Przewodnik 3 – praca z tablicami w języku JAVA - podstawy.

dr inż. Łukasz Sosnowski
WIT Wyższa Szkoła Informatyki Stosowanej i Zarządzania
pod auspicjami Polskiej Akademii Nauk

1 Utworzenie klasy

W pakiecie **pl.wit.lab2** utwórz klasę o nazwie: **Lab2ArraysExample**. Dodaj komentarz wieloliniowy dla klasy: „Klasa z przykładami metod operującymi na tablicach”

2 Zadeklarowanie zmiennych składowych klasy

Zadeklaruj w klasie zmienną prywatną o nazwie „intNumbers” która będzie tablicą jednowymiarową typu int, np. w postaci:

```
private int intNumbers[]=null;
```

Zadeklaruj drugą zmienną prywatną o nazwie „stringElements”, która będzie tablicą jednowymiarową typu String.

Zadeklaruj trzecią zmienną prywatną o nazwie „boolElements”, która będzie tablicą jednowymiarową typu boolean.

Dla wyżej zadeklarowanych zmiennych wygeneruj automatycznie gettery (Source → Generate Getters and Setters...)

3 Konstruktory

Zdefiniuj konstruktor prywatny bezparametryczny. Zaimplementuj go w taki sposób aby zainicjować wszystkie zmienne klasy pustymi tablicami;

private `Lab2ArraysExample (){`

```
//ciało konstruktora
intNumbers = new int[0];
...
}
```

Zdefiniuj 3 konstruktory parametryczne jednoargumentowe, dla następujących typów argumentów: *int*, *String*, *boolean*. Zaimplementuj je w taki sposób aby każdy z tych konstruktorów wywołał konstruktor prywatny bezparametrowy a następnie poszczególne typy argumentów zasiliły pierwszy element odpowiedniej tablicy, np. dla typu int w następujący sposób:

```
public Lab2ArraysExample(int intValue) {
		this();
		intNumbers = Arrays.copyOf(intNumbers, 1);
		intNumbers[0] = intValue;
	}
```

Użyj do implementacji klasy Arrays oraz metody statycznej „copyOf”.

4 Metody klasy

a. Zaimplementuj metodę publiczną bezparametrową *printArraysToLog* nie zwracającą wartości, która będzie wypisywać do logu w sposób bezpieczny w trybie „info” zawartość wszystkich tablic klasy.

Do tego celu powołaj w klasie zmienną:

```
protected static final Logger log =
LogManager.getLogger(Lab2ArraysExample.class.getName());
```

Do implementacji użyj instrukcji „*if*” do sprawdzenia oraz pętli rozszerzonej „*for*”. Dla pierwszej z tablic implementacja może wyglądać następująco:

```
if(intNumbers!=null) {
	for(int el:intNumbers)
		log.info(""+el+", ");
}...
```

Dla pozostałych tablic obsługę należy zrobić analogicznie.

b. Zaimplementuj 3 metody publiczne nie zwracające wartości o nazwie *addElement* przyjmująca odpowiednio parametry: *int*, *String* i *boolean*. Metody z parametrami odpowiednich typów powinny dodawać przekazany element na koniec właściwej tablicy.

Dla pierwszej metody kod może wyglądać następująco:

```
public void addElement(int element) {
		intNumbers = Arrays.copyOf(intNumbers, intNumbers.length+1);
		intNumbers[intNumbers.length - 1] = element;
}
```

c. Zaimplementuj trzy metody publiczne bezparametrowe zwracające wymiary poszczególnych tablic o nazwach: `getIntArraySize`, `getStringArraySize`, `getBooleanArraySize`

Każda z metod ma zwracać rozmiar odpowiedniej tablicy. Dla tablicy typu int implementacja może wyglądać następująco:

```
public int getIntArraySize() {
		if(intNumbers!=null)
			return intNumbers.length;
		else
			return -1;
	}
```

Wykonaj analogiczną implementację dla pozostałych tablic klasy.

d. Zaimplementuj trzy metody publiczne zwracające zadany element odpowiedniej tablicy o następujących sygnaturach: `public int getIntArrayElement(int index), public String getStringArrayElement(int index), public boolean getBooleanArrayElement(int index)`

Implementacja pierwszej metody może wyglądać następująco:

```
public int getIntArrayElement(int index) {
	if(intNumbers!=null && index<intNumbers.length && index >=0) {
		return intNumbers[index];
	}else {
		return -1;
	}
}
```

Kolejne dwie metody zaimplementuj analogicznie.

5 Test jednostkowy

W pakiecie **pl.wit.lab2** katalogu src/test/java utwórz klasę testu jednostkowego o nazwie: **Lab2ArraysExampleTest**.

Zaimplementuj metody testowe dla wykonanych metod w pkt. 4 (b,c,d)

Test pierwszej metody może wyglądać następująco:

```
@Test
public void addElementTest() {
	Lab2ArraysExample arrEx = new Lab2ArraysExample(1);
	Assert.assertArrayEquals(new int[] {1}, arrEx.getIntNumbers());

	arrEx.addElement(true);
	Assert.assertEquals(Arrays.toString(new boolean[] {true}),
	Arrays.toString(arrEx.getBoolElements()));

	arrEx.addElement("Test");
	Assert.assertArrayEquals(new String[] {"Test"}, arrEx.getStringElements());
}
```

Testy kolejnych metod wykonaj analogicznie.