Metoda Wytwórcza (Factory Method)

W sieci istnieje spory bałagan informacyjny związany z wzorcami kreacyjnymi. Przede wszystkim chodzi tutaj o stosowanie słowa „fabryka” (factory) w wielu różnych kontekstach i w odniesieniu do całkowicie odmiennych rozwiązań.

Prosta Fabryka (Simple Factory)

Przyjrzyjmy się najpierw koncepcji, którą określa się jako „prosta fabryka” (lub po prostu „fabryka”). **Uwaga: to nie jest wzorzec projektowy.** Generalnie jest to przykład rozwiązania, którego wolelibyśmy unikać.

„Prostą fabrykę” charakteryzuje szereg instrukcji if-else (lub switch-case) decydujących o tym, jaki typ obiektu zostanie stworzony. Na przykład:

```
Piwo* ProstaFabryka::produkujPiwo(string rodzaj)
{
        if (rodzaj == „perla”)
                return new Perla();
        else if (rodzaj == „harnas”)
                return new Harnas();
        else if (rodzaj == „tyskie”)
                return new Tyskie();
}
```

Takiego rozwiązania należy unikać. Przede wszystkim, to podejście **łamie zasadę Open/Closed Principle** (z SOLID). Jeżeli w przyszłości będziemy chcieli rozbudować naszą aplikację o kolejny rodzaj piwa, będziemy musieli wrócić do tej metody i ją zmodyfikować, dodając kolejnego if'a (a tego nie chcemy – kod powinien być napisany tak, aby dodawanie nowych elementów nie wymagało modyfikowania już zaimplementowanych i zamkniętych).

Metoda Wytwórcza (Factory Method)

Wzorzec metody wytwórczej stanowi odpowiedź na ten problem. Mamy tutaj dwie role:

- produkty (obiekty realizujące wspólny interfejs, np. różne rodzaje piwa)
- kreatory (często nazywane fabrykami – obiekty odpowiedzialne za tworzenie produktów)

Wszystkie kreatory realizują interfejs zawierający metodę wytwórczą (np. **stworzPiwo()**). Dla każdego rodzaju produktu implementujemy również kreator, np.:

- klasa DystrybutorPerly tworzy i zwraca obiekty klasy Perla
- klasa DystrybutorHarnasia tworzy i zwraca obiekty klasy Harnas
- klasa DystrybutorTyskiego tworzy i zwraca obiekty klasy Tyskie

I teraz najważniejsze: klient korzysta z tych kreatorów za pośrednictwem interfejsu. Wyobraźmy sobie np. klasę Barman, zawierającą następującą metodę:

```
Piwo* Barman::nalejZDystrybutora(Dystrybutor* dystrybutor)
{
	return dystrybutor->stworzPiwo();
}
```

Barman może współpracować z dowolnym dystrybutorem – wszystko zależy od tego, jaki obiekt przekażemy mu jako argument metody **nalejZDystrybutora()**. W ten sposób Barman jest niezależny od całego etapu produkcji piwa i jego różnych rodzajów. Barman musi jedynie umieć obsłużyć dystrybutor (czyli wywołać metodę **stworzPiwo()**). A jakie to będzie konkretnie piwo? Zależy od tego, jaki dystrybutor mu damy.

Zauważcie, że zasada **Open/Closed Principle** jest tutaj zachowana. Jeżeli w przyszłości będziemy chcieli dodać do aplikacji nowy rodzaj piwa, to zaimplementujemy nową klasę realizującą interfejs Piwo i nowy kreator dla niej. Klasy, które już istnieją, nie będą wymagały zmian.

Fabryka Abstrakcyjna

Fabryka abstrakcyjna to kolejny wzorzec, ideowo mocno związany z Metodą Wytwórczą. W tym przypadku jednak koncentrujemy się na tworzeniu całych **rodzin** produktów. W dużym skrócie, **mamy wiele różnych rodzajów produktów**, a fabryka abstrakcyjna zawiera **wiele metod do ich wytwarzania.**

Wyobraźmy sobie, że potrzebujemy obiektów wytwarzających zestawy komputerowe: jednostkę centralną, monitor, klawiaturę, myszkę... I teraz okazuje się, że w programie będzie istniało wiele różnych marek tego sprzętu.

W przykładzie z metodą wytwórczą mieliśmy jeden rodzaj produktu – Piwo. Teraz mamy JednostkaCentralna, Monitor, Klawiatura, Myszka... To wszystko są interfejsy, które będą realizowane przez konkretne klasy. Na przykład: JednostkaCentralnaDell, JednostkaCentralnaHP, MonitorDell, MonitorHP...

Interfejs fabryki abstrakcyjnej zawiera wiele metod wytwórczych – po jednej dla każdego rodzaju produktu:

```
class FabrykaAbstrakcyjna
{
public:
	virtual JednostkaCentralna* stworzJednostkeCentralna() = 0;
	virtual Monitor* stworzMonitor() = 0;
	virtual Klawiatura* stworzKlawiature() = 0;
	virtual Myszka* stworzMyszke() = 0;
};
```

Dla każdej marki sprzętu tworzymy oddzielną, konkretną fabrykę. Fabryka Dell'a produkuje wszystkie rodzaje sprzętu Dell'a, fabryka HP analogicznie – produkuje sprzęt HP.