**Java Efektywne Programowanie notatki:**

**Tworzenie i usuwanie obiektów**

***Temat 1. Tworzenie statycznych metod fabrycznych zamiast konstruktorów***

Statyczna metoda fabryczna – zwyczajna metoda statyczna zwracająca obiekt danej klasy.

Ex.

public static Boolean valueOf(boolean b){  
 return (b ? Boolean.*TRUE* : Boolean.*FALSE*);  
}

Zalety statycznych metod fabrycznych zamiast publicznych konstruktorów:

* Metody fabryczne posiadają nazwy – przez co kod jest łatwiejszy do analizy.

Zamiast konstruktora BigInteger (int,int,Random), który zwraca obiekt typu BigInteger, która prawdopodobnie jest liczbą pierwszą , lepiej zastosować statyczną metodę fabryczną o nazwie BigInteger.probablePrime(…)

Klasa może posiadać tylko jeden konstruktor o odpowiedniej sygnaturze, można obejść poprzez stworzenie dodatkowego konstruktora o innej kolejności typów parametrów, ale powoduje to potencjalne użycie nie tego konstruktora, którego się chce. Metody statyczne posiadają nazwy, więc można powtarzać sygnatury.

* W przeciwieństwie do konstruktorów, nie jest wymagane utworzenie nowegoobiektu podczas ich wywołania – unika powielania niepotrzebnie powielających się obiektów.

Wywołanie tego samego obiektu podczas kolejnych wywołań , może być wykorzystane do ścisłej kontroli nad istniejącymi w danej chwili obiektami – klasy,które tak robią nazywa się **kontrolowanymi przez instancję.** Pozwala to na zagwarantowanie, że klasa jest klasą typu singleton, pozwala upewnić się klasie niezmiennej, że istnieją dwa identyczne obiekty a.equals(b) jest spełnione tylko wtedy gdy a==b ( Podstawa wzorca Flyweight).

* Możliwość zwracania typu, który jest podtypem zwracanego typu – ukrycie implementacji klas pozwala na tworzenie niewielkich API. Tak działa np. java.util.Collections
* W przypadku metod fabrycznych, w zależności od tego, jakie wartości parametrów są przekazywane do metody, ulegać zmianie może też klasa zwracanego obiektu – można zwracać obiekty dowolnej klasy pochodnej do klasy zadeklarowanej jako zwracana. Klasa zwracanego obiektu może różnić się od jej odpowiednika w kolejnych wersjach pakietu. Przykładem jest Klasa EnumSet, która nie ma konstruktorów jedynie statyczne metody fabryczne. W implementacji JDK w zależności od ilości elementów do 64 zwracany jest RegularEnumSet , gdy typ wyliczeniowy zawiera ponad 64 elementy zwracany jest obiekt JumboEnumSet. Dla klienta jest to niewidoczne a może wpłynąć na wydajność.
* Klasa obiektu zwracanego przez statyczną metodę fabryczną może jeszcze nie istnieć, kiedy pisana jest klasa zawierająca metodę statyczną – taka elastyczność wykorzystywana jest w *bibliotekach dostawcy usług* np. JDBC. Biblioteka dostawcy usług jest systemem, w którym dostawcy tworzą implementację interfejsu API, dostępnego dla użytkowników biblioteki, izolując ich od implementacji. Biblioteka dostawcy usług zawiera : interfejs usługi(implementowany przez dostawce), API rejestracji dostawcy(rejestracja implementacji dająca klientom dostęp do niej), API dostępu do usług(uzyskanie instancji usługi)

Wady statycznych metod fabrycznych zamiast publicznych konstruktorów:

* Klasy nieposiadające public/protected konstruktora nie mogą być dziedziczone – Nie można na przykład dziedziczyć po żadnej z klas Collections.
* Nie można ich od razu odróżnić od innych metod statycznych – nie są umieszczane w dokumentacji wraz z konstruktorami, ciężko się zorientować po dokumentacji klasy jak skonstruować obiekt.

***Temat 2. Zastosowanie budowniczego do obsługi wielu parametrów konstruktora***

W sytuacji gdy mamy parametry wymagane oraz pola opcjonalne. Programiści wykorzystują wzorzec **konstruktora teleskopowego**, w którym tworzy się konstruktor z parametrami wymaganymi oraz kolejny z dodatkowym jednym parametrem opcjonalnym, kolejny z dwoma i tak dalej. Wywołanie konstruktora wymaga wielu parametrów czasem przekazujemy wartość zero gdy nie ma odpowiedniej wartości opcjonalnej. Trudno jest pisać kod klienta w przypadku wielu parametrów oraz kod jest nieczytelny. Łatwo o powstanie subtelnych błędów( np. zamiana miejscami parametrów).

Innym sposobem radzenia sobie z takimi sytuacjami jest wzorzec **JavaBeans** – w którym wykorzystuje się konstruktor bezparametrowy do tworzenia obiektu, a następnie wywołuje się metody ustawiające wartości każdego z wymaganych i opcjonalnych parametrów. Wadą JavaBeans jest to, że w czasie tworzenia obiektu , jest on przejściowo w stanie niespójnym. A wywołanie obiektu w stanie niespójnym może powodować błąd, w momencie dalekim od kodu problemu. Dodatkowo wymaga dodatkowego nakładu pracy by zapewnić bezpieczeństwo dla wątków.

Alternatywą dla powyższych sposobów łączącą bezpieczeństwo wzorca konstruktora teleskopowego z czytelności wzorca JavaBeans jest **Builder**.

Wzorzec Builder jest dobrym wyborem przy projektowaniu klas, których konstruktorzy lub statyczne metody fabryczne mają więcej niż kilka parametrów, szczególnie gdy większość tych parametrów jest opcjonalna.