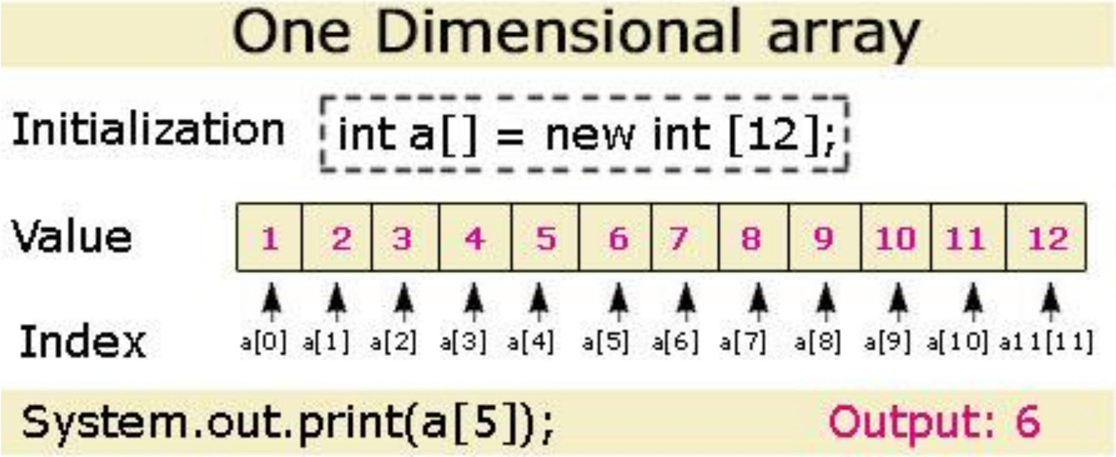
VILNIAUS TECHNOLOGIJŲ IR VERSLO PROFESINIO MOKYMO CENTRAS

6- MASYVAI

Jaroslav Grablevski / Justina Balsė

Masyvai

- Masyvai skirti saugoti dideliam kiekiui vienodo tipo reikšmių
- Masyvo elementai skaičiuojami nuo 0 (žymi pirmą elementą)
- Masyvo dydis nurodomas jį sukuriant ir nebegali kisti. Jį galima sužinoti per savybę „length“



One Dimensional array

Initialization

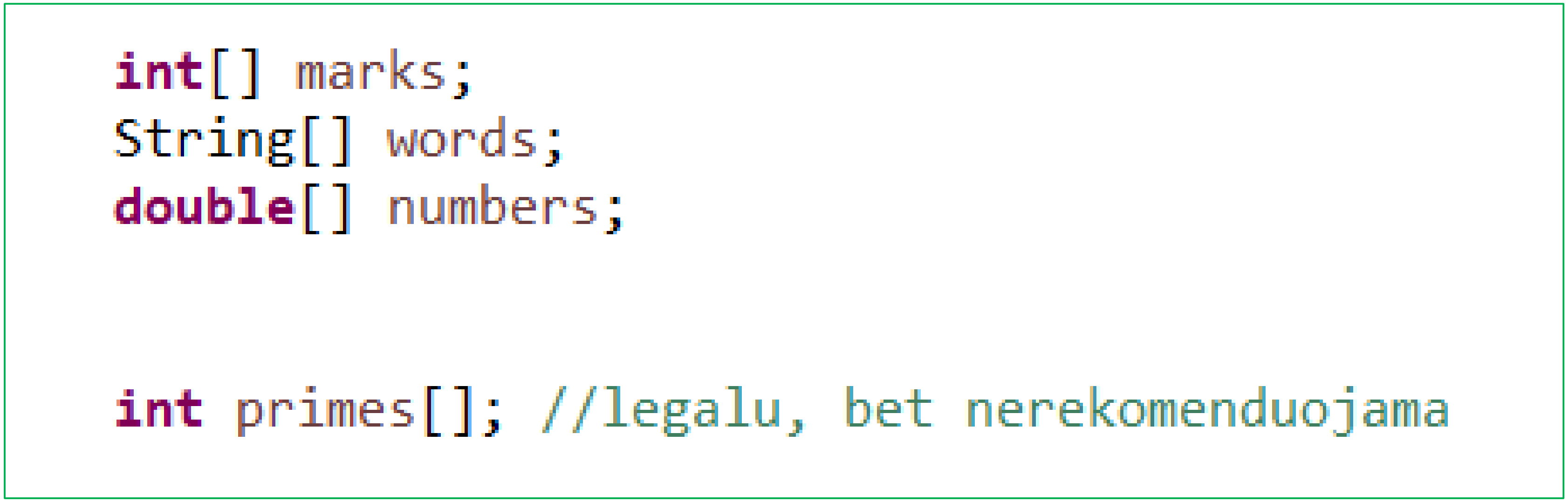
```
int a[] = new int [12];
```

Value	1	2	3	4	5	6	7	8	9	10	11	12
Index	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a11[11]

```
System.out.print(a[5]);
```

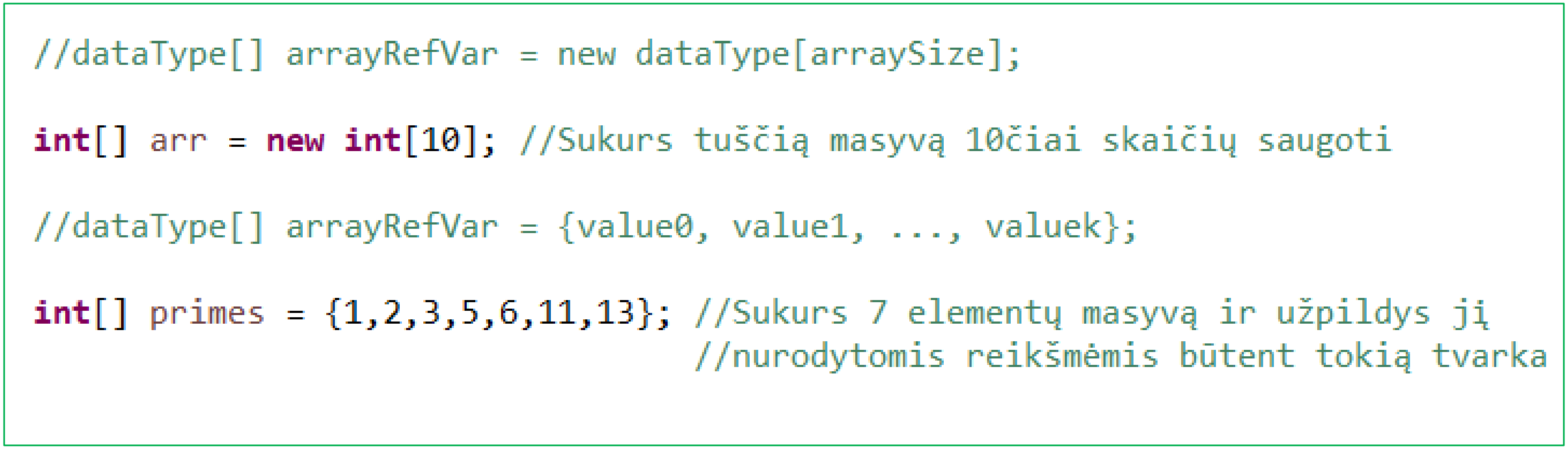
Output: 6

Deklaravimas

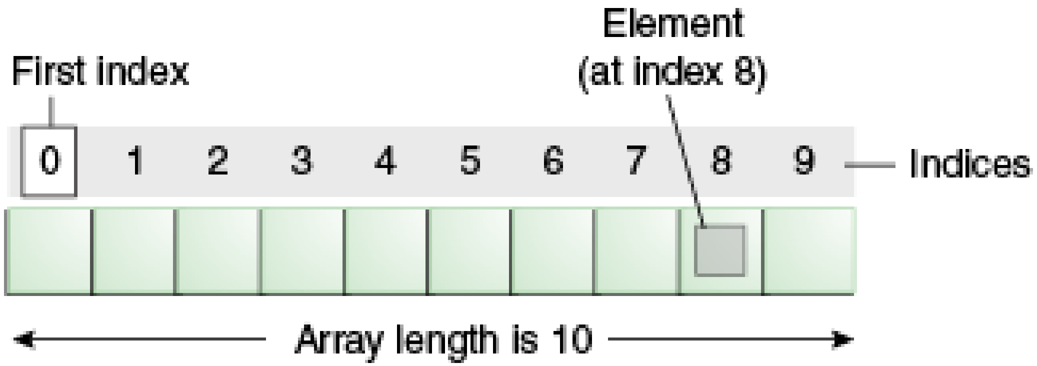


```
int[] marks;
String[] words;
double[] numbers;


int primes[]; //legalu, bet nerekomenduojama
```

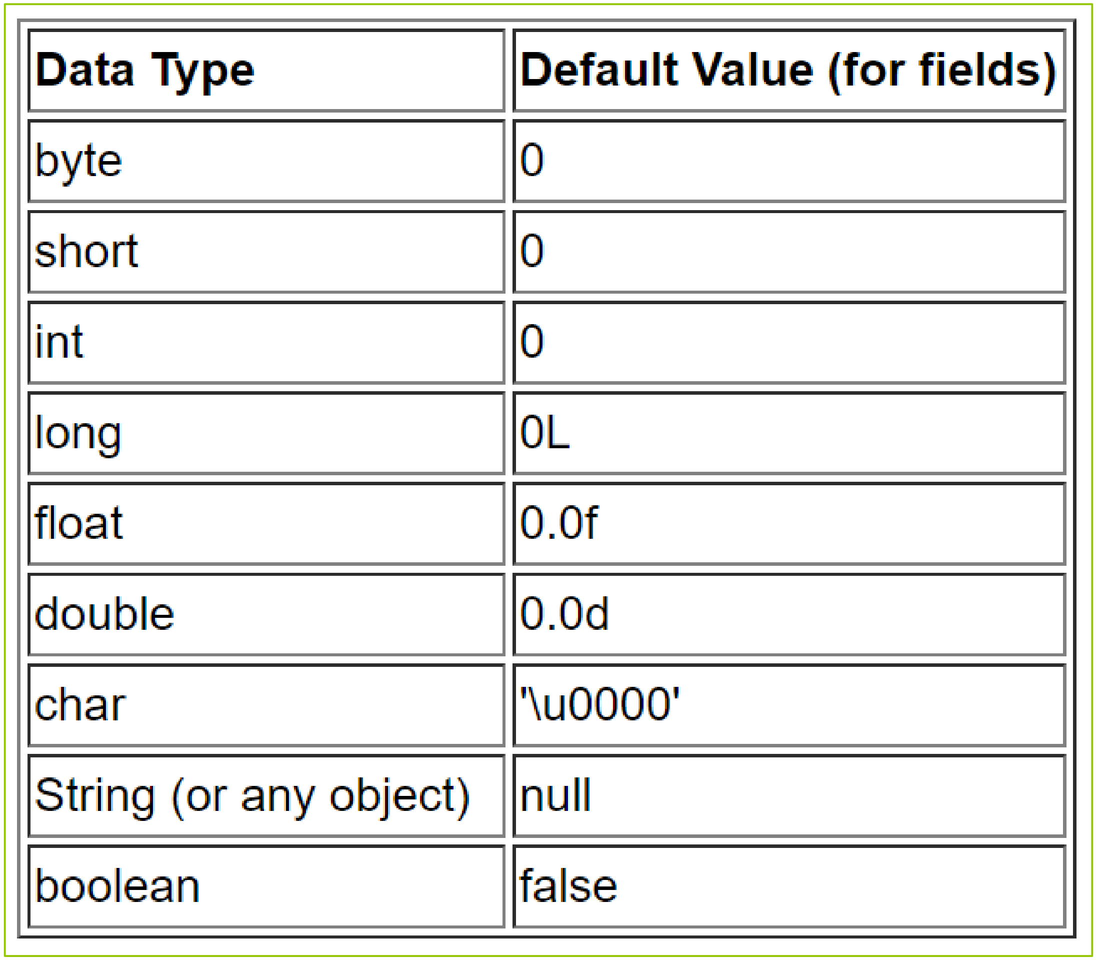
Inicializavimas



```
//dataType[] arrayRefVar = new dataType[arraySize];

int[] arr = new int[10]; //Sukurs tuščią masyvą 10čiai skaičių saugoti

//dataType[] arrayRefVar = {value0, value1, ..., valuek};

int[] primes = {1,2,3,5,6,11,13}; //Sukurs 7 elementų masyvą ir užpildys jį
                                  //nurodytomis reikšmėmis būtent tokią tvarka
```

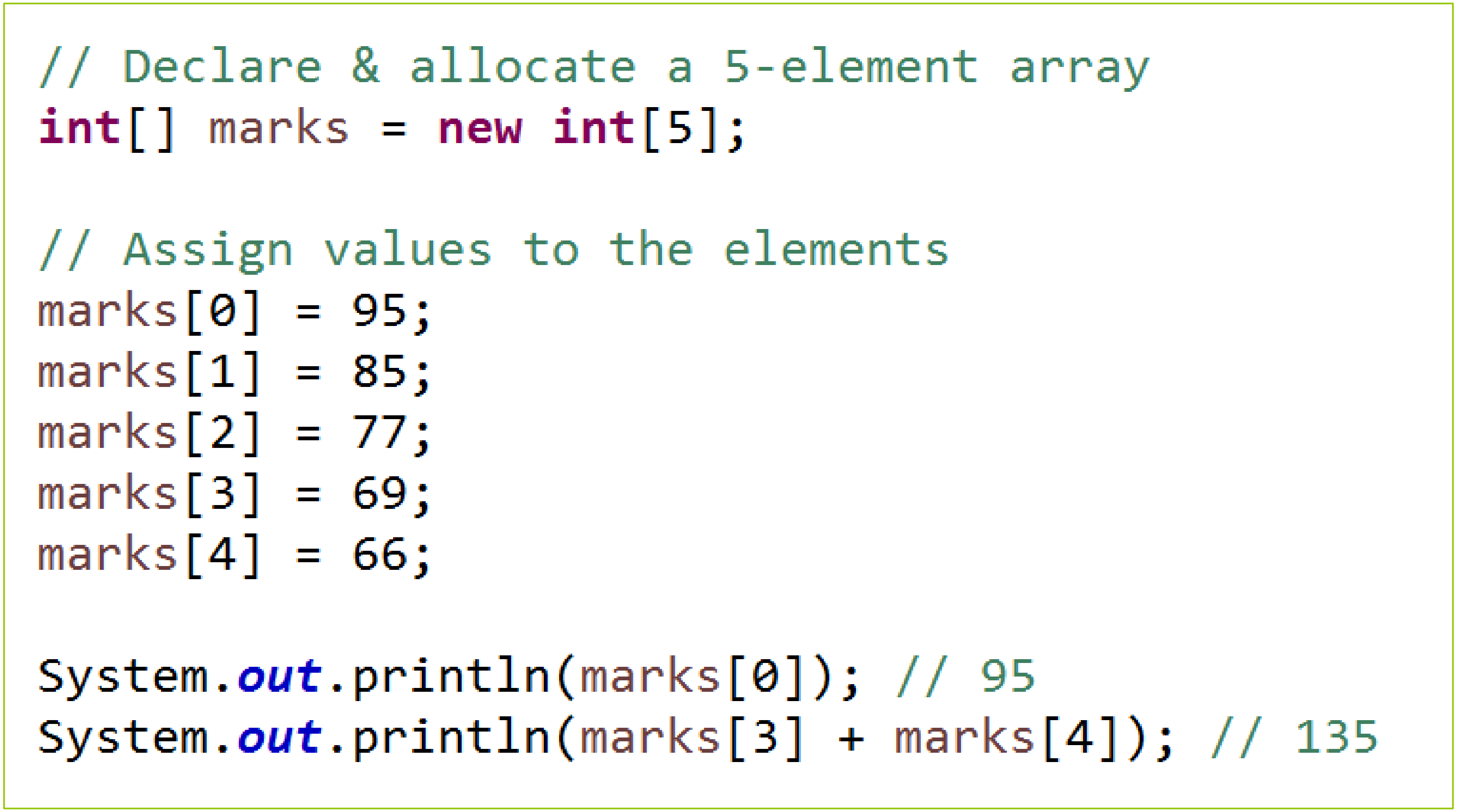


Reikšmės pagal nutylėjimą



Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Reikšmių priskyrimas

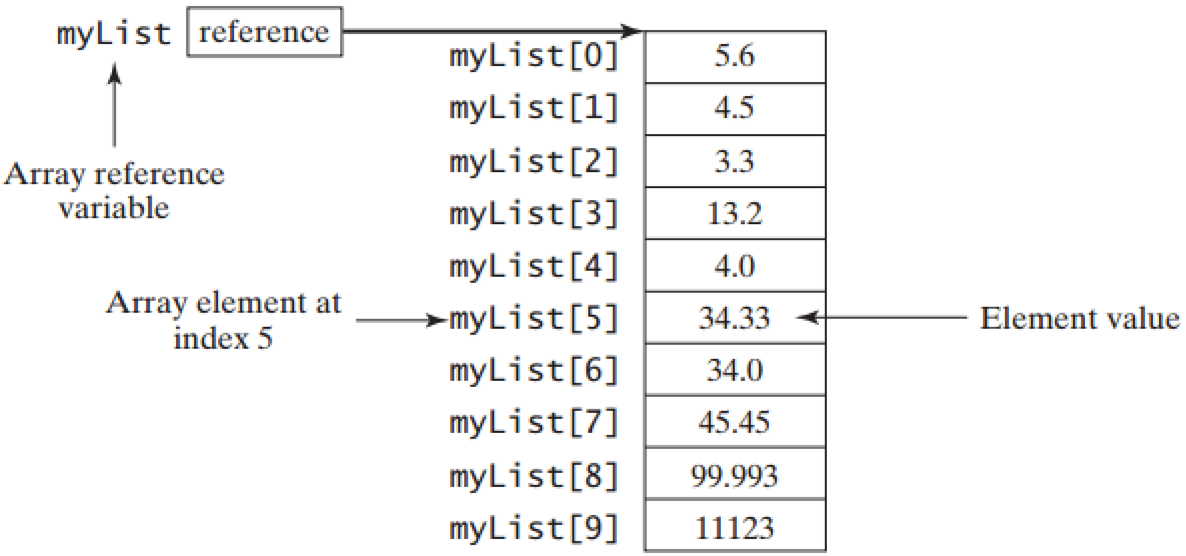


```
// Declare & allocate a 5-element array
int[] marks = new int[5];

// Assign values to the elements
marks[0] = 95;
marks[1] = 85;
marks[2] = 77;
marks[3] = 69;
marks[4] = 66;

System.out.println(marks[0]); // 95
System.out.println(marks[3] + marks[4]); // 135
```

Reikšmių priskyrimas

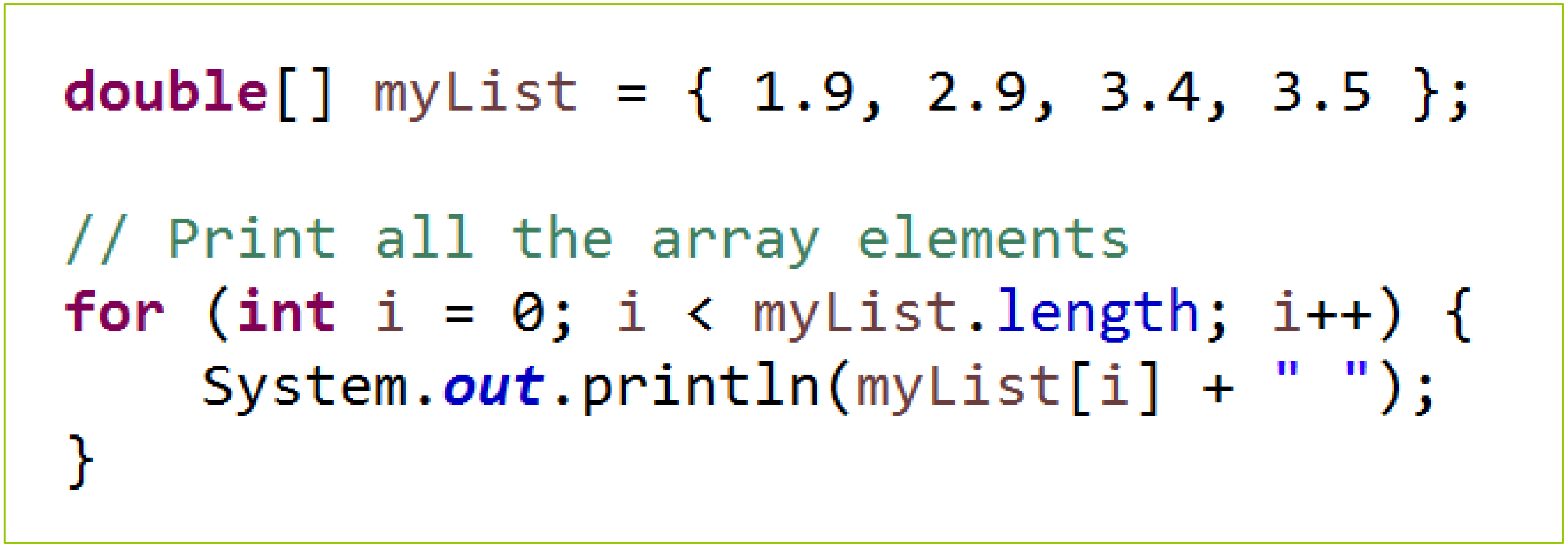
```
myList[0] = 5.6;
myList[1] = 4.5;
myList[2] = 3.3;
myList[3] = 13.2;
myList[4] = 4.0;
myList[5] = 34.33;
myList[6] = 34.0;
myList[7] = 45.45;
myList[8] = 99.993;
myList[9] = 11123;
```

```
double[] myList = new double[10];
```



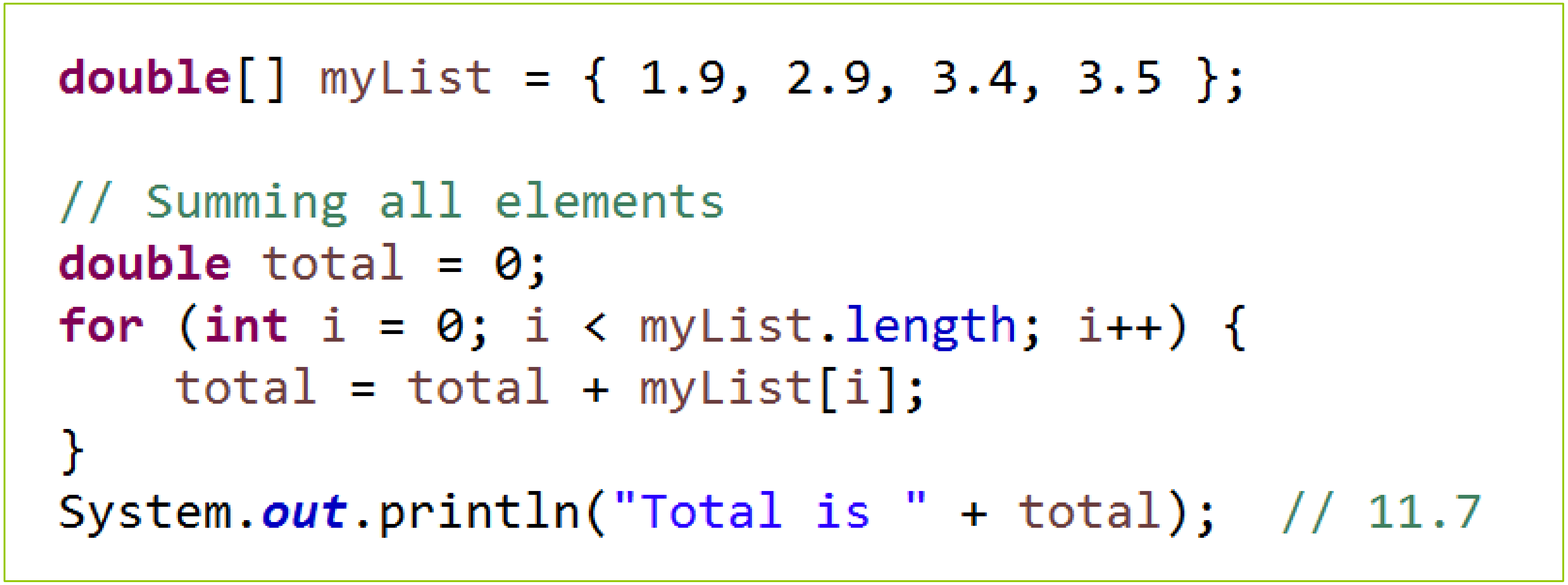
myList[0]	5.6
myList[1]	4.5
myList[2]	3.3
myList[3]	13.2
myList[4]	4.0
myList[5]	34.33
myList[6]	34.0
myList[7]	45.45
myList[8]	99.993
myList[9]	11123

Masyvo elementų spausdinimas



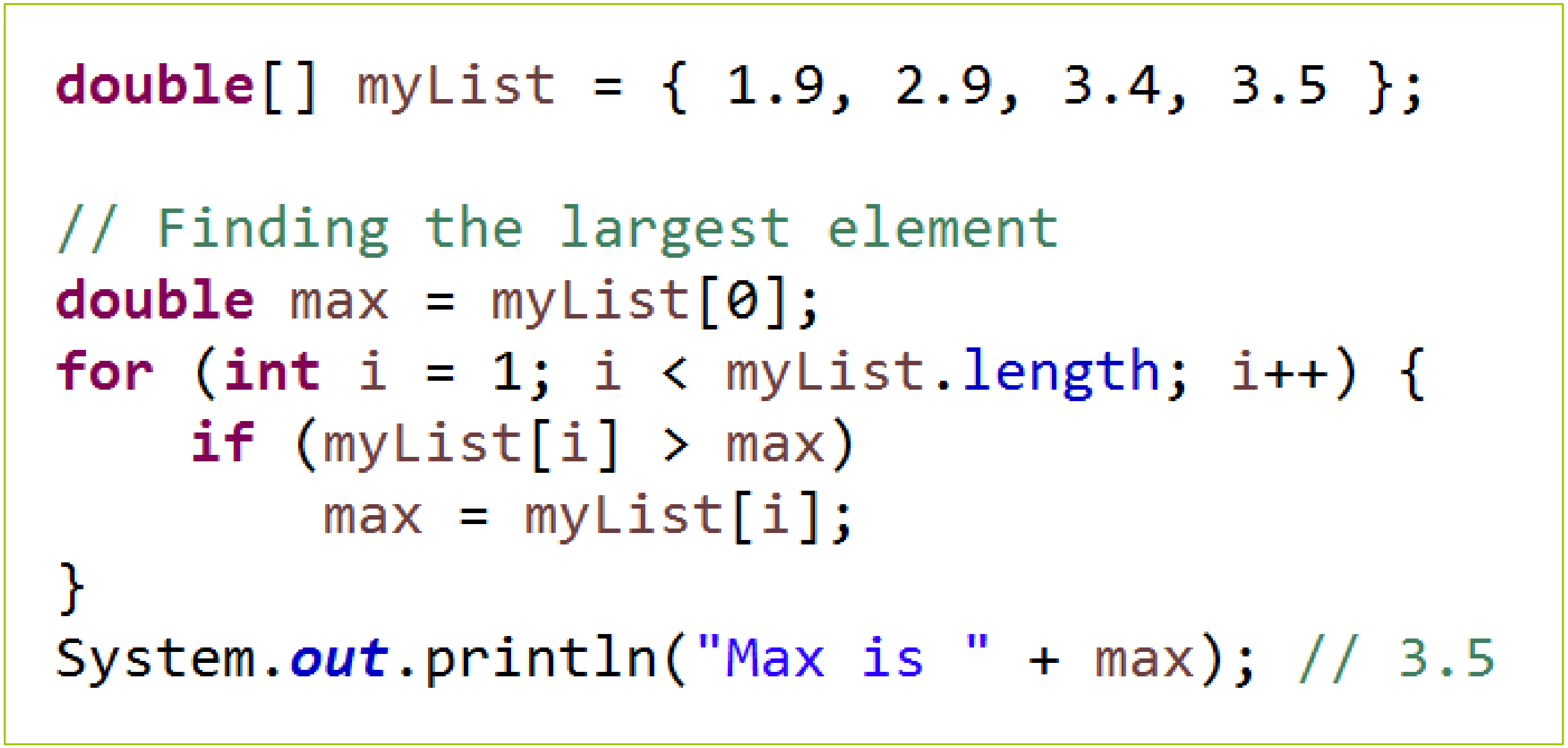
```java
double[] myList = { 1.9, 2.9, 3.4, 3.5 };

// Print all the array elements
for (int i = 0; i < myList.length; i++) {
    System.out.println(myList[i] + " ");
}
```

Masyvo elementų suma



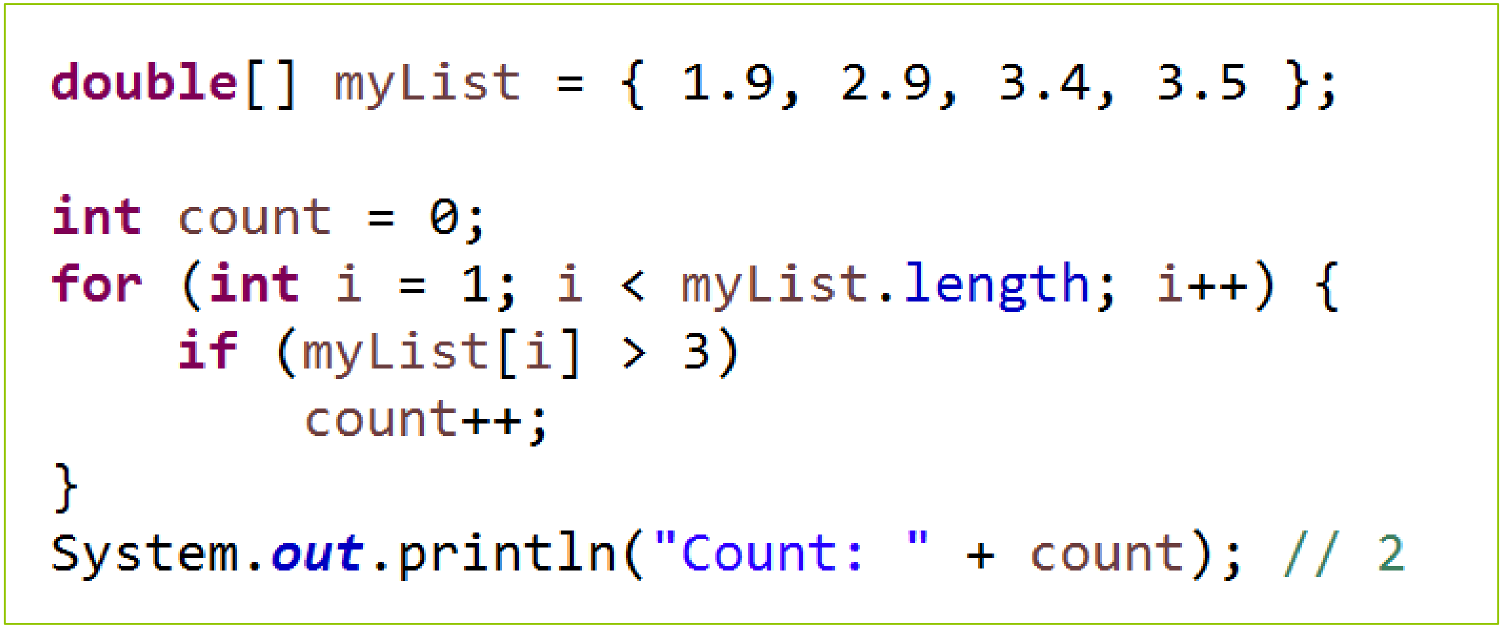
```
double[] myList = { 1.9, 2.9, 3.4, 3.5 };

// Summing all elements
double total = 0;
for (int i = 0; i < myList.length; i++) {
    total = total + myList[i];
}
System.out.println("Total is " + total);  // 11.7
```

Didžiausios reikšmės paieška



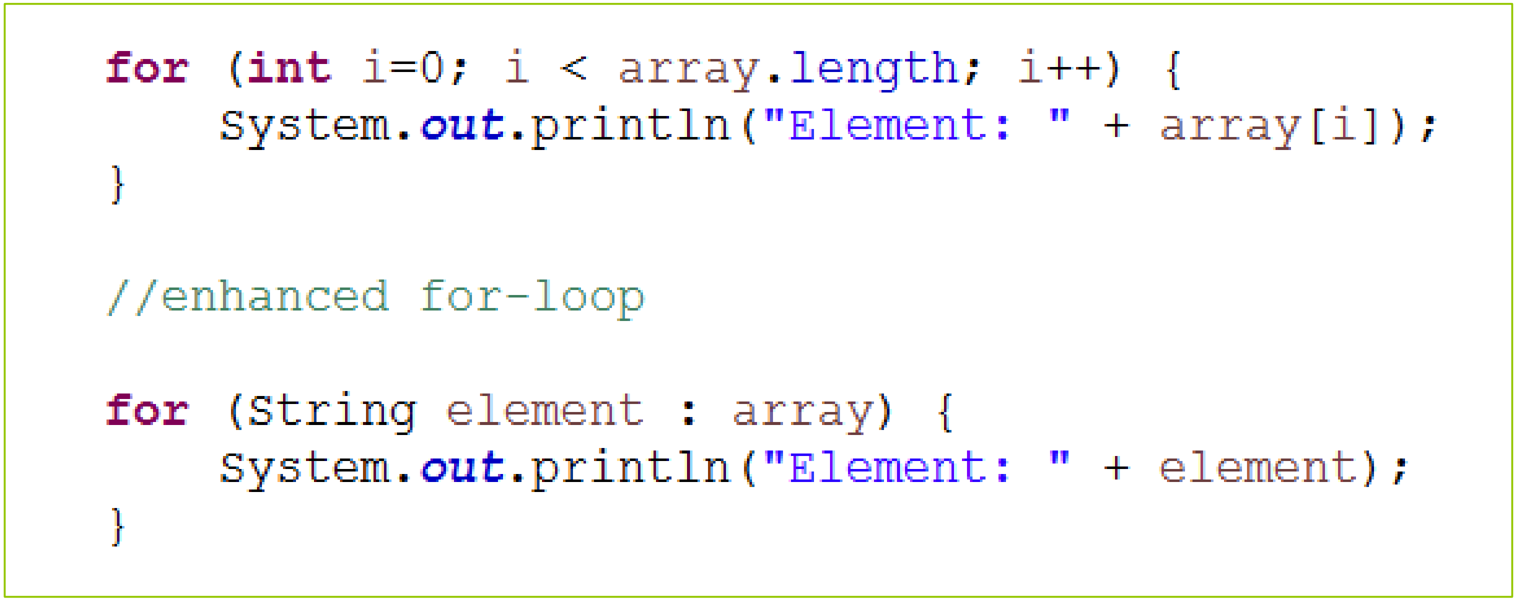
```java
double[] myList = { 1.9, 2.9, 3.4, 3.5 };

// Finding the largest element
double max = myList[0];
for (int i = 1; i < myList.length; i++) {
    if (myList[i] > max)
        max = myList[i];
}
System.out.println("Max is " + max); // 3.5
```

Elementų, tenkinančių sąlygą, kiekis



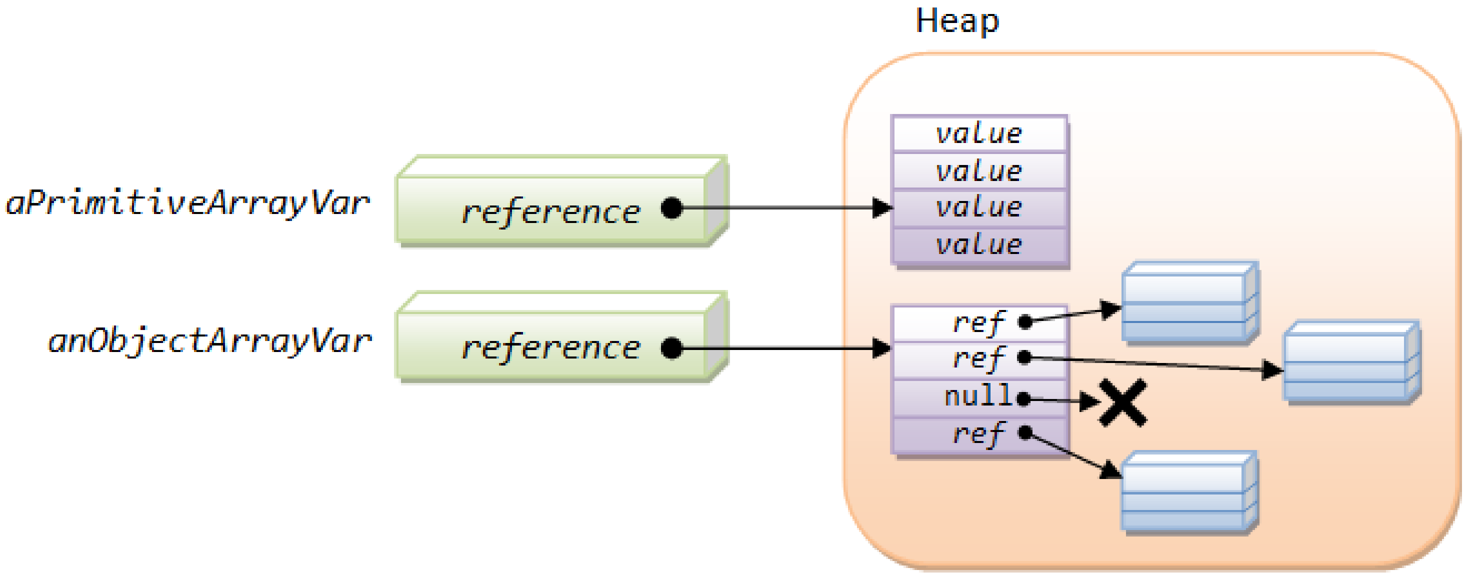
```
double[] myList = { 1.9, 2.9, 3.4, 3.5 };

int count = 0;
for (int i = 1; i < myList.length; i++) {
    if (myList[i] > 3)
        count++;
}
System.out.println("Count: " + count); // 2
```

enhanced for-loop

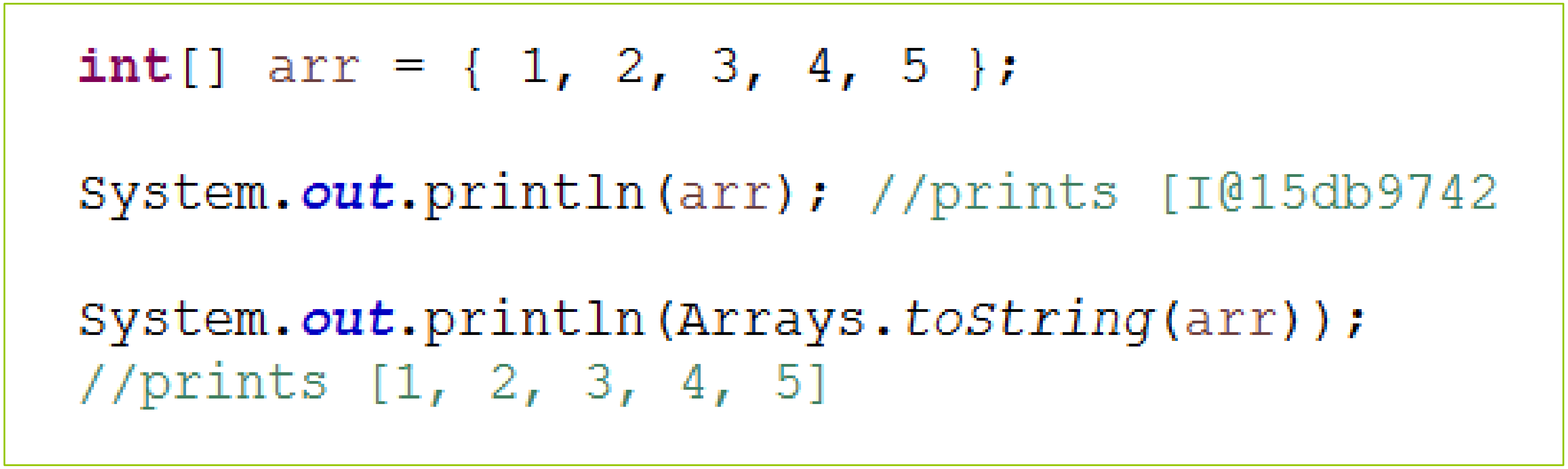


```java
for (int i=0; i < array.length; i++) {
    System.out.println("Element: " + array[i]);
}

//enhanced for-loop

for (String element : array) {
    System.out.println("Element: " + element);
}
```

Masyvai

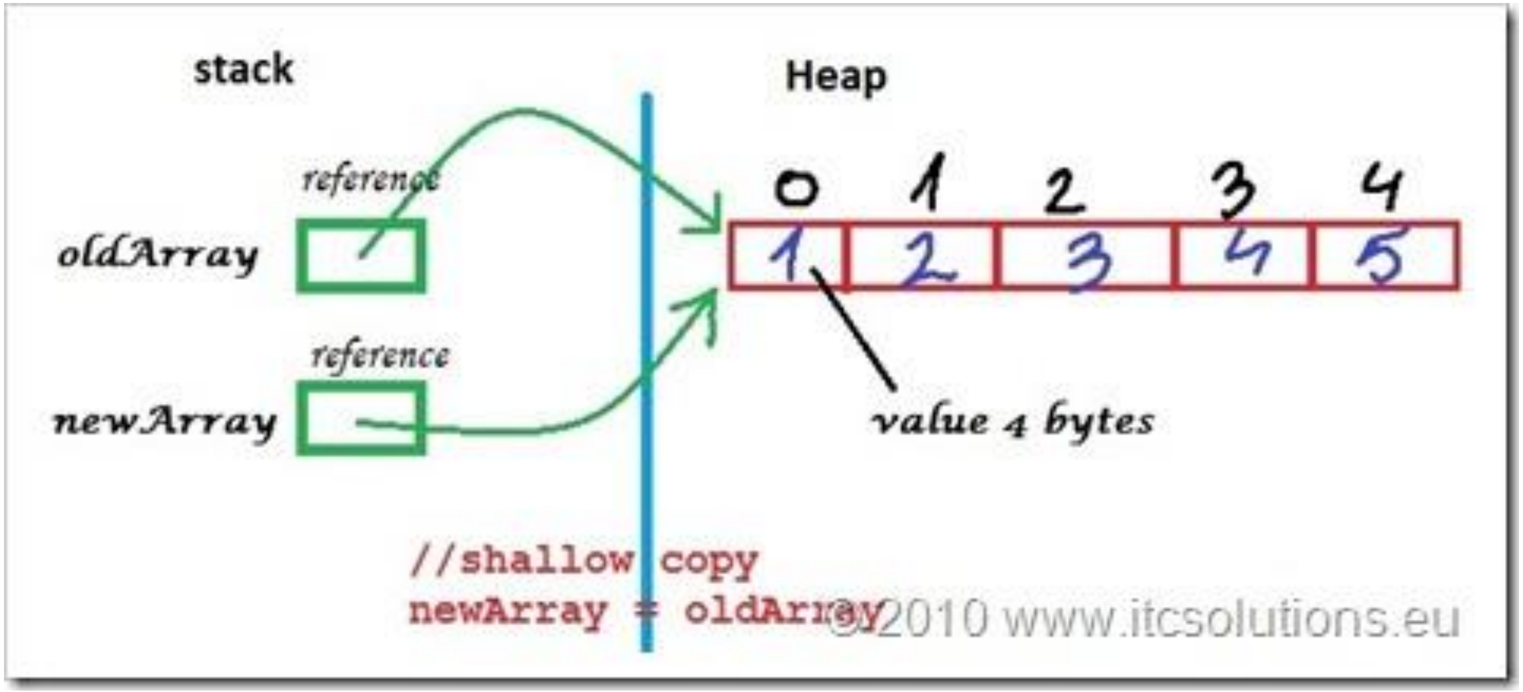

Heap
aPrimitiveArrayVar
reference
anObjectArrayVar
reference
value
value
value
value
ref
ref
null
ref

Arrays.toString



```
int[] arr = { 1, 2, 3, 4, 5 };

System.out.println(arr); //prints [I@15db9742

System.out.println(Arrays.toString(arr));
//prints [1, 2, 3, 4, 5]
```

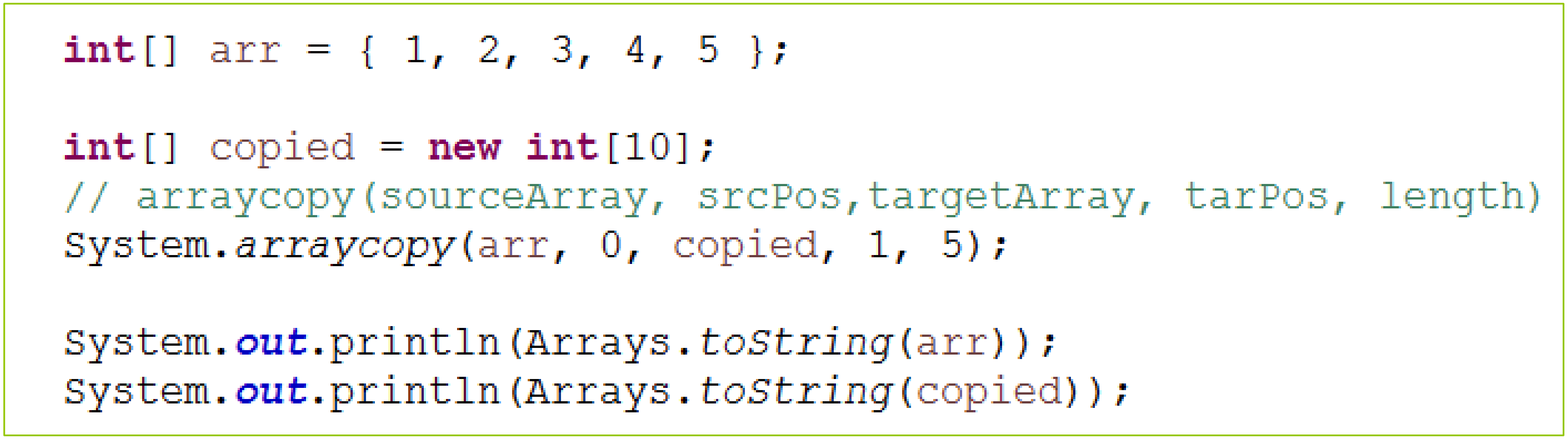
Kopijavimas



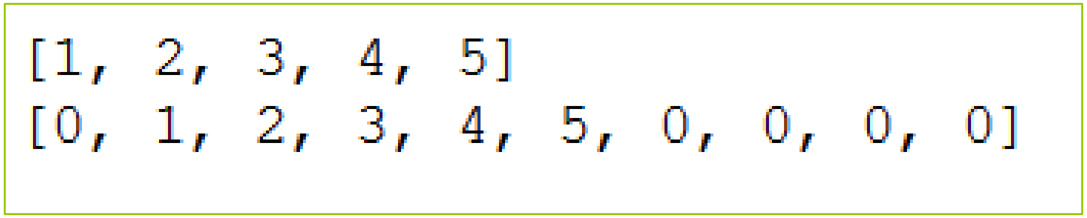
```
//shallow copy
newArray = oldArray
```

©2010 www.itcsolutions.eu

System.arraycopy

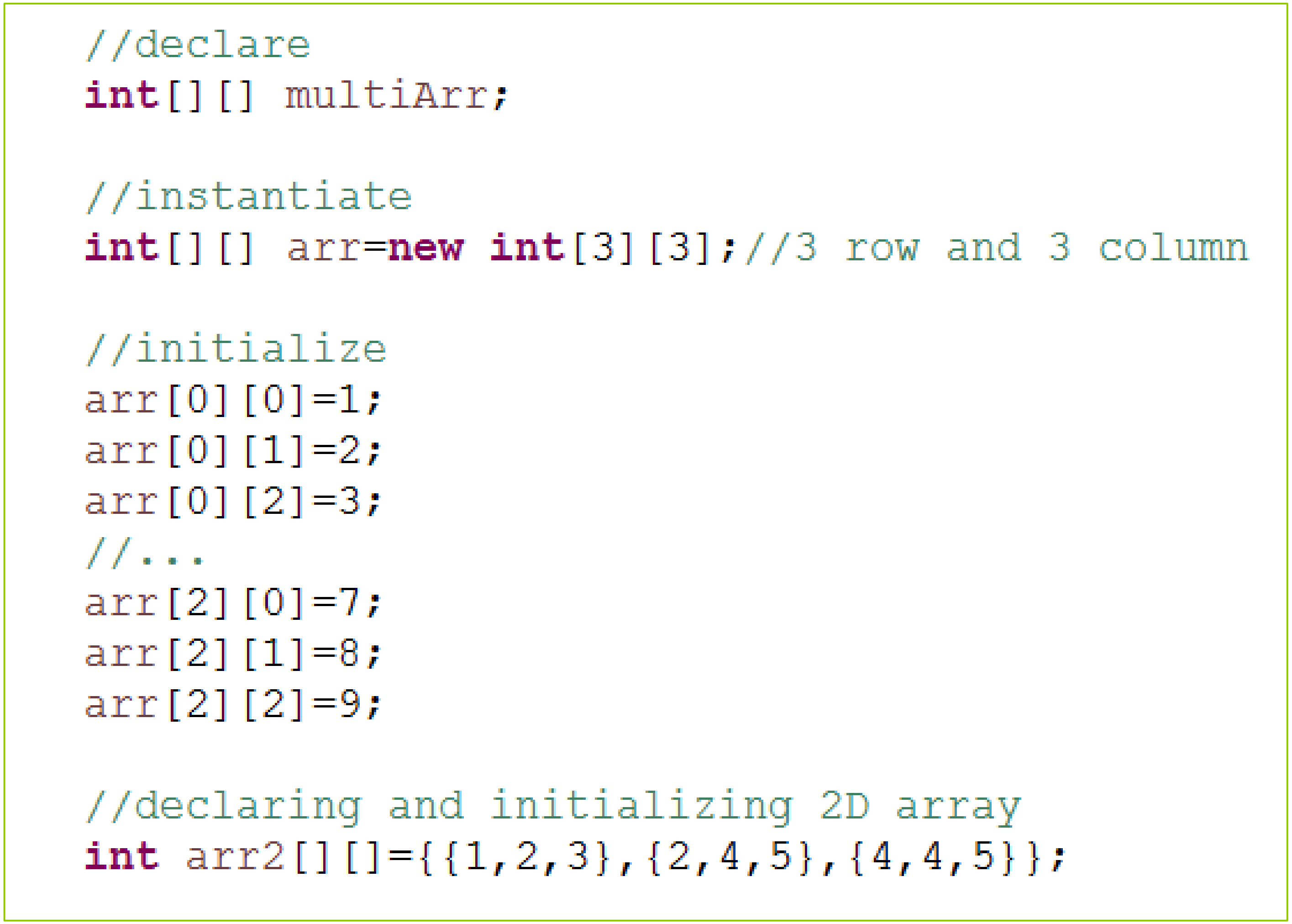


```
int[] arr = { 1, 2, 3, 4, 5 };

int[] copied = new int[10];
// arraycopy(sourceArray, srcPos,targetArray, tarPos, length)
System.arraycopy(arr, 0, copied, 1, 5);

System.out.println(Arrays.toString(arr));
System.out.println(Arrays.toString(copied));
```



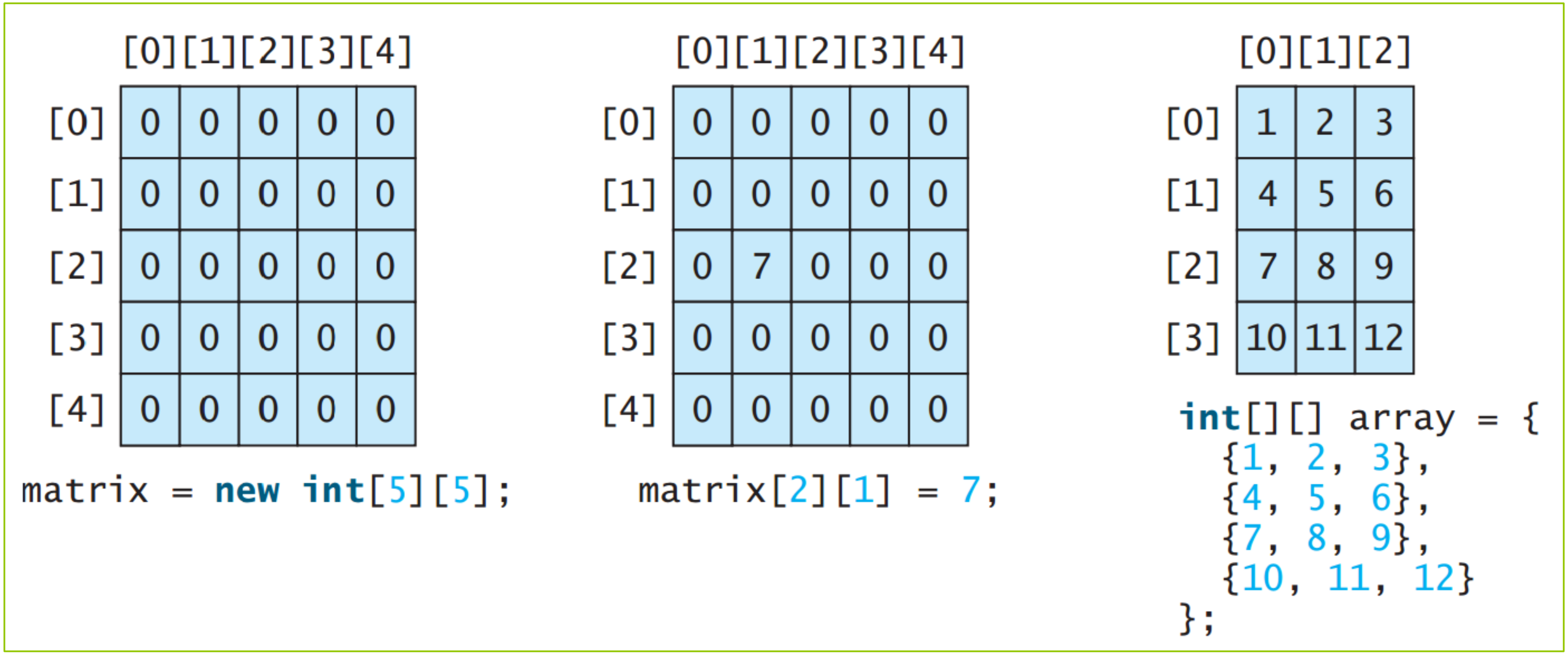
```
[1, 2, 3, 4, 5]
[0, 1, 2, 3, 4, 5, 0, 0, 0, 0]
```

Daugiamatis masyvas



```
//declare
int[][] multiArr;

//instantiate
int[][] arr=new int[3][3];//3 row and 3 column

//initialize
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
//...
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;

//declaring and initializing 2D array
int arr2[][]={{1,2,3},{2,4,5},{4,4,5}};
```

Daugiamatis masyvas



	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

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

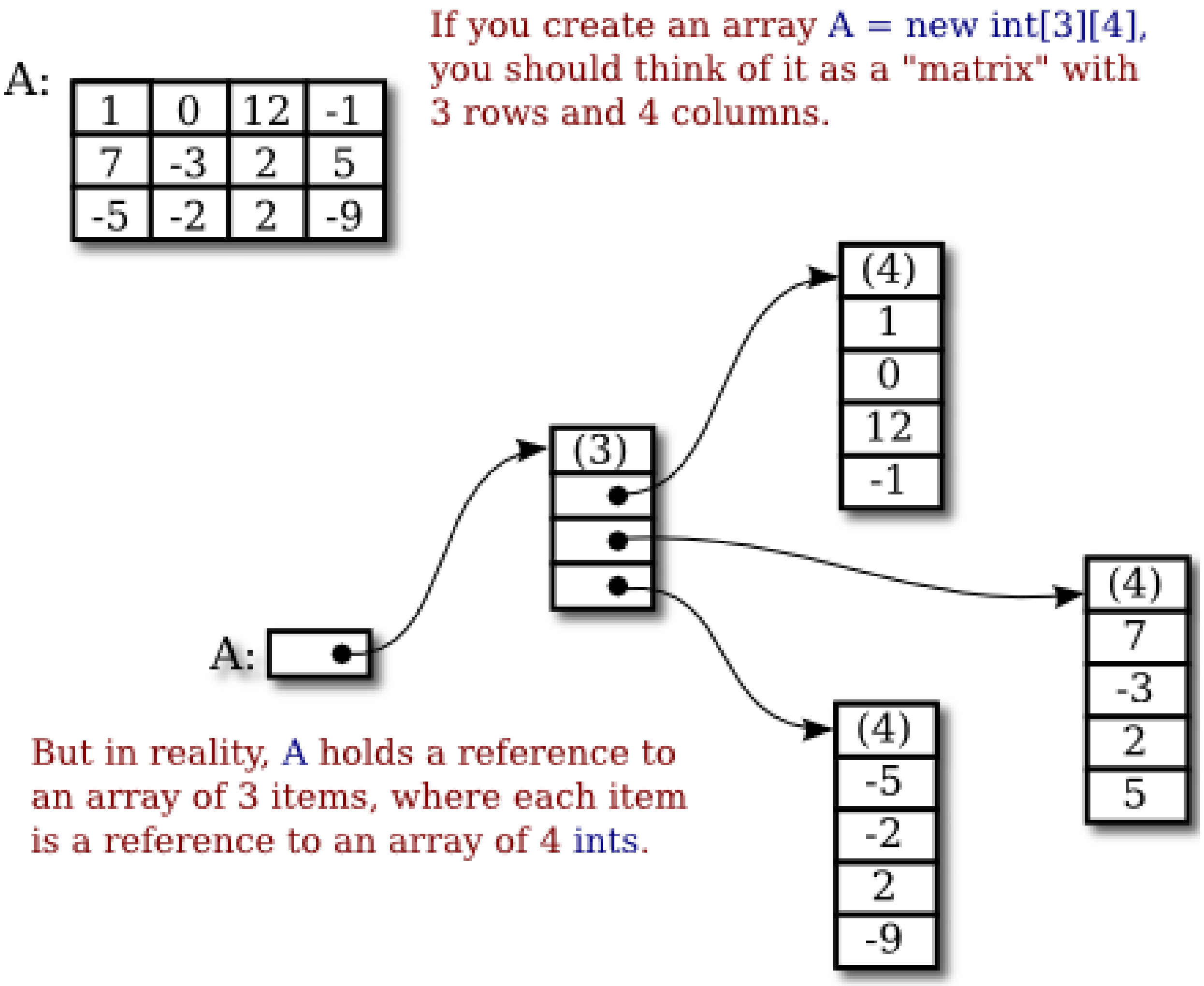
	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

```
matrix[2][1] = 7;
```

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

```
int[][] array = {
  {1, 2, 3},
  {4, 5, 6},
  {7, 8, 9},
  {10, 11, 12}
};
```

Daugiamatis masyvas



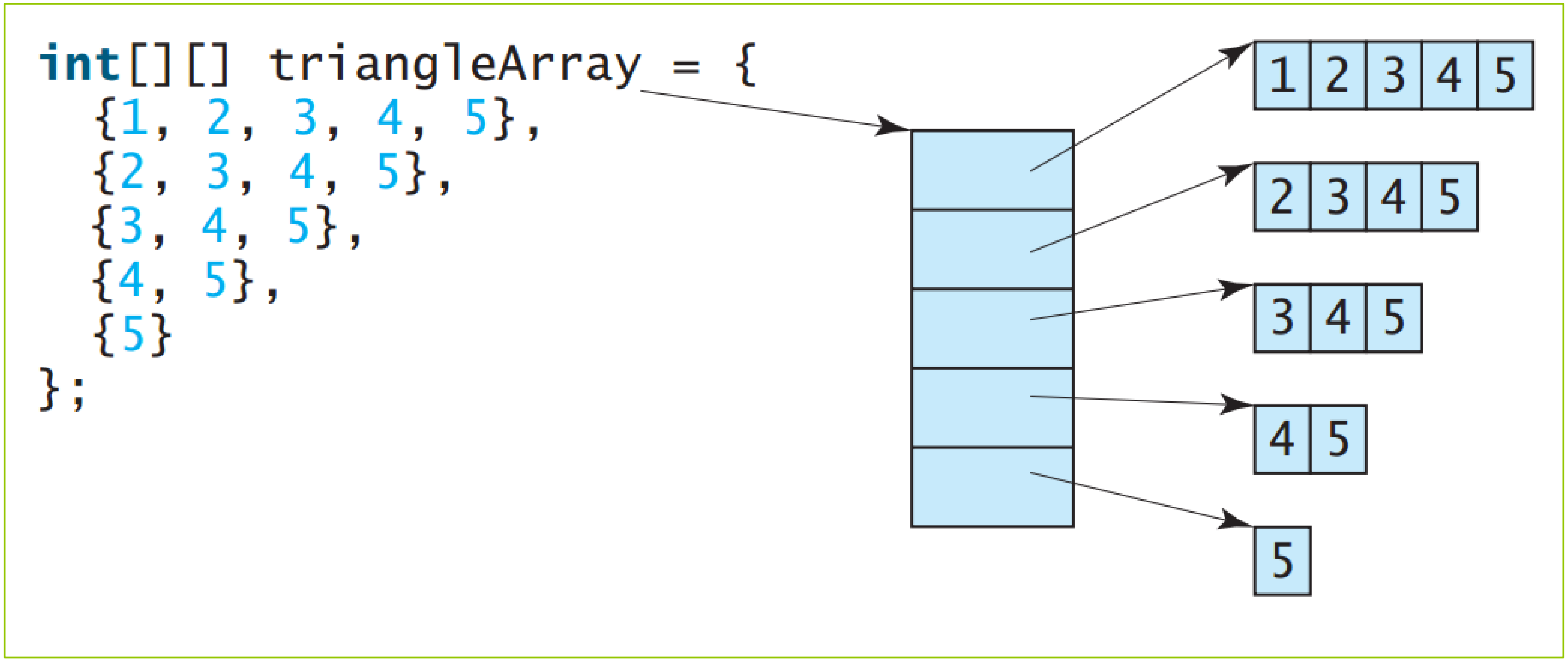
If you create an array A = new int[3][4], you should think of it as a "matrix" with 3 rows and 4 columns.

A:

1	0	12	-1
7	-3	2	5
-5	-2	2	-9

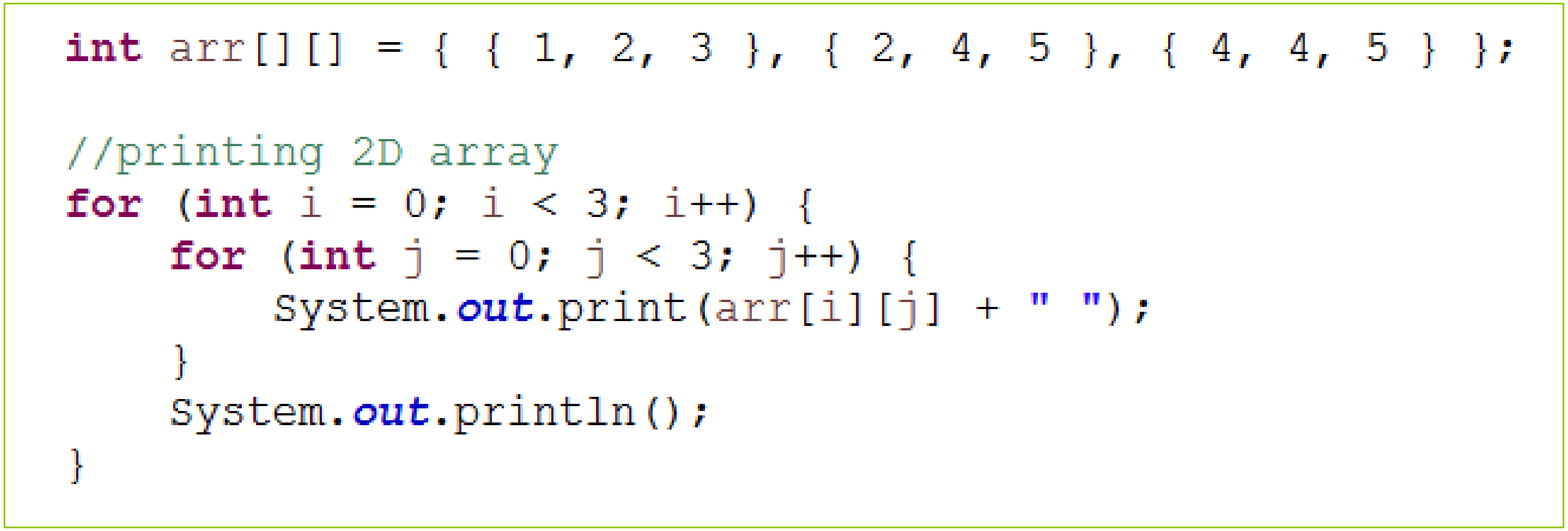
But in reality, A holds a reference to an array of 3 items, where each item is a reference to an array of 4 ints.

Ragged array

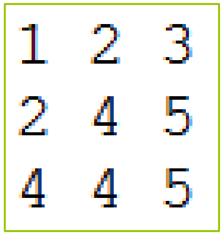


```
int[][] triangleArray = {
  {1, 2, 3, 4, 5},
  {2, 3, 4, 5},
  {3, 4, 5},
  {4, 5},
  {5}
};
```

Dvimačio masyvo spausdinimas

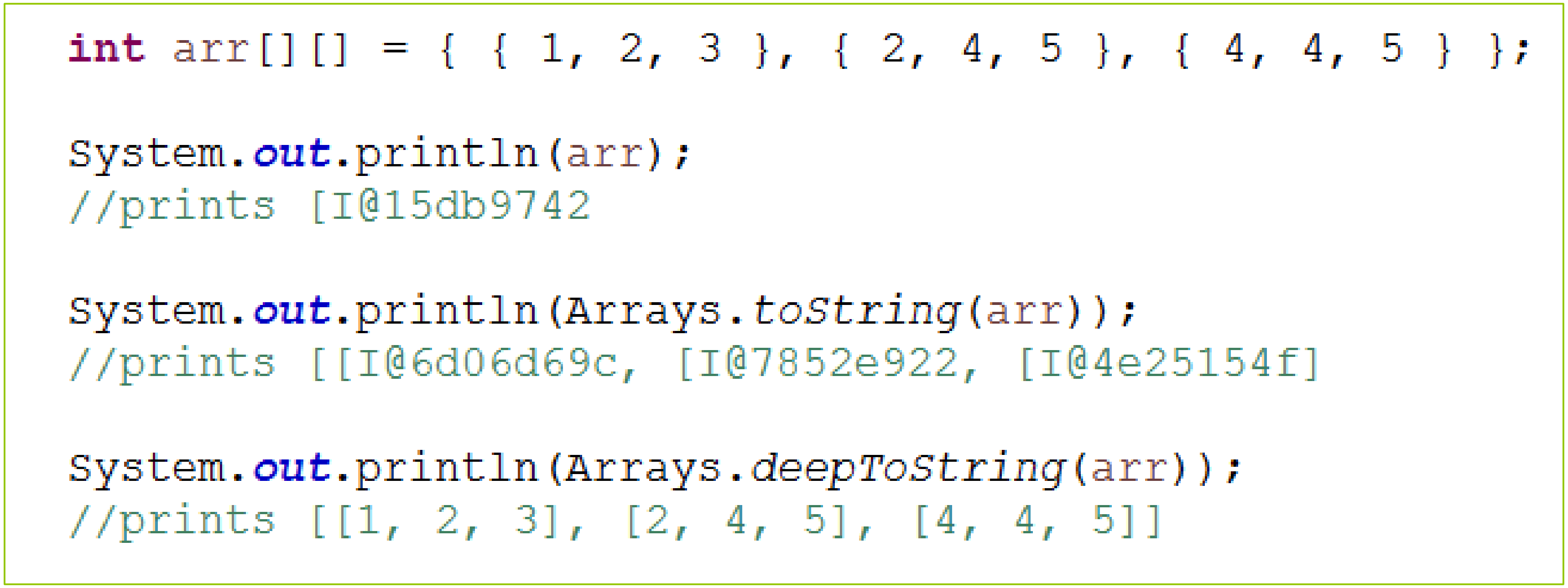


```
int arr[][] = { { 1, 2, 3 }, { 2, 4, 5 }, { 4, 4, 5 } };

//printing 2D array
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        System.out.print(arr[i][j] + " ");
    }
    System.out.println();
}
```



```
1 2 3
2 4 5
4 4 5
```

Dvimačio masyvo spausdinimas (2)



```
int arr[][] = { { 1, 2, 3 }, { 2, 4, 5 }, { 4, 4, 5 } };

System.out.println(arr);
//prints [I@15db9742

System.out.println(Arrays.toString(arr));
//prints [[I@6d06d69c, [I@7852e922, [I@4e25154f]

System.out.println(Arrays.deepToString(arr));
//prints [[1, 2, 3], [2, 4, 5], [4, 4, 5]]
```