Juniorprogrammierer.de

Java 6: Array methods

2024/25 – Sascha Stojanovic

Agenda

- What is are array methods?
- copyOf()
- equals() and “deepEquals()”
- fill()
- mismatch()
- sort()
- Exercises

What is are array methods?

- On our last session you tried unsuccessfully “cars.sum” and ”cars.add”. These are unvalid methods for arrays
- We already got to know one valid array-method which is ”.length”
- However there are more, see => `import java.util.Arrays;`
- Lets check together the most important ones

copyOf()

- This method creates copy of array with new length

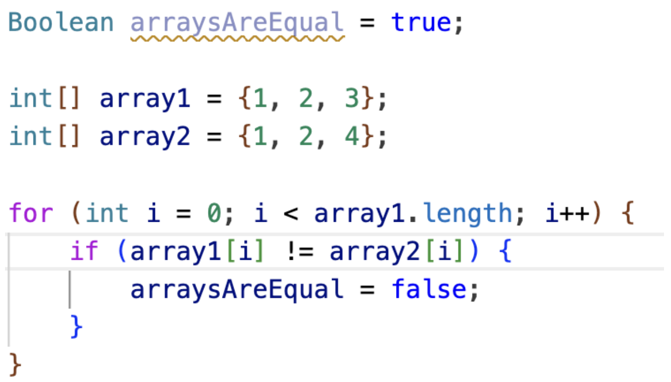
```java
// Initial array
int[] originalArray = {1, 2, 3};

// Element to add
int newElement = 4;

// Create a new array with one extra space
int[] newArray = new int[originalArray.length + 1];

// Copy elements from the old array to the new array
for (int i = 0; i < originalArray.length; i++) {
    newArray[i] = originalArray[i];
}

// Add the new element to the new array
newArray[newArray.length - 1] = newElement;

// Print the new array
for (int element : newArray) {
    System.out.print(element + " ");
}
```

With copyOf()

```java
// Original array
int[] originalArray1 = {1, 2, 3};

// Element to add
int newElement1 = 4;

// Create a new array with an additional space
int[] newArray1 = Arrays.copyOf(originalArray1, originalArray1.length + 1);

// Add the new element to the last position of the new array
newArray1[newArray1.length - 1] = newElement1;

// Print the new array
System.out.println("New array: " + Arrays.toString(newArray1));
```

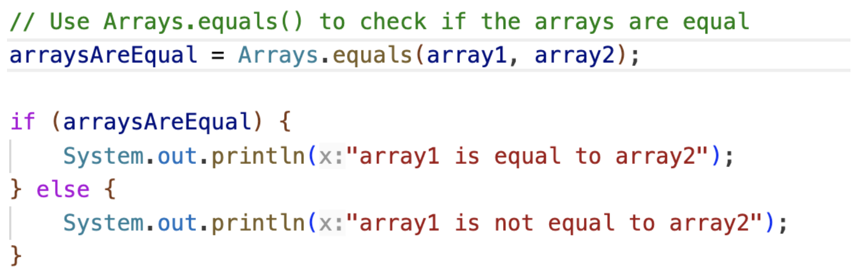
equals() and deepEquals()

- It compares two arrays



```java
Boolean arraysAreEqual = true;

int[] array1 = {1, 2, 3};
int[] array2 = {1, 2, 4};

for (int i = 0; i < array1.length; i++) {
    if (array1[i] != array2[i]) {
        arraysAreEqual = false;
    }
}
```

With equals()



```java
// Use Arrays.equals() to check if the arrays are equal
arraysAreEqual = Arrays.equals(array1, array2);

if (arraysAreEqual) {
    System.out.println(x:"array1 is equal to array2");
} else {
    System.out.println(x:"array1 is not equal to array2");
}
```

- For a comparision of two multidimensional arrays use “deepEquals”

fill()

- Fills array with values

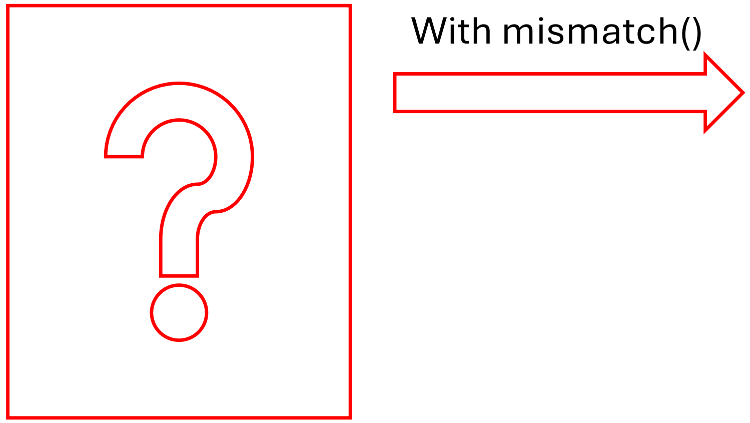
```java
int[] numbers = new int[3];

numbers[0] = 3;
numbers[1] = 3;
numbers[2] = 3;

// Print array
System.out.println("New array: " + Arrays.toString(numbers));

int[] numbers1 = new int[8];
numbers1[0] = 0;
numbers1[1] = 0;
numbers1[2] = 3;
numbers1[3] = 3;
numbers1[4] = 3;
numbers1[5] = 3;
numbers1[6] = 0;
numbers1[7] = 0;

System.out.println("New array: " + Arrays.toString(numbers1));
```

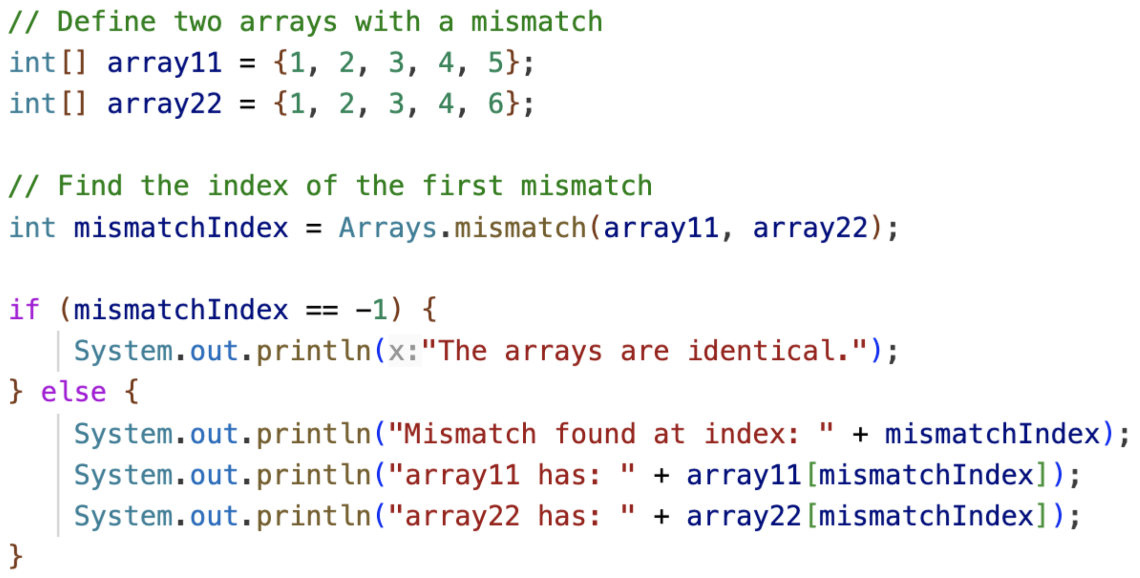
With fill()

```java
// Fill the entire array with the value 7
Arrays.fill(data1, val:7);

// Print the array to see the result
System.out.println("Array after fill: " + Arrays.toString(data1));

// Create another array and partially fill it
int[] data = new int[8];

// Fill part of the array (indices 2 to 5 with the value 3)
Arrays.fill(data, fromIndex:2, toIndex:6, val:3);

// Print the array to see the partial fill result
System.out.println("Partially filled array: " + Arrays.toString(data));
```

mismatch()

- Shows you the index of the mismatch()





```java
// Define two arrays with a mismatch
int[] array11 = {1, 2, 3, 4, 5};
int[] array22 = {1, 2, 3, 4, 6};

// Find the index of the first mismatch
int mismatchIndex = Arrays.mismatch(array11, array22);

if (mismatchIndex == -1) {
    System.out.println(x:"The arrays are identical.");
} else {
    System.out.println("Mismatch found at index: " + mismatchIndex);
    System.out.println("array11 has: " + array11[mismatchIndex]);
    System.out.println("array22 has: " + array22[mismatchIndex]);
}
```

sort()

- Sorts an array in ascending order

```java
int[] intArray = {5, 2, 8, 3, 7};
Arrays.sort(intArray);
System.out.println("Sorted integers: " + Arrays.toString(intArray));
```

- Super special tricky question: Every slide contains one “Arrays-Method if have not explained yet. Which one is it?

Exercise sort array in descending order

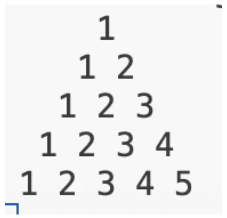
- The following array was provided in ascending order:

```java
int[] intArray = {5, 2, 8, 3, 7};
Arrays.sort(intArray);
System.out.println("Sorted integers: " + Arrays.toString(intArray));
```

- Please sort it now in descending order

Exersice print a pyramid (legendary level)

- The endresult should look like this:



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

- Define a “int level” to describe the levels of your pyramid
- Next create a pyramid multidimensional , e.g. “int [][] pyramid = new int[level] []
- Fill this multidimensional array
- Last print the multidimensional array correctly