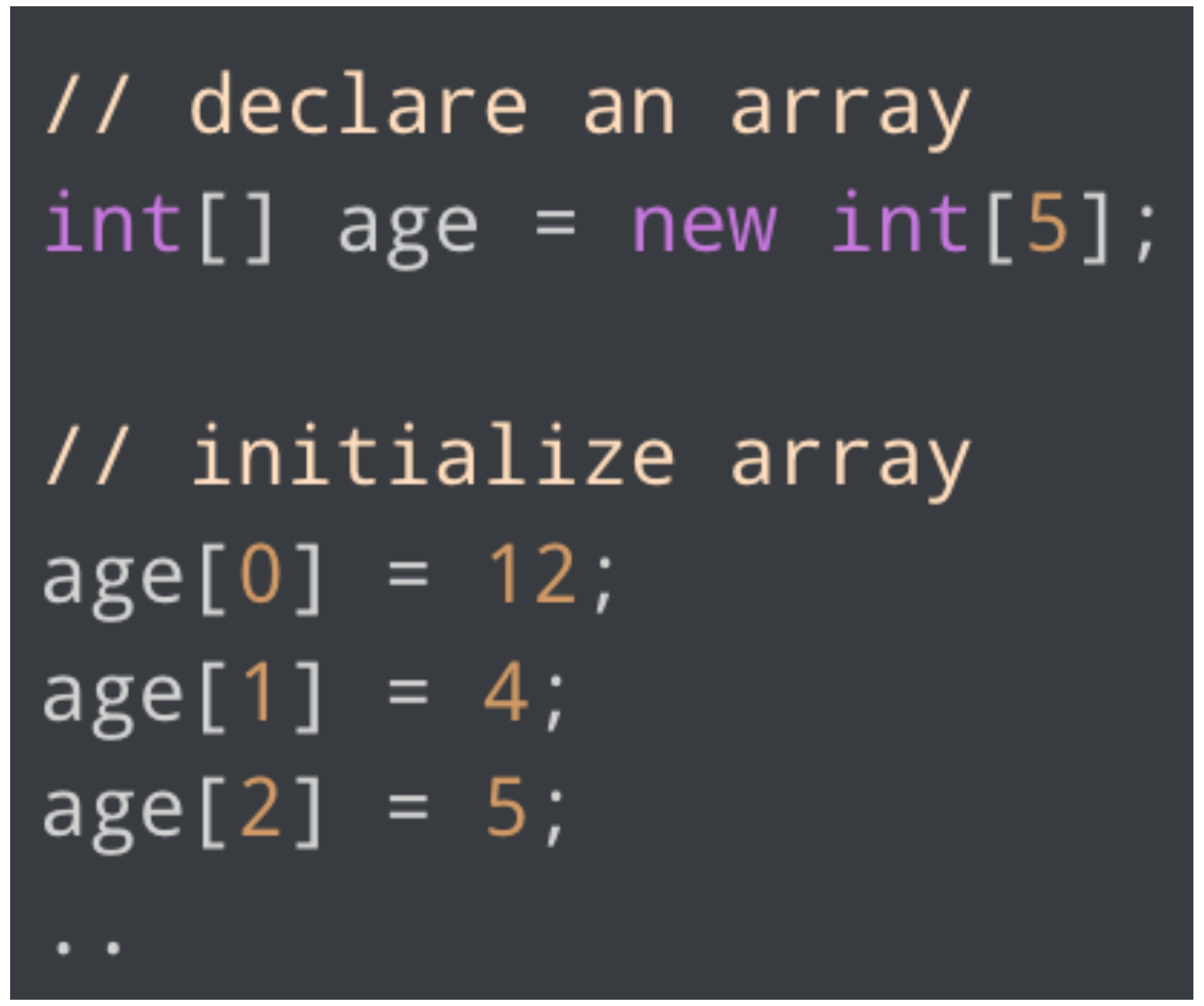
Java Course

Arrays

Week no. 2

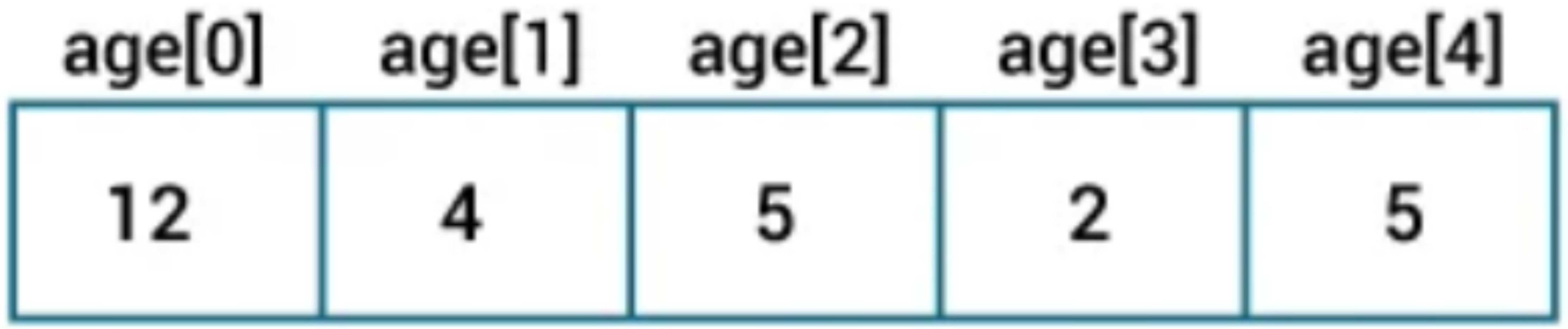
Java Arrays

- Java Array is a collection of same-type data
- Arrays are not primitive types in Java
- But, Arrays can store primitive type elements, as well as non-primitive types



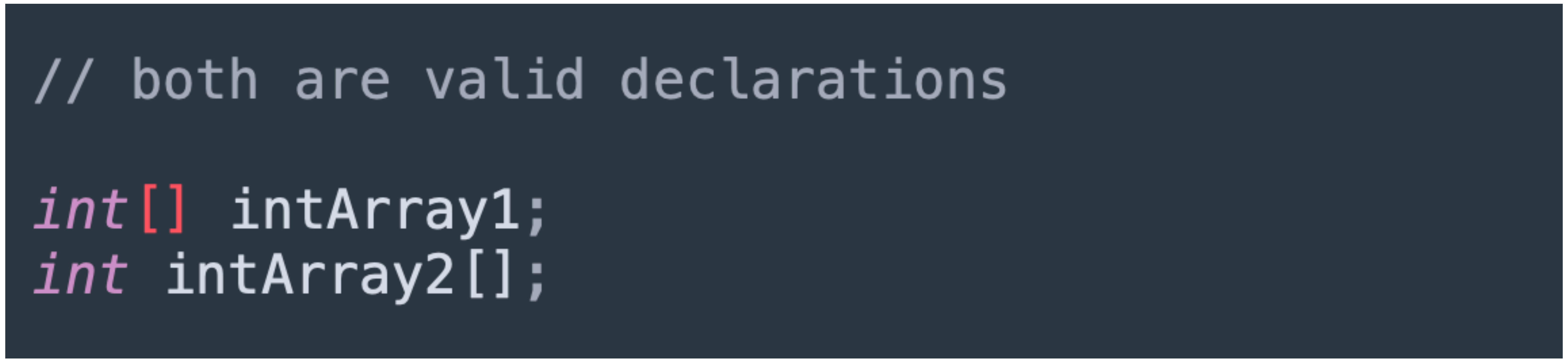
```java
// declare an array
int[] age = new int[5];

// initialize array
age[0] = 12;
age[1] = 4;
age[2] = 5;
..
```



Array Declaration & initialisation

- Syntax: `dataType[] arrayName;`
    - dataType - primitive data type like int, char, double, etc. or Java objects
    - arrayName - identifier
    - Declaration only tells the compiler that variable will store an array of dataType elements, but there is no actual array just yet

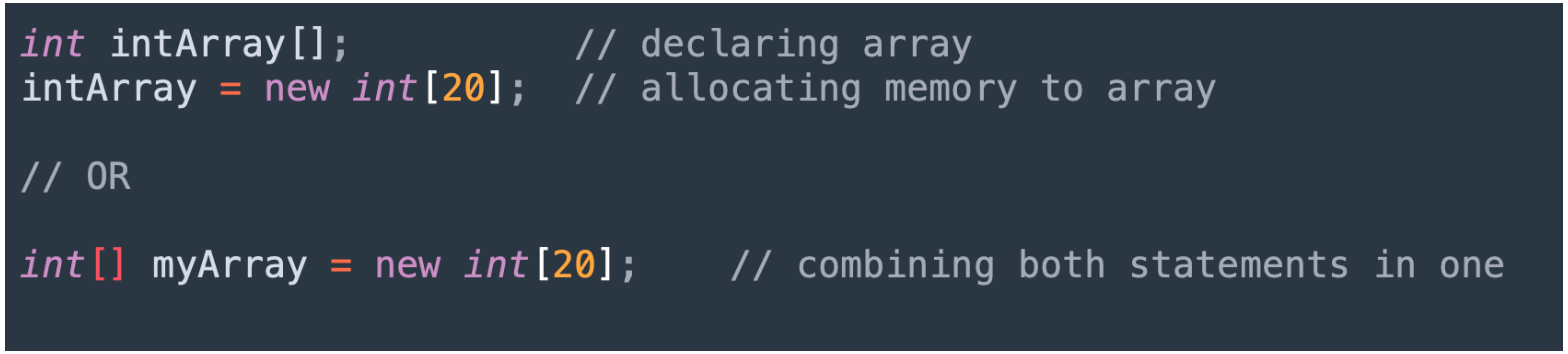


```java
// both are valid declarations

int[] intArray1;
int intArray2[];
```

Array Declaration & initialisation

- When an array is declared, only a reference of an array is created
- To create an actual array, we need to a allocate memory for it



```java
int intArray[];            // declaring array
intArray = new int[20];  // allocating memory to array

// OR

int[] myArray = new int[20];    // combining both statements in one
```

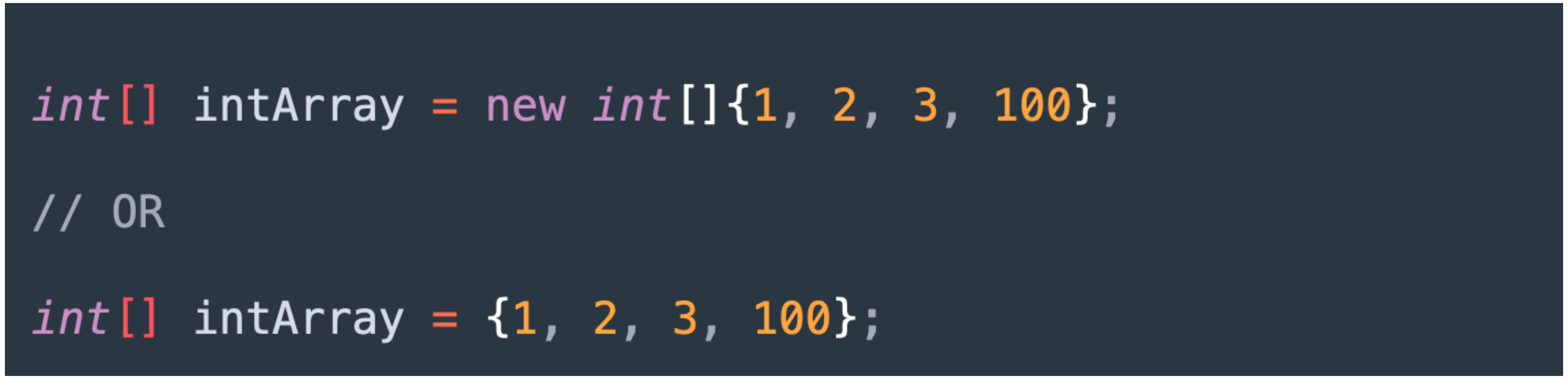
```java
int intArray[];          // declaring array
intArray = new int[20];  // allocating memory to array

// OR

int[] myArray = new int[20];    // combining both statements in one
```

- Note:
  - The elements in the array allocated by new will automatically be initialised to **zero** (for numeric types), **false** (for boolean) or **null** (for reference types - non-primitive types)
  - First, you must declare a variable of the desired array type.
  - Second, you must allocate the memory to hold the array (using **new**), and assign it to the array variable.

- When the size of the array and elements of the array are already known at the time we are creating an array, we can use:



```java
int[] intArray = new int[]{1, 2, 3, 100};

// OR

int[] intArray = {1, 2, 3, 100};
```

- Indexes in Java arrays start from 0

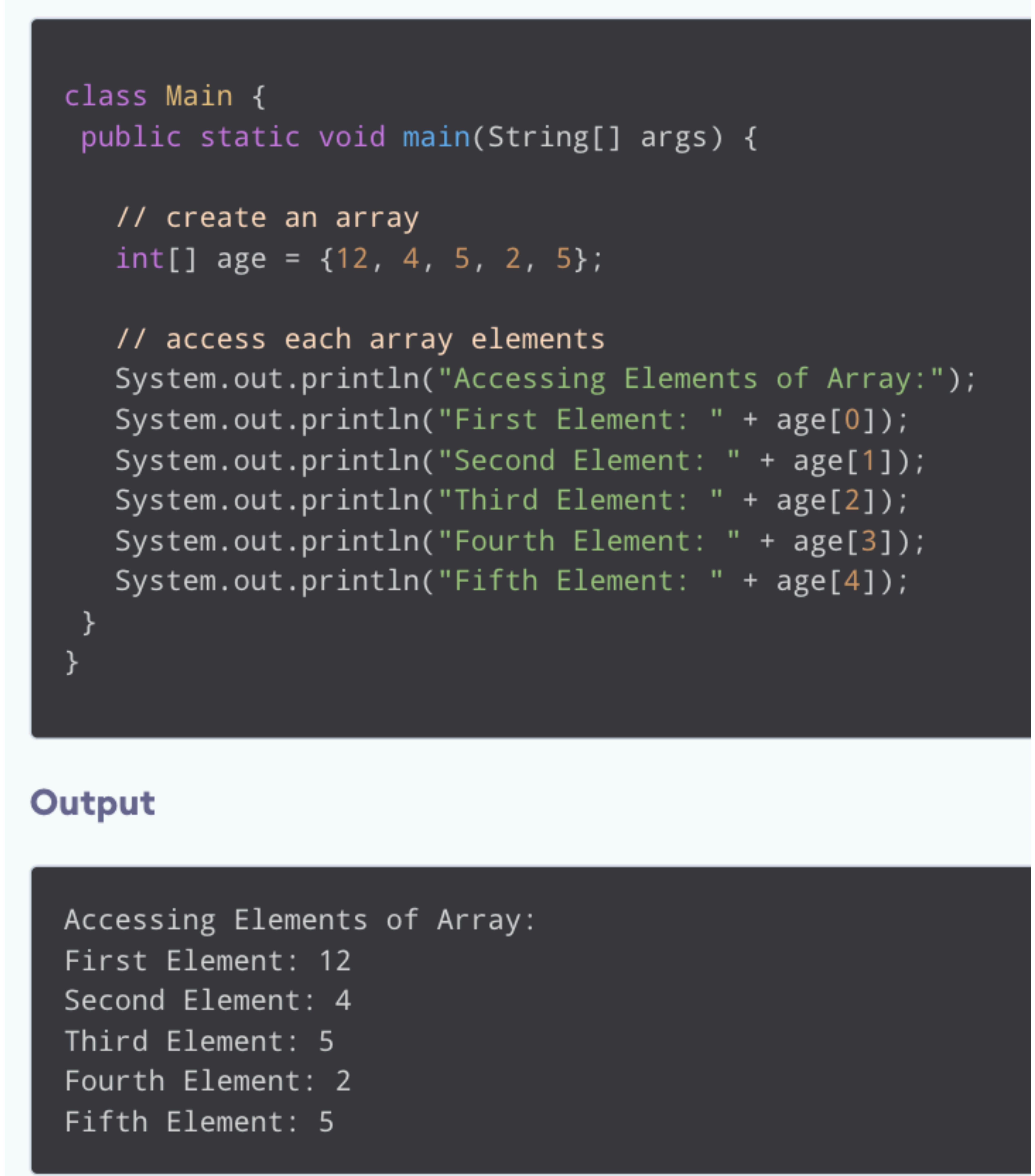
```java
// declare an array of 5 int elements
int[] age = new int[5];

// initialize array
age[0] = 35;
age[1] = 21;
age[2] = 3;
age[3] = 44;
age[4] = 50;

// will give us java.lang.ArrayIndexOutOfBoundsException
age[5] = 12;
```

Accessing array elements

- Array elements can be accessed using square brackets `[]` or using `for-each` loop



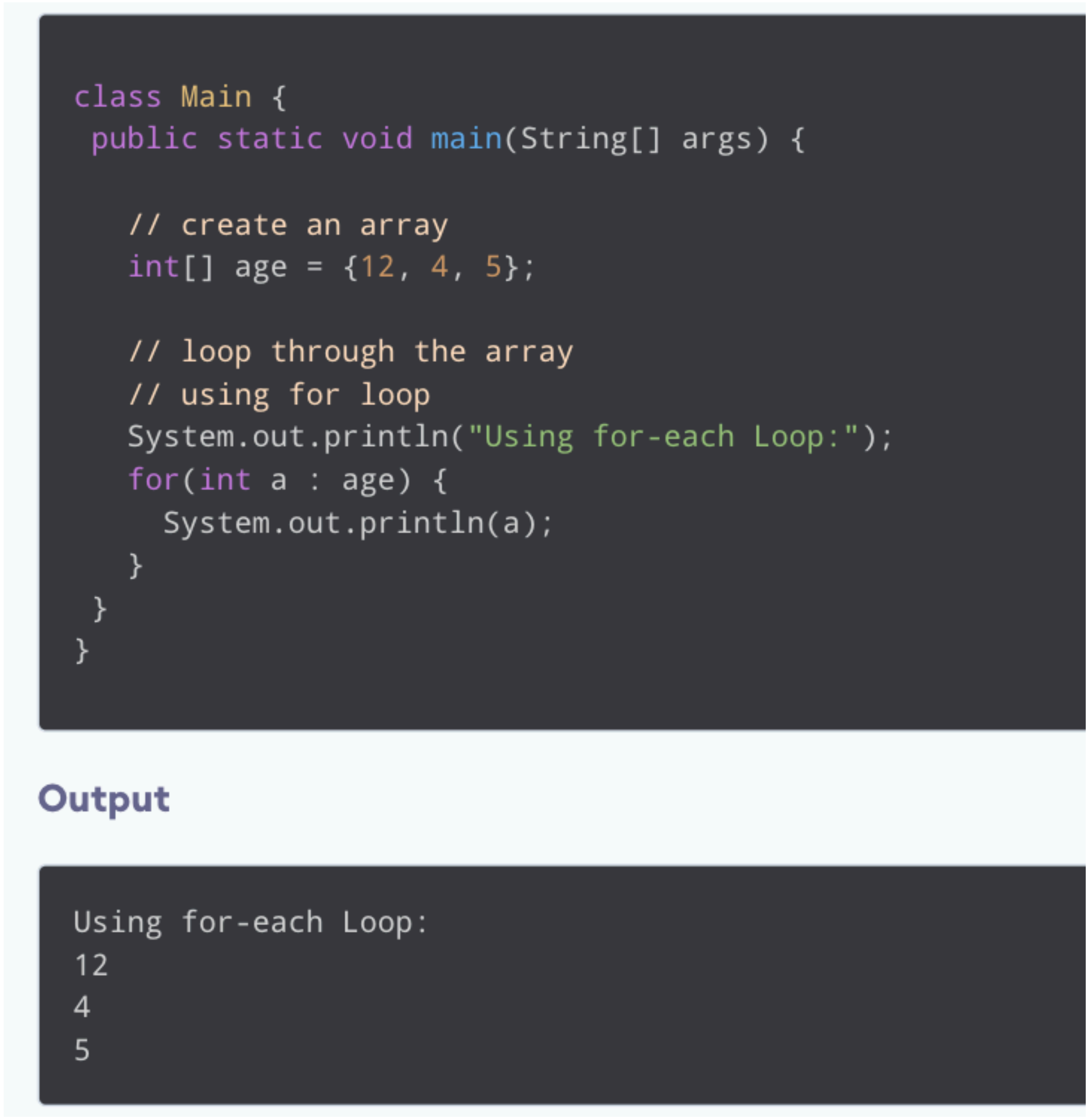
```java
class Main {
 public static void main(String[] args) {

   // create an array
   int[] age = {12, 4, 5, 2, 5};

   // access each array elements
   System.out.println("Accessing Elements of Array:");
   System.out.println("First Element: " + age[0]);
   System.out.println("Second Element: " + age[1]);
   System.out.println("Third Element: " + age[2]);
   System.out.println("Fourth Element: " + age[3]);
   System.out.println("Fifth Element: " + age[4]);
 }
}
```

Output

```
Accessing Elements of Array:
First Element: 12
Second Element: 4
Third Element: 5
Fourth Element: 2
Fifth Element: 5
```



```java
class Main {
 public static void main(String[] args) {

   // create an array
   int[] age = {12, 4, 5};

   // loop through the array
   // using for loop
   System.out.println("Using for-each Loop:");
   for(int a : age) {
     System.out.println(a);
   }
 }
}
```

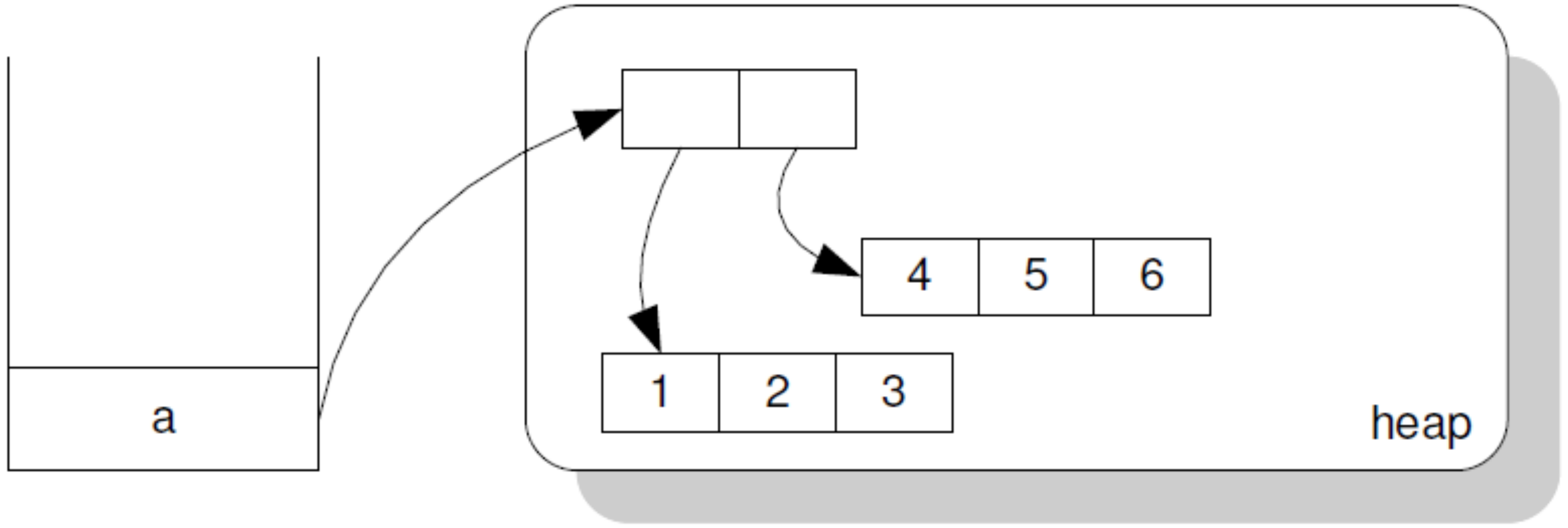
Output

```
Using for-each Loop:
12
4
5
```

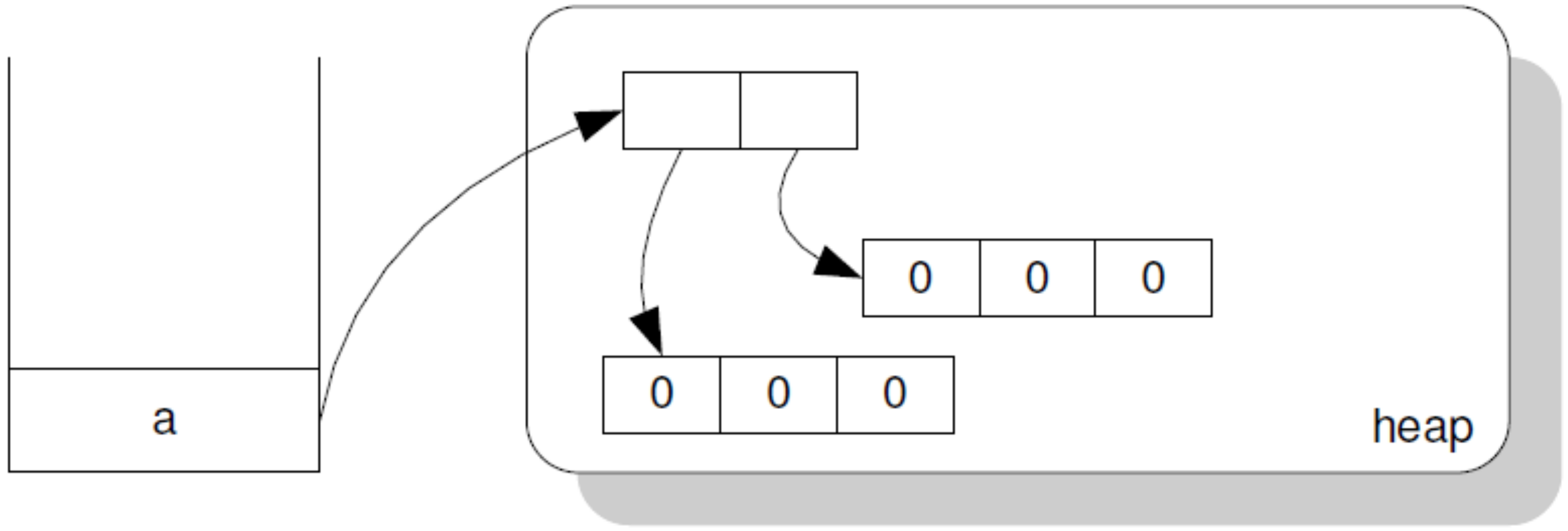
Multidimensional Arrays

- Array of arrays
- Each element of the array holds the reference of other arrays

```
int[][] a = { {1, 2, 3}, {4, 5, 6} };
```



```
int[][] a = new int[2][3];
```

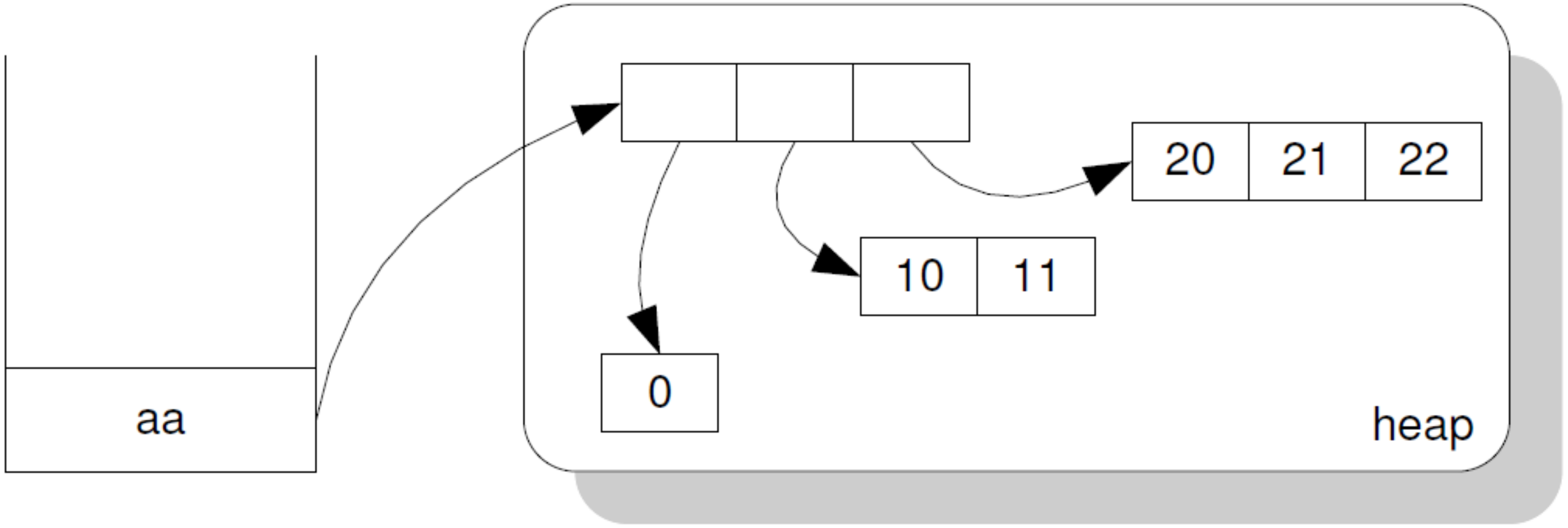


- We can create a multidimensional array with different number of columns:

```
int[][] aa = new int[3][];
for (i = 0; i < aa.length; i++) {
  aa[i] = new int[i + 1];
  for (int j =0; j < aa[i].length; j++)
    aa[i][j] = i*10 + j;
}
```

Output

```
0
10 11
20 21 22
```


aa
20
21
22
10
11
0
heap

Exercise

- https://www.hackerrank.com/challenges/java-1d-array-introduction/problem