Static

Spis treści

Zapiski prowadzącego Karola Rogowskiego i uczestnika Bootcampu Zajavka Bartek Borowczyk aka Samuraj Programowania.

Co to i po co to?

Nareszcie się doczekaliśmy. Cały czas pojawia się to słówko (`static`) w materiałach i za każdym razem wspominamy, że zostanie wyjaśnione później ☺. Ten moment właśnie nastąpił.

Samo słówko `static` oznacza w rozumieniu obiektowym, że pole, bądź metoda, która jest statyczna, nie należy do obiektu klasy, który tworzymy poprzez słówko new, tylko do samej klasy. Innymi słowy, dane pole nie należy do instancji, tylko należy do typu.

Pole statyczne

Nie ma to znaczenia, ile instancji danej klasy utworzymy, wszystkie będą miały tę samą wartość pola, które jest statyczne. Jeżeli będziemy mieli klasę, która jest zdefiniowana następująco:

```
public class Student {

    private String name;

    public Student(final String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public static void main(String[] args) {
        Student student1 = new Student("Stefan");
        Student student2 = new Student("Agnieszka");
        Student student3 = new Student("Roman");
        Student student4 = new Student("Bartek");
        Student student5 = new Student("Karolina");
        System.out.println(student1.getName());
        System.out.println(student2.getName());
        System.out.println(student3.getName());
        System.out.println(student4.getName());
        System.out.println(student5.getName());
    }
}
```

Będzie to oznaczało, że możemy utworzyć 5 instancji studentów, każdego nazwać inaczej i faktycznie każdy z nich będzie nosił inne imię. Jeżeli jednak przy definicji pola dodalibyśmy słówko static i kod wyglądałby tak:

```
public class Student {

    private static String name;

    // reszta pliku
}
```

I wykonalibyśmy tę samą metodę main co powyżej, to nagle się okaże, że każdy student nazywa się Karolina. Dlaczego?

Pole statyczne jest współdzielone przez wszystkie obiekty, które utworzyliśmy na podstawie danej klasy. Przy każdym wywołaniu konstruktora `Student` ustawiamy wartość pola statycznego ponownie i wartość ta jest w każdym obiekcie taka sama (właściwość pola statycznego). Skoro jest taka sama i dojdziemy do wywołania konstruktora z imieniem "Karolina", to każdy obiekt, który już istnieje, będzie miał zmienione imię na "Karolina". Tak jak wcześniej każdy obiekt miał zmieniane imię na "Agnieszka", "Roman", czy "Bartek".

Z racji, że już o tym wiemy, widać, że jest to o tyle niebezpieczny mechanizm, że jak nie rozumiemy, co on robi, to można sobie zrobić kuku. Jeżeli natomiast rozumiemy, co on robi, to jest to bardzo przydatne w kwestiach, w których chcemy zapewnić, żeby w naszym programie jakaś wartość pojawiła się tylko raz i była współdzielona przez wszystkie obiekty. Jednocześnie zmniejszy nam to zużycie pamięci, bo zarezerwujemy ją tylko raz, a nie za każdym utworzeniem obiektu. Natomiast trzeba ten mechanizm rozumieć i używać go z głową ☺.

Kolejny przykład zastosowania tego mechanizmu - możemy do klasy Student dodać pole zliczające ilość wywołań konstruktora:

```
public class Student {

    private static String name;

    private static int counter = 0;

    public Student(final String name) {
        this.name = name;
        counter++;
    }

    public String getName() {
        return name;
    }

    public static void main(String[] args) {
        Student student1 = new Student("Stefan");
        Student student2 = new Student("Agnieszka");
        Student student3 = new Student("Roman");
        Student student4 = new Student("Bartek");
        Student student5 = new Student("Karolina");
        System.out.println(Student.counter); // 5
    }
}
```

Jak już widzisz, odnieśliśmy się do pola statycznego nie przez obiekt, który został utworzony, a przez klasę. Odnosimy się do pola counter w sposób Student.counter, gdyż tak jak mówiłem, jest to właściwość klasy, a nie obiektu. Oczywiście można odnosić się do tych pól przez stworzony obiekt, ale nie jest to zalecane.

Do zapamiętania:

- pola statyczne mogą być używane bez tworzenia obiektu, bo przynależą do klasy, a nie do obiektu,
- skoro przynależą do klasy, to odwołujemy się do nich poprzez nazwę klasy, a nie przez referencję do obiektu,
- pola statyczne są niezależne od obiektu, bo przynależą do klasy,
- wartości pól statycznych są współdzielone między obiektami.

Metoda statyczna

Podobnie jak pola, metody statyczne należą do klasy, a nie do obiektu, który na podstawie tej klasy zostanie utworzony. Oznacza to, że wykonujemy te metody bez tworzenia obiektów, metoda taka jest wykonana bezpośrednio na klasie. Czyli innymi słowy, jeżeli mamy jakąś metodę, która ma być wykonywana na zmiennych statycznych, a jej efekt niezależny od instancji klasy, to taką metodę definiujemy jako statyczną. Przykład:

```
public class Student {

    private static int counter = 0;

    public Student() {
        counter++;
    }

    public static void changeCounter(int counter) {
        Student.counter = counter;
    }

    public static void main(String[] args) {
        Student student1 = new Student();
        Student student2 = new Student();
        Student student3 = new Student();
        Student student4 = new Student();
        Student student5 = new Student();

        System.out.println(Student.counter);
        Student.changeCounter(12);

        System.out.println(Student.counter);
        System.out.println(student3.counter);
        System.out.println(student5.counter);
    }
}
```

Pole `counter` jest statyczne, tak samo, jak metoda `changeCounter()`. Wywołując ją zmieniamy wartość zmiennej `counter` na poziomie klasy `Student`, czyli wszystkie instancje klasy `Student` będą widziały tę samą wartość. Obrazuje to fragment kodu:

```
System.out.println(Student.counter);
System.out.println(student3.counter);
System.out.println(student5.counter);
```

W pierwszej linijce odwołujemy się do zmiennej statycznej poprzez klasę i dostajemy wartość `12`. Następnie odwołujemy się do tej samej zmiennej w sposób niezalecany (ale kompilujący się), czyli przez referencję utworzonego obiektu i również dostajemy `12`. To pokazuje ponownie, że zmiana pola statycznego jest propagowana we wszystkich obiektach.

Ważne aspekty używania `static`:

- metody statyczne nie mogą odwoływać się do pól instancyjnych klasy. Pole instancyjne jest inicjowane na etapie tworzenia obiektu i istnieje tylko razem z obiektem. Metoda statyczna przynależy do klasy, zatem próba odwołania się z metody statycznej do pola instancyjnego jest jak odwołanie się do pola, które jeszcze nie istnieje i dlatego powoduje błąd kompilacji,
- w metodach static nie możemy używać słówka this. Powód jest ten sam, metoda statyczna należy do klasy, a nie do obiektu. W metodzie statycznej nie mamy kontekstu obiektu, na którym możemy wywołać this, więc nie możemy tego zrobić. Użycie this w metodzie statycznej powoduje błąd kompilacji.

Do zapamiętania:

- metody statyczne nie mogą zależeć od obiektu, na którym operują, dlatego nie można z metod statycznych odwołać się ani do pól, ani metod niestatycznych (instancyjnych),
- metody statyczne często są używane do szybkiego policzenia czegoś, przykładowo sumy lub różnicy 2 liczb,
- metody statyczne nie mogą być `@Override`, bo jest to część polimorfizmu. Polimorfizm oznacza użycie kontekstu obiektowego, którego w przypadku static nie mamy, bo operujemy na klasach,
- metody abstrakcyjne nie mogą być statyczne, bo abstract oznacza, że metoda ma być `@Override`, a statyczna przecież nie może,
- metody statyczne nie mogą używać słówek `this` ani `super`, bo znowu, byłby to kontekst obiektu, a static działa na klasach.

Warto zapamiętać jakie kombinacje wywołania metod są dozwolone:

- metody statyczne nie mogą odwoływać się bezpośrednio do pól lub metod instancyjnych. Możemy natomiast w metodzie statycznej stworzyć obiekt i na nim operować. Cały czas zresztą robimy tak z `main()`, która przecież jest statyczna. Tworzymy w niej obiekty i operujemy na nich,
- metody statyczne mogą odwoływać się do innych metod statycznych i pól statycznych,
- metody instancyjne mogą odwoływać się do pól i metod statycznych,
- metody instancyjne mogą odwoływać do innych pól i metod instancyjnych.

Metody statyczne w interfejsach

W Javie 8 została wprowadzona możliwość definiowania metod statycznych w interfejsach. Taka metoda musi wtedy zawierać kompletne ciało, nie może być zdefiniowana jako `public abstract` tak jak inne metody w interfejsach. Z racji, że taka metoda jest statyczna, nie możemy jej też później `@Override`.