

Operacje na tablicach - klasa Arrays

Czego się dowiesz

- Jakie problemy możesz napotkać pracując z tablicami,
- jak wykorzystać klasę Arrays do podstawowych operacji na tablicach.

Wstęp

Pisząc dowolne programy bardzo często wykorzystuje się w nich tablice do przechowywania kolekcji danych. Sprawia to, że niemal w każdym projekcie musielibyśmy definiować metody, które pozwolą nam je sortować, kopiować, wyświetlać czy wyszukiwać w nich jakieś informacje i byłoby to co najmniej męczące.

Na szczęście w Javie niemal na początku jej istnienia powstał pakiet `java.util`, a w nim wygodna klasa **Arrays** do wykonywania podstawowych operacji na tablicach, które wymieniono powyżej. Przykładowe metody, które tam znajdziemy to:

- **sort()** - sortuje elementy zgodnie z ich naturalnym porządkiem,
- **binarySearch()** - wyszukuje w tablicy podaną wartość, za pomocą wyszukiwania binarnego. Wymaga, aby tablica była posortowana (np. metodą `sort()`),
- **copyOf()** - pozwoli skopiować całość lub fragment tablicy, do innej tablicy. W przypadku typów obiektowych kopiowane są referencje (nie są tworzone kopie obiektów), więc modyfikując obiekty w jednej tablicy, zmodyfikujemy także obiekty w utworzonej kopii,
- **equals()** - porównuje równość tablic, jednak nie jest sprawdzana równość strukturalna poszczególnych obiektów, a jedynie równość ich referencji. Jeśli chcesz sprawdzić równość strukturalną dwóch tablic, wykorzystaj metodę `deepEquals()`,
- **fill()** - pozwala wypełnić całą tablicę jedną, podaną wartością.

Niektóre z wymienionych metod posiadają tylko wersję generyczną dla tablic typów obiektowych, ale niektóre posiadają też przeciążone wersje dla tablic typów prostych.

Poniżej pokazana jest klasa, w której przedstawiono operacje za pomocą powyższych metod na różnych typach tablic.

ArraysExample.java

```
import java.util.Arrays;

class ArraysExample {
    public static void main(String[] args) {
        String[] names = {"Basia", "Kasia", "Wojtek", "Agnieszka", "Kacper"};
        Integer[] numbers = {4, 7, 2, 1, 14, 23, 10, 5};

        System.out.println("Names: ");
        printArray(names);

        System.out.println("Numbers: ");
        printArray(numbers);
        System.out.println();

        //sortowanie
        System.out.println("Arrays.sort(numbers): ");
        Arrays.sort(numbers);
        printArray(numbers);

        System.out.println("Arrays.sort(names): ");
        Arrays.sort(names);
        printArray(names);
        System.out.println();

        //copyOf
        System.out.println("Numbers2, Arrays.copyOf()");
        Integer[] numbers2 = Arrays.copyOf(numbers, numbers.length);
        printArray(numbers2);
        System.out.println();

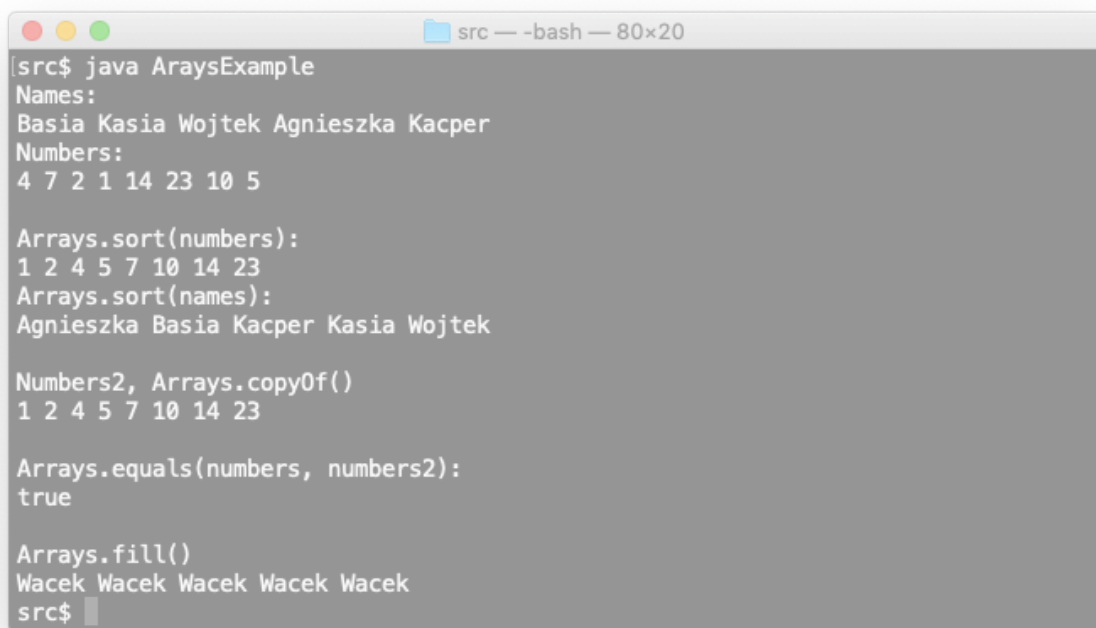
        //equals
        System.out.println("Arrays.equals(numbers, numbers2): ");
        System.out.println(Arrays.equals(numbers, numbers2));
        System.out.println();

        //fill
        System.out.println("Arrays.fill()");
        String[] strings = new String[5];
        Arrays.fill(strings, "Wacek");
        printArray(strings);
    }

    public static <T> void printArray(T[] arr) {
        for(T t: arr) {
            System.out.print(t + " ");
        }
        System.out.println();
    }
}
```

Powyższy przykład pokazuje zastosowanie dwóch typów tablic - `Integer[]` i `String[]`. Używamy typu `Integer[]`, a nie `int[]`, dzięki czemu mogliśmy przekazać ją jako argument generycznej metody `printArray()` wyświetlającej kolejne elementy tablicy.

Metody klasy `Arrays` nie zwracają wyniku (są typu `void`), modyfikowana jest oryginalna tablica. Ponieważ jest to operacja nieodwracalna, to lepiej zastanowić się przed wykorzystaniem którejkolwiek z zaprezentowanych funkcjonalności.



```
src — -bash — 80x20
[src$ java AraysExample
Names:
Basia Kasia Wojtek Agnieszka Kacper
Numbers:
4 7 2 1 14 23 10 5

Arrays.sort(numbers):
1 2 4 5 7 10 14 23
Arrays.sort(names):
Agnieszka Basia Kacper Kasia Wojtek

Numbers2, Arrays.copyOf()
1 2 4 5 7 10 14 23

Arrays.equals(numbers, numbers2):
true

Arrays.fill()
Wacek Wacek Wacek Wacek Wacek
src$
```

Istnieje jeszcze jedna metoda kopiująca elementy z jednej tablicy do drugiej, jednak znajduje się ona w klasie `System`.

Statyczna metoda `System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length)` przyjmuje 5 parametrów:

- tablicę źródłową (`src`),
- indeks od którego chcemy skopiować elementy z tablicy źródłowej (`srcPos`),
- tablicę, do której kopiujemy (`dest`),
- indeks od którego mają być wstawiane skopiowane elementy w tablicy wynikowej (`destPos`),
- ilość elementów do skopiowania (`length`).

W rzeczywistości metoda `Arrays.copyOf()` wykorzystuje metodę `System.arraycopy()` pod spodem.

ArraysCopyExample.java

```
class ArraysCopyExample {  
    public static void main(String[] args) {  
        int[] numbers1 = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  
  
        int[] numbers2 = new int[numbers1.length];  
  
        System.arraycopy(numbers1, 4, numbers2, 0, numbers1.length - 4);  
  
        for (int num : numbers2) {  
            System.out.print(num + " ");  
        }  
    }  
}
```

W ten prosty sposób do tablicy `numbers2` skopiowaliśmy elementy z tablicy `numbers1`, zaczynając od indeksu 4 włącznie i wstawiając je na miejsca od indeksu 0 w tablicy `numbers2`. Liczba skopiowanych elementów to `number1.length-1`.