Prof. Dr. Christoph Bockisch
MSc Steffen Dick
Fachbereich Mathematik und Informatik
AG Programmiersprachen und -werkzeuge

Philipps Universität Marburg

Companion exercises
Objektorientierte Programmierung: Wintersemester 2021

No. 1, due until 1.11.2021

Attention: This exercise is supposed to guide you through the steps to set up your environment and get a basic understanding of ILIAS. You should work on this exercise individually.

Task 1.1: Yanna kuzu peekay — 1 Point

This task is meant to help you with setting up Java.

a) First of all, we require at least version 13 of the Java SE Development Kit (JDK 13). If you do not have Java installed, you may install the latest version (JDK 17). We recommend Oracle's implementation: `https://www.oracle.com/de/java/technologies/javase-downloads.html`

b) If you use Windows, you need to change some of your system settings to be able to use java from your terminal. You can use this guide provided by Oracle: `https://www.java.com/en/download/help/path.html`

c) Open your terminal. If you use Windows, press *Windows button* + *R* simultaneously and type `cmd` into the window that appears. Alternatively you can just press *Windows button* and just type cmd and the correct programme will be found. Use the terminal to execute the `java -version` command.

   Copy the result of the command into a new text file. You may copy text from your terminal by pressing *CMD* + *M* on your keyboard. You then need to mark the text you want to copy with your cursor. Lastly, press *Enter* to copy the text.

You will need the JShell for future tasks. Should you have set Java up correctly, you will be able to call the JShell from your terminal by typing *jshell* into your terminal. If you have had any compilcations during your installation, you may call the JShell from the directory where you installed your JDK to.

Normally, you can find it in *C:\Programme Files\Java\jdk<version>\bin\jshell.exe*

Task 1.2: Going E-Postal — 0 Points

All important updates regarding the lecture and exercises will be sent via ILIAS. It is important that you can be reached on your students-address. You can enable automatic forwarding via `https://admin.students.uni-marburg.de/change-forward.html` should you not use your HRZ-account on a daily basis.

Task 1.3: String theory

4 Points

For this task you will need the JShell.

Copy your input and output of the following tasks into a new text file.

a) You already know the statement `System.out.println("Hello World");` from the lecture. Change that statement in such a way that your favourite starter Pokemon will be displayed instead of *Hello World*.

b) Java has a few different types that represent numbers to achive calculations. One of these is **`int`** which represents whole numbers.

   Use `System.out.println` to output `42 + 23` to the console.

c) Change your calculation to compute `"42" + "23"`.

   Which outcome did you expect? What actually happened?

d) Change the `"Hello World"`-example to display the current time in milliseconds as well. To get the current time in milliseconds, you may use `System.currentTimeMillis()`. Append the time to `"Hello World"` by using `+`.

Type */v* into the JShell and copy its output into a new text-file.

Task 1.4: Of apples and oranges

2 Points

You will need the JShell to complete this task.

Copy your input and output of the following tasks into a new text file.

a) Java uses variables to store values for later use. Every variable gets a name and a type when declared, though the value can be set and changed later.

   Declare and initialize variables with the following names, types and values:

   - `thirtyfive` of type **`int`** with `35` as its value.
   - `initial` of type **`char`** with the first character of your last name as its value.
   - `pi` of type **`double`** with `3.14` as its value.
   - `product` of type **`int`** with `42 * 23` as its value.

b) With the help of logical values, you can check if statements are true or false. Java uses the **`boolean`**-type for that.

   Write down your expected outcomes and then use **`boolean`** expressions to compare the following:

   - `thirtyfive` with `5 * 7`, using `==`
   - `pi` with `3.14`, using `==`

   Use **`boolean`** expressions to check if

   - `thirtyfive` is bigger than `34`. Use `<` or `>` to accomplish this.
   - `thirtyfive` is bigger than or equals to `pi`. Use `<=` or `>=` to accomplish this.

Type */v* into the JShell and copy its output into a new text-file.

Task 1.5: Circus of Values | 4 Points

Algorithms have uses in different daily activities. An algorithm always needs a prerequisite and a postcondition to function properly.

Just think about vendor machines. You select a snack, you pay for said snack and you get your change. For simplicities' sake we assume that every snack costs about 72 cents. Let's also assume that this particular vendor machine only accepts coins or a 5€ note.

Describe (not implement) an algorithm that takes any amount of money (up to 5€) and outputs the change in coins. Should the amount not suffice to buy a snack, the output should simply be the input amount in coins.

Furthermore, think about how you would ensure that your algorithm does what it is supposed to. Which values would you need to call your algorithm with in order to do so?

Available coins:

1ct, 2ct, 5ct, 10ct, 20ct, 50ct, 1€, 2€

Example:

Input: 3.14€
Output: 2ct, 20ct, 20ct, 2€

Task 1.6: Resistance is futile | 1 Point

For your submission you will have to create **one** ZIP-file that contains every text-file that you created in your tasks. The title of the ZIP-file should be your last name. Within your exercise group within ILIAS you can hand your ZIP-file in.

Please use "Upload File" instead of "Upload multiple files as ZIP" to hand in your submission.

Attention: You can find an exercise called "Abgaben" within *your tutorial*. Upload your ZIP-file to "Übung 01" via "Submit Exercise". You may change your submission until the due date.

Should you be having trouble or questions while setting things up, please visit your tutorial. Your tutor will help you make things work.