Exercises:

1. Get and Install IntelliJ IDEA Community Edition.
2. What happens, if you compare two *String* values with the "=="-operator?
3. What happens, if you output a *String* containing umlauts to the console? – If strange letters are appearing on the console, what do you have to configure for the OS console and the IntelliJ IDEA console? Please document this clearly!
4. Write an infinity loop. How can program execution be stopped from the console, so that the infinity loop breaks?
5. What is the difference between this construct:

```
for (int i = 1; 5 >= i; ++i) {
        for (int j = 1; 5 >= j; ++j) {
            System.out.println(i * j);
        }
}
```

   and this construct:

```
for (int i = 1; 5 >= i; i++) {
        for (int j = 1; 5 >= j; j++) {
            System.out.println(i * j);
        }
}
```

   Explain the different outputs of those examples!
6. Execute any program stepwise with the debugging features of the IDE you're using.
   a) Show, how debugging works with log-output.
   b) Show, how conditional breakpoints work.
7. Write a program asking the user for five integers (Don't use arrays!). The program then outputs the sum of squares.
   a) Learn how you can start a Java program from the Windows/macOS console, i.e. without the IDE (i.e. without IntelliJ IDEA)! – Document clearly how it works! – Your knowledge about using the console should help you.
   b) A menu should be implemented with following options: [input numbers], [calculate sum of squares] and [quit]. All options (except [quit]) return to the menu after their action has taken place. Erroneous user inputs (e.g. text instead of an integer) should be checked by the program and require new input from the user.
8. Write a new program, which prints following output to the console:

```
Terminal
PC:src hers$      22
PC:src hers$     4444
PC:src hers$    666666
PC:src hers$   88888888
```

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!
- 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!
- 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.