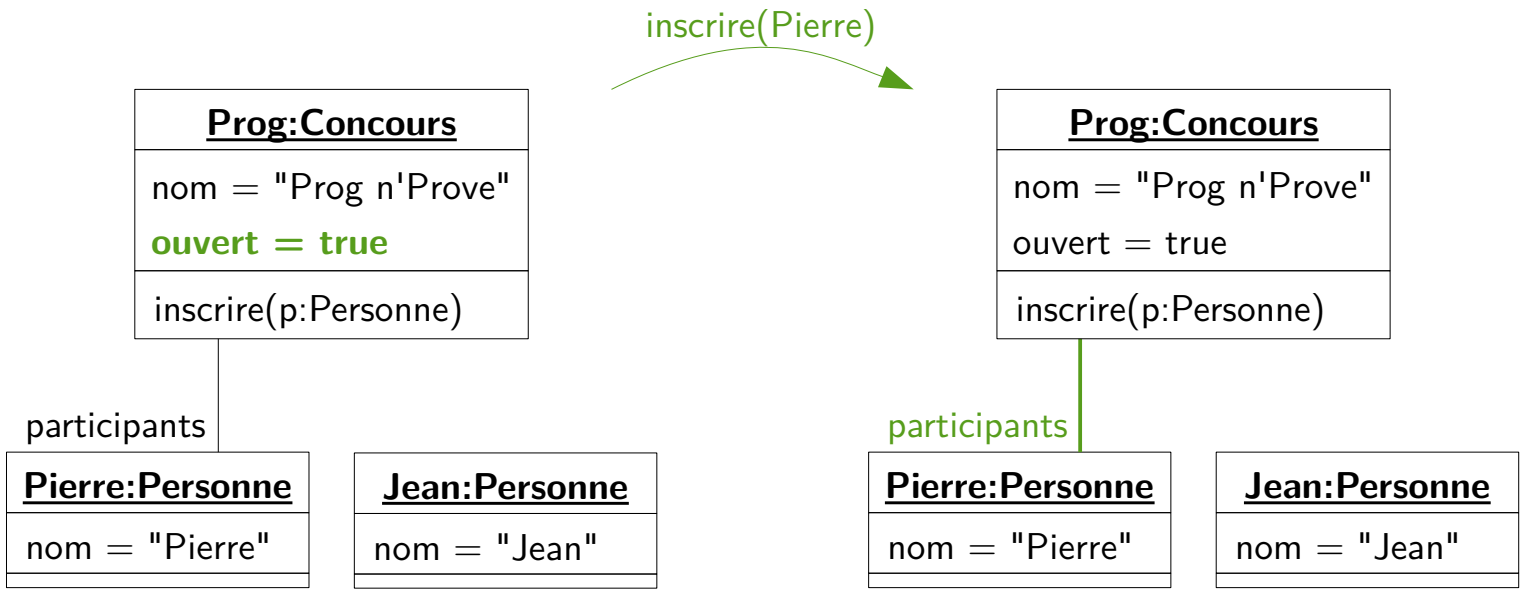
Spécification d'une opération

Concours :: inscrire(p : Personne)

pre : le concours est ouvert aux inscriptions

post : la personne est inscrite au concours

Comportement possible de l'implantation



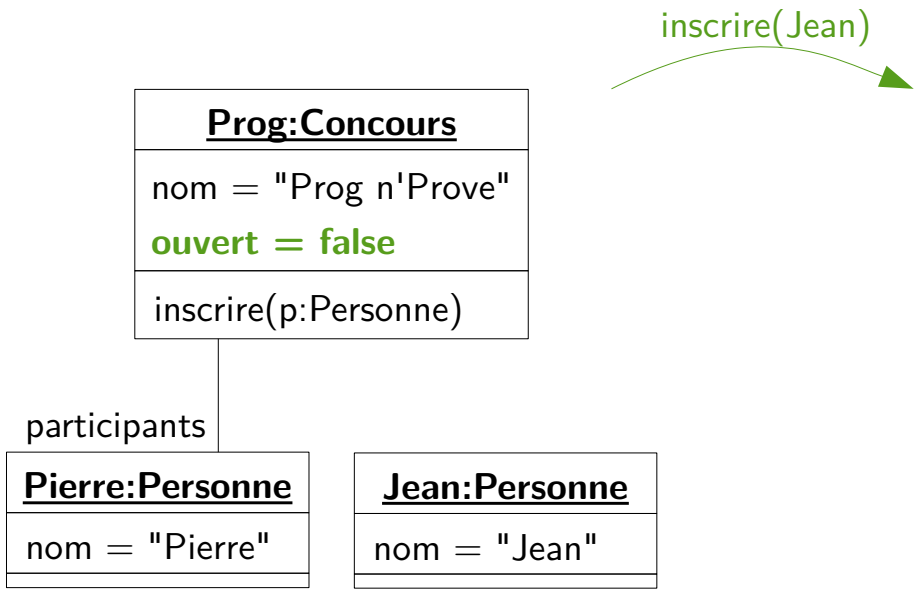
Spécification d'une opération

Concours :: inscrire(p : Personne)

pre : le concours est ouvert aux inscriptions

post : la personne est inscrite au concours

Comportement possible de l'implantation



?

Etat inchangé

Jean inscrit

Plus aucun inscrit

Levée d'une exception

Crash du système...

Implantations possibles d'une spécification

Spécification d'une opération représente un ensemble d'implantations possibles de cette opération

Liste :: insérer(e : Élément)
pre : aucune
post : l'élément est présent dans la liste

Implantations possibles :

- l'élément est toujours ajouté en tête de liste
- l'élément est inséré de façon triée
- l'élément est inséré seulement s'il n'était pas déjà présent

et même :

- la liste est réduite à cet élément
- l'élément remplace tous les éléments déjà présents

Correction d'une spécification

Spécification correcte : permet toutes les implantations voulues, mais peut-être plus

Liste :: insérer(e : Élément)
post : l'élément est présent dans la liste

Trop permissive : permet des implantations non voulues

↳ Permet que le reste de la liste soit modifié

Complétude d'une spécification

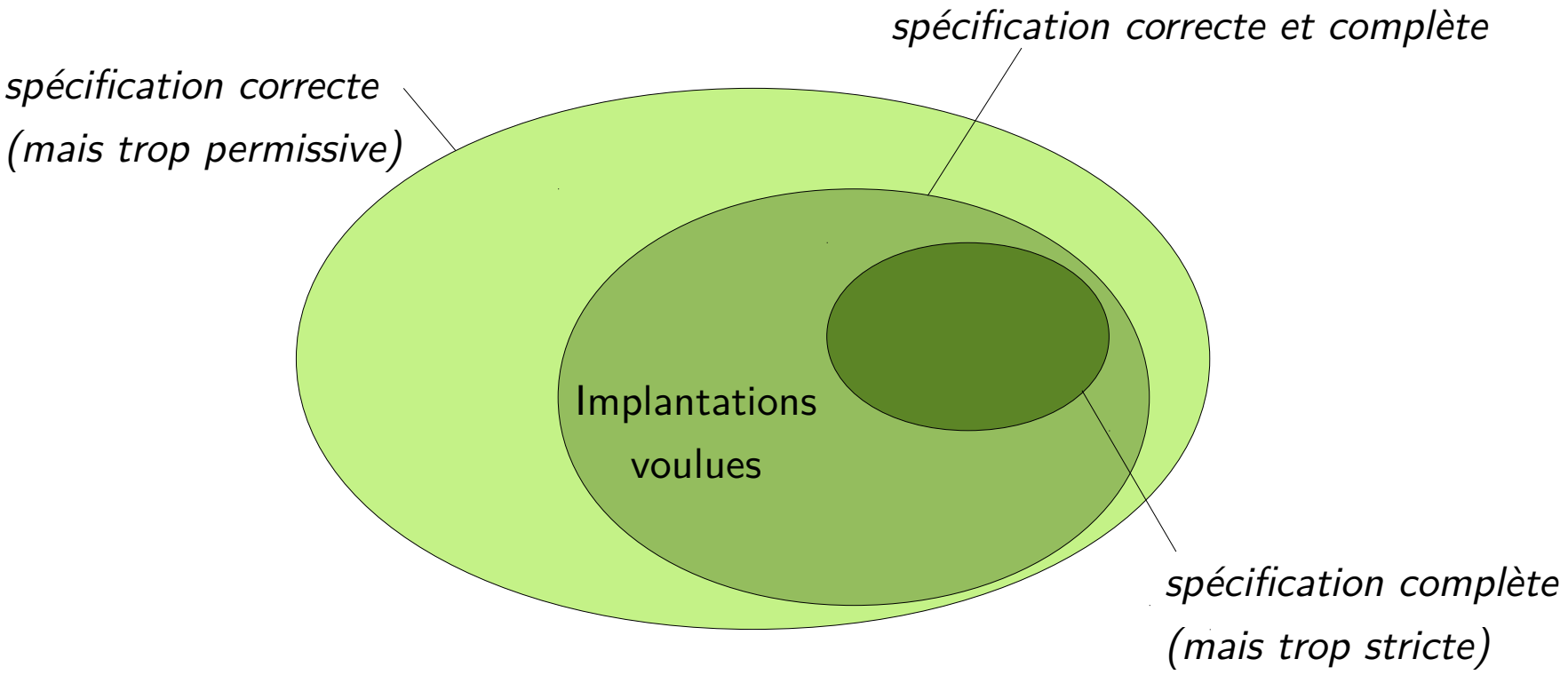
Spécification compléte : permet seulement des implantations voulues, mais peut-être pas toutes

Liste :: insérer(e : Élément)
post : l'élément est inséré en tête de liste
post : le reste de la liste n'est pas modifié

Trop stricte : interdit des implantations voulues

↳ Ne laisse pas le choix de la place de l'élément dans la liste

Implantations possibles d'une spécification



Formalisation des pré et post-conditions

Signature :

Classe :: opération(arg_1 : $type_1$, ..., arg_n : $type_n$) : type_retour

Mots-clés :

this : instance de **Classe** sur laquelle est vérifiée la condition

old(exp) : valeur de l'expression **exp** à l'état précédent

result : valeur renvoyée par l'opération

} à utiliser uniquement dans les post-conditions

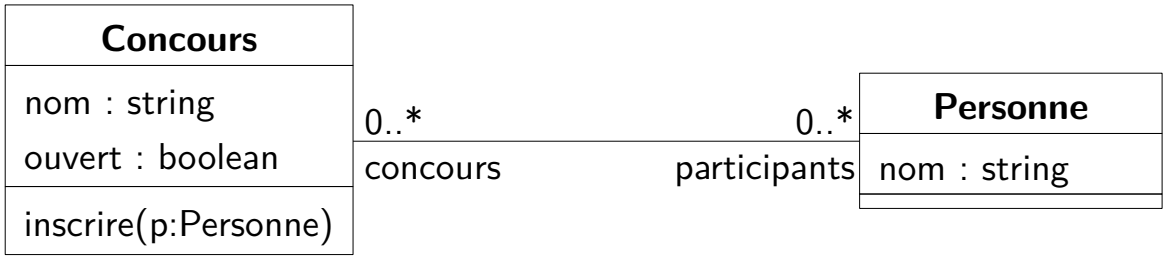
Formalisation des pré et post-conditions

Inscrire une personne à un concours

Pré-condition : le concours est ouvert aux inscriptions

Post-condition : la personne est inscrite au concours

Concours :: inscrire(p : Personne)
pre : this.ouvert
post : p ∈ this.participants



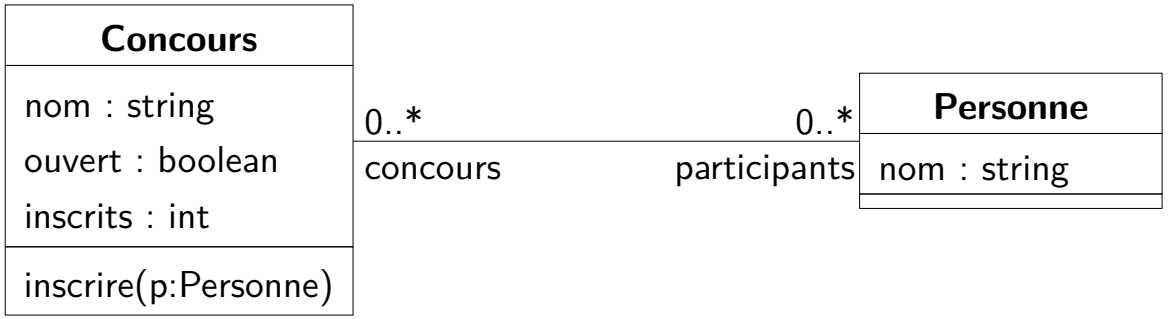
Formalisation des pré et post-conditions

Inscrire une personne à un concours

Pré-condition : le concours est ouvert aux inscriptions

Post-condition : la personne est inscrite au concours et le nombre d'inscrits a augmenté de 1, si la personne n'était pas déjà inscrite

Concours :: inscrire(p : Personne)
pre : this.ouvert
post : $p \notin \text{old}(\text{this.participants}) \Rightarrow p \in \text{this.participants} \wedge p.\text{inscrits} = \text{old}(p.\text{inscrits}) + 1$



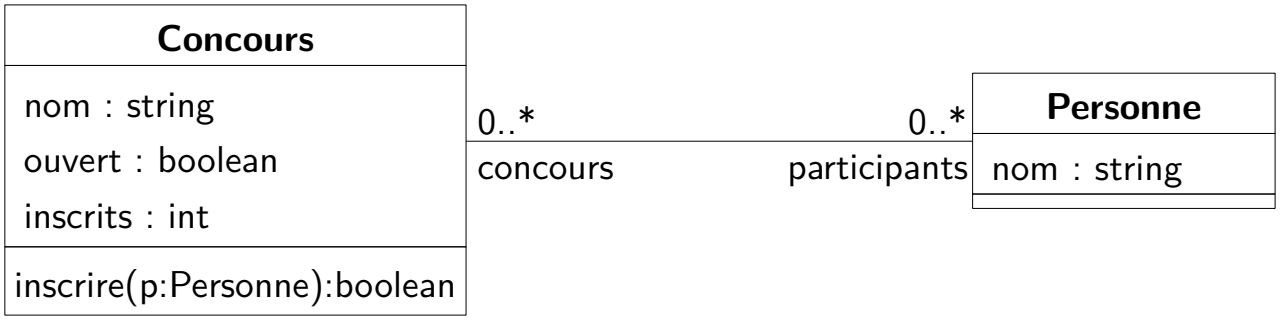
Formalisation des pré et post-conditions

Inscrire une personne à un concours

Pré-condition : le concours est ouvert aux inscriptions

Post-conditions : la personne est inscrite au concours, le nombre d'inscrits a augmenté de 1 et le résultat renvoyé est *true*, si la personne n'était pas déjà inscrite. Le résultat renvoyé est *false* si la personne était déjà inscrite

$$
\begin{aligned}
&\text{Concours :: inscrire(p : Personne) : boolean}\\
&\textbf{pre} : \text{this.ouvert}\\
&\textbf{post} : p \notin \text{old(this.participants)} \Rightarrow (p \in \text{this.participants}\\
&\qquad\qquad \wedge\ \text{this.inscrits} = \text{old(this.inscrits)} + 1\\
&\qquad\qquad \wedge\ \text{result} = \text{true})\\
&\textbf{post} : p \in \text{old(this.participants)} \Rightarrow \text{result} = \text{false}
\end{aligned}
$$



Pré-condition

Vérifiée à l'entrée par l'appelant, doit être réaliste

Concours :: inscrire(p : Personne)
pre : this.ouvert ← comment le garantir ?
post : p ∈ this.participants

Autre solution : affaiblissement de la pré-condition et traitement des différents cas par l'opération

Concours :: inscrire(p : Personne)
pre : true
post : old(this.ouvert) ⇒ p ∈ this.participants
post : ¬ old(this.ouvert) ⇒ ...

Contrats différents avec la classe appelante

Quelques exemples

Minimum d'un ensemble

Pré-condition : l'ensemble n'est pas vide

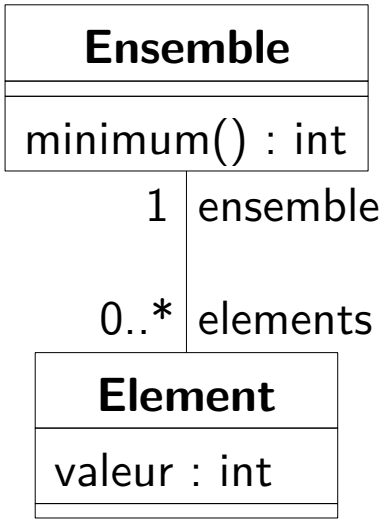
Post-condition : le résultat est la valeur la plus petite de l'ensemble

Ensemble :: minimum() : int

pre : this.elements $\neq \emptyset$

post : result $\in$ this.elements.valeur

post : $\forall$ e $\in$ this.elements, result $\leq$ e.valeur



Quelques exemples

Fermer la station essence

Pré-condition : elle est ouverte

Post-condition : si un poste manuel est utilisé, elle est restée ouverte et le résultat renvoyé est *false* ; si tous les postes manuels sont libres, la station est fermée ainsi que tous les postes manuels, et le résultat renvoyé est *true*

Station :: fermer() : boolean

pre : this.ouverte = true

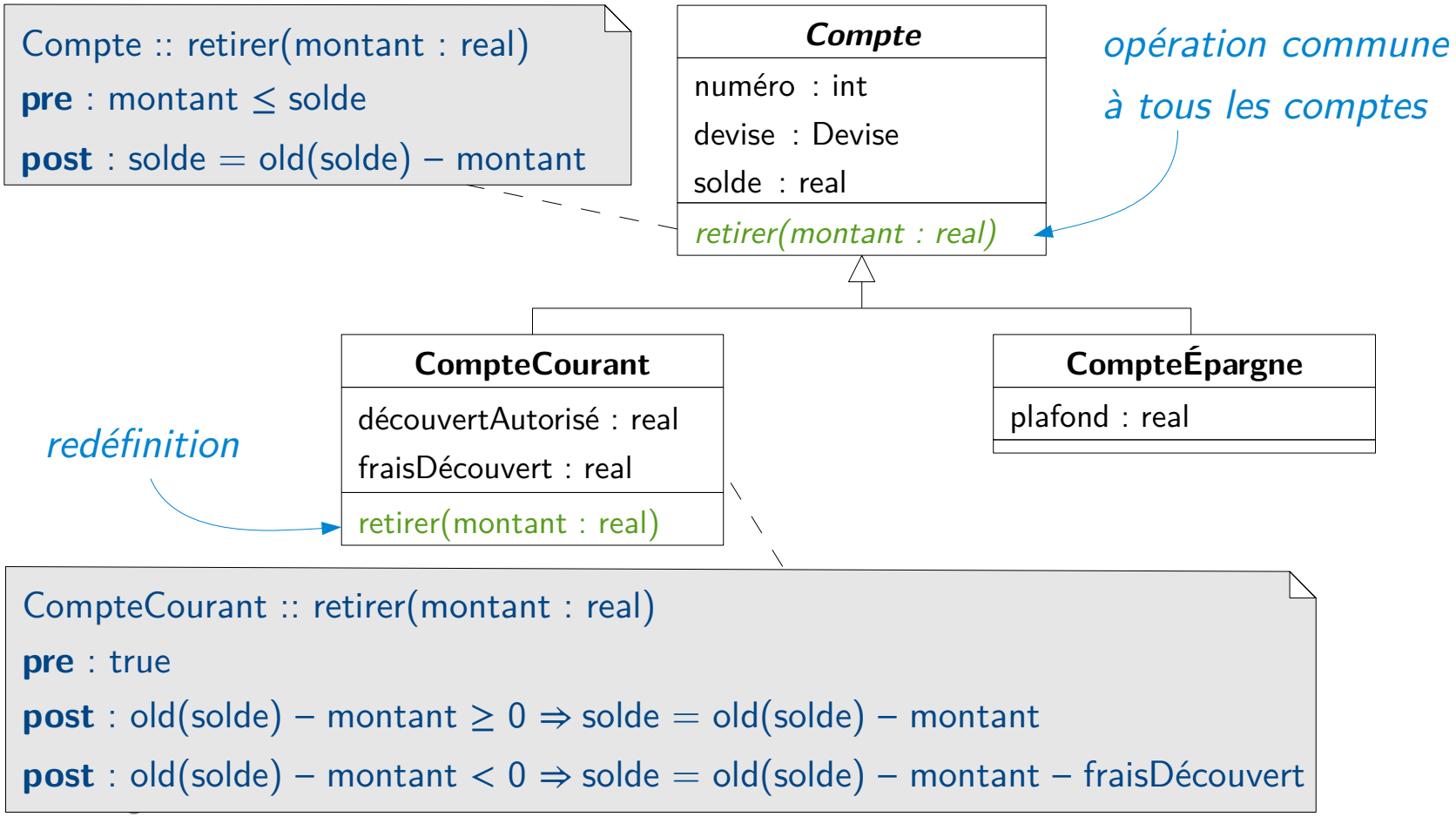
post : $(\exists\ p \in \text{PosteManuel},\ p.\text{citerne} \neq \emptyset) \Rightarrow \text{this.ouverte} = \text{true} \wedge \text{result} = \text{false}$

post : $(\forall\ p \in \text{PosteManuel},\ p.\text{citerne} = \emptyset) \Rightarrow \text{this.ouverte} = \text{false} \wedge \text{result} = \text{true}$
$\wedge\ \forall\ p \in \text{PosteManuel},\ p.\text{ouvert} = \text{false}$

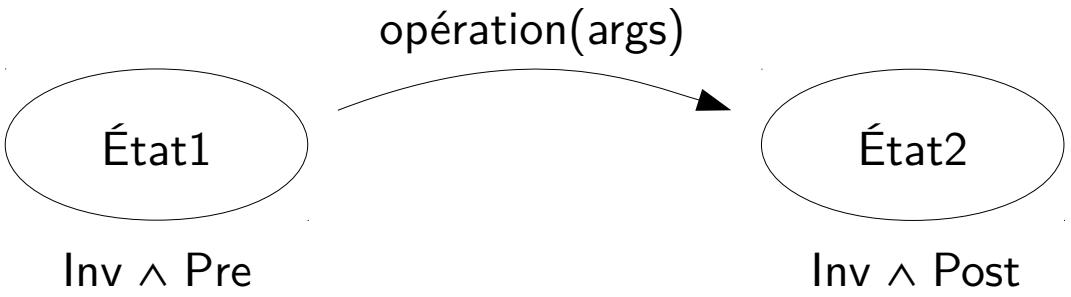
Héritage des spécifications

Opération redéfinissant une opération **op** d'une super-classe :

- Pré-condition égale ou plus faible que **op**
- Post-condition égale ou plus forte que **op**



Aspects sémantiques



Inv : invariants du diagramme de classes

Pre : pré-condition de l'opération

Post : post-condition de l'opération

Sémantique de la spécification de l'opération, exprimée dans État2 :

$$\text{old}(\text{Inv}) \wedge \text{old}(\text{Pre}) \wedge \text{Inv} \wedge \text{Post}$$

Spécifications par contrats

Langages : OCL pour UML, ACSL pour C, JML pour Java, Spec# pour C#, Eiffel...

Utilisations

- Runtime assertion checking : vérification des spécifications à l'exécution
- Extended static checking : recherche statique d'erreurs courantes (indice hors des bornes, problèmes de pointeurs...)
- Vérification déductive : preuve mathématique que le programme satisfait sa spécification