Welcome.TU.Code

23.11.2016

Agenda

- Discussion of homework
- (Final) recap of functions
- Recap of Arrays
- Exercises
- Two-Dimensional Arrays

How was the homework?

Recap - Functions

A function that calculates the sum of two (integer) numbers:

```java
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}
```

Recap - Functions

Let’s ignore this for now

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}
```

Recap - Functions

This is the **return type** of your function

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}
```

The return type tells you what *“comes out”* of a function. In this case, it’s an **integer** (i.e., a number)

Recap - Functions

This is the name of your function. It specifies what you have to write in order to call it (use it) from somewhere in your program

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}
```

Example:

```
public static void main(String[] args) {
        calculateSum(3, 4);
}
```

Recap - Functions

These are the parameters for your function.
They also have a type (here, **integer**) and a
name (here, a and b).

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}
```

Parameters are useful to help “abstract” or “generalize” a functionality, like here the computation of a sum. The function computes the sum of two numbers, it doesn’t care about the actual values.

With the same function, you can compute the sum of 3 and 4, 9 and 12, 1112 and 2534, and so on.

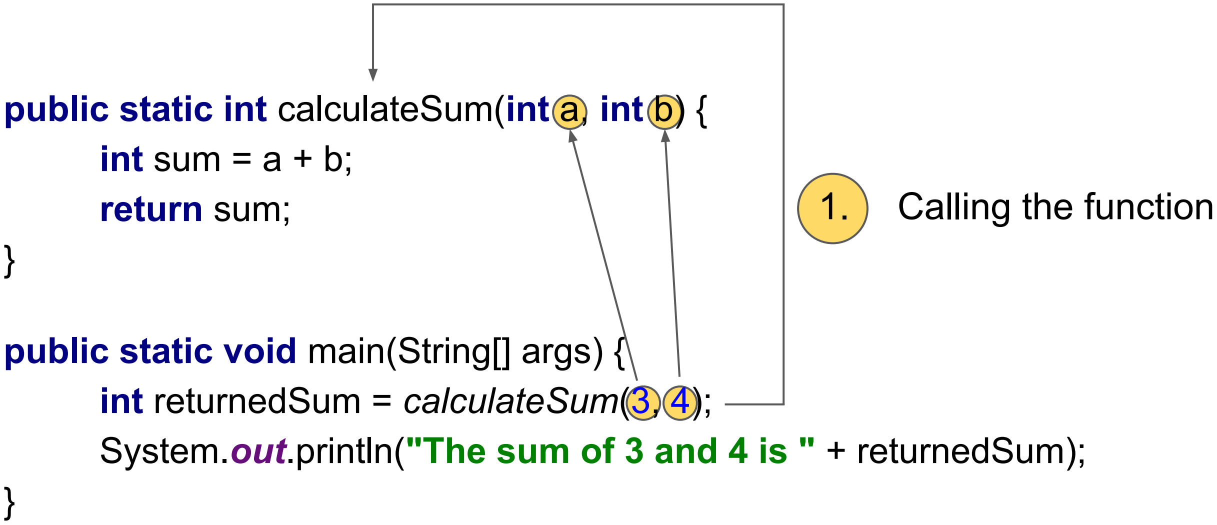
```
calculateSum(3, 4); -> 7
calculateSum(9, 12); -> 21
calculateSum(1112, 2534); -> 3656
```

Recap - Functions

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}

public static void main(String[] args) {
        int returnedSum = calculateSum(3, 4);
        System.out.println("The sum of 3 and 4 is " + returnedSum);
}
```

Recap - Functions



```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}

public static void main(String[] args) {
        int returnedSum = calculateSum(3, 4);
        System.out.println("The sum of 3 and 4 is " + returnedSum);
}
```

1. Calling the function

Recap - Functions

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}
```

2. Execution of the function
(i.e., calculate 3 + 4)

```
public static void main(String[] args) {
        int returnedSum = calculateSum(3, 4);
        System.out.println("The sum of 3 and 4 is " + returnedSum);
}
```

Recap - Functions

3. Return of the result (after this step, *returnedSum* holds the value *7*)

```
public static int calculateSum(int a, int b) {
        int sum = a + b;
        return sum;
}

public static void main(String[] args) {
        int returnedSum = calculateSum(3, 4);
        System.out.println("The sum of 3 and 4 is " + returnedSum);
}
```

Questions?

Recap - Arrays

You can think of an Array as a “box” that holds different values of the same type, for example, a box of Strings (words).

```
String[] words = {"One", "Two", "Three", "Four"};
```

Recap - Arrays

Each item in the “box” has a number assigned we can use to access it. This number is called the “index”.

```
String[] words = {"One", "Two", "Three", "Four"};
```

index	0	1	2	3
word	**One**	**Two**	**Three**	**Four**

Recap - Arrays

When we write this out in code, we put the index in between the square brackets:

```
String one = words[0];
String two = words[1];
String three = words[2];
String four = words[3];

String oneTwo = words[0] + words[1];
```

Recap - Arrays

Arrays are useful if we have multiple things of the same type. Let’s say we want to apply a function on each of the Strings we just saw:

```
for(int i = 0;i < words.length;i++) {
        System.out.println(words[i]);
}
```

Here, we just print out each word on a single line. This is not too impressive, but if we had more than four words (let’s say, one thousand), we would only need three lines of code instead of 1000.

Recap - Arrays

If we don’t know what items we want to put in the box beforehand, we can also do something like this:

```
String[] emptyBox = new String[10];
```

Now we have a “box” (an array) with space for 10 items.

Questions?

Let’s do an exercise together

Exercise

Let’s write a program that takes an array of words as input, prints out every word on its own line, and wraps everything in a “frame”.

For example, “Hello World in a frame” would become:

```
*********
* Hello *
* World *
* in    *
* a     *
* frame *
*********
```

Two-Dimensional Arrays

So far, we only dealt with one-dimensional arrays. What if we do an array of arrays?

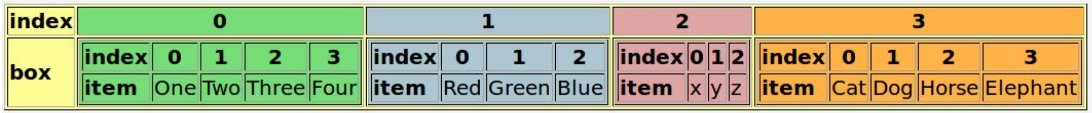
```
String[][] storage = {
        {"One", "Two", "Three", "Four"},
        {"Red", "Green", "Blue"},
        {"x", "y", "z"},
        {"Cat", "Dog", "Horse", "Elephant"}
};
```

Two-Dimensional Arrays

You can think of this as a “box of boxes”. Again, we can use indices that let us access particular items in the box, but this time we have two different ones:

- The first index refers to the box we want to access
- The second index refers to the item in that box

```
String[][] storage = {
        {"One", "Two", "Three", "Four"},
        {"Red", "Green", "Blue"},
        {"x", "y", "z"},
        {"Cat", "Dog", "Horse", "Elephant"}
};
```

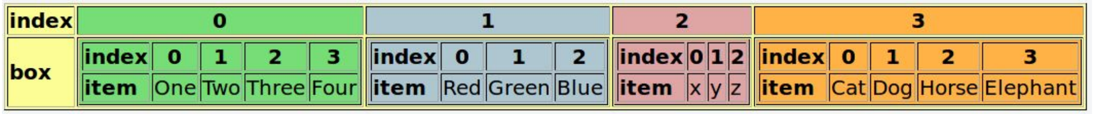


index	0					1				2				3				
box	index	0	1	2	3	index	0	1	2	index	0	1	2	index	0	1	2	3
	item	One	Two	Three	Four	item	Red	Green	Blue	item	x	y	z	item	Cat	Dog	Horse	Elephant

Two-Dimensional Arrays

```
storage[0] -> {“One”, “Two”, “Three”, “Four”}
storage[0][0] -> “One”
```

```
String[][] storage = {
        {"One", "Two", "Three", "Four"},
        {"Red", "Green", "Blue"},
        {"x", "y", "z"},
        {"Cat", "Dog", "Horse", "Elephant"}
};
```



<table>
<tr><th>index</th><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><th>box</th>
<td><table><tr><th>index</th><th>0</th><th>1</th><th>2</th><th>3</th></tr><tr><th>item</th><td>One</td><td>Two</td><td>Three</td><td>Four</td></tr></table></td>
<td><table><tr><th>index</th><th>0</th><th>1</th><th>2</th></tr><tr><th>item</th><td>Red</td><td>Green</td><td>Blue</td></tr></table></td>
<td><table><tr><th>index</th><th>0</th><th>1</th><th>2</th></tr><tr><th>item</th><td>x</td><td>y</td><td>z</td></tr></table></td>
<td><table><tr><th>index</th><th>0</th><th>1</th><th>2</th><th>3</th></tr><tr><th>item</th><td>Cat</td><td>Dog</td><td>Horse</td><td>Elephant</td></tr></table></td>
</tr>
</table>