Java blocks/body

Mindaugas Karpinskas
2017

Prisiminti

- *Apsirašant masyvą galima iškart priskirti masyvo elementų reikšmes*
- Objekto/Klasės inicijavimo blokai: {}, static { }

Declare And Initialize Java Array In One Statement

Initialize Java Array Using Assignment

```
public class TestArray {

    public static void main(String[] args) {
     int[] testArray = new int[4];
     testArray[0] = 2;
     testArray[1] = 3;
     testArray[2] = 5;
     testArray[3] = 7;
    }

}
```

Declare And Initialize Java Array In One Statement

Initialize Java Array In One Statement

```
public class TestArray {

        public static void main(String[] args) {
         int[] testArray = {5, 7, 11, 13, 17};
         int[][] testArray2 = {{5, 7}, {11, 13}, {1, 17}};
        }

}
```

Declare And Initialize Java Array In One Statement

Initialize Java Array In One Statement

```
public class TestArray {

        public static void main(String[] args) {
         int[] testArray = {5, 7, 11, 13, 17};
         int[][] testArray2 = {
                                    {5, 7},
                                    {11, 13},
                                    {1, 17}       };
        }

}
```

Declare And Initialize Java Array In One Statement

Initialize Java Array Using Loops

```
public class InitializeJavaArray {
	public static void main(String[] args) {
		int[] testArray = new int[10];
		for (int i = 0; i < testArray.length; i++) {
			testArray[i] = i + 1;
		}
		System.out.println(Arrays.toString(testArray));
	}
}
```

http://javadevnotes.com/java-initialize-array-examples

Declare And Initialize Java Array In One Statement

Initialize Java Array With Same Value

```
import java.util.Arrays;

public class TestArray {

        public static void main(String[] args) {
         int[] testArray = new int[4];
         Arrays.fill(testArray, 50);
         System.out.println(Arrays.toString(testArray));
        }

}
```

Initialize Java Array From Console

```
class InitializeJavaArray {
	public static void main(String[] args) {
		int[] testArray = new int[10];
		for (int i = 0; i < 10; i++) {
			String stringValue = System.console().readLine("Enter number: ");
			testArray[i] = Integer.parseInt(stringValue);
		}
		System.out.println(Arrays.toString(testArray));
	}
}
```

http://javadevnotes.com/java-initialize-array-examples

Užduotis Array1 prisiminkim...

- Sukurkite programą, kuri paprašytų įvesti savaitės dienos pavadinimą ir išspausdintų:
  - kelinta savaitės diena
  - jei savaitės dienos pavadinimas netinkamas, programa turi paprašyti vėl įvesti savaitės dieną
  - panaudokite savaitės dienų masyvą, elementus priskirkite inicijuojant masyvą
- Masyvą inicijuoti:
  - Deklaruojant kintamąjį
  - Po viena elementą

Dvimatis masyvas

Masyvas 4x4 int tipo.

1. Užpildome jį naudodami for, for ir random.nextInt(10). Išspausdiname rezultatą
2. Užpild. inicijavimo metu (random.nextInt(10)). Išspausdinti rezultatą
3. Užpild. for Arrays.*fill*(m, random.nextInt(10)). Išspausdinti rezultatą.

5	8	9	4
4	6	4	5
5	6	7	1
7	6	5	4

```
public class DvimatisMasyvas {
	static Random random = new Random();
	public static void main(String[] args) {
		int[][] array = new int [5][5];
		for (int i = 0; i < array.length; i++) {
			for (int j = 0; j < array[i].length; j++) {
				array[i][j] = r();
			}
		}
		print(array);
	}
	static int r() {
		return random.nextInt(10);
	}
	static void print(int[][] array) {
		for (int i = 0; i < array.length; i++) {
			System.out.println(Arrays.toString(array[i]));
		}
	}
}
```

```
public class DvimatisMasyvas {
	static Random random = new Random();

	public static void main(String[] args) {
		int[][] array = {
				{ r(), r(), r(), r(), r() },
				{ r(), r(), r(), r(), r() },
				{ r(), r(), r(), r(), r() },
				{ r(), r(), r(), r(), r() },
				{ r(), r(), r(), r(), r() }   };
		print(array);
	}
	static int r() {
		return random.nextInt(10);
	}
	static void print(int[][] array) {
		for (int i = 0; i < array.length; i++) {
			System.out.println(Arrays.toString(array[i]));
		}
	}
}
```

```
public class DvimatisMasyvas {
    static Random random = new Random();
    public static void main(String[] args) {
        int[][] array = new int [5][5];
        for (int i = 0; i < array.length; i++) {
            Arrays.fill(array[i], r());
        }
        print(array);
    }
    static int r() {
        return random.nextInt(10);
    }
    static void print(int[][] array) {
        for (int i = 0; i < array.length; i++) {
            System.out.println(Arrays.toString(array[i]));
        }
    }
}
```

- Objekto/Klasės inicijavimo blokai: {}, static { }

Initializing Fields

As you have seen, you can often provide an initial value for a field in its declaration:

```
public class BedAndBreakfast {
    // initialize to 10
    public static int capacity = 10;
    // initialize to false
    private boolean full = false;
    // initialize new object and assign to varaible
    private BedAndBreakfast object = new BedAndBreakfast();
}
```

https://docs.oracle.com/javase/tutorial/java/javaOO/

Static Initialization Blocks

A *static initialization block* is a normal block of code enclosed in braces, `{ }`, and preceded by the `static` keyword. Here is an example:

```
static {
    // whatever code is needed for initialization goes here
}
```

A class can have any number of static initialization blocks, and they can appear anywhere in the class body. The runtime system guarantees that static initialization blocks are called in the order that they appear in the source code.

Static Initialization Blocks

```
class Whatever {
    static {

        // initialization code goes here
    }
}
```

Static Initialization Blocks

There is an alternative to static blocks — you can write a private static method:

```
class Whatever {
    public static varType myVar = initializeClassVariable();

    private static varType initializeClassVariable() {

        // initialization code goes here
    }
}
```

The advantage of private static methods is that they can be **reused** later if you need to reinitialize the class variable.

Initializing Instance Members

Normally, you would put code to initialize an instance variable in a constructor. There are two alternatives to using a constructor to initialize instance variables: initializer blocks and final methods.

Initializer blocks for instance variables look just like static initializer blocks, but without the `static` keyword:

```
{
    // whatever code is needed for initialization goes here
}
```

The Java compiler copies initializer blocks into every constructor. Therefore, this approach can be used to share a block of code between multiple constructors.

Initializing Instance Members

Here is an example of using a method for initializing an instance variable:

```
class Whatever {
    private varType myVar = initializeInstanceVariable();
    protected varType initializeInstanceVariable() {
        // initialization code goes here
    }
}
```

This is especially useful if subclasses might want to reuse the initialization method. *The method should be final because calling non-final methods during instance initialization can cause problems.*

Užduotis Blokai1

- Tegu jūsų programa išspausdina tekstą:
  - dvejuose staic {} blokuose,
  - trijuose ~~*instance*~~ {} blokuose,
  - construktoriuje
  - main metode

?