Assignment 1

Dagur Logi Ólafsson - dlo4@hi.is

Bergur Ísak Ólafsson - bio19@hi.is

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)?**

   Vscode, Windows: TAB

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)?**

   By pressing Ctrl and . we get a menu with options on how to fix the problem.
   Vscode, Windows: Ctrl + .

3. **In your IDE, how do your re-format the source code (e.g. fix indentation)?**

   Vscode, Windows: Shift + Alt + F

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)?**

   For my computer the F-keys have a secondary function so I need to hold Fn while pressing the F-keys
   Vscode, Windows: Fn + F12

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)?**

   Vscode, Windows: Fn + Shift + F12

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.**

   In a project that contains a large amount of code I think that using Fn + F12 (the jump to a declaration of an identifier) or even Fn + Shift + F12 to find the references of an identifier will be of huge value in the future.
   As well as reforming the source code with Shift + Alt + F, by not having to stress to much of indentation.

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**

      With Vscode open and navigated into the folder where we want the project to be stored, we navigate into the top left corner of the screen and choose:
      Terminal -> New Terminal
      In the new Terminal we write:
      "git clone **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

      With the project now in our folder we find the SimpleTest in the files.
      We expand the "JUnit381debugExample"
      Then we expand the "Sample" section
      From there we click on the SimpleTest.java to open up our code.

      Now in the code we run it with the commands Ctrl + Alt + N
      We get the error message.
      Now we have to set the breakpoint, we find line 45 and click on the empty space on the left side of the line number.
      Now we start the debug by pressing Fn + F5
      We see that result is 5.000000

      Now to find this.fname we navigate to the left of the screen into the debugger menu and find the Variable line, by expanding that we get two options and one of them is this = SimpleTest@19
      We expand that by clicking on the "eye" or right clicking and choosing Add to Watch.
      By expanding that we get the name which is "fname = testAdd"

      result = 5.000000
      this.fname = testAdd

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 Javainternal 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.**

We start by going to the top of the list and observing the SimpeTest.testAdd function and by hovering with our mouse over the name of the right side of the debugger menu, over the SimpleTest.java we can see where it was called from. By going further down we see that junit stops on the next 3 calls and on the fourth “TestCase.runTest()” we can see that it accesses junit.framework.TestCase.
Therefore we have found what the method name and class name is.

Method name:.runTest
Class name: junit.framework.TestCase

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!**

We navigate to the top of the debugger menu to Variables -> Local -> runMethod
We can expand that by either clicking the “eye” or right clicking and choose “Add to Watch”.
By expanding that we navigate to clazz and expand that
By going down from there we find “name = “junit.samples.SimpleTest””

Local Variable Name = “junit.samples.SimpleTest”

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!**

We navigate into the Call Stack and make sure that SimpleTest.testAdd() is chosen.
After that we see on the top right side there are four symbols and we find the “step into” symbol and press that 4 times.
After stepping into 4 times we have changed the call stack into Assert.fail(String)

Method Name = Assert.fail

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

   Stepping in will take you deeper into the code by entering the method that is currently being called, allowing you to see how the code behaves step by step.

   Step out will finish executing the current method, including any remaining iterations of loops or statements, and then return to the line in the calling method where the current method was invoked. It effectively moves back up one level in the call stack.

g. **You are now finished and can stop your debugger.**
   Vscode, Windows: Ctrl + C