HÁSKÓLI ÍSLANDS
Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild

HBV202G: Software Design and Construction · Spring 2025

Andri Valur Guðjohnsen · Dr. Helmut Neukirchen

Assignment 1 · Due Tuesday 28.1.2025 / Friday 31.1.2025

Formalities:

- Submit your PDF solution as a team via Gradescope. Remember to select your other team member during upload and to mark where each answer of the questions can be found. (This all speeds up grading.)

Objectives: demonstrate that you learned some of the most important features of your IDE while going through the IDE's tutorials.

- Please add the name of your *IDE* and *Operating System* (keyboard shortcuts are sometimes different for, e.g., Mac OS) in front of **each** of your answers (while grading, Gradescope shows only the current answer, i.e. any other context gets lost).
- In most cases, it is sufficient to answer by naming the keyboard shortcuts or the mouse actions to activate the features mentioned below. Answer by mentioning the faster approach, i.e. *favour keyboard shortcuts*/key presses over mouse clicks, e.g. for copying text, you should rather answer "CTRL-C" instead of "Click on the Edit main menu entry, then click on Copy".
- An example answer might therefore look like: "IntelliJ, Windows: Ctrl-F10" (Mac users would rather write: "IntelliJ, Mac: Cmd-F10").
- If you work with different IDEs in your team: decide on one IDE to provide the answers for; but each team member should find the solution for their respective IDE (so that they are able to use that feature in future).

1. In your IDE, how do you use *code completion* (e.g. type just the first letter(s) and then the IDE offers to complete the rest)?
2. In your IDE, how can you let the IDE *fix problems* for you (e.g. make the IDE create a skeleton of a new method if you call a method that has not yet been defined)?
3. In your IDE, how do your *re-format the source code* (e.g. fix indentation)?
4. In your IDE, how do you *jump to the declaration* of an identifier (e.g. your cursor is on a method name and you want to go the source code where that method is defined)?
5. In your IDE, how do you *find* out who *references* an identifier (e.g. your cursor is on a method name and you want to find out who calls that method)?
6. From those IDE *features that you did not know before going through the tutorials*, but then learned by going through the tutorial: which one do you expect to be in future *most useful* for you? Describe with one sentence what that feature is about and the keyboard shortcut or the mouse actions to execute it. Briefly (ca. 1–3 sentences) justify your answer, i.e. why your think this feature is so useful.

7. Demonstrate that you can use the debugger of your IDE:

   (a) Clone from GitHub the following project as Java project into your IDE:
   `https://github.com/helmutneukirchen/JUnit381debugExample`

   (b) Run `junit.samples.SimpleTest.main(String[])` – on the IDE's text output, you should see `1 error` and `2 failures` being displayed. Go to the source code location of the first failure, i.e. line 45 of the `testAdd()` method and set a breakpoint in that line and run that main method now with your IDE's debugger. When the debugger stops at that breakpoint, inspect some variables and write down their values in your assignment solution:

   - `result`
   - `this.fname`

   (c) The debugger shows also the stack frames, i.e. the *call stack* of each method that calls another method and the local variables stored in that stack frame. When you go down from top of the call stack, you find at the top of that stack the current method (*testAdd* from the package `junit.samples`) that was called by some Java-internal methods (from Java packages such as `java.lang` or `jdk.internal`).

   Before all these Java-internal methods were called, what is the name of the last method that made a call from a class that is from package `junit.framework`? Write in your assignment solution down the name of the method and its class.

   (d) From the call stack, the method `runTest` in class `TestCase` has a local variable `runMethod` (of type `Method`) that in turn has a local variable `clazz` that in turn has a local variable `name`. Write down the value of that local variable `name` in your assignment solution!

   (e) Return now to the top element stack frame of your call stack, i.e. the `testAdd` method: If you now do with your debugger *exactly four times* a "Step into", in what method do you end up? Write down the method name in your assignment solution!

   (f) Describe in your *own words*, what "Step out" (or sometimes called "Step return") does!

   (g) You are now finished and can stop your debugger.