Міністерство освіти і науки України
Хмельницький національний університет
Кафедра комп’ютерної інженерії та інформаційних систем

ЛАБОРАТОРНА РОБОТА № 1-2

з дисципліни: Об’єктно-орієнтовані технології програмування

Виконав студент: Роюк Р.В.
Група: КІ2м-24-2
Перевірив: Лисенко С.М.

2025

Завдання: написати програмне забезпечення, що описує застосування патерну «Фабричний метод» та породжуючого патерну «Прототип».

Отже маємо програму, яка моделює логістичну компанію, що використовує «Фабричний метод» для створення відповідного транспорту (вантажівка або корабель) та «Прототип» для клонування об'єктів транспорту.

```
from abc import ABC, abstractmethod
import copy

# Абстрактний клас для транспорту
class Transport(ABC):
    """
    Абстрактний клас, що визначає базовий інтерфейс для всіх типів транспорту.
    """
    @abstractmethod
    def deliver(self):
        """Метод для здійснення доставки."""
        pass

    @abstractmethod
    def clone(self):
        """Метод для клонування об'єкта."""
        pass

# Конкретні класи транспорту
class Truck(Transport):
    """
    Клас, що представляє вантажівку.
    """
    def __init__(self, cargo_capacity):
        """Ініціалізує вантажівку з заданою місткістю."""
        self.cargo_capacity = cargo_capacity

    def deliver(self):
        """Реалізація доставки вантажівкою."""
        return f"Доставка вантажівкою з місткістю {self.cargo_capacity} тонн"

    def clone(self):
        """Створення копії вантажівки."""
        return copy.deepcopy(self)

class Ship(Transport):
    """
    Клас, що представляє корабель.
    """
    def __init__(self, cargo_capacity):
```

```
        """Ініціалізує корабель з заданою місткістю."""
        self.cargo_capacity = cargo_capacity

    def deliver(self):
        """Реалізація доставки кораблем."""
        return f"Доставка кораблем з місткістю {self.cargo_capacity} тонн"

    def clone(self):
        """Створення копії корабля."""
        return copy.deepcopy(self)

# Фабричний метод
class Logistics(ABC):
    """
    Абстрактний клас, що визначає фабричний метод для створення транспорту.
    """
    @abstractmethod
    def create_transport(self):
        """Фабричний метод для створення транспорту."""
        pass

# Конкретні фабрики транспорту
class RoadLogistics(Logistics):
    """
    Клас, що представляє логістику для автомобільного транспорту.
    """
    def create_transport(self):
        """Створює вантажівку з місткістю 20 тонн."""
        return Truck(20)

class SeaLogistics(Logistics):
    """
    Клас, що представляє логістику для морського транспорту.
    """
    def create_transport(self):
        """Створює корабель з місткістю 200 тонн."""
        return Ship(200)

# Демонстрація роботи програми
if __name__ == "__main__":
    # Використання автомобільної логістики
    road_logistics = RoadLogistics()
    truck = road_logistics.create_transport()
    print(truck.deliver())

    # Використання морської логістики
    sea_logistics = SeaLogistics()
    ship = sea_logistics.create_transport()
    print(ship.deliver())

    # Демонстрація патерну "Прототип" (клонування об'єктів транспорту)
```

```
cloned_truck = truck.clone()
print(cloned_truck.deliver())

cloned_ship = ship.clone()
print(cloned_ship.deliver())
```

Фабричний метод є породжуючим патерном, який дозволяє створювати об'єкти без вказання їхніх конкретних класів. Це досягається шляхом визначення загального інтерфейсу для створення об'єктів, а конкретні класи реалізують його для створення відповідних типів. У коді це реалізовано як:

- Абстрактний клас Logistics – має абстрактний метод create_transport(), який повинен повертати екземпляр транспорту.
- Конкретні фабрики RoadLogistics та SeaLogistics: RoadLogistics створює вантажівку (Truck), а SeaLogistics створює корабель (Ship). Вони реалізують фабричний метод create_transport().

Перевагами такого підходу є спрощення розширення, тобто якщо потрібно додати новий вид транспорту, достатньо створити новий підклас Logistics, а також данний підхід відокремлює створення об’єктів від їхнього використання.

Прототип — це породжуючий патерн, який дозволяє створювати нові об'єкти шляхом копіювання вже існуючих екземплярів. Це особливо корисно, коли створення об'єкта є складним або ресурсомістким процесом. У коді це реалізовано як:

- Абстрактний клас Transport – Визначає метод clone(), який реалізують його підкласи.
- Класи Truck і Ship – Мають метод clone(), який використовує copy.deepcopy(self), щоб створити повну копію об'єкта.

У роботі, створюється об'єкт транспорту (вантажівка або корабель), далі викликається clone(), що створює копію об'єкта, яка має ті ж характеристики, що й оригінали.

Перевагами такого підходу є швидке створення нових екземплярів, особливо якщо створення з нуля є ресурсомістким, а також збереження стану об'єкта при його копіюванні.