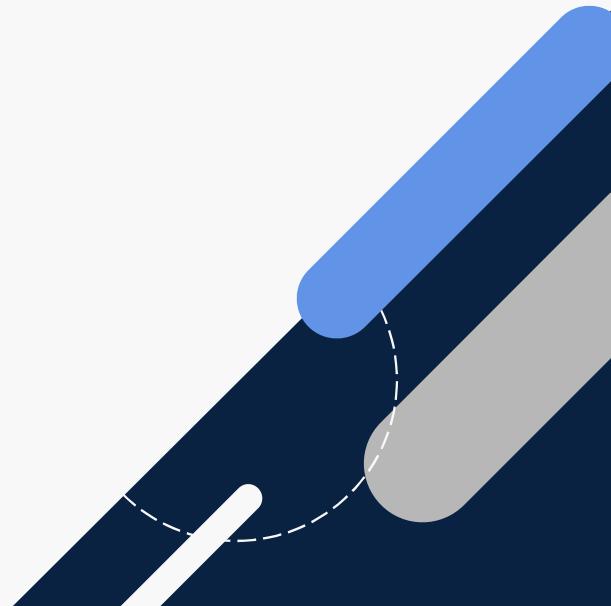


Pattern Prototype

...:::..



Définition

- Le Pattern Prototype est un patron de conception qui permet de copier des objets existants plutôt que de les créer à partir de zéro.
- Il repose sur le principe que la création d'un objet peut être coûteuse ou complexe, et qu'une duplication d'un prototype existant est plus efficace.
- Principe clé :

“Créer de nouveaux objets en clonant un objet prototype existant.”

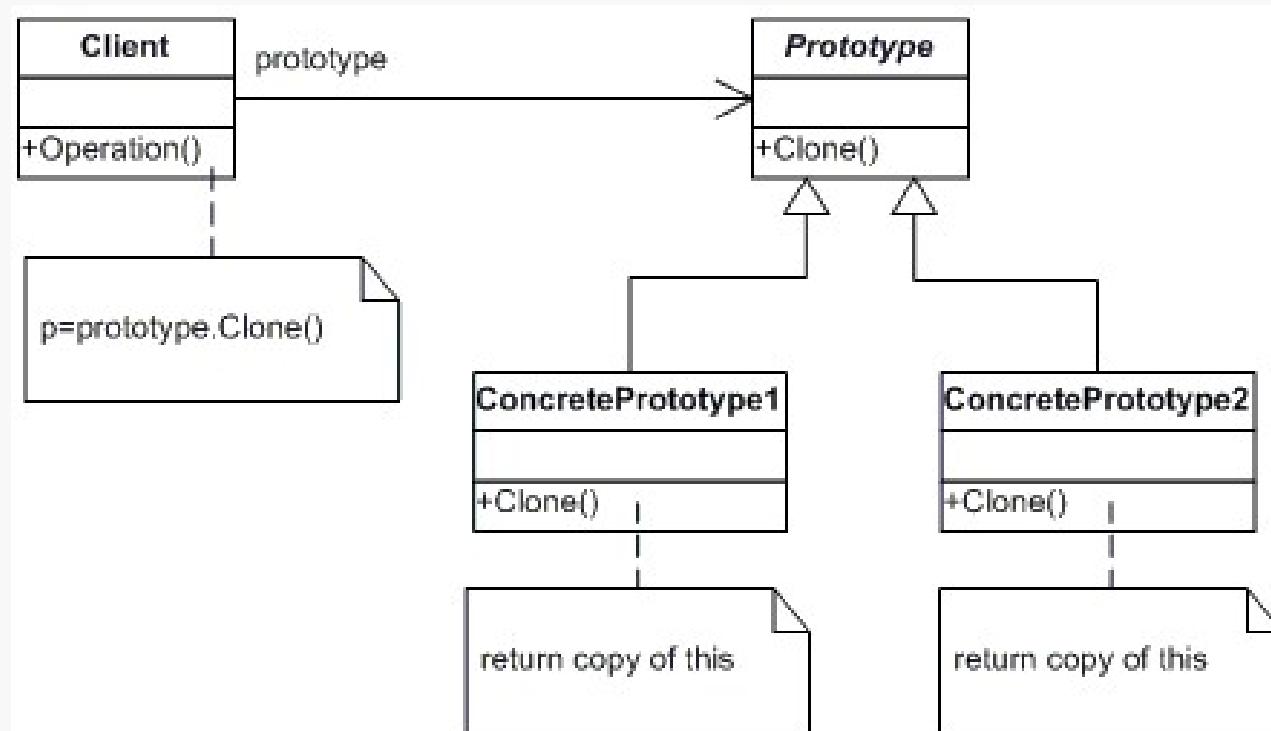
Objectifs

1. Réduire le coût de création d'objets complexes ou lourds à instancier.
2. Permettre la duplication d'objets avec leurs états actuels, en conservant les attributs configurés.
3. Fournir un mécanisme flexible pour créer des objets sans dépendre de leurs classes concrètes.
4. Faciliter la création d'objets dérivés à partir d'un modèle commun.

• • : : : •



Structure UML



• • • • •

Exercice

Vous développez un jeu de gestion d'animaux dans un zoo.

Chaque animal a des attributs comme :

- nom (String)
- espèce (String)
- âge (int)
- santé (String)

Au lieu de créer chaque animal à partir de zéro, vous voulez cloner des animaux existants pour créer rapidement de nouveaux individus.

• • : : : : •

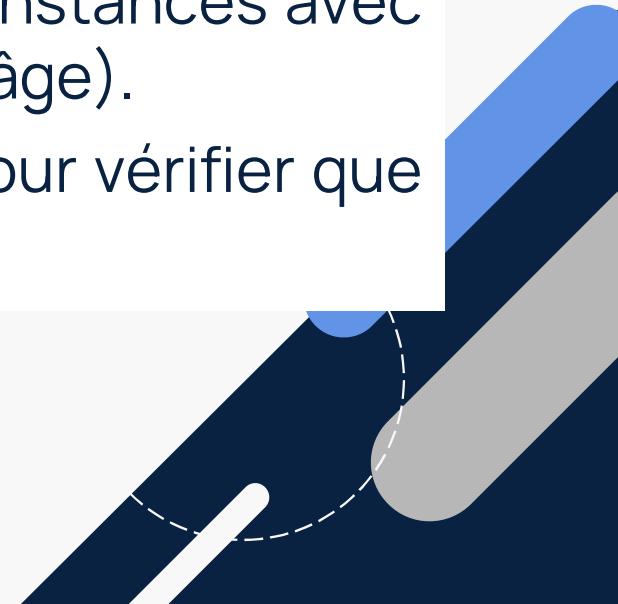


Exercice

Travail demandé

1. Créer une classe Animal qui implémente le pattern Prototype (méthode clone).
2. Créer 2 prototypes d'animaux : un lion et un éléphant.
3. Cloner ces animaux pour créer de nouvelles instances avec éventuellement des attributs modifiés (nom, âge).
4. Afficher les informations de chaque animal pour vérifier que le clonage fonctionne correctement.

• • : : : : •



Solution

```
class Animal implements Cloneable {  
    private String nom;  
    private String espece;  
    private int age;  
    private String sante;  
  
    public Animal(String nom, String espece, int age,  
String sante) {  
        this.nom = nom;  
        this.espece = espece;  
        this.age = age;  
        this.sante = sante;  
        this.maladies = new ArrayList<>();  
    }  
  
    . . . . .
```

```
// Méthode clone pour Prototype (clonage profond)  
@Override  
public Animal clone() {  
    try {  
        Animal clone = (Animal) super.clone();  
        // Clonage profond de la liste  
        return clone;  
    } catch (CloneNotSupportedException e) {  
        e.printStackTrace();  
        return null;  
    }  
}  
  
// Affichage des infos  
public void afficherInfos() {  
    System.out.println("Nom: " + nom);  
    System.out.println("Espèce: " + espece);  
    System.out.println("Âge: " + age);  
    System.out.println("Santé: " + sante);  
    System.out.println("Maladies: " + maladies);  
    System.out.println("-----");  
}
```

Solution

```
public class ZooPrototypeDemo {  
    public static void main(String[] args) {  
        // Création des prototypes  
        Animal lionPrototype = new Animal("Lion  
Prototype", "Lion", 5, "Bon");  
  
        Animal elephantPrototype = new Animal("Éléphant  
Prototype", "Éléphant", 10, "Bon");  
  
        // Clonage des animaux  
        Animal lion1 = lionPrototype.clone();  
        lion1.setNom("Simba");  
        lion1.setAge(3);  
  
        Animal elephant1 = elephantPrototype.clone();  
        elephant1.setNom("Dumbo");  
        elephant1.setAge(8);  
  
        // Affichage  
        lionPrototype.afficherInfos();  
        lion1.afficherInfos();  
        elephantPrototype.afficherInfos();  
        elephant1.afficherInfos();  
    }  
}
```

