DESIGN PATTERNS - CHAPITRE 8

Bridge en Python

Séparer deux axes de variation en reliant une abstraction à un objet d'implémentation.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Séparer deux axes de variation en reliant une abstraction à un objet d'implémentation.
- Faire varier forme et couleur sans multiplier leurs sous-classes combinées.

Prérequis. Savoir utiliser classes, héritage, composition, listes, dictionnaires et exceptions Python. Les mécanismes spécifiques sont expliqués avant leur emploi.

Bridge : intention et expérience

Bridge : séparer deux axes de variation

Si formes et couleurs varient indépendamment, créer une sous-classe pour chaque paire multiplie les combinaisons. Le **Bridge** donne à une forme un objet couleur auquel elle délègue le rendu. C'est une composition : un objet contient une référence vers un collaborateur.

Python

```python
# example.py
# Abstraction
class Shape:
    def __init__(self, color):
        self.color = color

    def draw(self):
        pass

# Refined Abstraction
class Circle(Shape):
    def __init__(self, color, radius):
        super().__init__(color)
        self.radius = radius

    def draw(self):
        return f"Circle with radius {self.radius} drawn in {self.color.fill()}"

# Implementor
class Color:
    def fill(self):
        pass

# Concrete Implementor
class RedColor(Color):
    def fill(self):
        return "Red"

class BlueColor(Color):
    def fill(self):
        return "Blue"

# Client code
if __name__ == "__main__":
    red = RedColor()
    blue = BlueColor()
```

```
    circle1 = Circle(red, 5)
    circle2 = Circle(blue, 10)

    print(circle1.draw())
    print(circle2.draw())
```

Vérifier et interpréter

Utilisez la même instance de couleur pour deux cercles de rayons différents, puis changez seulement l'objet couleur d'un cercle. La géométrie ne doit pas changer. Cette séparation permet deux axes indépendants, au prix d'une délégation supplémentaire qui serait inutile si la couleur ne variait jamais.

Version web : https://www.clearcode.fr/courses/programming/design-patterns/structure/bridge