Wzorce projektowe, cz. 7 – Factory method

Wzorzec ***metody wytwórczej*** dostarcza abstrakcji do tworzenia obiektów nieokreślonych, ale powiązanych typów. Umożliwia także dziedziczącym klasom decydowanie jakiego typu ma to być obiekt.

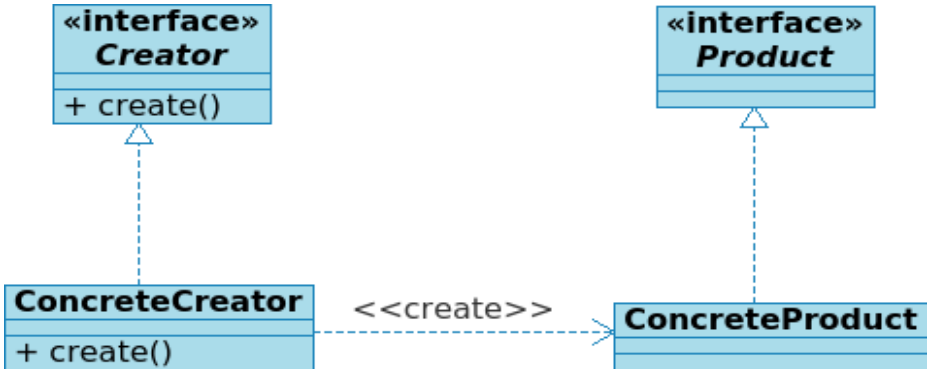

Diagram klasy wzorca Factory method

Wzorzec składa się z dwóch ról: produktu ***Product*** definiującego typ zasobów oraz kreatora ***Creator*** definiującego sposób ich tworzenia. Wszystkie typy produktów (***ConreteProduct1, ConreteProduct2*** itp.) muszą implementować interfejs ***Product***. Z kolei ***ConcreteCrator*** dostarcza mechanizm umożliwiający stworzenie obiektu produktu danego typu.

Przykładowa implementacja

```
<?php

// Produkty
interface Product{
    public function getName();
}
class ConceteProduct1 implements Product{
    public function getName() {
        return "Produkt 1";
    }
}
class ConceteProduct2 implements Product{
    public function getName() {
        return "Produkt 2";
    }
```

```
}

// Kreator tworzacy obiekt produktu
interface Creator{
    public function create($type);
}
class ConcreteCreator implements Creator{
    public function create($type) {
        switch($type) {
            case 'Product 1':
                return new ConceteProduct1();
                break;
            case 'Product 2':
                return new ConceteProduct2();
                break;
        }
    }
}

// testy
$creator = new ConcreteCreator();
$prod1 = $creator->create("Product 1");
$prod2 = $creator->create("Product 2");
echo $prod1->getName(); // wyswietli "Produkt 1"
echo $prod2->getName(); // wyswietli "Produkt 2"

?>
```

Przykład z życia wzięty

Tworzymy system zamówień dla pizzerii. W ofercie są różne typy pizz. Podstawowym pytaniem jest: jak stworzyć wydajny mechanizm do tworzenia obiektów różnych rodzai pizz? Posłużmy się metodą wytwórczą...

```
<?php

// Produkty
interface Pizza{
    public function getName();
}
class HawaiianPizza implements Pizza{
    public function getName() {
        return "Hawalian pizza";
    }
}
class DeluxePizza implements Pizza{
    public function getName() {
        return "Deluxe pizza";
    }
}

// Kreator tworzacy obiekt produktu
interface Creator{
    public function create($type);
}
class ConcreteCreator implements Creator{
    public function create($type) {
        switch($type) {
            case 'Hawalian':
                return new HawaiianPizza();
                break;
            case 'Deluxe':
                return new DeluxePizza();
```

```
                break;
        }
    }
}

// testy
$creator = new ConcreteCreator();
$prod1 = $creator->create("Hawalian");
$prod2 = $creator->create("Deluxe");
echo $prod1->getName(); // wyswietli "Hawalian pizza"
echo $prod2->getName(); // wyswietli "Deluxe pizza"

?>
```

Dzięki zastosowaniu factory method możemy w łatwy sposób dołączać kolejne pizze. Zamiast używania konstrukcji ***switch*** (korzystam z tego, gdyż nie chcę komplikować przykładu) warto byłoby stworzyć bardziej abstrakcyjny mechanizm.

Zalety i wady

Zalety:

- Niezależność od konkretnych implementacji zasobów oraz procesu ich tworzenia.
- Wzorzec hermetyzuje proces tworzenia obiektów, zamykając go za ściśle zdefiniowanym interfejsem.
- Spójność produktów - w sytuacji, gdy pożądane jest, aby klasy produkty były z określonej rodziny.

Wady:

- Nie znam...

Zastosowanie

Wzorzec metody wytwórczej można wykorzystać między innymi przy tworzeniu systemów zamówień, gdzie oferta może się zmieniać, ale składa z jednakowego typu produktów.

Innym zastosowaniem może być system pluginów. Dzięki zastosowaniu metody wytwórczej możemy łatwo rozbudowywać nasz skrypt o kolejne funkcjonalności (np. o obsługę kolejnych formatów plików).