# Metoda wytwórcza

Emil Śmiech,
Amelia Królczyk,
Michał Palwal,
Sabina Adamska

Czerwiec 2021

# Wstęp

Wzorzec projektowy to sprawdzona koncepcja, która opisuje problem powtarzający się wielokrotnie w określonym kontekście, działające na niego siły, oraz podaje istotę jego rozwiązania w sposób abstrakcyjny.

Wzorce projektowe najczęściej tworzone są w oparciu o programowanie obiektowe. Zakres tego pojęcia stał się problemem rozważanym od połowy lat 90. XX wieku. Ostatecznie ustalono, że algorytmy nie są wzorcami projektowymi, jako że rozwiązują problemy obliczeniowe, a nie projektowe. Wzorce często są łączone w celu rozwiązania bardziej złożonego problemu.

Zamiast skupiać się na funkcjonowaniu poszczególnych elementów, wzorce projektowe stanowią abstrakcyjny opis zależności pomiędzy klasami, co w efekcie wprowadza pewną standaryzację kodu oraz zwiększa jego zrozumiałość, wydajność i niezawodność. Wartość wzorców projektowych stanowi nie tylko samo rozwiązanie problemu, ale także dokumentacja, która wyjaśnia cel, działanie, zalety danego rozwiązania, co pomaga w łatwiejszym stosowaniu i adaptacji wzorców w danym zastosowaniu.

Wzorce projektowe mogą przyspieszyć proces rozwoju oprogramowania przez dostarczenie wypróbowanych rozwiązań dla problemów, które mogą nie być oczywiste na początku procesu projektowego. Często zagadnienia te wiążą się z ewolucją oczekiwań względem projektowanego systemu: rozszerzeniem jego funkcjonalności, zmianą sposobu i formatu wprowadzanych danych czy dostosowaniem aplikacji do różnych klas użytkowników. Nieuwzględnienie ich na początku procesu rozwoju produktu programistycznego powoduje często konieczność gruntownego przebudowywania zaawansowanego lub gotowego już oprogramowania.

# Metoda wytwórcza

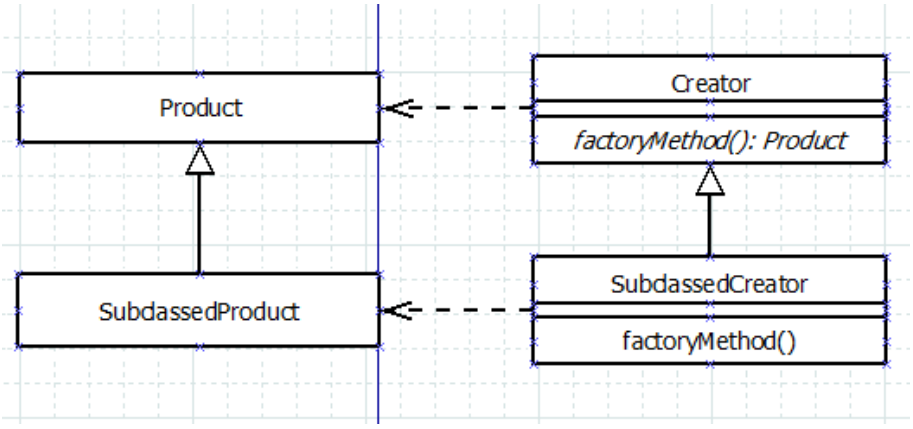
Metoda wytwórcza z angielskiego factory method jest to kreacyjny wzorzec projektowy, którego celem jest dostarczenie interfejsu do tworzenia obiektów nieokreślonych jako powiązanych typów. Tworzeniem egzemplarzy zajmują się podklasy.

We wzorcu występują dwie ogólne klasy bądź interfejsy definiujące pewien typ zasobów (Product) oraz sposób ich tworzenia (Creator, metoda factoryMethod()). Od nich wyprowadza się konkretne klasy zasobów (ConcreteProduct) wraz z tworzącymi je klasami wytwórczymi (ConcreteCreator), które dostarczają odpowiednią implementację metody factoryMethod(). Komponent pragnący tworzyć zasoby i operować na nich, korzysta z ogólnych interfejsów Product oraz Creator, umożliwiając wybór konkretnej implementacji w sposób dynamiczny.

# Opis problemu

Wyobraźmy sobie sytuację, w której mamy aplikacje służącą do zamawiania biletów online. Do tej pory program pozwalał wyłącznie na zamawianei biletów na pociąg ale został rozszeżony o możliwość zamówienia biletu na autobus i tramwaj. Problem polega na tym, że program pozwala na zamówienie tylko jednego rodzaju biletów.

## Wykres UML wykonany w Dia




## Implementacja

```python
class Train:
    def deliver(self, ticket):
        print(f"Bilet {ticket} na pociąg.")

class Bus:
    def deliver(self, ticket):
        print(f"Bilet {ticket} na autobus.")


class Tram:
    def deliver(self, ticket):
        print(f"Bilet {ticket} na tramwaj.")


class OrderLifecycle:
    def process_order(self, order_id):
        ticket = self.prepare_ticket(order_id)
        delivery_service = self.delivery_service()
        delivery_service.deliver(ticket)

    def prepare_ticket(self, order_id):
        ticket = f"[zamówienie:{order_id}]"
        print(f"Bilet {ticket} został zamówiony")
        return ticket

    def delivery_service(self):
        return Train()

```

```
29
30  class BusOrderLifecycle(OrderLifecycle):
31      def delivery_service(self):
32          return Bus()
33
34
35  class TramOrderLifecycle(OrderLifecycle):
36      def delivery_service(self):
37          return Tram()
38
39
40  if __name__ == "__main__":
41      tram_order = TramOrderLifecycle()
42      bus_order =BusOrderLifecycle()
43      train_order = OrderLifecycle()
44
45      train_order.process_order("order_1")
46      tram_order.process_order("order_2")
47      bus_order.process_order("order_3")
```

Po uruchomieniu tego programu na konsoli pokaże się:

```
Bilet [zamówienie:order_1] został zamówiony
Bilet [zamówienie:order_1] na pociąg.
Bilet [zamówienie:order_2] został zamówiony
Bilet [zamówienie:order_2] na tramwaj.
Bilet [zamówienie:order_3] został zamówiony
Bilet [zamówienie:order_3] na autobus.
PS C:\Users\Admin\Desktop\Wzorce Projektowe>
```

## Bibliografia


1. https://pl.wikipedia.org/wiki/Wzorzec_projektowy_(informatyka)
2. https://pl.wikipedia.org/wiki/Metoda_wytw%C3%B3rcza_(wzorzec_projektowy)