DESIGN PATTERNS - CHAPITRE 1

Partir d'un problème de variation

Isoler une construction variable et définir ce que le client attend.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Définir un contrat commun pour plusieurs objets.
- Tester le choix de produit et le cas inconnu.

Prérequis. Savoir utiliser les fonctions, les classes, les listes, les dictionnaires et les exceptions Python.

Partir d'un problème de variation

Comment ajouter un type de véhicule sans disperser sa construction dans le programme ? Une **fabrique simple** centralise ce choix. Le contrat commun est ici `drive() -> str` : chaque véhicule fournit un message. Le client demande un type à la fabrique, puis appelle cette opération.

Un contrat décrit ce que l'appelant peut attendre : opérations disponibles, entrées acceptées, résultat et erreurs. Une interface est l'ensemble des opérations exposées ; elle peut être respectée sans héritage explicite en Python.

Python

```python
# example.py
# Factory Pattern Example
class Vehicle:
    def __init__(self, model):
        self.model = model

    def drive(self):
        raise NotImplementedError

class Car(Vehicle):
    def drive(self):
        return "Driving a car"

class Motorcycle(Vehicle):
    def drive(self):
        return "Riding a motorcycle"

class VehicleFactory:
    def create_vehicle(self, type):
        if type == "car":
            return Car("Sedan")
        elif type == "motorcycle":
            return Motorcycle("Sportbike")
        else:
            raise ValueError("Unknown vehicle type")

# Usage
factory = VehicleFactory()
car = factory.create_vehicle("car")
print(car.drive())  # Output: Driving a car
motorcycle = factory.create_vehicle("motorcycle")
print(motorcycle.drive()) # Output: Riding a motorcycle
```

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