Adapter

- Fetter Léo
- Laval Anthony
- Guerin Nathan
- Marcuzzi Quentin

Sommaire

Les designs patterns

Définition

Méthodes de résolution de problèmes récurrents de conception en développement informatique

- Bonnes pratiques de développement
- Documentés et connus des développeurs

Types de patterns

- Créationnels

Instanciation de la classe

- Comportementaux

Facon dont une classe communique avec les autres

- Structurels

Structure et composition de la classe

Les principes SOLIDs

Les principes SOLIDs

- SRP : Single Responsability Principle
  - Une classe doit avoir une et une seule responsabilité
- OCP : Open Closed Principle
  - Une classe doit être ouverte aux extensions mais fermé aux modifications
- LSP : Liskov Substitution Principle
  - Les sous-types doivent pouvoir être substitués à leurs types de base
- ISP : Interface Segregation Principle
  - Coder avec une interface plutôt qu'avec une implémentation
- DIP : Dependancy Inversion Principle
  - Les modules de haut niveau ne doivent pas dépendre des modules de plus bas niveau

Les principes SOLIDs respectés

- SRP : Single Responsability Principle
  - Une classe doit avoir une et une seule responsabilité
- OCP : Open Closed Principle
  - Une classe doit être ouverte aux extensions mais fermé aux modifications
- LSP : Liskov Substitution Principle
  - Les sous-types doivent pouvoir être substitués à leurs types de base
- ISP : Interface Segregation Principle
  - Coder avec une interface plutôt qu'avec une implémentation
- DIP : Dependancy Inversion Principle
  - Les modules de haut niveau ne doivent pas dépendre des modules de plus bas niveau

Le pattern Adapter

- Pattern Structurel
- Objectif simple
- Sans perturber le fonctionnement

Exemple d'utilisation

- Enoncé du besoin

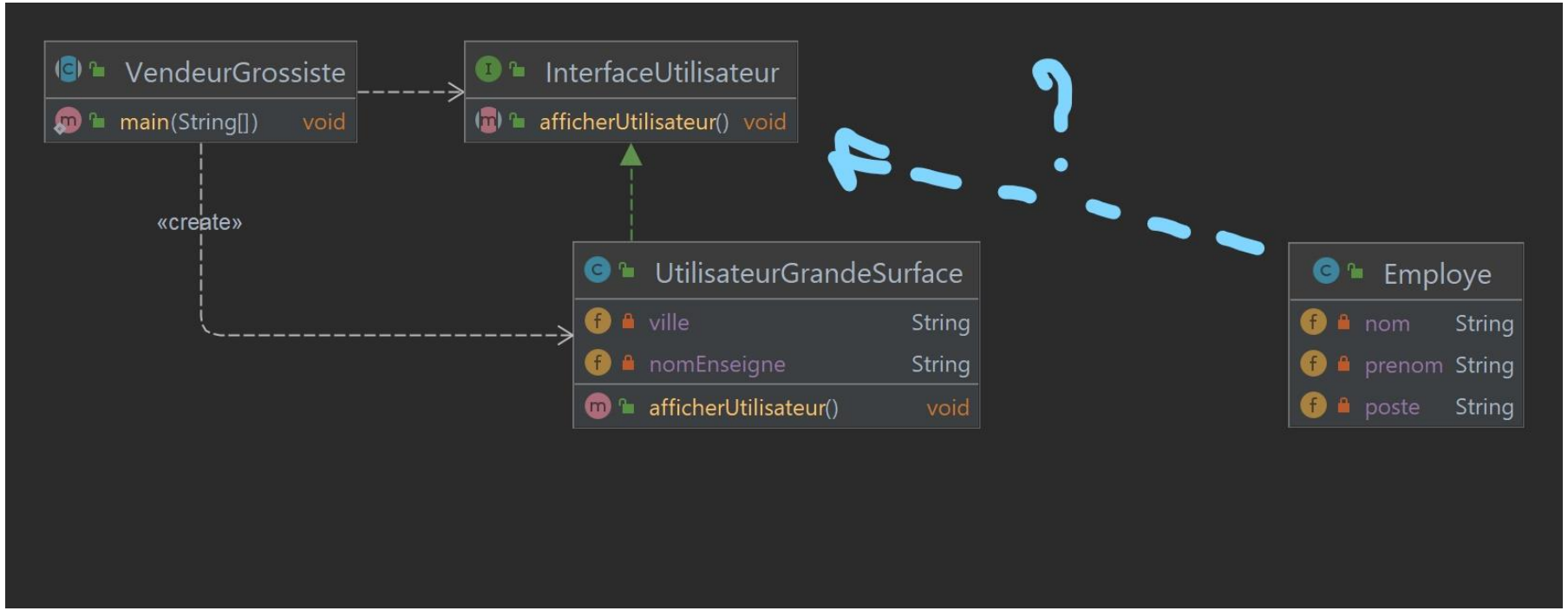
  Nous voulons faire évoluer la fonctionnalité lister l’ensemble de nos clients.

- Problématique

  Nous avons une gestion identique des clients et des salariés et nous ne voulons pas refaire entièrement la gestion des utilisateurs et souhaitons que cette évolution ne perturbe pas le fonctionnement actuel de l’application

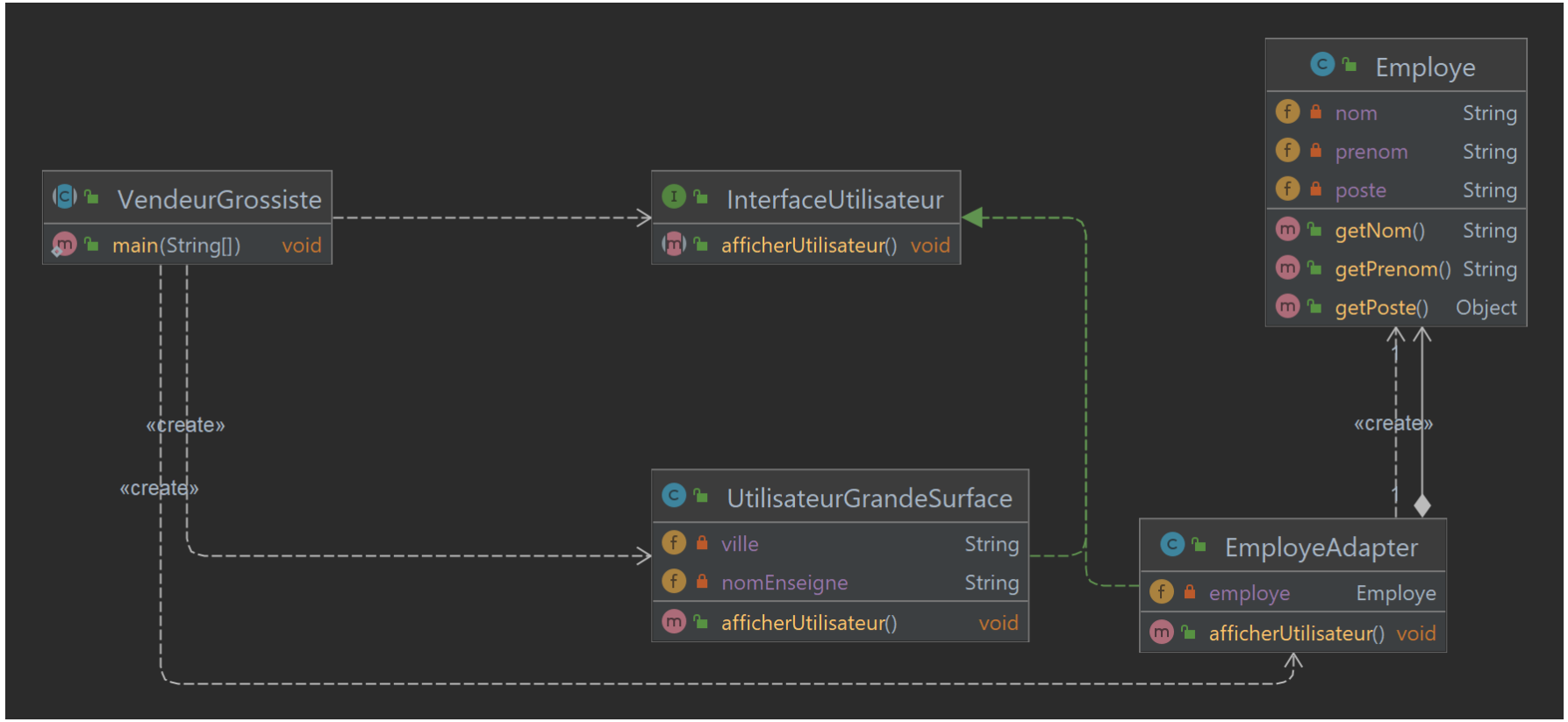
Exemple d'utilisation

- Comment passer de ça…



Exemple d'utilisation

- À ça ?



Live coding

- https://youtu.be/3IlWFyn9YDc

Relations avec d’autres patterns

Pattern Decorator

Adapter

- Modifie l'interface d'un objet existant
- Ne prend pas en charge la composition récursive

Decorator

- Améliore un objet sans toucher à son interface
- Prend en charge la composition récursive

Pattern State

Adapter

- Modifie un comportement afin de rendre compatible un objet avec un autre

State

- Permet de modifier l'état de l'objet, et ainsi changer son comportement

Pattern Strategy

Adapter

- Modifie un comportement afin de rendre compatible un objet avec un autre

Strategy

- Permet de choisir parmi plusieurs stratégies lors de l'exécution, qui va modifier le comportement de l'objet