Java 10 update

Spis treści

Java 10 update

Java 10 została wydana w marcu 2018 i jest wersją **non-LTS**. Poniżej omówimy niektóre funkcjonalności udostępnione w tym wydaniu. Przy aktualizacji wersji Javy często poprawianych jest o wiele więcej funkcjonalności i dodawanych o wiele więcej klas lub metod niż te, które wymieniamy tutaj. W obrębie tych materiałów poruszamy tylko te kwestie, które są adekwatne do naszego poziomu zaawansowania jako Java developerów.

Niektóre z poruszanych zagadnień będą dla Ciebie tylko przypomnieniem, bo poruszaliśmy je już wcześniej. Z jednej strony chcę Ci przez to pokazać, ile już umiesz, a z drugiej strony zaznaczyć, które funkcjonalności były dodawane do języka na przestrzeni kolejnych wydań Javy.

var

Java 10 wprowadziła możliwość pominięcia pisania typu zmiennych. Oczywiście istnieje kilka ograniczeń, które teraz sobie omówimy. Od Javy 10 możemy stosować słowo kluczowe `var` podczas deklaracji zmiennej. Słówko to daje możliwość uproszczenia pewnych kombinacji. Przykładowo zamiast pisać taki kod:

```
Map<String, Map<String, List<String>>> result = someMethod();
```

Możemy od Javy 10 zapisać to samo w ten sposób:

```
var result = someMethod();
```

W kwestii ograniczeń stosowania `var`:

1. Możemy używać `var` do deklaracji różnych typów danych.

```
var a = 1;
var b = 1.1;
```

```
var c = 'a';
var d = "zajavka";
var e = true;
```

2. Możemy używać `var` do deklaracji zmiennych lokalnych, czyli np. zmiennych deklarowanych w obrębie jakiejś metody.
3. `var` nie może być użyte jako pole w klasie.
4. `var` nie może być używane zamiast typów generycznych.

```
// tak nie wolno
var<String> list = new ArrayList<>();
// tak też nie
List<var> list = new ArrayList<>();
// i tak też nie wolno
var<var> list = new ArrayList<>();

// ale tak już można
var list = new ArrayList<>();
```

5. `var` nie może być użyte bez inicjalizacji.

```
// tak nie wolno
var a;
a = 2;
```

```
// tak już można
var b = 1;
```

6. `var` nie może być użyte z lambdami.

```
// tak nie wolno
var lambda = () -> "zajavka";
// tak też nie
Consumer<String> lambda = (var a) -> System.out.println(a);
```

7. `var` nie może być używane jako parametry albo typy zwracane w metodach.

```
// tak nie wolno
private static void method(var a) {}
// tak też nie wolno
private static var method1(String a) {}
```

Są zwolennicy i są przeciwnicy tego rozwiązania ☺.

Kolekcje

W Java 10 do interfejsów `List`, `Set` oraz `Map` została dodana metoda `copyOf()`. Pozwala ona na stworzenie niemutowalnej kolekcji z danymi identycznymi jak kolekcja kopiowana. Przykład:

```
public class Example {
    public static void main(String[] args) {
        List<String> food = new ArrayList<>();
        food.add("Banana");
        food.add("Apple");
        food.add("Orange");
        System.out.println("Original food: " + food);

        List<String> copyOfFood = List.copyOf(food);
        System.out.println("Copied food: " + copyOfFood);

        food.add("Pineapple"); ①
        System.out.println("Original food: " + food);
        System.out.println("Copied food: " + copyOfFood);
    }
}
```

① Tutaj trzeba zwrócić uwagę na jedną kwestię. Możemy modyfikować zawartość źródłowej kolekcji. Nie możemy modyfikować zawartości kolekcji `copyOfFood`. Metoda `copyOf()` tworzy **immutable** collection, zatem wywołanie metody `add()` na `copyOfFood` spowodowałoby wyrzucenie wyjątku `UnsupportedOperationException`.

Na ekranie zostanie wydrukowane:

```
Original food: [Banana, Apple, Orange]
Copied food: [Banana, Apple, Orange]
Original food: [Banana, Apple, Orange, Pineapple] ①
Copied food: [Banana, Apple, Orange]
```

① Zmodyfikowana została tylko oryginalna kolekcja, ale skopiowana już nie.

Optional

Dopiero Java 10 wprowadziła do klasy `Optional` metodę `orElseThrow()`. Metoda ta wyrzuci określony przez nas wyjątek lub `NoSuchElementException`, jeżeli `Optional` okaże się pusty. Możemy ją wywołać zamiast metody `get()`, która dla pustego `Optionala` wyrzuci `NoSuchElementException`. Przykład:

```
Optional.ofNullable(null)
    .get(); ①

Optional.ofNullable(null)
    .orElseThrow(); ②

Optional.ofNullable(null)
    .orElseThrow(() -> new RuntimeException("My custom message")); ③
```

① Zostanie wyrzucony wyjątek `NoSuchElementException`.

② Zostanie wyrzucony wyjątek `NoSuchElementException`.

③ Zostanie wyrzucony wyjątek `RuntimeException`.

Stream - Collectors

W Java 10 zostały również dodane nowe kolektory do klasy `Collectors`. Przykład:

```
List<String> food = new ArrayList<>();
food.add("Banana");
food.add("Apple");
food.add("Orange");

List<String> collect = food.stream().collect(Collectors.toUnmodifiableList()); ①
collect.add("Pineapple"); ②
```

① Otrzymamy w ten sposób kolekcję **immutable**.

② Wywołanie tej linijki wyrzuci wyjątek `UnsupportedOperationException`.

Podsumowanie

Wymienione funkcjonalności nie są wszystkimi, jakie zostały wprowadzone w Javie 10. Przy aktualizacji wersji Javy często poprawianych jest o wiele więcej funkcjonalności i dodawanych o wiele więcej klas lub metod niż te, które wymieniamy tutaj. Z kolejnymi wersjami wprowadzane są również rozmaite poprawki lub usprawnienia w samym działaniu JVM albo przykładowo Garbage Collectora (w tym przypadku mogą to być, chociażby różne algorytmy, o których działanie oparty jest GC). Zmianom mogą ulegać również kwestie dotyczące zarządzania pamięcią. Oprócz tego kolejne wersje Javy mogą również wprowadzać dodatkowe narzędzia, które programista może wykorzystywać w swojej pracy. W obrębie tych materiałów poruszamy tylko te kwestie, które są adekwatne do naszego poziomu zaawansowania jako Java developerów. Nie poruszamy też zagadnień, co do których twórcy Zajavki uznali, że z naszego punktu widzenia zmiany te nie są aż tak istotne i lepiej poświęcić ten sam czas na skupienie się na dalszych zagadnieniach.

Jeżeli natomiast interesuje Cię, jakie jeszcze zmiany są wprowadzane z każdą wersją - wystarczy, że wpiszesz w Google np. "Java 10 features" i znajdziesz dużo artykułów opisujących wprowadzone zmiany. Możesz również zerknąć na tę stronę JDK 10. Zaznaczam jednak, że wiele funkcjonalności będzie niezrozumiałych. ☺