Design Patterns, State, Observer

Sébastien Jean

IUT de Valence
Département Informatique

v5.2, 22 novembre 2023

IUT Valence
Université Grenoble Alpes

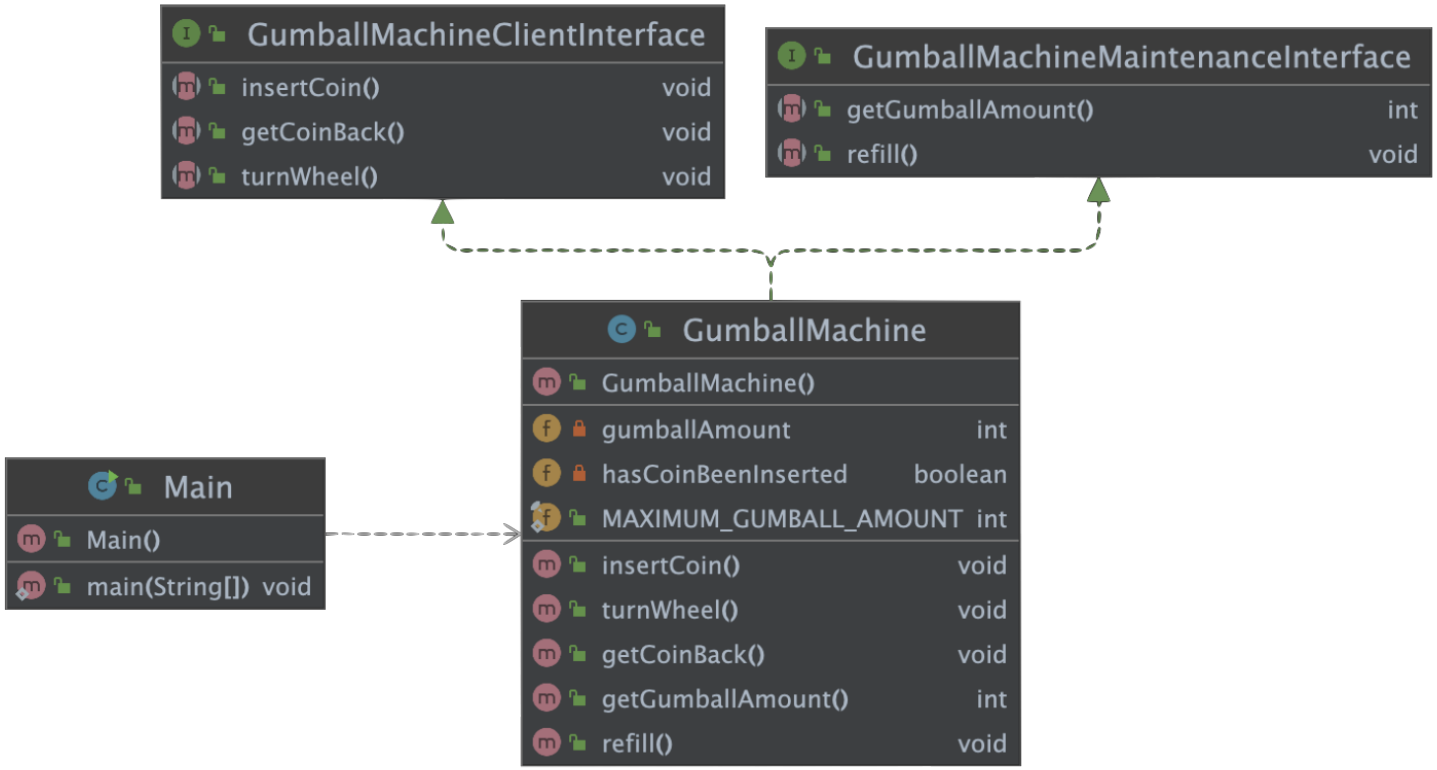
State, problème

- Modélisation d'une machine à bonbons
  - Interface client
    - **stock** (limité) de bonbons
    - emplacement permettant d'**insérer une pièce** à la fois
    - molette permettant de **retirer un bonbon du stock**
    - bouton permettant de **récupérer la pièce**
  - Interface maintenance
    - trappe de maintenance pour refaire le plein et vider le monnayeur

IUT A Valence
Université Grenoble Alpes

State, problème

- Implémentation simple de l'interface client, « orientée transitions »



- Monolithique, peu évolutif/flexible dans son comportement.

IUT Valence Université Grenoble Alpes

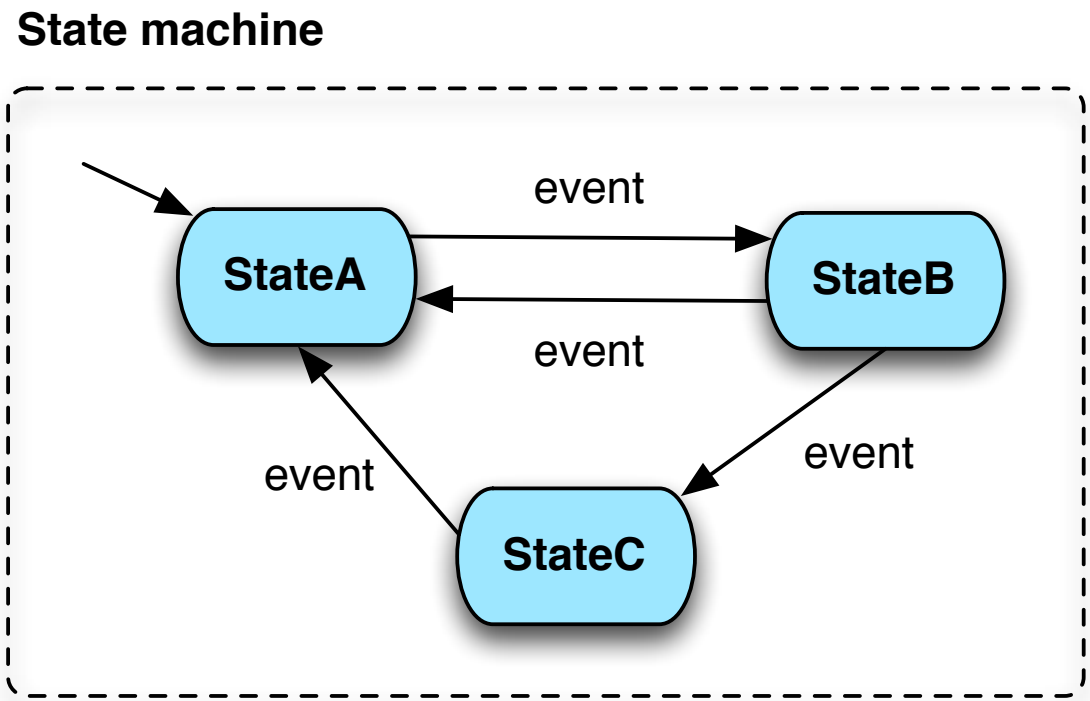
State, définition

- Implémenter, de manière flexible, un comportement
  - **dépendant de l'état interne d'un objet** . . .
  - . . .lorsque cet état prend un **ensemble fini et limité de valeurs possibles**

IUT Valence
Université Grenoble Alpes

State, définition (suite)

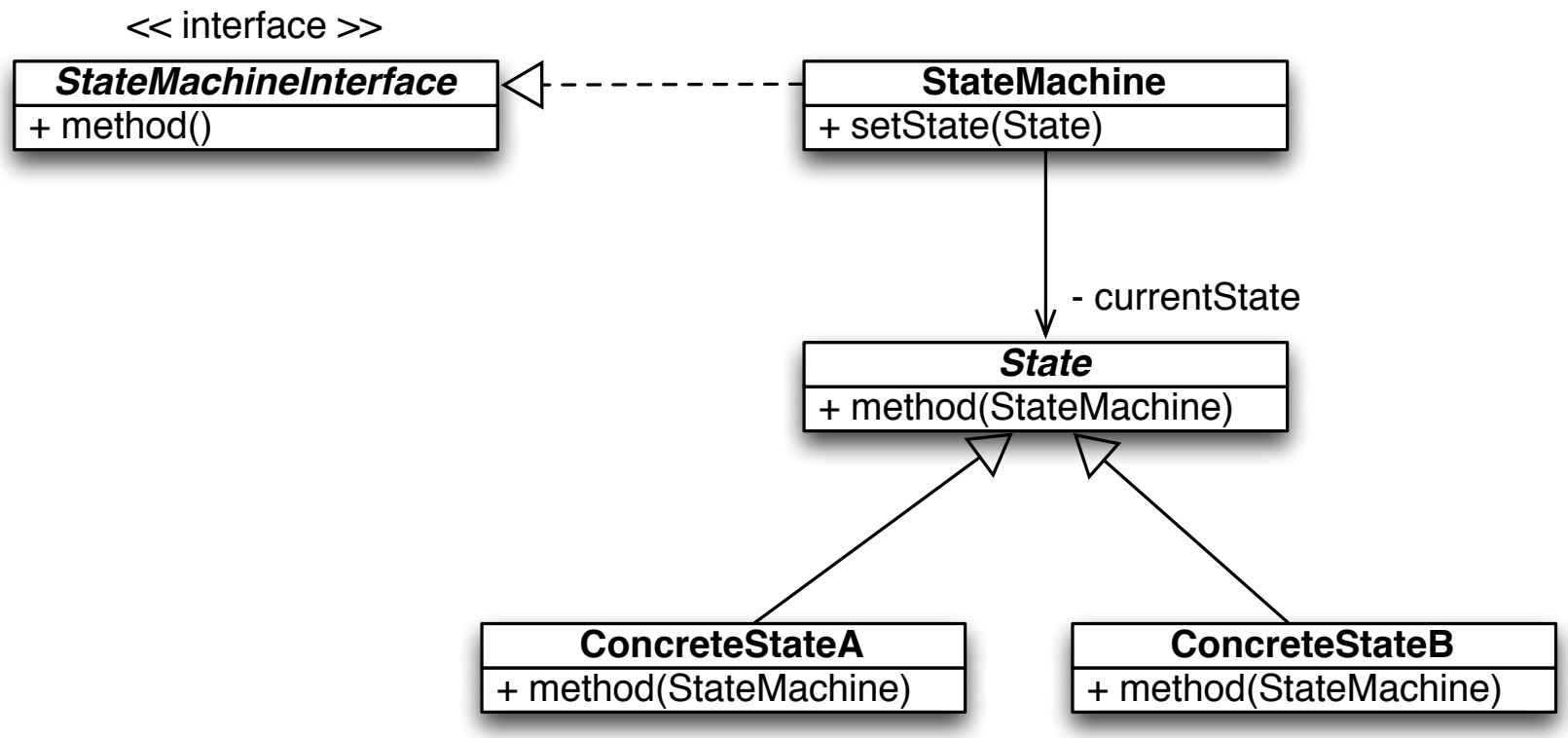
- State met en œuvre des **automates d'états finis**



- Un **état** décrit une situation
- Les **transitions** entre états s'effectuent sur l'occurrence d'**évènements**

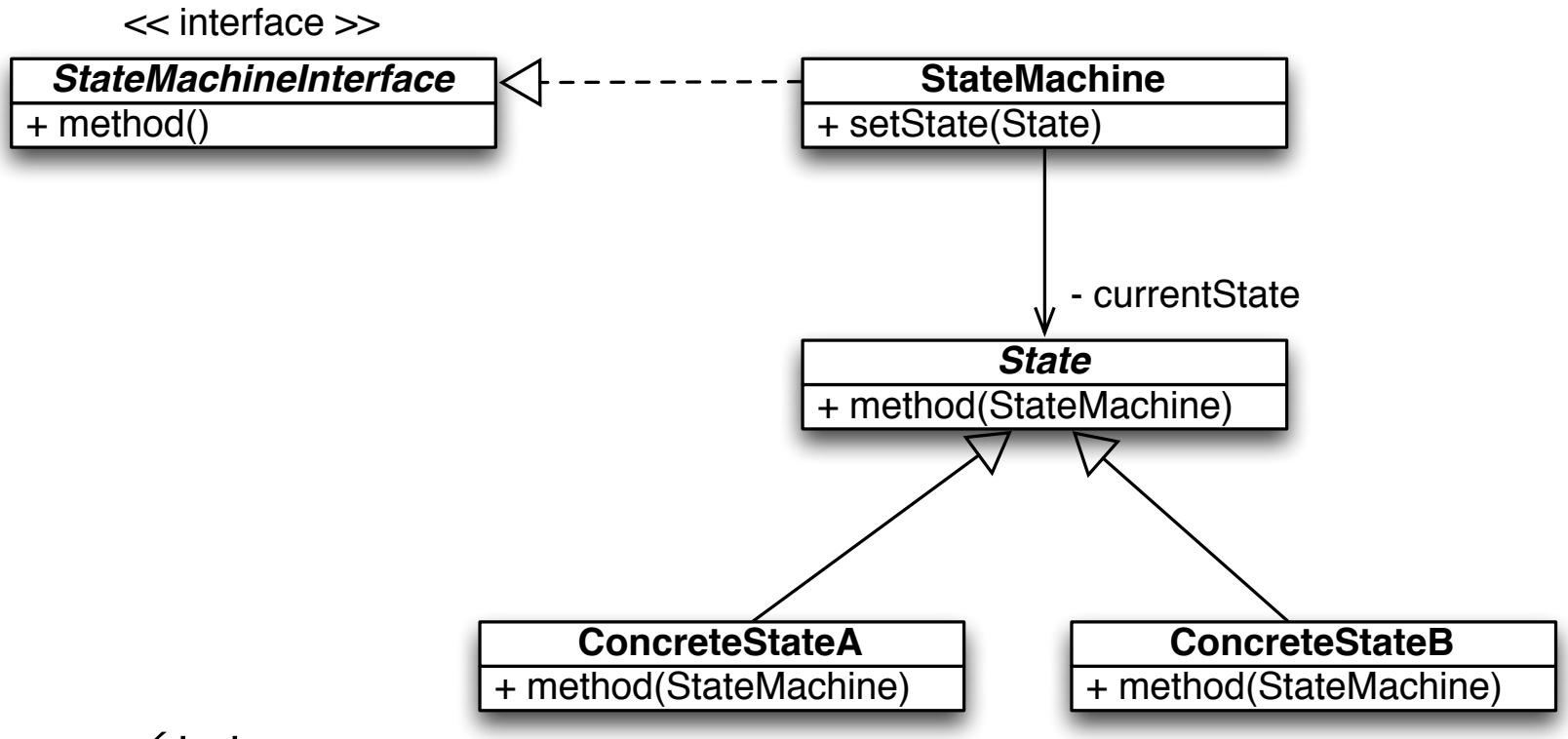
IUT A Valence
Université Grenoble Alpes

State, définition (suite)



- Chaque état est représenté par une classe
- Le client interagit avec la machine d'état à travers une interface
  - La machine d'état **définit le contexte global** et **route les appels vers l'état courant**

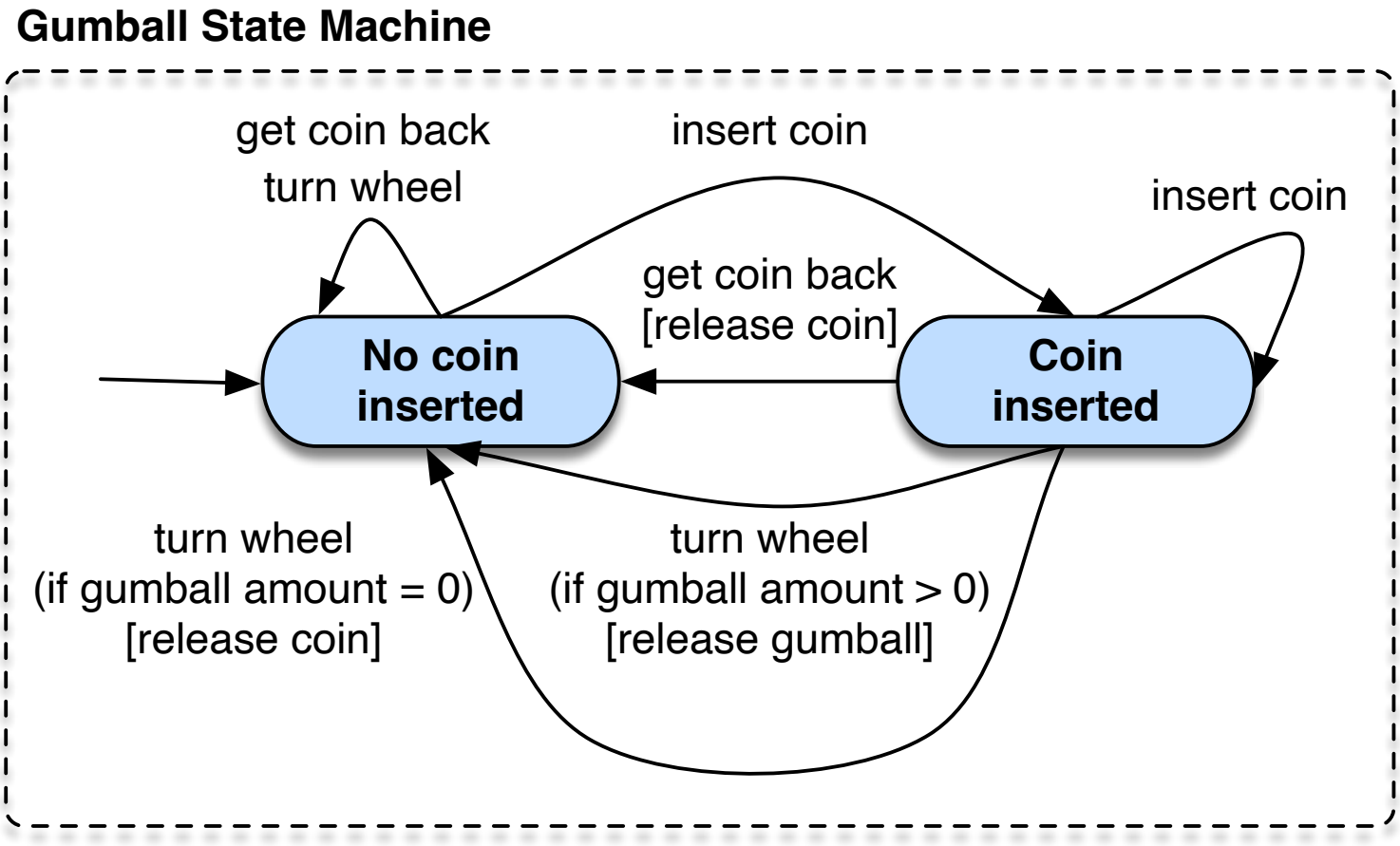
State, définition (suite)



- Chaque état :
  - fournit (sensiblement) **la même interface que la machine d'état**
  - dispose d'une référence sur la machine d'états (attribut ou paramètre) pour **effectuer les transitions** et **mettre à jour le contexte global**

IUT A Valence
Université Grenoble Alpes

State, automate d'états de l'exemple



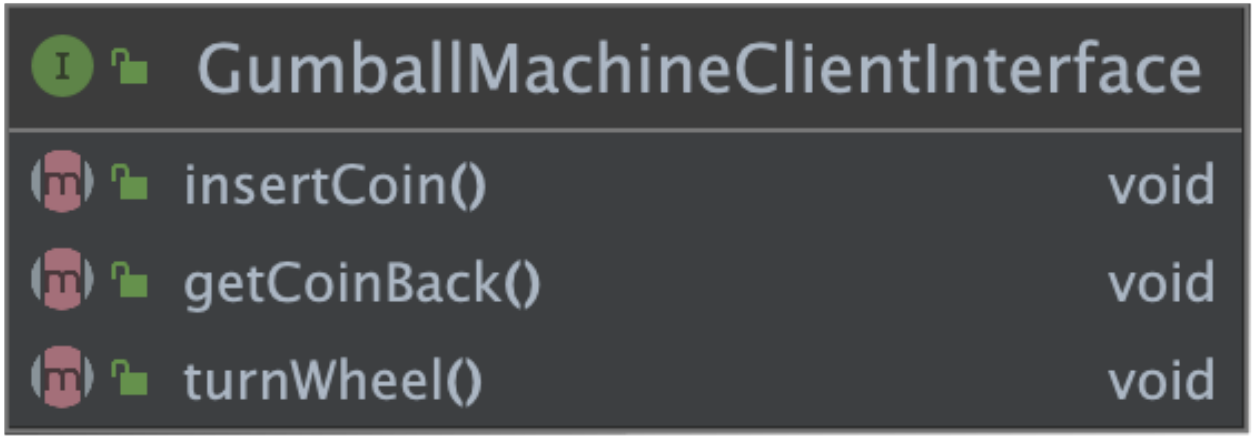
- N.B. : ici les transitions sont étiquetées avec le formalisme *Evènement (Condition d'application) [Actions avant la transition]*

State, exemple

- Interface de la machine d'états ?

State, exemple

- Identique à celle de départ ;-)

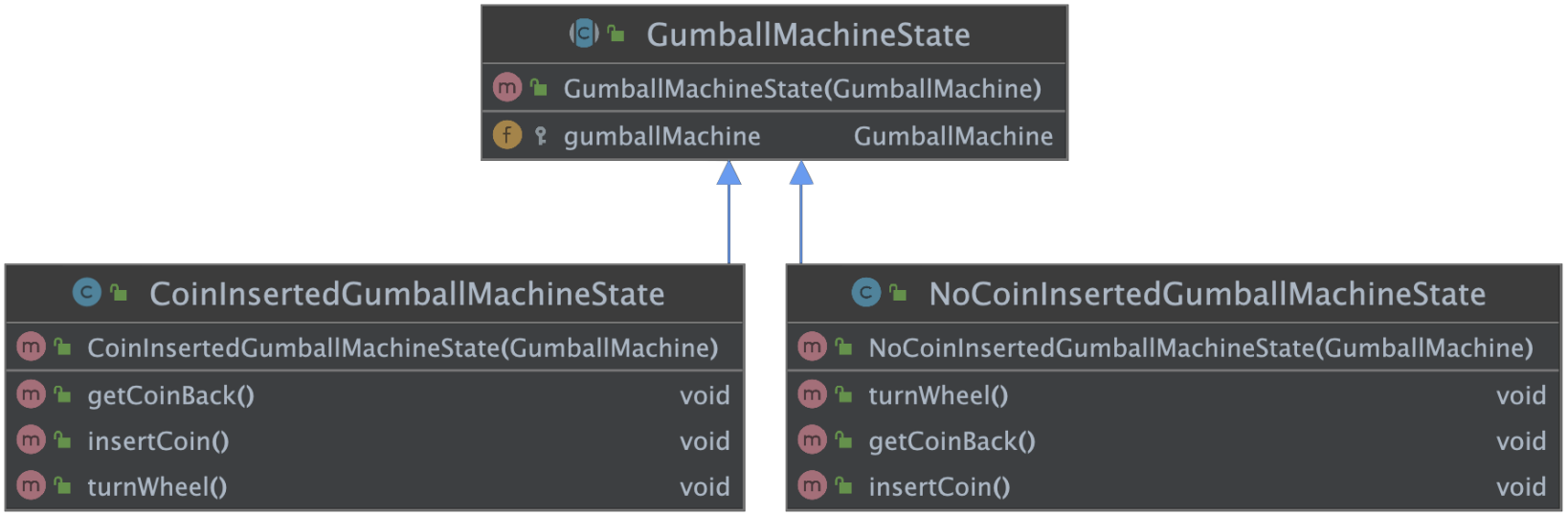


State, exemple

- Etats abstraits, états concrets ?

State, exemple

- Les états exposent la même interface que la machine d'états, la référence de la machine d'états est passée en paramètre du constructeur et stockée dans un attribut
  - La classe abstraite `GumballMachineState` implémente `GumballMachineClientInterface`

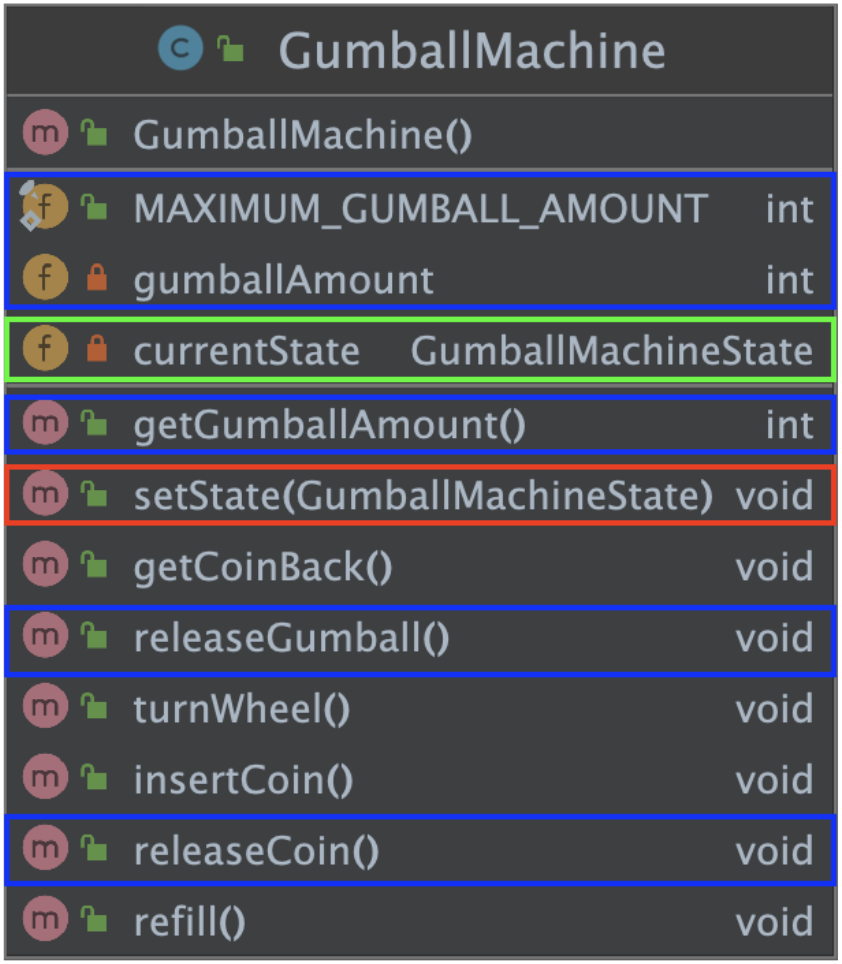


IUT Valence Université Grenoble Alpes

State, exemple

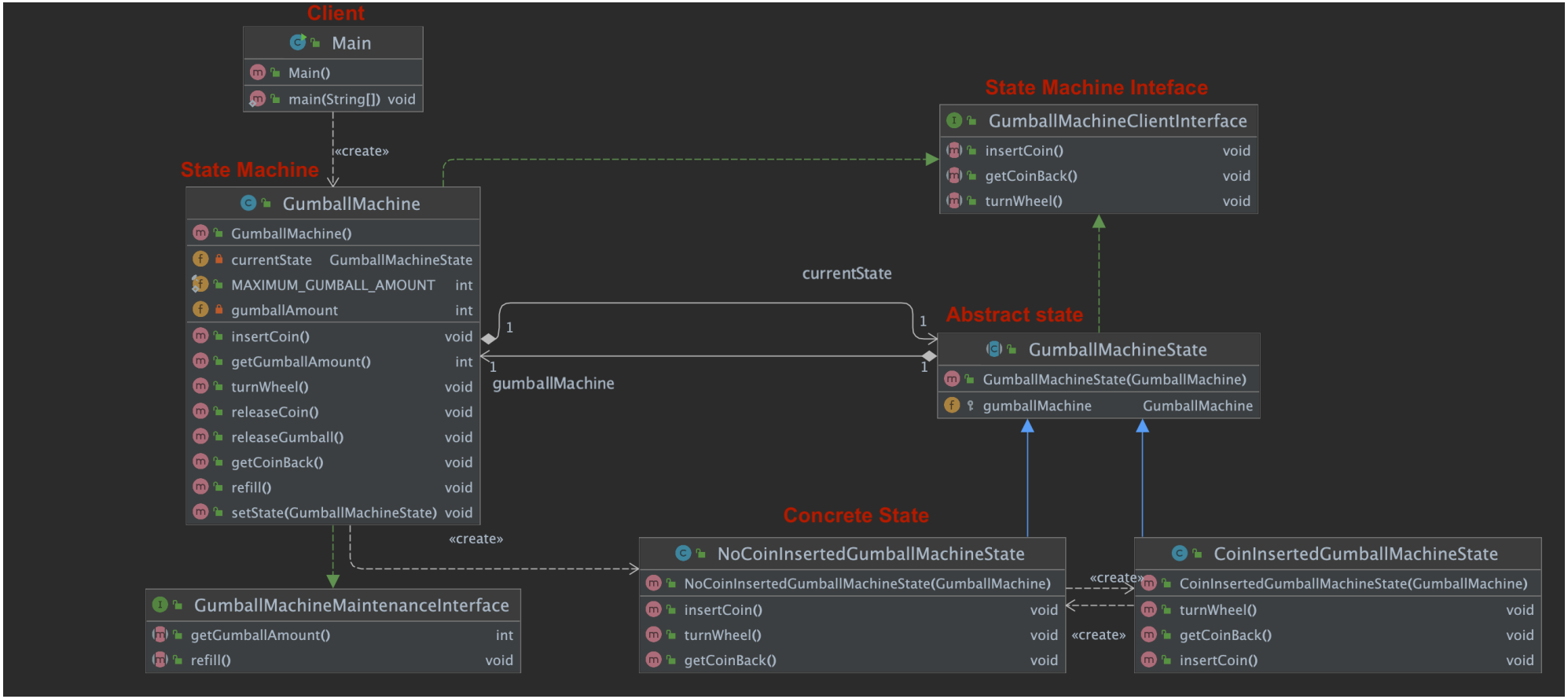
- Machine d'état ?

State, exemple



State, exemple

- Diagramme de classes final



IUT Valence
Université Grenoble Alpes

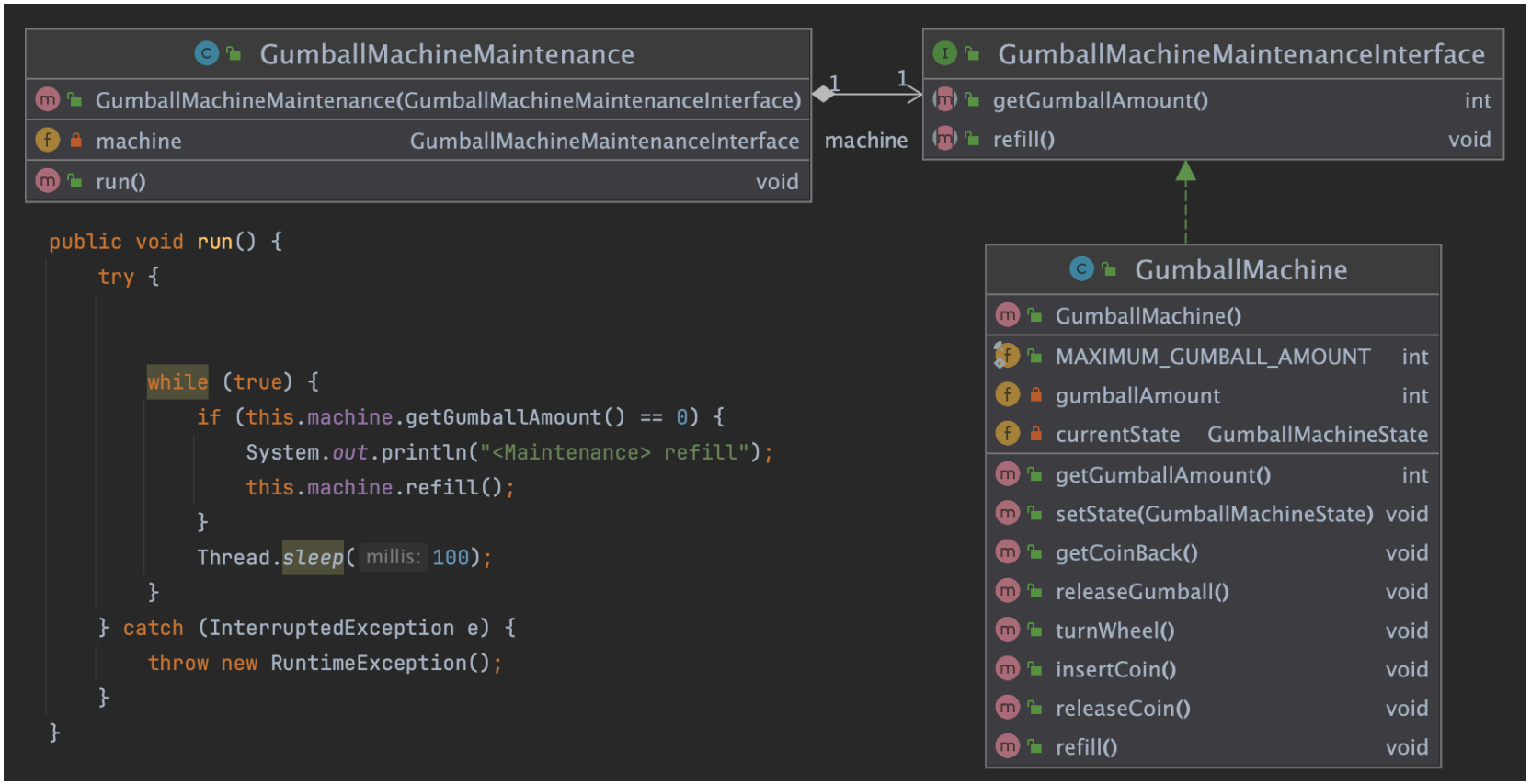
Observer, problème

- On veut **savoir quand le stock de bonbons est épuisé**, pour pouvoir remplir la machine à nouveau

IUT A Valence
Université Grenoble Alpes

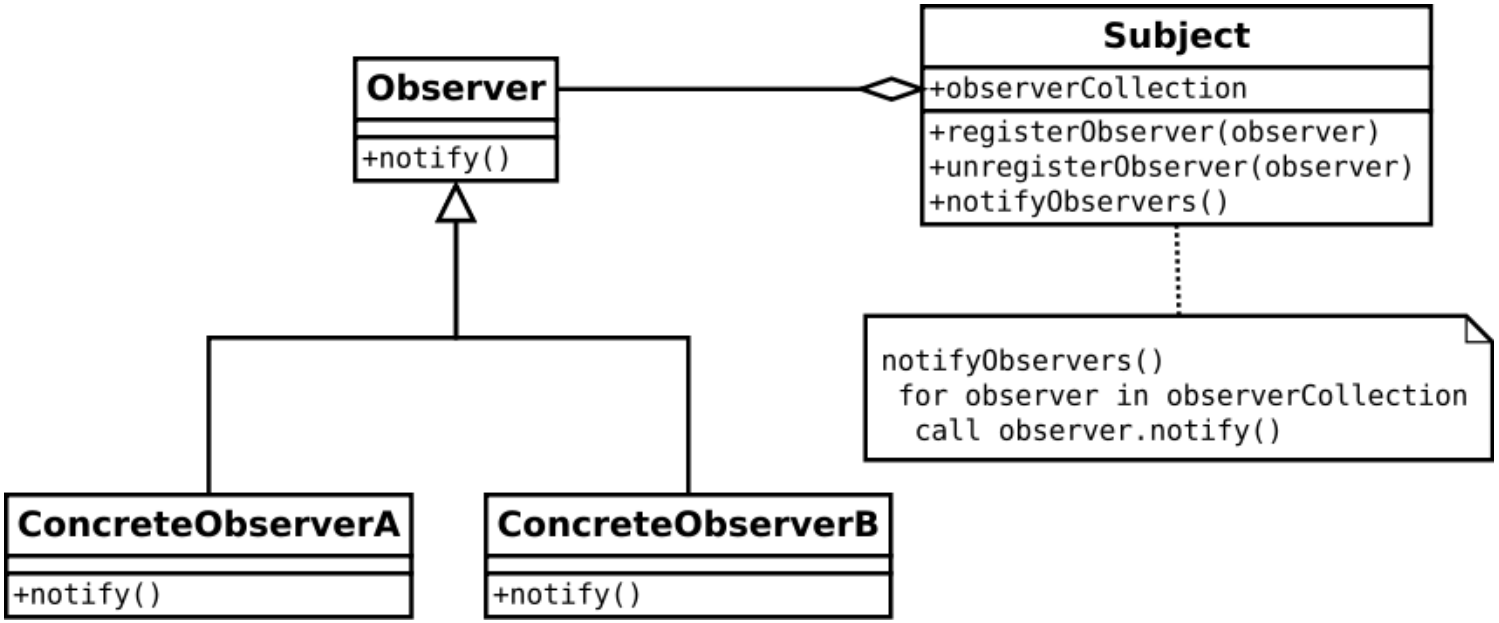
Observer, problème

- Une solution simple, mais inefficace, utilisant l'attente active



Observer, définition

- Séparer la production d'évènements de leur traitement
  - « *Don't call us, we'll call you* »



- Schéma **producteur/consommateur**
  - Le sujet produit des évènements
  - Les observateurs intéressés par les évènements peuvent s'abonner et fournissent interface pour être notifiés par le sujet (*callback*)

IUT Valence Université Grenoble Alpes

Observer, exemple

- Sujet(s), observateur(s), évènement(s) ?

IUT Valence
Université Grenoble Alpes

Observer, exemple

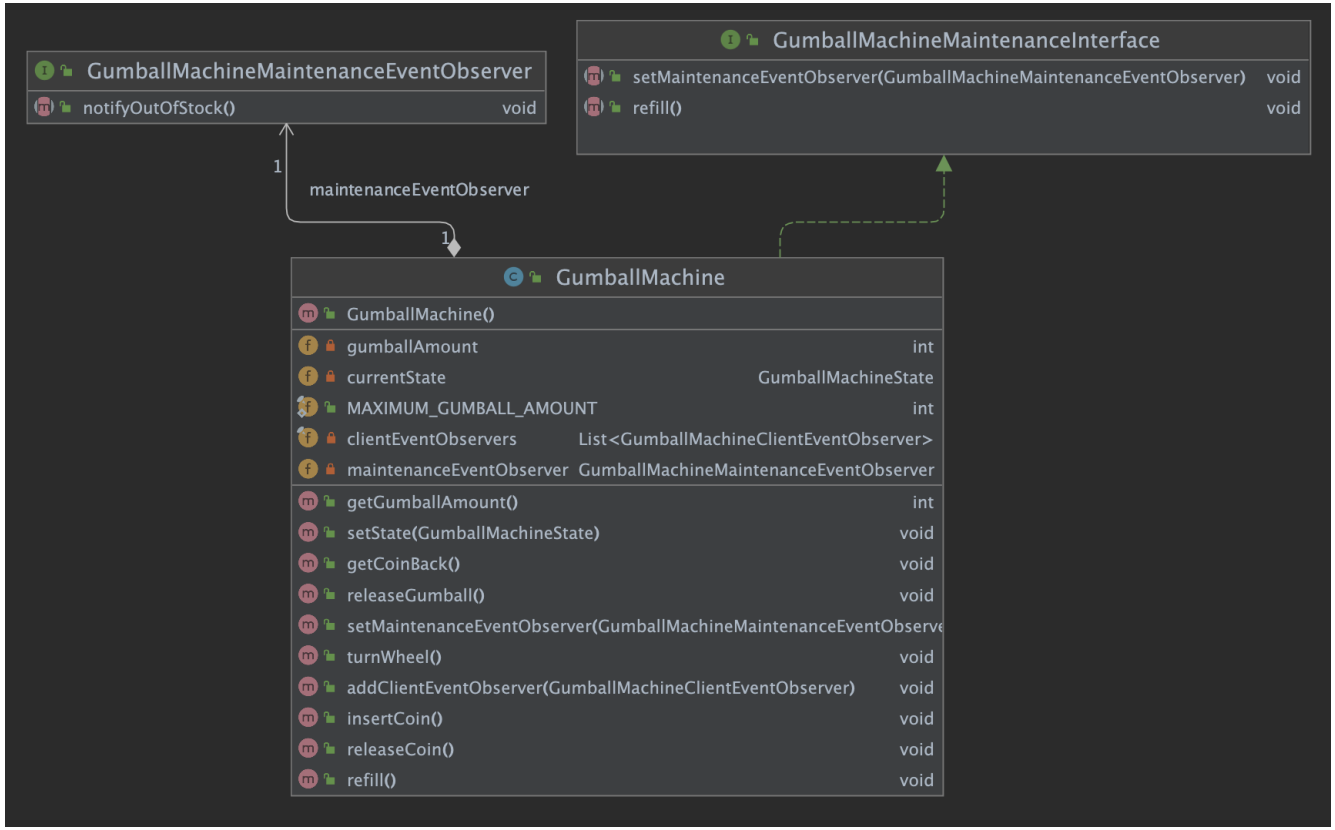
- La machine est le **sujet**
- La machine produit l'évènement « le stock est épuisé »
  - Ici l'interface est simple car l'évènement est unique et invariable

- Le service de maintenance est l'**observateur**

IUT Valence Université Grenoble Alpes

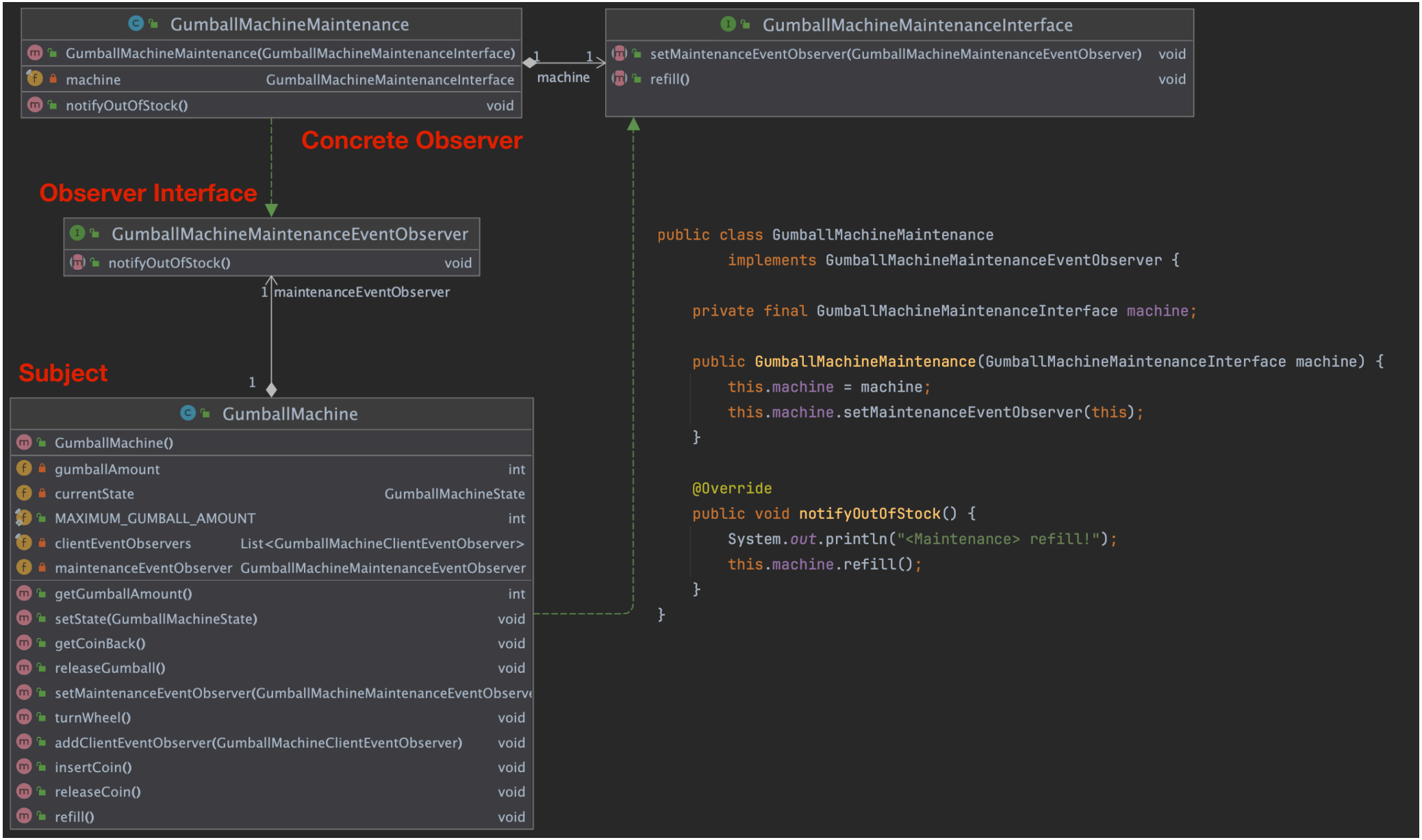
Observer, multiplicité, Abonnement / Désabonnement

- Ici on suppose que l'observateur est unique
- On fait évoluer l'interface de maintenance pour que l'observateur s'enregistre auprès du sujet



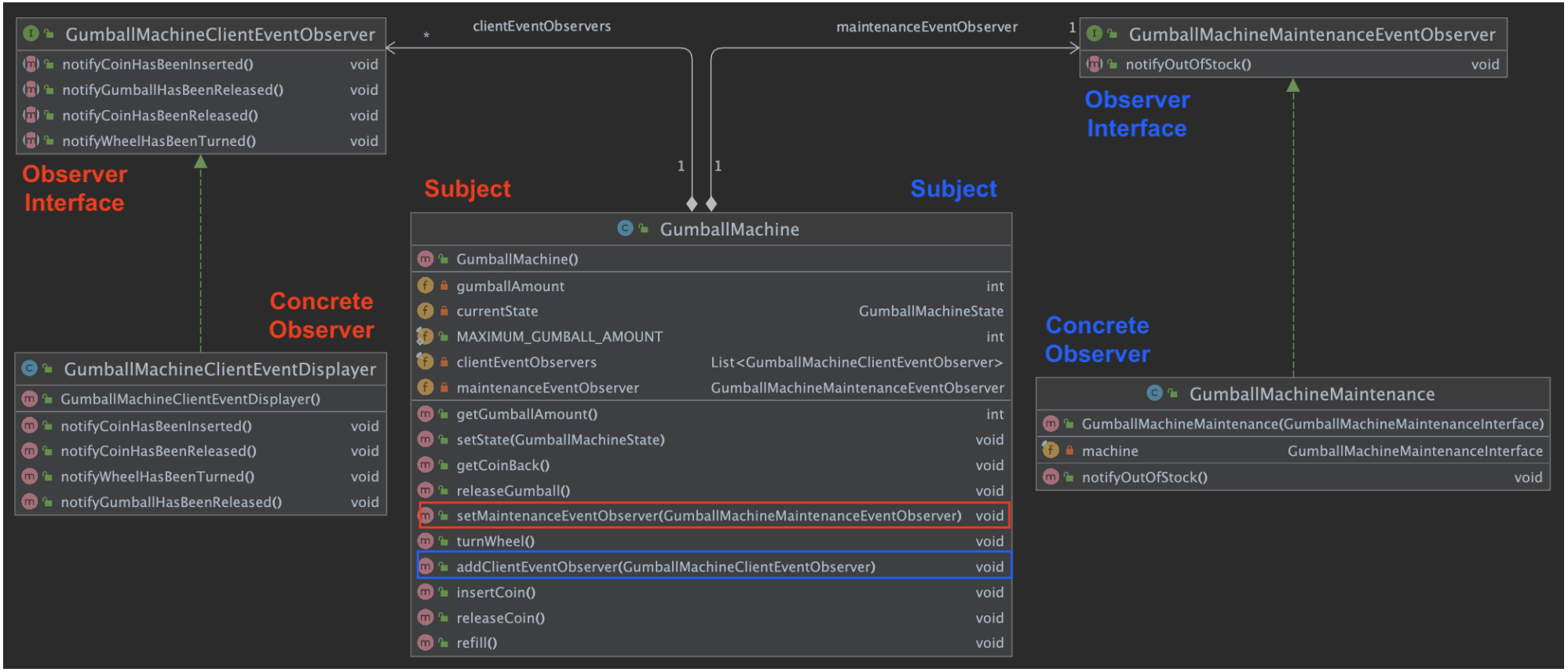
IUT Valence Université Grenoble Alpes

Observer, diagramme de classes final



Observer, exemple

- Mise en place une 2e fois du pattern pour les évènements client.
  - observateurs multiples



Fin !