Arrays

Lecture 6 of TDA 540
Object-Oriented Programming

Jesper Cockx

Fall 2018

Chalmers University of Technology — Gothenburg University

Last week: recap

Last week

- Abstraction
- Abstraction
- Abstraction

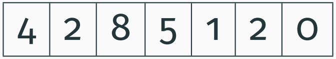
TELEPHONE

SONY
60

WM

Arrays

An **array** = a list of a fixed length, containing values of the same type



In Java syntax:

```
int[] myList = { 4, 2, 8, 5, 1, 2, 0 };
```

Accessing values in an array

Get the value at a given position:

```
int[] myList = { 4, 2, 8, 5, 1, 2, 0 };
int x = myList[3]; // x == 5
```

Warning: first position is 0!

Accessing values in an array

Get the value at a given position:

```
int[] myList = { 4, 2, 8, 5, 1, 2, 0 };
int x = myList[3]; // x == 5
```

Warning: first position is 0!

Update the value at a given position:

```
myList[2] = 9;
// myList is now
// { 4, 2, 9, 5, 1, 2, 0 }
```

Example: days of the week

```java
public static String getWeekday(int dayNumberOfWeek) {
  if      (dayNumberOfWeek == 1) return "Monday";
  else if (dayNumberOfWeek == 2) return "Tuesday";
  else if (dayNumberOfWeek == 3) return "Wednesday";
  else if (dayNumberOfWeek == 4) return "Thursday";
  else if (dayNumberOfWeek == 5) return "Friday";
  else if (dayNumberOfWeek == 6) return "Saturday";
  else if (dayNumberOfWeek == 7) return "Sunday";
  else return "Illegal day number!";
}
```

can be turned into

```java
public static String getWeekday(int dayNumberOfWeek) {
  final String[] weekdays =
      {"Monday", "Tuesday", "Wednesday",
       "Thursday", "Friday", "Saturday", "Sunday"};
  return weekdays[dayNumberOfWeek - 1];
}
```

The `args` of the `main` method

The `String[] args` in the main method contain the program arguments:

```java
public static void main(String[] args) {
  System.out.println(args[0]);
}
```

This prints the first program argument.

Creating a new array

```
// An array that can hold 20 doubles
double[] numbers = new double[20];

// An array that can hold 4 Strings
String[] numbers = new String[4];
```

Creating a new array

```
// An array that can hold 20 doubles
double[] numbers = new double[20];

// An array that can hold 4 Strings
String[] numbers = new String[4];
```

The initial values depend on the type:

- Numbers (`int`, `float`, ...) get value 0
- Booleans (`boolean`) get value `false`
- Objects (`String`, ...) get special value *null*

Iterating over an array

Print out all elements of an array in order:

```
int[] myList = { 4, 2, 8, 5, 1, 2, 0 };
for (int i = 0; i < myList.length; i++) {
  System.out.print( myList[i] );
}
```

Question: What does this print? How can you improve the output?

Example: searching an element in an array

```
static boolean search(int[] list, int x) {
  boolean found = false;
  for (int i = 0; i < list.length, i++) {
    if (list[i] == x) {
      found = true;
    }
  }
  return found;
}
```

Question: How can we also return the *position* of `x`?

How can we make this more efficient?

The enhanced for loop

The enhanced for loop enumerates all elements in an array:

```
static boolean search(int[] list, int x) {
  boolean found = false;
  for (int element : list) {
    if (element == x) {
      found = true;
    }
  }
  return found;
}
```

When (not) to use arrays

Use arrays to...

- store and process a list of user inputs
- write a method that has a variable number of inputs or outputs

Don't use arrays when...

- the values in the list have different types
- you only need the sum / minimum / maximum / ...and not all individual values

Important warning incoming

Array variables are references

A variable of type `int[]` is NOT an array but a reference to the location of the array.

```
int[] list  = { 1 , 2 , 3 };
int[] list2 = { 1 , 2 , 3 };
boolean test = list == list2;
                      // test is False!

int[] list3 = list;
list3[2] = 4;
int x = list[2]; // x is 4!
```

Copying an array ('deep copy')

Don't use `int[] array2 = array;`

Instead, use `Arrays.copyOf` from the standard library, or do it yourself:

```
// create new array of the same length
int[] array2 = new int[array.length];

// copy values one by one
for (int i = 0; i < array.length; i++) {
  array2[i] = array[i];
}
```

Checking if two arrays are equal

Don't use `array == array2;`

Instead, use `Arrays.equals` from the standard library, or do it yourself:

```
boolean equal = true
if (array.length != array2.length) {
  equal = false;
} else {
  for (int i = 0; i < array.length; i++) {
    if (array[i] != array2[i])
      equal = false;
  }
}
```

java.utils.Arrays

- `boolean equals(int[] a, int[] b)`
- `int[] copyOf(int[] f, int length)`

-
  ```
  int[] copyOfRange(int[] f,
                    int from, int to)
  ```
- `String toString(int[] f)`
- `void fill(int[] f, int value)`
- `void sort(int[] f)`

Note: there are similar methods for `float[]`, `double[]`, `boolean[]`, `char[]`, ...

15 min. break

Kahoot!

Arrays in Java

Some array algorithms

Example 1: Calculating the average and standard deviation

Task: implement the following methods:

- `double average(double[] values)`

$$\textbf{average}(x_1, \ldots, x_n) = \bar{x} = \frac{x_1 + \ldots + x_n}{n}$$

- `double stdDeviation(double[] values)`

$$\sigma(x_1, \ldots, x_n)^2 = \frac{(x_1 - \bar{x})^2 + \ldots (x_n - \bar{x})^2}{n - 1}$$

Example 2: Removing duplicates

Task: implement a method

```
int[] removeDuplicates(int[] values)
```

that returns an array with all duplicates removed.

Example 2: Removing duplicates

Task: implement a method

```
int[] removeDuplicates(int[] values)
```

that returns an array with all duplicates removed.

How to avoid creating a new array at each step?

Example 3: Binary search

Task: implement a more efficient version of

```
static boolean search(int[] values, int x)
```

for when the list `values` is sorted.

Example 3: Binary search

Task: implement a more efficient version of

```
static boolean search(int[] values, int x)
```

for when the list `values` is sorted.

This can also be found in `java.utils.Arrays` as `binarySearch`.

What’s next?

Next lecture: **Multi-dimensional arrays.**

To do:

- Read the book:
  - Today: sections 6.1-6.6
  - Next lecture: sections 6.7-6.8
- Hand in the third lab assignment