Null Pour Les Nuls

Ferdina ADRACHI - Evan ARRANGA - Eliott BAYART - Mostapha AYEB

Sommaire

La problématique du null en Java

Null?
Qu’est-ce que c’est ?

```
String exemple = null;
```

Ne référence aucun objet

Absence de valeur

Null Pointer Exception

```java
public class Exemple {

    public static void main(String[] args) {
        String nom = null;
        System.out.println(nom.length());
    }

}
```

```
Exception in thread "main" java.lang.NullPointerException
```

Comment éviter les NPE ?

Toujours initialiser vos variables

```
String nom;
```

```
String nom = "";
```

Vérifier si la variable n'est pas null

```
if (nom != null) {
    System.out.println(nom.length());
}
```

Gérer le null avec le Java moderne

```
Exception in thread "main" java.lang.NullPointerException
```

Optional → “Je remplace le null par une présence contrôlée”

Objects.requireNonNull() → “Je valide ce qui est essentiel”

record → “Je crée des objets sûrs et immuables”

L'objet optional

```java
Customer customer = database.findCustomer("Mostapha");
if (customer == null) {
    plan = BillingPlan.basic();
} else {
    plan = customer.getPlan();
}
```

Ici si le client n'existe pas : on utilise NullCustomer à la place de Null

```java
Customer customer = Optional.ofNullable(database.findCustomer("Mostapha"))
    .orElse(new NullCustomer());
Plan plan = customer.getPlan();
```

Objects.requireNonNull et les records

```java
public class Person {
    private final String name;
    public Person(String name) {
        this.name = Objects.requireNonNull(name, "Le nom ne peut pas être null");
    }
}
```

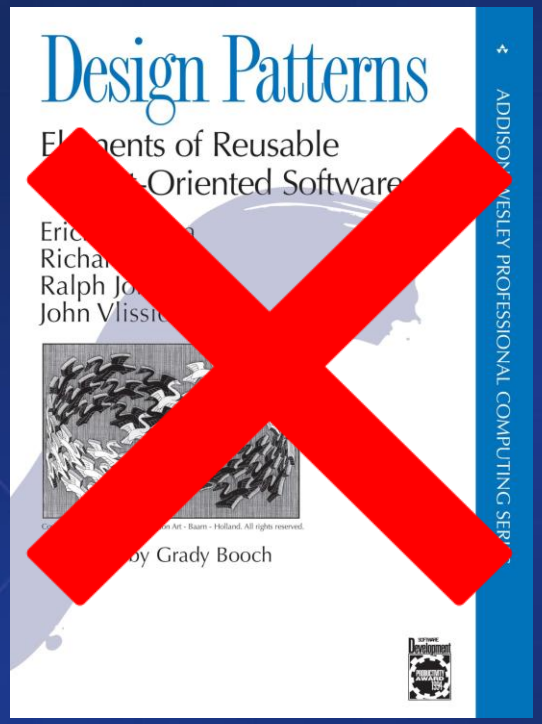
Le constructeur compact permet d'ajouter toute les vérifications lors de la création de l'objet

```java
public record Person(String name, int age) {
    public Person {
        Objects.requireNonNull(name, "Le nom ne peut pas être null");
        if (age < 0) throw new IllegalArgumentException("Âge invalide");
    }
}
```

Le Pattern Null Object

Le Pattern Null

Appartient aux patterns comportementaux, mais est un pattern non officiel, on dit plutôt que c'est une **bonne pratique dérivée**



Le **pattern Null Object** est un pattern qui fournit un objet avec un comportement neutre (par défaut) à la place de null ou undefined.

Utilisation du Pattern Null

Problème

Fonction() : null

- Vérifications de null
- Duplication de code

Solution

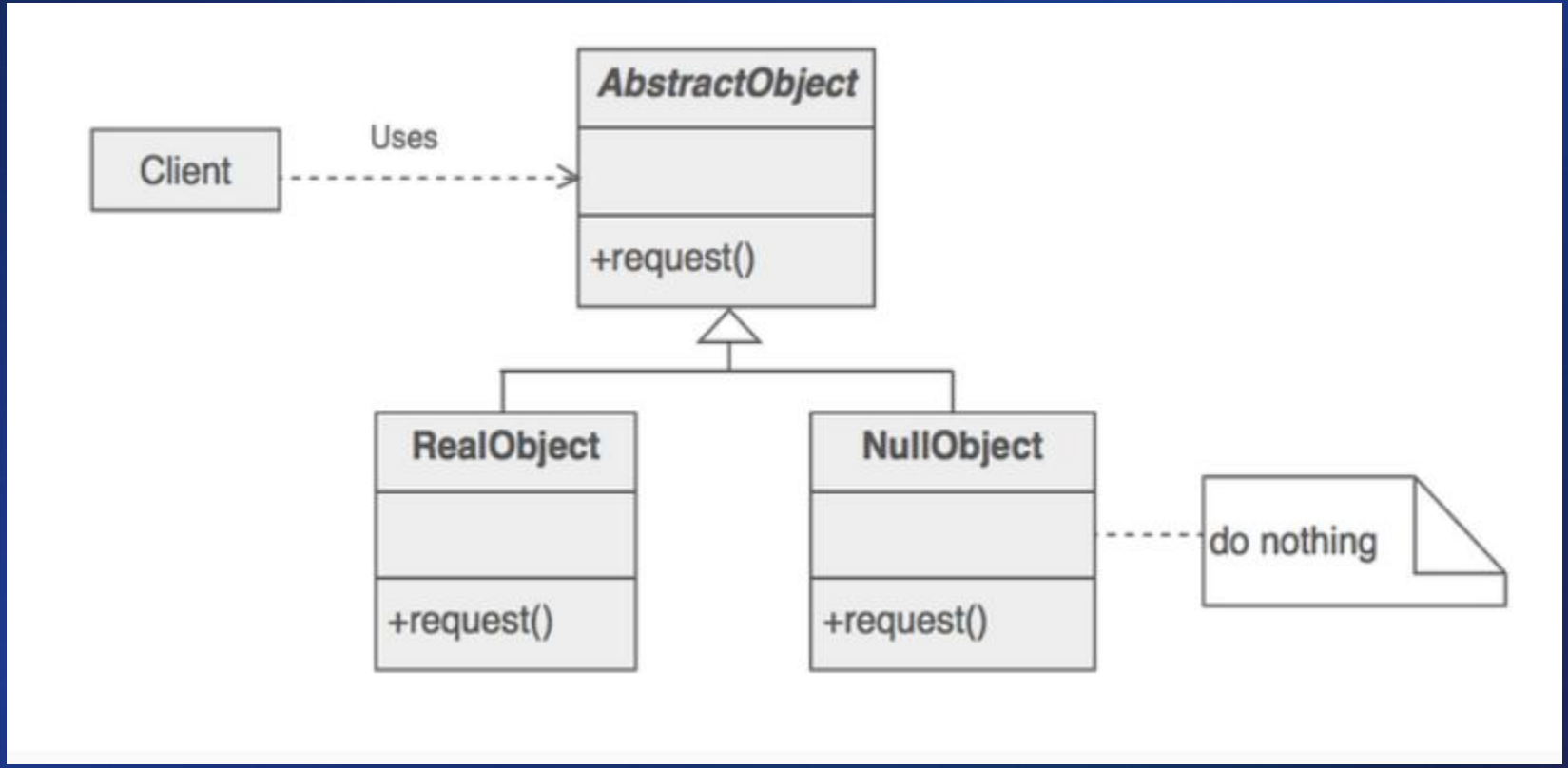
Fonction() : Object->null

- Aucune vérification
- Aucune vérification

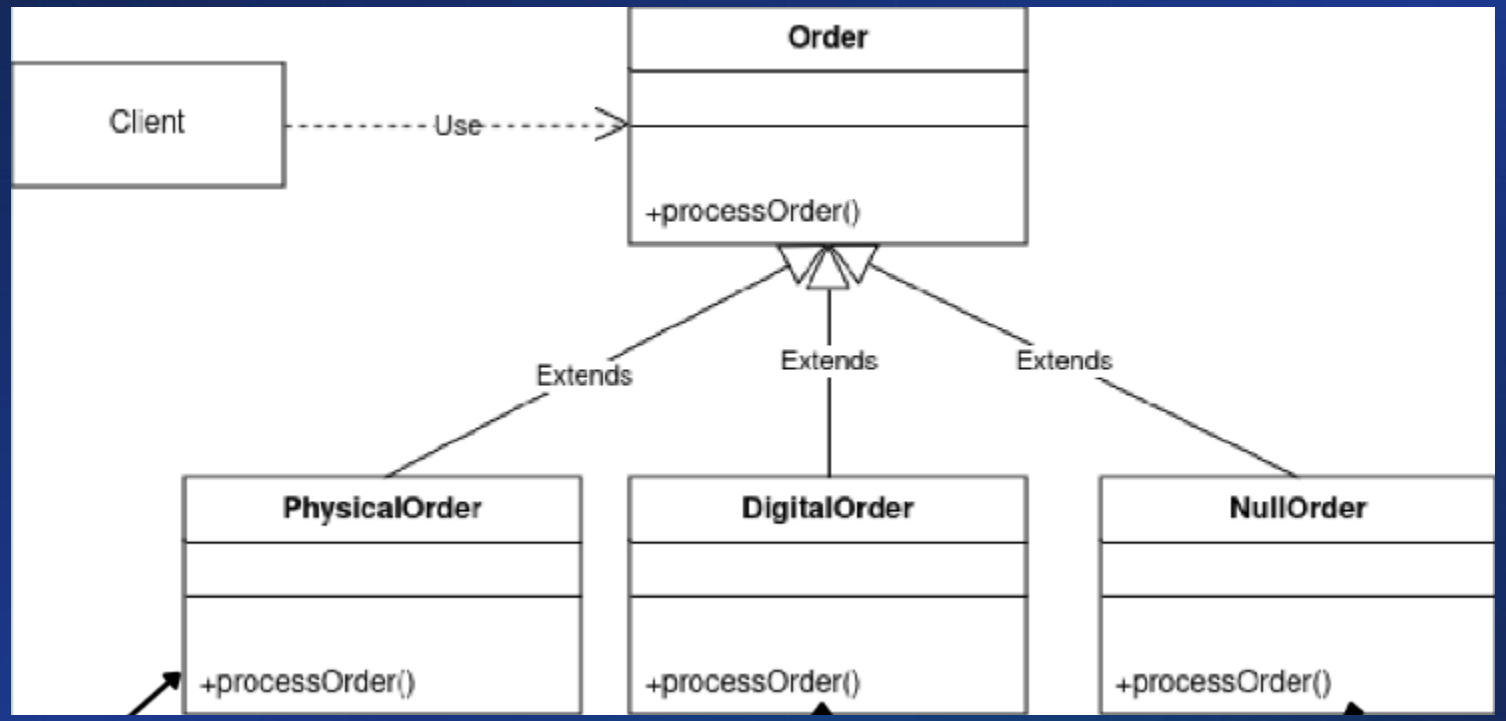
Principes du Pattern Null

- Substitution par un objet neutre
- Prévention des exceptions de nullité
- Suppression des contrôles de nullité

Diagramme de classe



Exemple d'utilisation

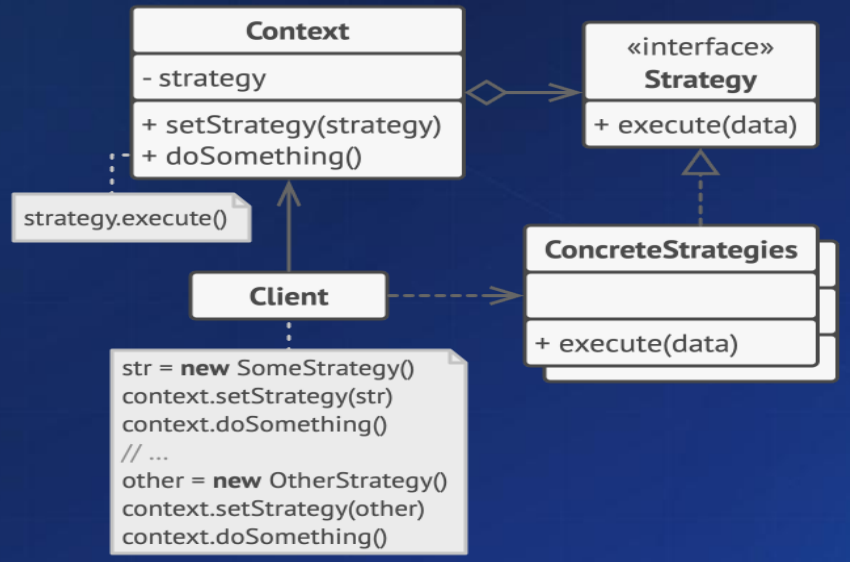


Principaux avantages et inconvénients du pattern Null

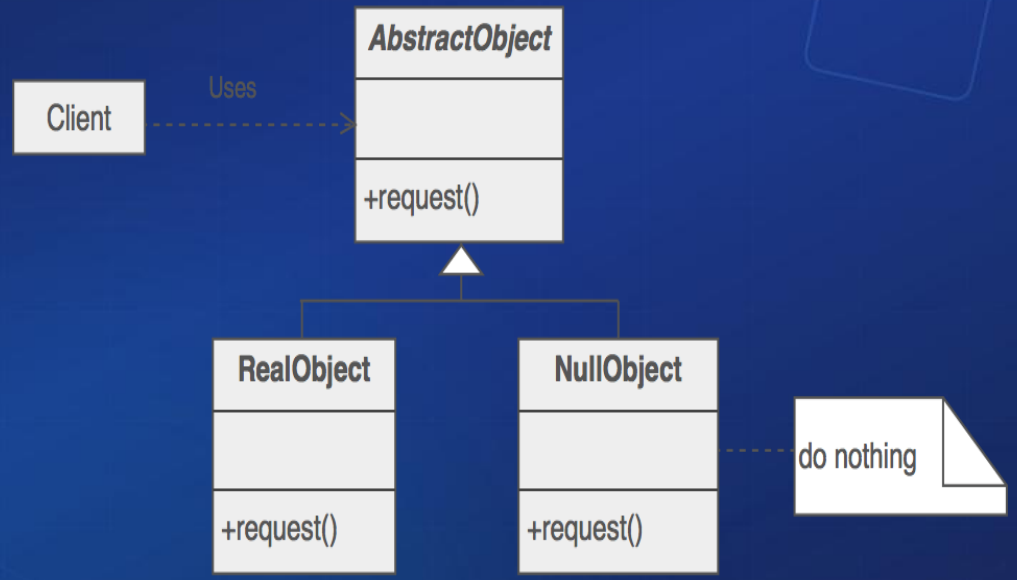
Lisibilité	Utilisation limitée
Maintenabilité	Complexité structurelle
Réduction des erreurs	Masquage des erreurs

Relation avec le pattern Strategy

Pattern Strategy

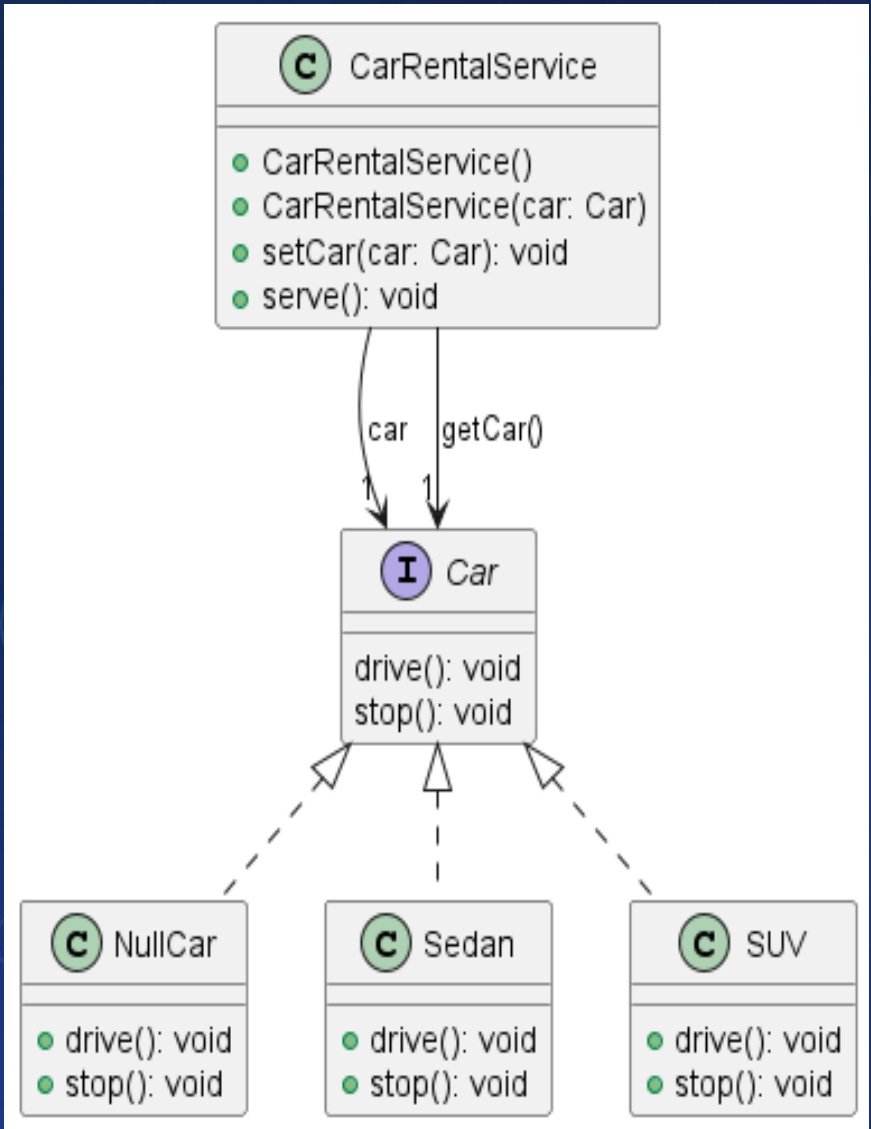


Pattern Null Object



Strategy **Concrete Strategy** **Null Strategy** **Context**	**AbstractObject** **RealObject** **NullObject** **Client**

Le cas de la Location de Voiture

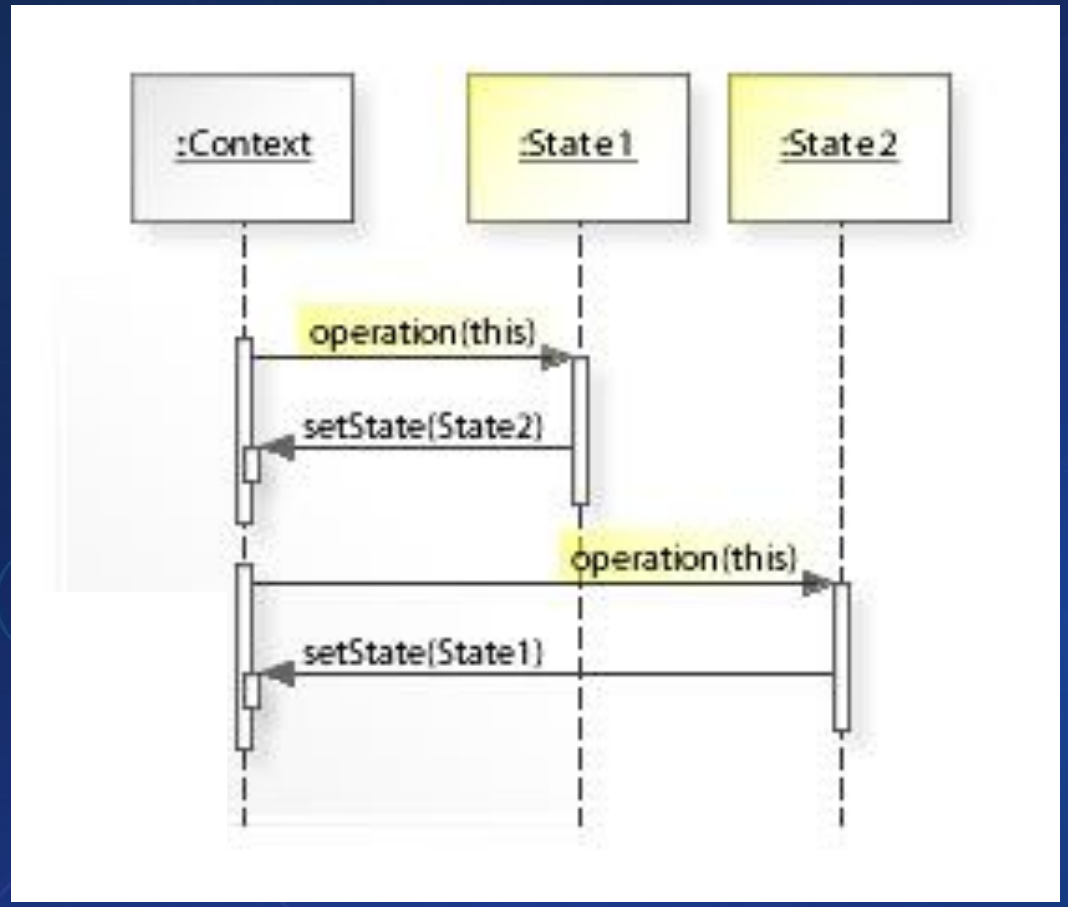


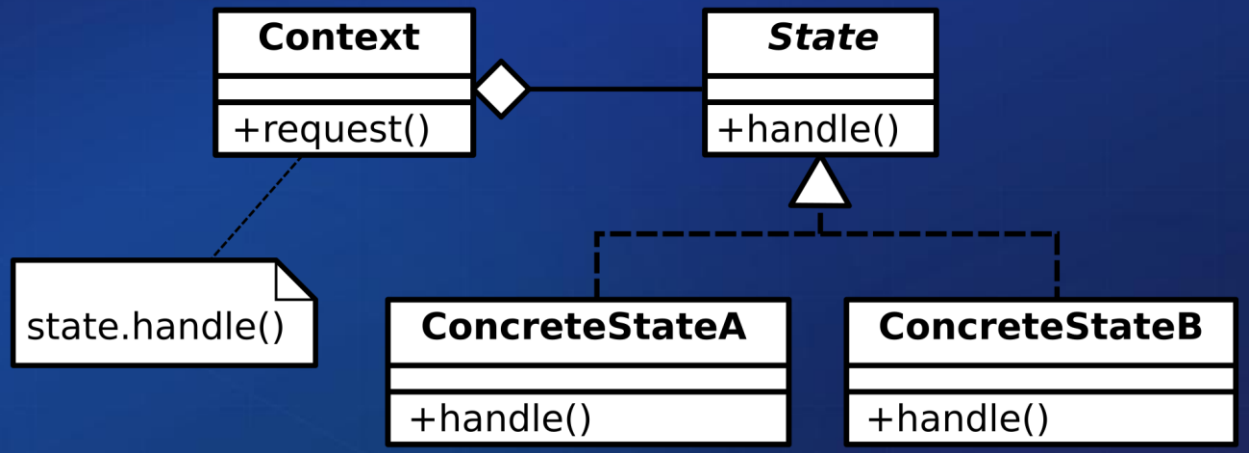
Composant dans le Pattern Null Object	Composant dans le Pattern Strategy	Élément dans l'Exemple	Description
AbstractObject (Interface)	**Strategy** (Interface)	Car (avec méthodes drive() et stop())	Définit le contrat
RealObject	**Concrete Strategies**	SUV, Sedan	Implémentent le comportement réel
NullObject	**Null Strategy** (Concrete Strategy)	NullCar	Fournit un comportement neutre
Client	**Context**	CarRentalService	Appelle simplement car.drive() sans vérification de nullité

```java
package model;

public class CarRentalService {
    private Car car;

    public CarRentalService() {
        this.car = new NullCar();
    }

    public CarRentalService(Car car) {
        this.car = car;
    }

    public void setCar(Car car) {
        this.car = car;
    }

    public Car getCar() {
        return this.car;
    }

    public void serve() {
        System.out.println("CarRentalService : début de la location");
        car.drive();
        car.stop();
        System.out.println("CarRentalService : location terminée");
    }
}
```

```java
package model;

public class NullCar implements Car {
    @Override
    public void drive() {
    }

    @Override
    public void stop() {
    }
}
```

Relation avec le pattern State


:Context
:State1
:State2
operation(this)
setState(State2)
operation(this)
setState(State1)


Context
+request()
State
+handle()
state.handle()
ConcreteStateA
+handle()
ConcreteStateB
+handle()

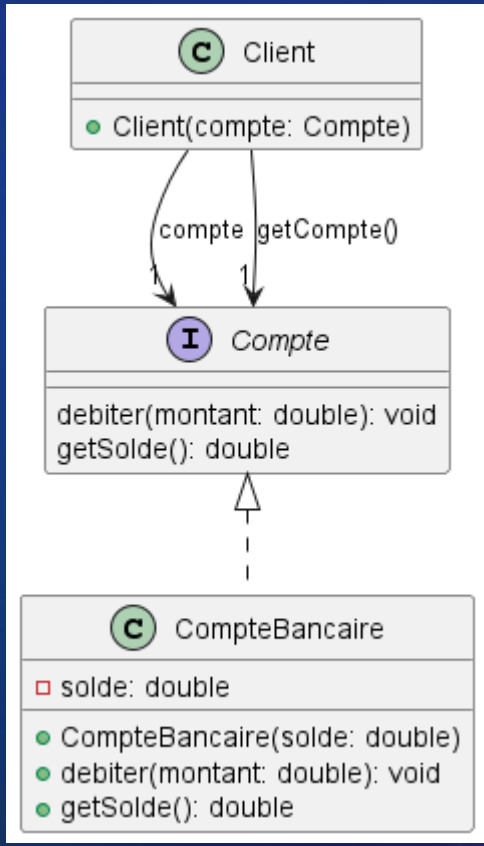
```java
interface Etat {
        void action();
}

class EtatActif implements Etat {
        public void action() { System.out.println("Actif !"); }
}

class EtatInactif implements Etat {
        public void action() { System.out.println("Inactif."); }
}

class Contexte {
        private Etat etat = new EtatInactif();

        public void setEtat(Etat e) { this.etat = e; }
        public void action() { etat.action(); }
}
```

Rappel du problème avant le Live Coding

```java
package com.gestionclientelesanspattern;

public class Main {

    public static void main(String[] args) {

        Client client1 = new Client(new CompteBancaire(1000.0));

        if (client1.getCompte() != null) {
            System.out.println("Client 1 - Solde initial : " + client1.getCompte().getSolde() + " €");
            client1.getCompte().debiter(200.0);
            System.out.println("Client 1 - Solde après débit : " + client1.getCompte().getSolde() + " €");
        } else {
            System.out.println("Client 1 - Pas de compte");
        }

        System.out.println();

        Client client2 = new Client(null);

        try {
            System.out.println("Client 2 - Tentative d'accès sans vérification...");
            System.out.println("Client 2 - Solde : " + client2.getCompte().getSolde() + " €");
        } catch (NullPointerException e) {
            System.out.println("ERREUR : NullPointerException ! Le compte est null.");
        }
    }
}
```

Console

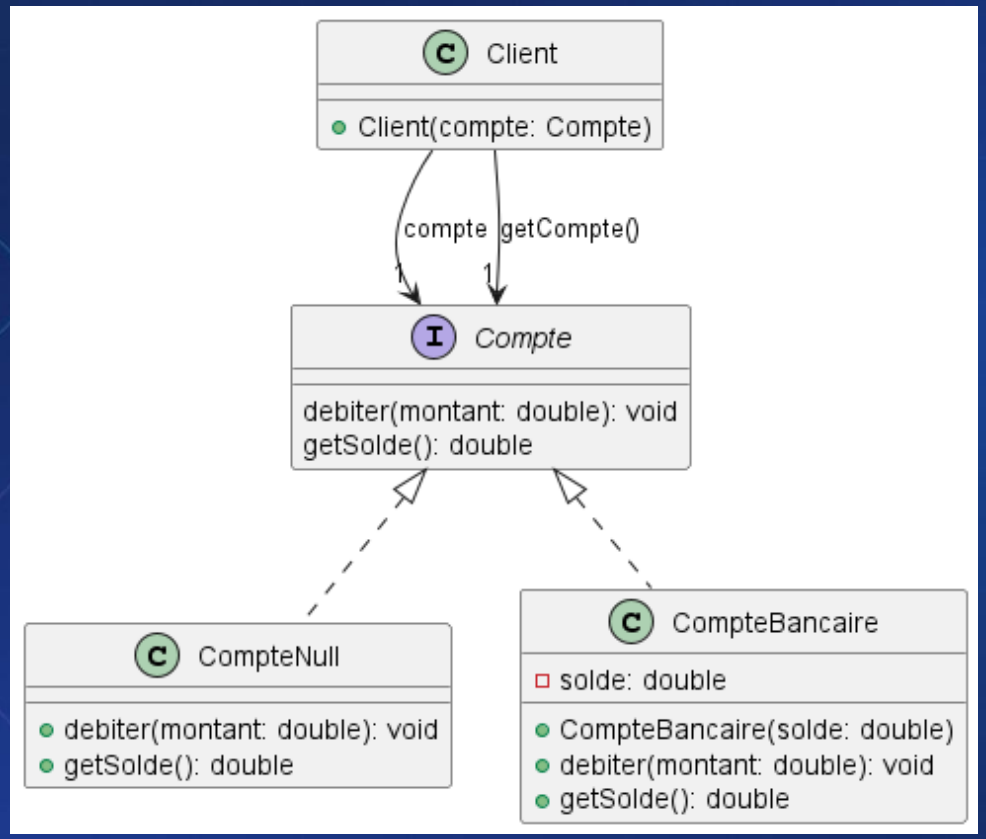
```
<terminated> Main (1) [Java Application] C:\Users\ferdi\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe (29 Oct
Client 1 - Solde initial : 1000.0 €
Client 1 - Solde après débit : 800.0 €

Client 2 - Tentative d'accès sans vérification...
ERREUR : NullPointerException ! Le compte est null.
```

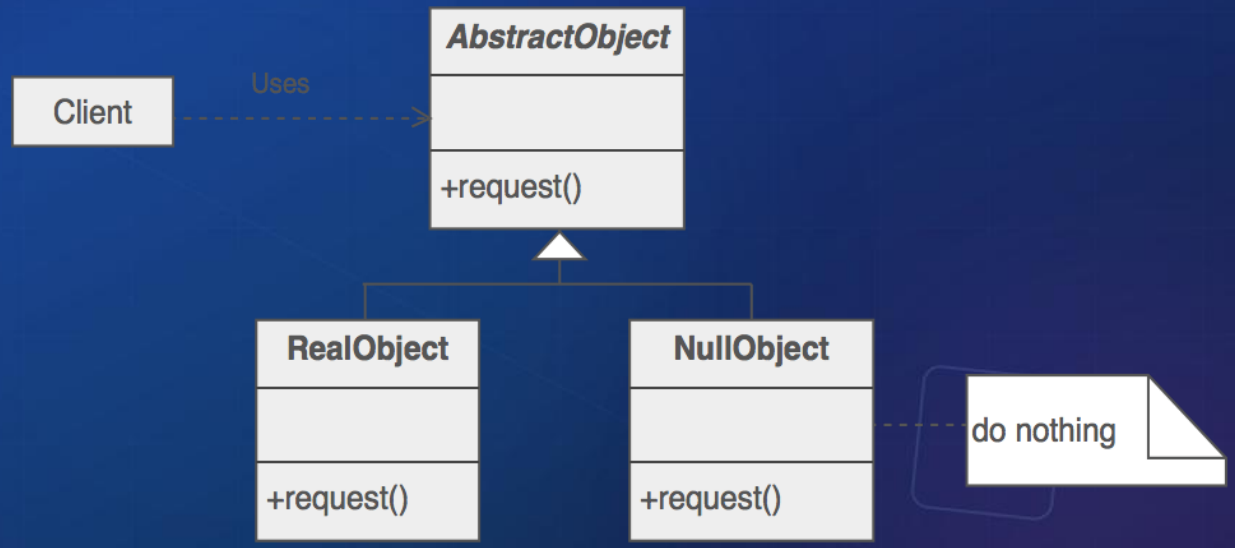


Live Coding

https://youtu.be/PVZ08267qts

Après rétro-conception





QCM

Webographie

- https://refactoring.guru/fr/introduce-null-object
- https://www.baeldung.com/java-null-object-pattern
- https://sourcemaking.com/design_patterns/null_object
- https://www.geeksforgeeks.org/system-design/null-object-design-pattern/
- https://refactoring.guru/fr/design-patterns/strategy
- https://refactoring.guru/fr/design-patterns/state
- https://fr.wikipedia.org/wiki/Objet_null_(patron_de_conception)