



## Génie Logiciel : Conception

### Batailles de choix de conception

Sébastien Mosser  
INF-5153, Hiver 2019, Cours #11.2



## A quoi ressemblera l'examen ?

- Trois niveaux de compétences différents, en 3h.
- Des “petites questions” de cours (sur 20 points)
  - Sur la totalité des notions couvertes par INF-5153
  - Conception, UML, SOLID, GRASP, Patrons, Anti-patrons, Métriques
- Un exercice d'application (sur 30 points)
  - Appliquer les principes vu en cours
- Une étude de cas libre (sur 50 points)
  - Ou vous devez mettre en avant VOS PROPRES choix, et les JUSTIFIER

## Aujourd'hui ...

- Trois niveaux de compétences différents, en 3h.
- Des “petites questions” de cours (sur 20 points)
  - Sur la totalité des notions couvertes par INF-5153
  - Conception, UML, SOLID, GRASP, Patrons, Anti-patrons, Métriques

- Un exercice d'application (sur 30 points)
  - Appliquer les principes vu en cours

### La semaine prochaine

- Une étude de cas libre (sur 50 points)
  - Ou vous devez mettre en avant VOS PROPRES choix, et les JUSTIFIER

# 1 Création d'objets

## Masquage d'objets

# 3 Héritage vs Composition

# Création




## Qu'est-ce qu'un problème de création ?

- En d'autres mots ...
  - Qui contrôle la création des instances d'une classe ?
- Pourquoi est-ce un problème ?
  - Maintenance
  - Lisibilité
  - Extensibilité

## Quelles sont les solutions à disposition

- Faire des "new" directement
- Jouer sur la visibilité des constructeurs
- Patrons de conceptions :
  - Singleton
  - Factory
  - Builder
  - Prototype

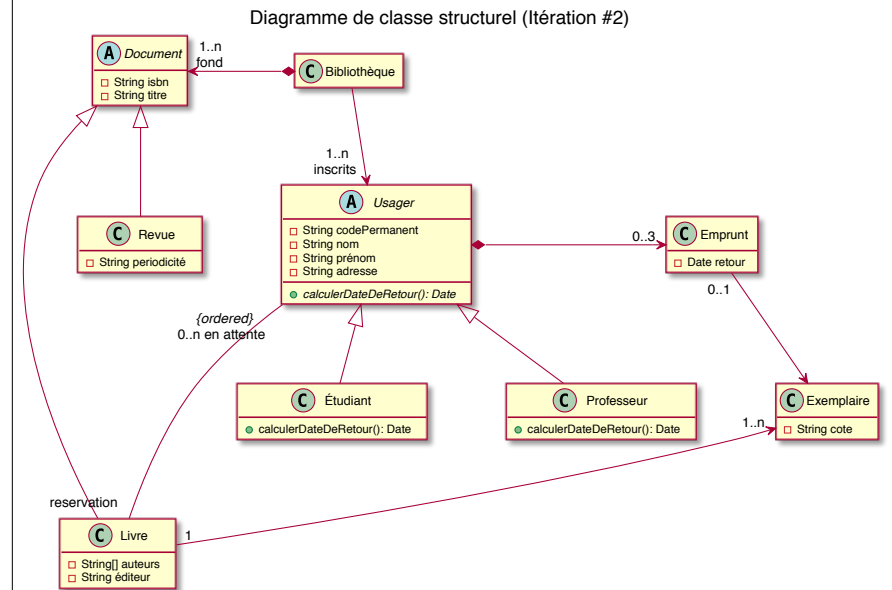
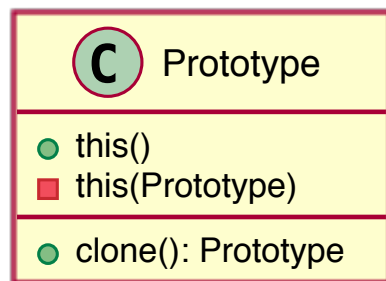
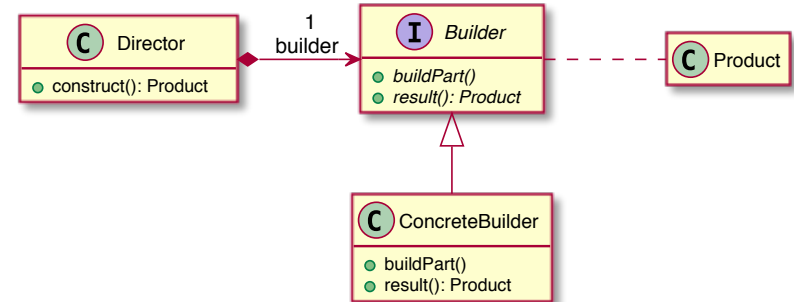
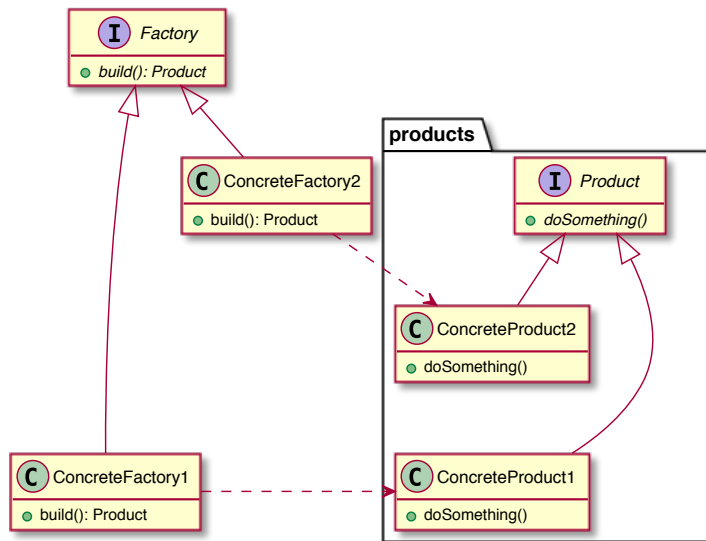
```
public class ChocolateBoiler {  
  
    private boolean empty;  
    private boolean boiled;  
  
    private static ChocolateBoiler uniqueInstance = null;  
  
    private ChocolateBoiler() {  
        empty = true;  
        boiled = false;  
    }  
  
    public static ChocolateBoiler getInstance() {  
        if (uniqueInstance == null)  
            uniqueInstance = new ChocolateBoiler();  
        return uniqueInstance;  
    }  
  
}
```

 ChocolateBoiler

☐ bool empty  
☐ bool boiled  
☐ ChocolateBoiler uniqueInstance

☒ ChocolateBoiler()  
☒ getInstance(): ChocolateBoiler

☒ fill()  
☒ boil()  
☒ drain()



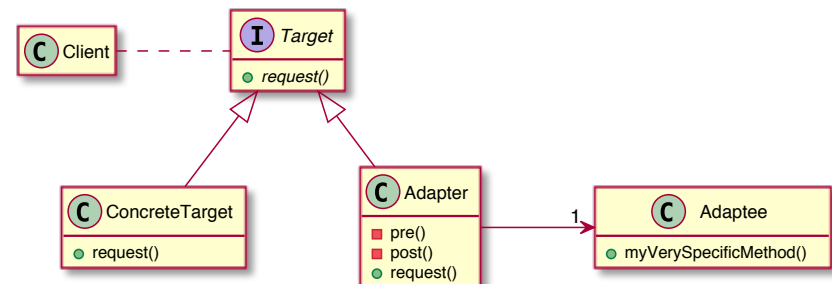
# Masquage d'objets 2

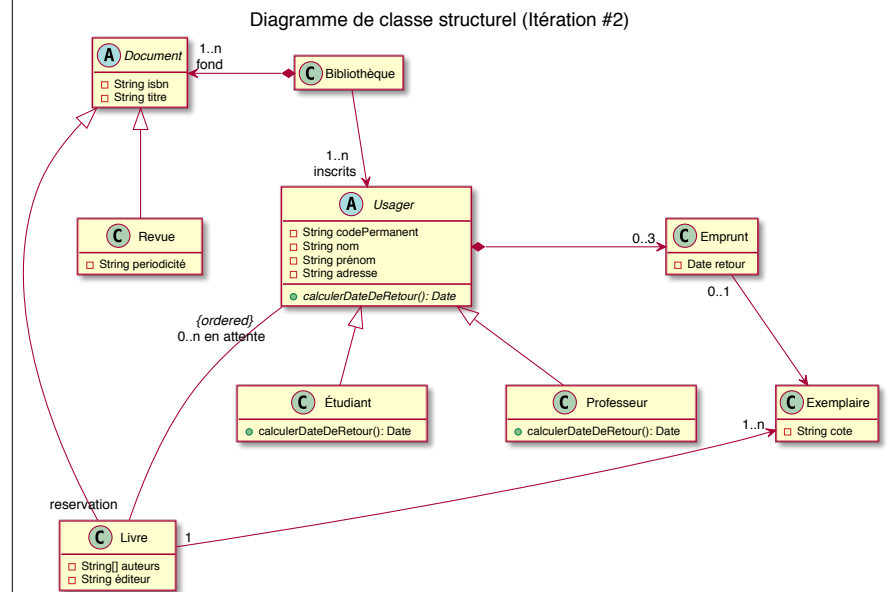
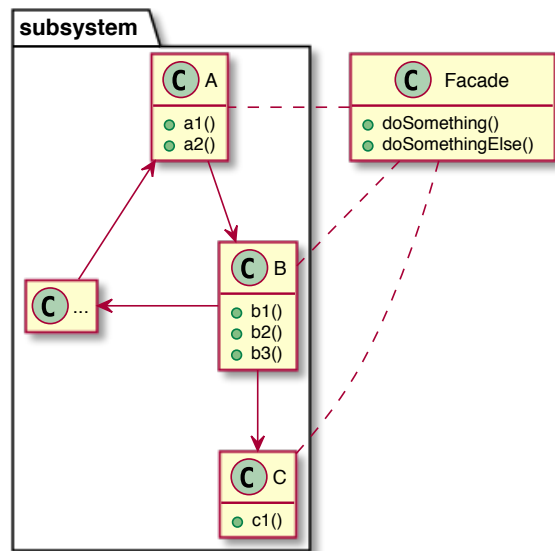
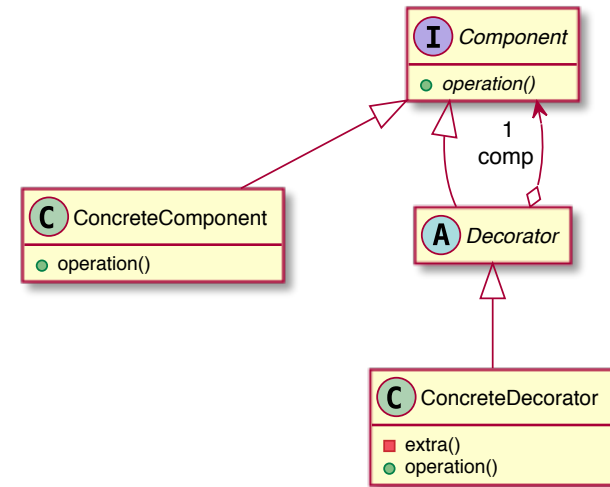
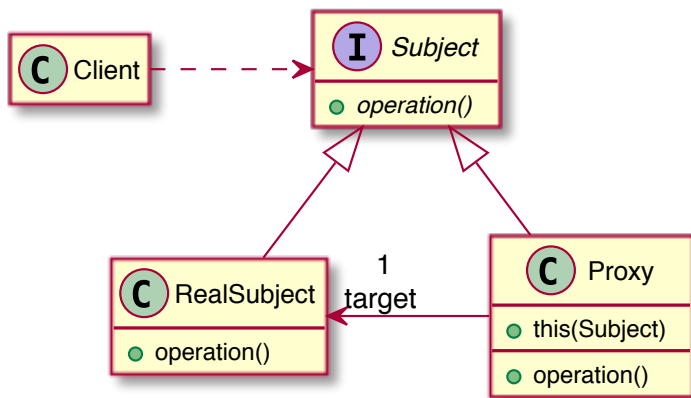
## Qu'est-ce qu'un problème de masquage ?

- En d'autres mots ...
  - Que manipulent réellement les consommateurs des objets ?
- Pourquoi est-ce un problème ?
  - Adaptation dynamique du comportement
  - Sécurité (accès aux données)
  - Maintenance, Séparation des préoccupations

## Quelles sont les solutions à disposition

- Jouer sur la visibilité des méthodes et des classes
- Utiliser des bibliothèques "boîtes noires"
- Patrons de conceptions :
  - Adapter
  - Proxy
  - Decorateur
  - Facade





# Héritage & Composition 3

## Comment choisir ?

- En d'autres mots ...
  - Qu'est-ce qui motive le choix de l'héritage sur le choix de la composition ?
- Pourquoi est-ce un problème ?
  - L'héritage doit représenter une relation "est un".
  - La composition doit représenter une relation "possède un"
  - Ça paraît simple ... et pourtant, "ça dépend"

## Quelles sont les solutions à disposition

- Du point de vue "fondateur" :
  - Pilier "Ouvert / Fermé" de SOLID
- Trois patrons caractéristiques de ce problème :
  - Stratégie
  - State
  - Template Method

