Tablice

Spis treści

Zapiski uczestnika Bootcampu Zajavka.pl w 12 tygodni by Bartek Borowczyk aka Samuraj Programowania. Dopiski na zielono od Karola Rogowskiego.

Czym są tablice?

Czym są tablice? Tablice to obiekty, które służą do przechowywania uporządkowanego zbioru danych i pomagają w pracy z nimi. Tak, zgadza się, nie zawsze wystarczą nam zmienne ☺.

Co warto wiedzieć o tablicach:

- przechowują ten sam typ elementów. Czyli mamy tablicę `String`ów, `int`ów, itd. Po zadeklarowaniu typu tablicy (czyli typu przechowywanych danych) nie możemy już później tego zmienić. Oczywiście tablica może przechowywać też inne tablice (tablica tablic) czy obiekty.
- każda wartość przechowywana w tablicy ma swój indeks, liczony od zera.
- tablica jest obiektem.
- tablica ma określoną w chwili tworzenia wielkość (długość) i tej wielkości nie można potem zmienić.

Deklaracja zmiennych tablicowych

Tworząc zmienne, które będą przechowywać tablice (w istocie będą przechowywać referencje do obiektu tablicy) musimy skorzystać z określonego w Javie schematu.

Schemat deklaracji zmiennej tablicowej:

```
typPrzechowywanychDanych[] nazwaTablicy;
```

np.

```
String[] usersName;
int[] usersAge;
float[] usersWeight;
```

Warto zwrócić uwagę na to, że można też stworzyć tablice, trochę zmieniając ten schemat - poprzez przeniesienie nawiasów kwadratowych na koniec.

```
String usersName[];
int usersAge[];
```

To też jest prawidłowy sposób, ale jest on rzadziej spotykany i odradzany.

Tworzymy tablicę (obiekt tablicy)

Istnieje kilka sposobów na to, by stworzyć tablicę. Oczywiście możemy połączyć także deklarację zmiennej z tworzeniem tablicy.

Użycie operatora new

Przekonasz się wkrótce, że obiekty (a takim jest tablica) tworzymy w Javie za pomocą operatora `new`. Nie inaczej jest przy tablicy.

Sposób 1: tworzymy tablicę określając tylko jej wielkość

```
int[] age = new int[5];
```

Sposób 2: tworzymy tablicę wskazując, jakie ma mieć elementy. W zależności od tego ile ich wymienimy w nawiasach klamrowych, tak wielka (długa) będzie tablica.

```
double[] weight = new double[] { 2.22, 2, 40.11112 };
```

Zwróć uwagę, że w sposobie 1 zazwyczaj chcemy potem dodać jakieś wartości do tablicy. Zobaczmy przykładowe.

```
int[] age = new int[5];
// odwołujemy się do indeksu nazwa tablic i jej indeks definiuje element.
// W tablicy z 5 elementami mamy 5 indeksów: 0, 1, 2, 3, 4.
// Indeksy w tablicach zawsze zaczynają się od 0.
age[0] = 10;
age[1] = 11;
age[2] = 12;
age[3] = 13;
age[4] = 14;
```

To samo co powyżej możemy zautomatyzować za pomocą poznanej już pętli.

```
int[] age = new int[5];
for (int i = 0; i < age.length; i++) {
    /*  .length - to pole (właściwość) każdej tablicy,
        w których przechowywana jest długość tablicy.
        W naszym wypadku w praktyce mamy więc i < 5.
        Co oznacza, że iteracja powinna wykonać się 5 razy.
        To częsty zapis przy pracy z tablicami, bo jest to zapis uniwersalny,
        niezależny od wielkości tablicy. */
    age[i] = i + 10;
    /*  przy pierwszej iteracji i będzie równe 0 a więc:
        age[0] = 0 + 10 - czyli do age[0] przypisane będzie 10
        przy drugiej iteracji będzie już:
        age[1] = 1 + 10 - czyli do age[1] przypisane będzie 11. ITD */
}
```

Stworzenie tablicy za pomocą literału (w nawiasach klamrowych)

```
String[] cars = { "Opel", "Audi", "Fiat" };
```

Od razu w nawiasach klamrowych.

I przetestujemy czy wiesz już, o co chodzi:

```
cars.length // zawiera 3, wiesz dlaczego?
cars[1] // zawiera Audi, wiesz dlaczego?
```

Próba odczytania `cars[3]` w naszym wypadku spowoduje błąd, bo tablica nie ma elementu o indeksie `3`, a więc `4` elementu.

Pętla for do iteracji po tablicy

Opcja 1: for i tablica.length

Zapamiętaj schemat:

```
for (int i = 0; i < a.length ; i++)
```

Jak działa ten sposób iteracji po tablicy widzieliście już powyżej, ale zobaczmy jeszcze raz.

```
// age {20,30,12,90,1}
for (int i = 0; i < age.length; i++) {
    System.out.println(age[i]);
}
// Wydrukuje (w nowych liniach): 20 30 12 90 1
```

W ten sposób możemy pracować z każdym elementem tablicy.

Opcja 2: for i (item : items)

Możemy wykorzystać też pętle typu `foreach`. Zobaczmy schemat.

```
for(typElementu elementTablicy : tablica) {
    // ciało pętli
}
```

Tu już nie mamy indeksu, tylko wprost działamy na elemencie tablicy. Zobaczcie przykład, po którym zrozumiemy jak działa ten mechanizm pętli `for`.

```
String[] cars = { "Opel", "Audi", "Fiat" };

for(String car : cars) {
    System.out.println(car);
    // pod car przy każdej iteracji będzie znajdowała się inna, kolejna wartość tablicy.
    // W tym przypadku pętla wykona się 3 razy, bo mamy trzy elementy w tablicy.
}
// Wydrukuje (w nowych liniach): Opel Audi Fiat
```

Zobaczmy nawias po `for`. Po lewej stronie dwukropka podajemy naszą nazwę i typ danych, które przechowywane są w tablicy. W naszym wypadku mamy tablicę `String`ów, która musi przechowywać... `String`i ☺. Tak więc typ danych to `String`. Nazwa, której używamy jest dowolna, ale często jest ona liczbą pojedynczą słowa określające tablicę. Np. `człowiek: ludzie;`, `liczba : liczby`, `wiek : wieki` ☺. W naszym wypadku, skoro tablica to `cars`, to możemy użyć `car`, ale pamiętaj, że to tylko konwencja. Po prawej stronie wskazujemy tablicę, na której pracujemy.

Jaką mamy liczbę iteracji w takiej pętli? Taką samą jak liczbę elementów w tablicy. I nie musimy tego wiedzieć, bo program sam sobie poradzi. Najważniejszy mechanizm tutaj polega na tym, że przy każdej interakcji zmienna (`car` w naszym przykładzie) będzie zawierała referencje do innego elementu tablicy, czyli przy każdej iteracji będziemy pracować z innym elementem.