

# Null Object & Null pour les nuls

GUILLEMETEAU HIPPOLYTE, FORT GABIN,  
DESMARTIN JULIAN, NONY ANTOINE



# Histoire et définition d'un design pattern.

Premièrement qu'est ce qu'un pattern ?

Fait : les humains sont fainéants.

Dans tous les domaines de conception, il existe des problèmes récurrents spécifiques. L'idée d'un pattern est de proposer une solution générique, un modèle à suivre pour résoudre un problème de la meilleure des manières.

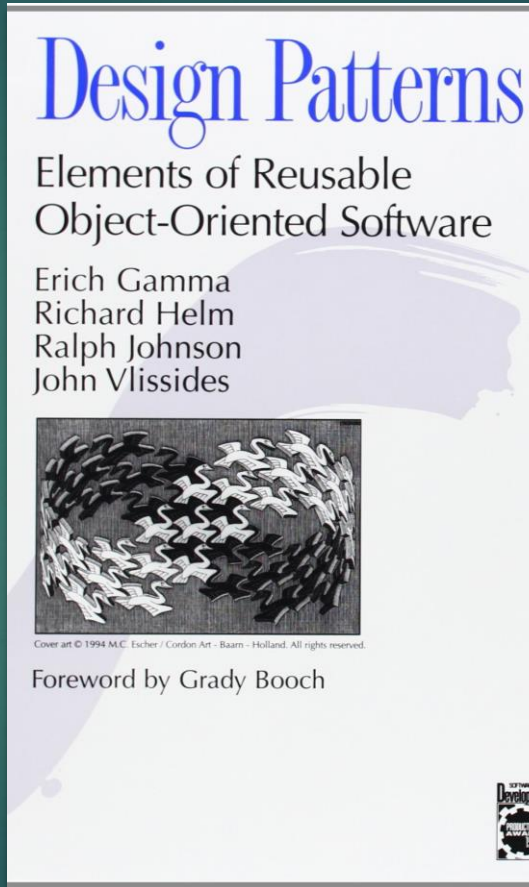
# Christopher Alexander

- ▶ Christopher Alexander a émis sa théorie des patrons de conception remet en question l'idée qu'il puisse y avoir une création ou une invention originale et individuelle dans le domaine de la conception.
- ▶ En résumé, pour lui, toute conception s'inspire de ce qui est déjà connu et suit un processus d'amélioration constante.



# Gang of Four et Design Patterns

► Les travaux d'Alexander ont inspiré quatre informaticiens répondant aux noms de Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides, surnommés le "Gang of Four" (d'où le terme GoF pour ces pattern). Ils proposent eux aussi des solutions élégantes, et toujours différentes pour résoudre différents problèmes récurrents rencontrés par les architectes logiciels.



# Quelques précisions...

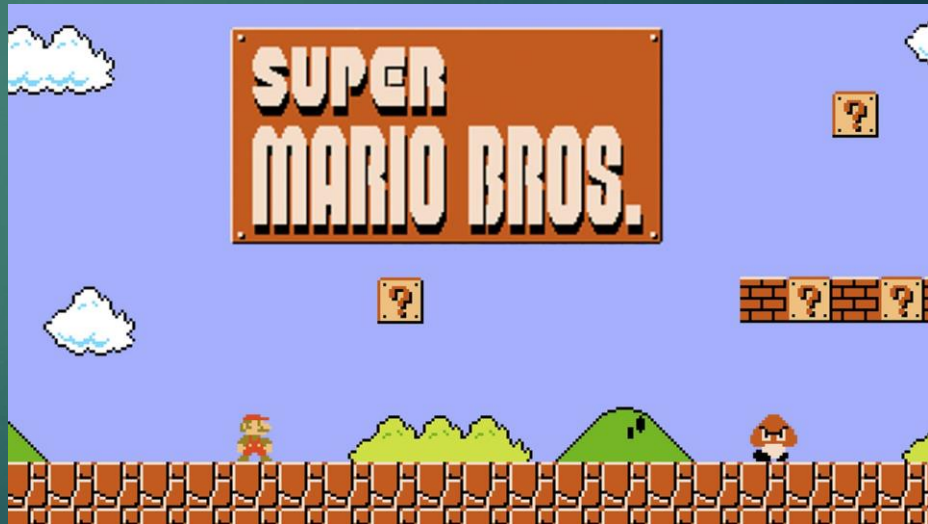
► 3 types de pattern :  
Création, structure,  
comportement

► Le pattern null object ne fait pas parti du GoF, car il n'est pas présent dans le livre "Design Patterns", néanmoins, il est mentionné dans le livre "Refactoring" de Martin Fowler, Architecte en informatique. Ainsi que dans "Insert Null Object Refactoring" de Joshua Kerievsky



Exemple :

Déplacement  
d'un  
personnage



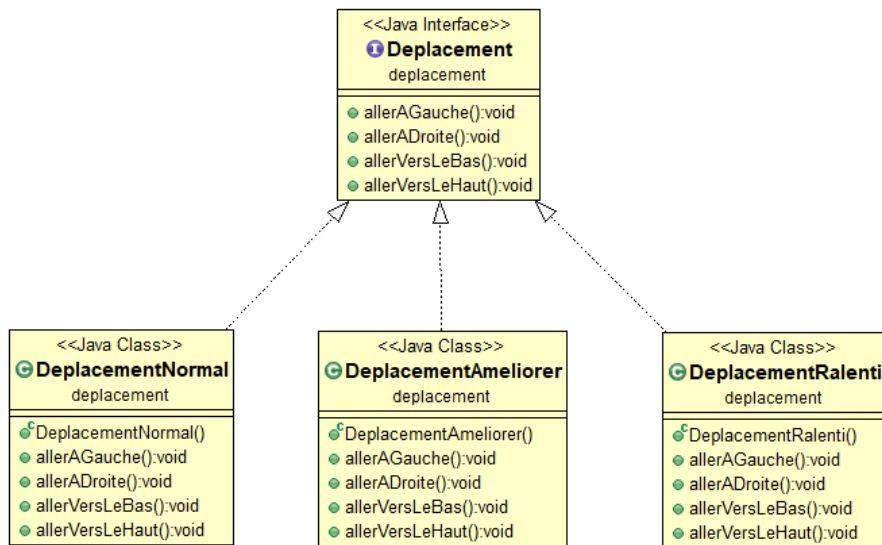


# Exemple :



```
public interface Deplacement {  
  
    void allerAGauche();  
    void allerADroite();  
    void allerVersLeBas();  
    void allerVersLeHaut();  
  
}
```

# Exemple : problème de cet exemple

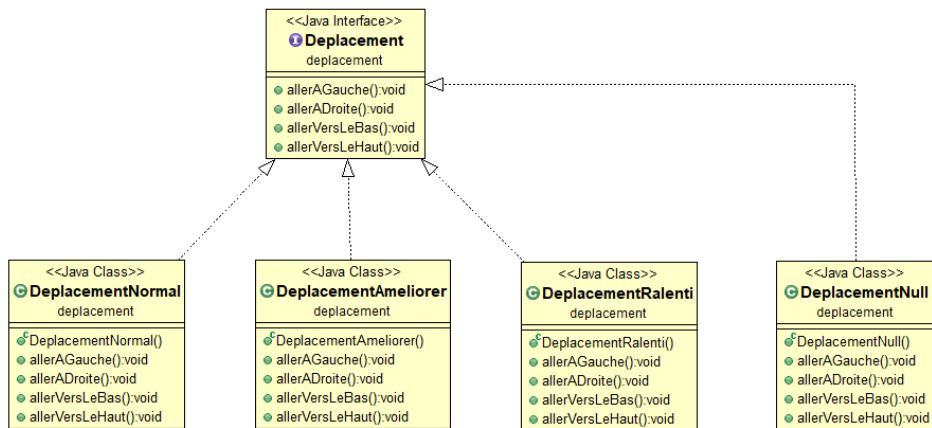




# Exemple : une solution



```
public class DeplacementNull implements Deplacement {  
  
    @Override  
    public void allerAGauche() {  
  
    }  
  
    @Override  
    public void allerADroite() {  
  
    }  
  
    @Override  
    public void allerVersLeBas() {  
  
    }  
  
    @Override  
    public void allerVersLeHaut() {  
  
    }  
  
}
```



# Exemple : Null Object

# Le Patron de conception NULL OBJET

## Problématique :

- Savoir comment simplifier la gestion des références nulles
- Comment l'absence de l'objet peut être traitée de façon transparente ?
- Éviter le risque de `NullPointerException`  
(Éviter les problèmes induits par le test `(object==null)` source de nombreux bugs)

## Intention :

Ce Pattern permet d'encapsuler l'absence d'un objet en fournissant un substitut qui implante le comportement de ne rien faire.

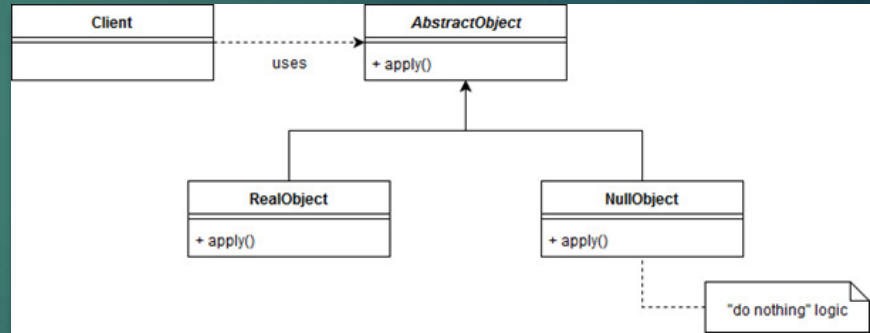


# SOLUTION et rôles des classe participants

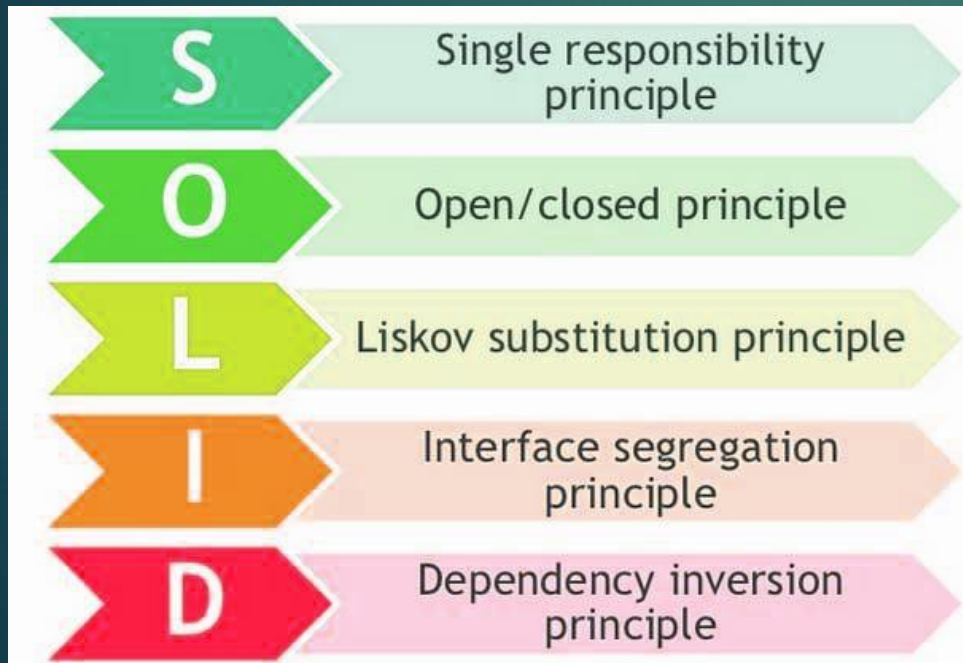
- ▶ Le Client requiert une instance d'un objet abstrait.
- ▶ L'objet Abstrait "AbstractObject" définit le contrat Client attend

L'objet Réel "RealObject" implémente l'objet Abstrait et fournit un comportement réel

L'objet Null "NullObject" implémente l'objet abstrait et fournit un comportement neutre



# PRINCIPE SOLID



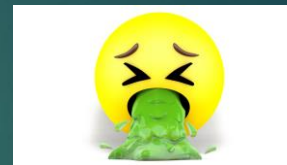
Notre pattern permet à notre code de respecter :

- ↪ Le principe OCP ( suppression des multiple condition if else pour les conditions null )
- ↪ Le principe DIP ( mise en place de ce pattern possible seulement à partir d'une class abstraite )

# Avantages et Inconvénients



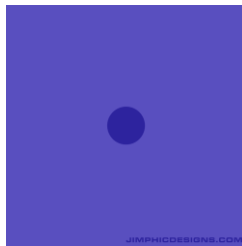
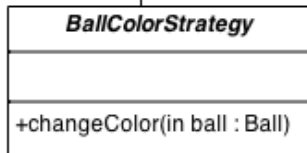
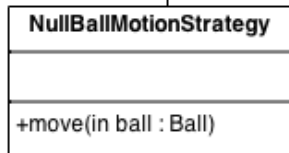
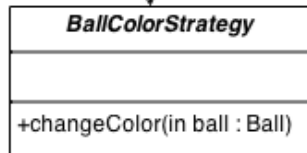
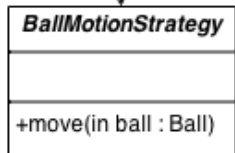
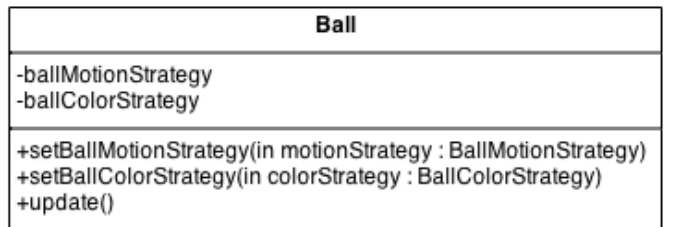
- Suppression de nombreux check pour null qui rendait le code long et mauvais.
- Améliore la lisibilité du code et respect les principes SOLID
- Faciliter à mettre en lien avec les pattern state, strategy et singleton.



- Ce pattern doit être utilisé avec précaution car il peut faire passer des erreurs pour un déroulement normal du programme.
- Le prix à payer pour se débarrasser des conditionnels crée encore une nouvelle classe.

# Les Patterns

## STATE et STRATEGY



► Ce patron peut être considéré comme un cas spécial du patron état et du patron stratégie.

► Permet d'en les deux cas d'initialiser les premières actions des objets créer.

► Exemple :

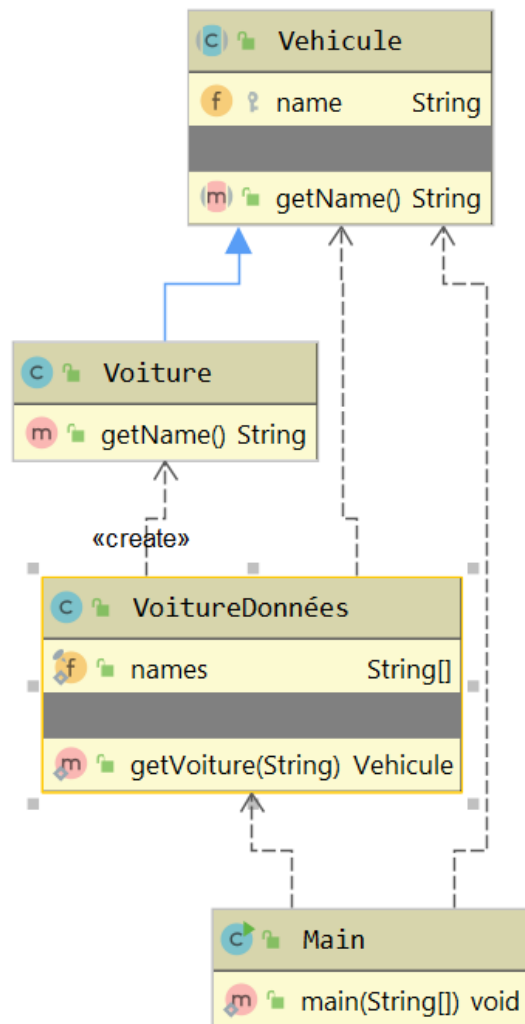
► Ici Le concept d'une balle pouvant changer couleur et pouvant bouger. Le concept le plus simple pouvant les initialiser et de ne rien faire tout l'emploi du pattern null object

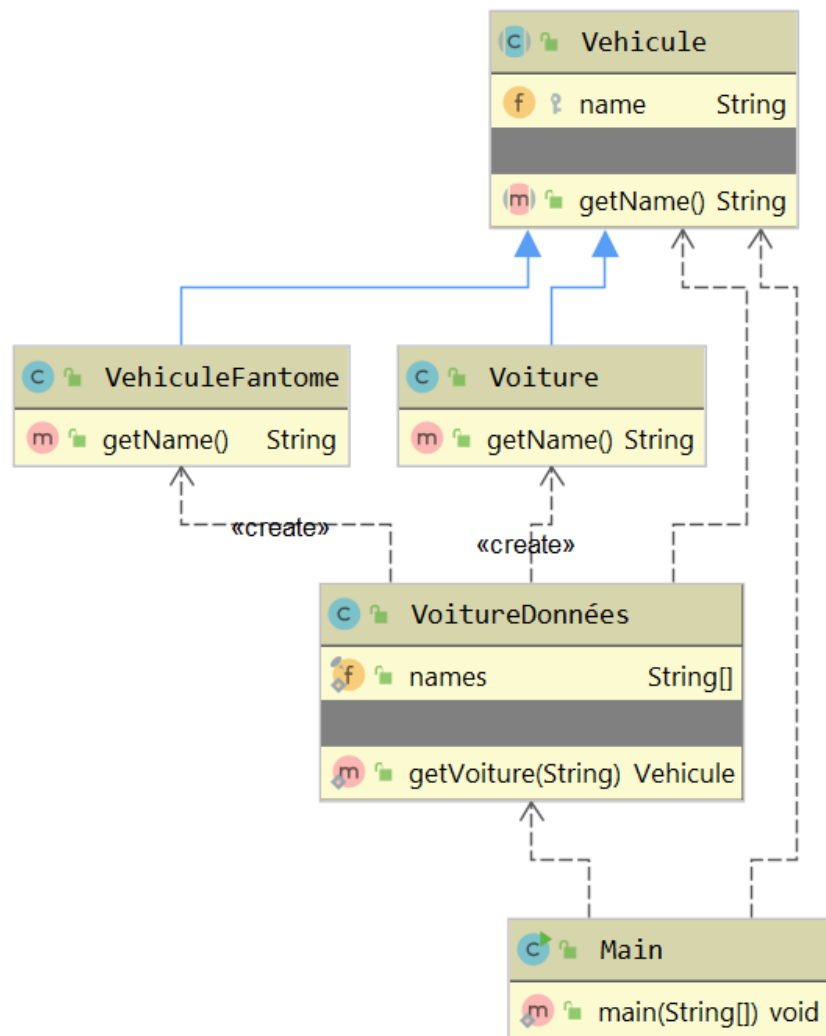




# Démonstration

LIVE-CODING





# Sitographie / Bibliographie

<https://www.youtube.com/watch?v=rQ7BzfRz7OY>

<https://www.geeksforgeeks.org/null-object-design-pattern/>

[http://mborne.github.io/cours-patron-conception/annexe/design\\_pattern/behavior/NullObject.html](http://mborne.github.io/cours-patron-conception/annexe/design_pattern/behavior/NullObject.html)

<https://refactoring.guru/design-patterns>

<https://www.codingame.com/playgrounds/491/avoiding-null-anti-patterns/the-null-object-pattern>

<https://foad.ensicaen.fr/pluginfile.php/1214/course/section/633/DP-03.pdf?time=1566973225228>


[https://en.wikipedia.org/wiki/Null\\_object\\_pattern](https://en.wikipedia.org/wiki/Null_object_pattern)

[https://sourcemaking.com/design\\_patterns/null\\_object](https://sourcemaking.com/design_patterns/null_object)

# QCM

► [cutt.ly/qcm2019](https://cutt.ly/qcm2019)





Null est il un pattern du Gang Of Four ? \*

- ☐ Oui
- ☐ Non, il n'est pas mentionné dans le livre "Design Patterns"


Question 1.

A quoi peut on comparer le Null Object Pattern ? \*

- ☐ A quelque chose de souple
- ☐ A un fantôme
- ☐ A quelque chose qui change de forme

Question 2.





Quel est le danger de l'implémentation du pattern Null Object ? \*


- ☐ Cela peut créer un null pointer exception
- ☐ Le programme peut se dérouler normalement alors qu'il y a une erreur
- ☐ Le PC peut surchauffer

Question 3.

Quels sont les principes solides respectés par le pattern Null Object ? \*

- ☐ OCP(Open closed principle) et DIP(Dependency inversion principle)
- ☐ LSP ( Liskov substitution principle ) et ISP ( Interface segregation principle )
- ☐ Tous
- ☐ La réponse D

Question 4.



Pourquoi utiliser les null object au lieu d'utiliser la référence null ? \*

- ☐ Les null objects ont un comportement prévisible et n'auront pas d'effets secondaire puisqu'ils ne font rien
- ☐ Cela permet d'éviter l'erreur NullPointerException
- ☐ Les null objects permettent une meilleure lisibilité du code

## Question 5.

Merci !