Exercises:

1. To use Gradle for own projects, the installation of Groovy is required. Document clearly, how you install Groovy and setup your computer and which artifacts where required.
   a) Additionally, write a small Groovy program, which proves, that Groovy is working correctly
2. Implement the method *swap(int, int)* with class scope variables.
3. Program a recursive variant of the method *factorial()*.
   a) Test and document the method. – Also try very large arguments and try to break the method!
4. Get some information about the Fibonacci series. Implement a method that prints the Fibonacci series up to a number, which is a parameter of the method.
   a) Test and document the method. – Also try very large arguments and try to break the method!
5. Create a method, to which a double is passed. This double should be rounded to an int and this int will be returned. Mind to write tests!
6. Implement the method boolean *promptAndReadBoolean(String prompt)* following the information from the lecture.
7. A program should ask the user for five integers. The program then outputs the sum of squares. – The program resulting from the last exercise is the basis for this exercise.
   a) The individual actions in the program should be divided into meaningful methods (e.g. *acceptUserInput()*, *output()*, *showMenu()* …). Readability and avoidance of redundancy (i.e. obeying the DRY-principle) should be your primary targets in the implementation.
8. Show, how **javadoc** can be used to generate an HTML documentation from Java code. (The flag **-private** must be specified on the command line.)

Remarks:

- If exercises ask to document something, a Word document with explanatory text, maybe incl. snippets and screenshots is awaited as companion artifact in the repository or sent as attachment to the solution of the exercise!
- All projects must be Gradle projects! Write your methods in a way, so that they can be executed by test-methods!
- Everything that was left unspecified can be solved as you prefer.
- In order to solve the exercises, only use known constructs, esp. the stuff you have learned in the lectures!
- The usage of class-scope variables is not allowed!
- Avoid magic numbers and use constants where possible.
- The results of the programming exercises need to be runnable applications! All programs have to be implemented as console programs.
- The programs need to be robust, i.e. they should output prompts, cope with erroneous input from the user.
- Mind to close *Scanner* instances!
- You should be able to describe your programs after implementation. Comments are mandatory.
- In documentations as well as in comments, strings or user interfaces make correct use of language (spelling and grammar)!
- Don't send binary files (e.g. class-files or the contents of debug/release folders) with your solutions! Do only send source and project files.
- Don't panic: In programming multiple solutions are possible.

- If you have problems use the help system of the IDE, books and the internet primarily.
- Of course you can also ask colleagues; but it is of course always better, if you find a solution yourself.