Step 2 - Get started with methods

In the previous step, you got IntelliJ installed and run your first Java application. Now it's time to program!
The goal of this step on the ladder:
- I understand how a method works.
- I can create methods that use primitive data types, arrays and strings.
- I can traverse an array using for and while.
- I can use conditions such as if, else and switch.
- I have become good friends with the class String.
- I have gained an insight into how unit tests work.

Preparations

The tasks for this step require that you use some files located in Canvas. I show in the intro video how to use the files. Take a look at the video and download the files from Canvas. Create your project (as shown in the video) before you start with the tasks.
Methods are described in the syllabus book, chapter 4. It is especially chapter 4.1 (with the exception of 4.1.1) that is relevant to us today. Take a look there if you want additional literature, but relevant resources from the web are linked to the various tasks.

Intro

Java is a **typed** language. We need to specify what type our variables should be. And if we do not comply with this, then the compiler will say no. In the first instance, we will explore the primitive data types. Here you will find a nice overview of those. And here you can read how we declare (create) them and assign them a value. Common to all the tasks today is that we only need local variables: variables we create inside our methods.

A look at methods. A method is one of the most important building blocks in a Java program. And they are not that difficult to get started with - especially when you know the *function* of JavaScript from before. Before you take a look at the link below, you should know that in the beginning we do not care so much about *modifiers* and *exceptions*. That's for later.

https://www.geeksforgeeks.org/methods-in-java/

So far, we care most that a method is something we use to accomplish something. We must specify what it may return, what it is called and what parameters it receives. And variables we use today thus only live in our methods.

In this step, you will implement methods. I've created them, but you need to fill them with content so that they work as intended. And how do we know they are working as intended? I've created tests that check it for you. I have explained this in the intro video for this session, so take a look at it, if you have not already done so.

Task 1

Ok, enough talk!

Implement the addThreeNumbers method in the Assignment class. You see that it basically always returns 0, but that is not correct 😊

Task 2

Conditions and If statements you know from before. But it can be good to repeat a little? If you have forgotten, take a look here.
Implement the method isNumberSmallMediumOrBig in the Assignment class.
Tips & Warnings This method returns a String. We will take a closer look at the String class in later assignments in this session. But so far, you need to know that we can return a String for example like this: return "Small";

Task 3

Arrays should also be familiar. But you have not used them in Java. You get the opportunity to do that now. Read here if you need some help. And we must be able to go through the elements with the help of a for-loop. You can read about it here.
Implement the printAllStrings method in the Assignment class. Use the standard for-loop to go through the elements in the array.

But why settle for just using a standard for-loop? Do the task again, but this time using a for-each loop. You will find a description of the for-each loop in the previous link.

Task 4

Ojoj, this is going away! Now you are so well underway that you can go on with a method for:
Implement arraySum.
But once we start going through all the elements in an array, we must not forget *while*. Here is some info, if you need it. Use a while loop to print one letter at a time until you meet a period. The period (which ends the sentence) must also be printed.
Implement the printFirstSentence method.

Task 5

You know the switch statement from before. But let's take a look at that too. In this exercise you will use a switch statement (although you can use another way). You will find relevant info here.
Implement printCourseName using switch statement.

Task 6

String is not a primitive data type, but a class. We will take a closer look at classes in the next session, but String is so common that we almost have to get better acquainted with it early. Classes often have methods, and this also applies to String. This means that we can call String methods (and thus perform an action) when we have a String. Read more about String here.
To implement the methods below, you will need help from some of the methods found in the String class. You will find some of them described in the link above. But it is also appropriate that you become familiar with the Java API. You can find it in several versions, but you can for example use version 15 (if you use Java 15). The Java API is so important that

I have linked it up in Canvas as well. In the API you will find information about classes in the Java library, and for example what methods they offer. You have a search box at the top right so you can search for the String class. There is a lot of information there, but the most important thing for us this time is to see what methods the class offers - ie Method summary.
A little tip: We use the equals method when we compare two strings. Ex: s1.equals (s2) will be true if s1 and s2 are equally stringed.
Implement the methods:
- printAllStringsNotCorona
- printUpperCaseStrings
- isColorInNorwegianFlag
- firstOccurrence
- combinedLength

Task 7

We can have a variable number of parameters in our method. You can read more about that here. So lucky that you get a task to train on here 😊
Implement the addNumbers method

Task 8

We can choose not to care about uppercase or lowercase letters when we compare strings. I guess you can find a handy method in the String class that can help you…
Implement the printAllStringsNotCoronaCaseInsensitive method

Finished all the tasks? That's great! Do not forget to answer how it went today! Here.

I have some extra tasks for you, if you want it.

Extra task 1

In this session, we have tested that the methods work using automated tests. It's a little mysterious. Maybe you can test if the methods you have created work by calling them from the main method? In that case, you must create an object of the type Assignment, and then call on them, for example as follows:

```
Assignment assignment = new Assignment ();
System.out.println (assignment.isNumberSmallMediumOrBig (15));
```

By the way, this is something we will take a closer look at next week 😊

Extra task 2

Was extra task 1 also too easy for you? Then you can create a few more device tests and create methods that make the tests pass. You can look at the existing tests as a starting point. This is far beyond what is expected for step 2, but getting acquainted with unit tests (for example via JUnit) is a good thing for those who need more difficult challenges.