Les patterns de création

- Pour créer des objets : *new*
  - La création ne doit pas être systématiquement libre et publique
    - Pattern Singleton (par exemple)
  - *new* permet l'instanciation de classes concrètes uniquement
    - *new* induit un couplage fort entre la classe « cliente » (c.-à-d. demandeuse de la création) et la classe de l'objet créé
      - Donc, en cas d'évolution de la classe qui définit l'objet créé, le code de la classe cliente doit être repris

    ☞ Essayer de séparer ce qui peut évoluer de ce qui ne change pas (externalisation du code de création)
  - Créer l'instance, ce n'est pas ni le « métier » ni la responsabilité de la classe cliente

    ☞ Mieux répartir les responsabilités entre les classes
- Principal objectif des patterns de création
  - Réduire le couplage entre la classe « cliente » et celle de l'objet créé

Les patterns de création

- Fabrique simple
  - La « responsabilité » (savoir-faire) de créer est externalisée
    - Le code est extrait de la classe cliente
    - Il est centralisé dans une classe « Fabrique simple »
      - Méthode de classe
      - Ou méthode d'un objet « fabrique » dédié (qui opère par délégation)
  - Avantages
    - Réduction du couplage
    - Séparation des préoccupations/métiers
    - Point de maintenance unique
  - Fabrique simple n'est pas un design pattern du GoF, plutôt une bonne pratique (GRASP)
    - Une **très bonne** pratique !!