

Step 1: Install Java and Subversion

We assume that you already have Ubuntu Linux 18.04 installed.

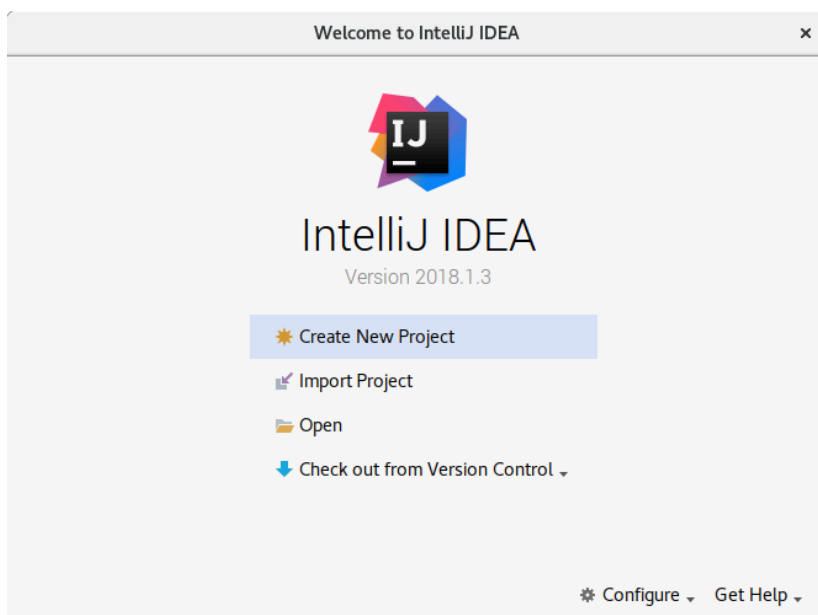
It is recommended to use Oracle Java 8. A tutorial on the installation steps can be found at <https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-on-ubuntu-18-04#installing-the-oracle-jdk>.

To install Subversion just use the command line by typing “sudo apt install subversion”.

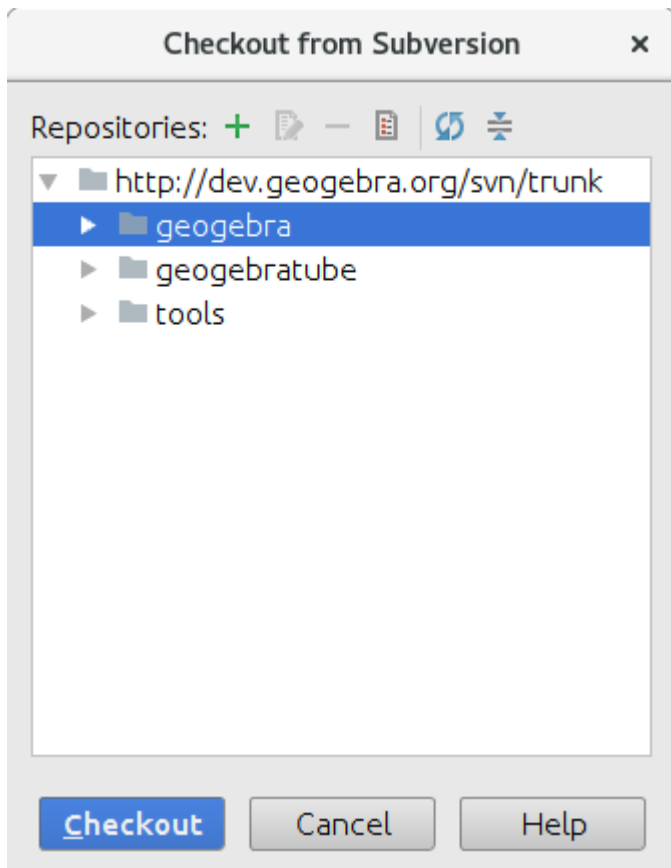
Step 2: Download and install IntelliJ IDEA

We will use the Community Version 2018.1.3.

Step 3: Start IntelliJ IDEA and import an existing repository



Choose **Check out from Version Control** > **Subversion** > **Repositories: +**.
Type: “<http://dev.geogebra.org/svn/trunk>”, press ENTER.



Select “geogebra” and press **Checkout**.

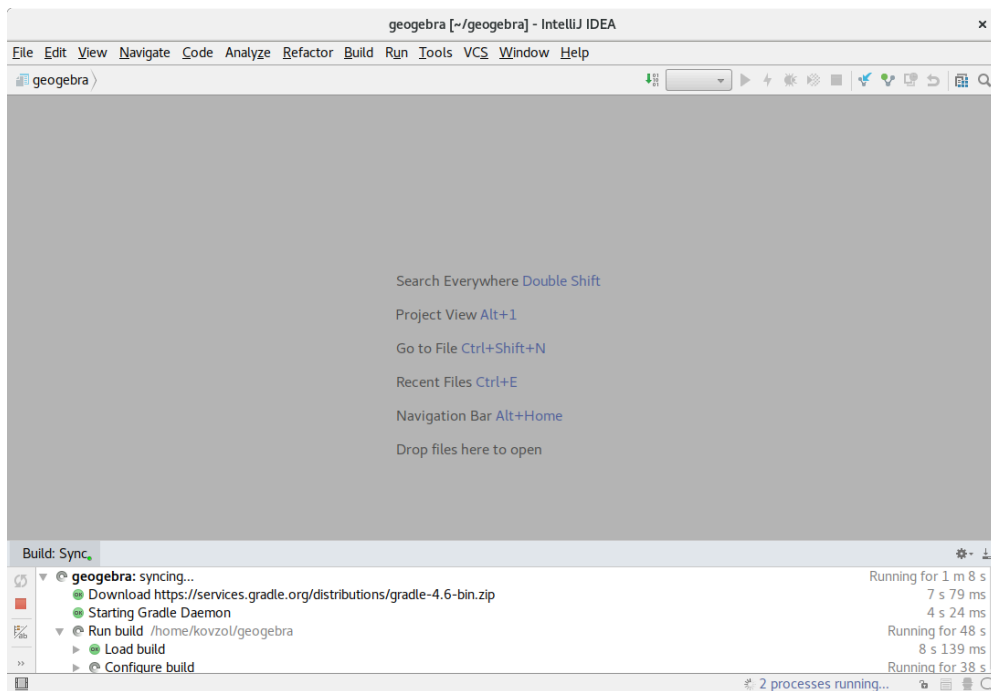
By using the New Folder icon, you may create a new folder “geogebra” and check out the remote repository into a local folder on your machine. It is your decision which folder name you prefer, it can be freely chosen. Also, at the next screen, you can fine tune how the directory layout will finally look like.

Choose the **1.8 format** when the **Subversion Working Copy Format** is asked.

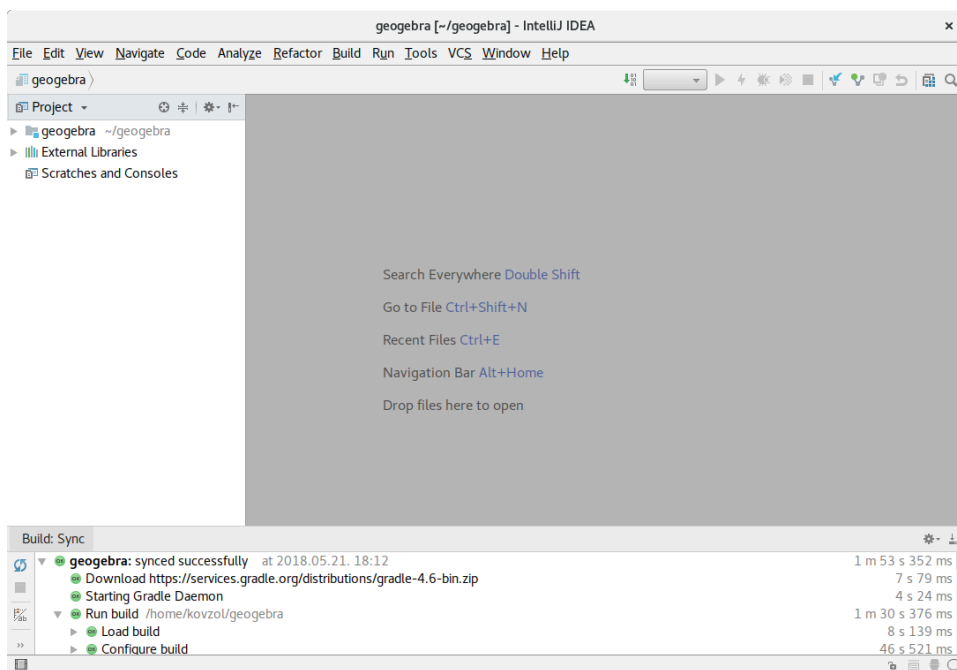
Now a question will be asked: “Would you like to create an IntelliJ IDEA project for the sources you have checked out to /home/USERNAME/geogebra?” Choose **Yes**. (Your USERNAME will be shown in the path.)

Finally choose **Gradle** at the **Import project from external model** option, and **Finish**. (In case IntelliJ is complaining about the **Gradle JVM**, just select the one you have, e.g. the folder **/usr/lib/jvm/java-8-oracle**.)

Now the GeoGebra project will be imported by IntelliJ IDEA. This will take a while.



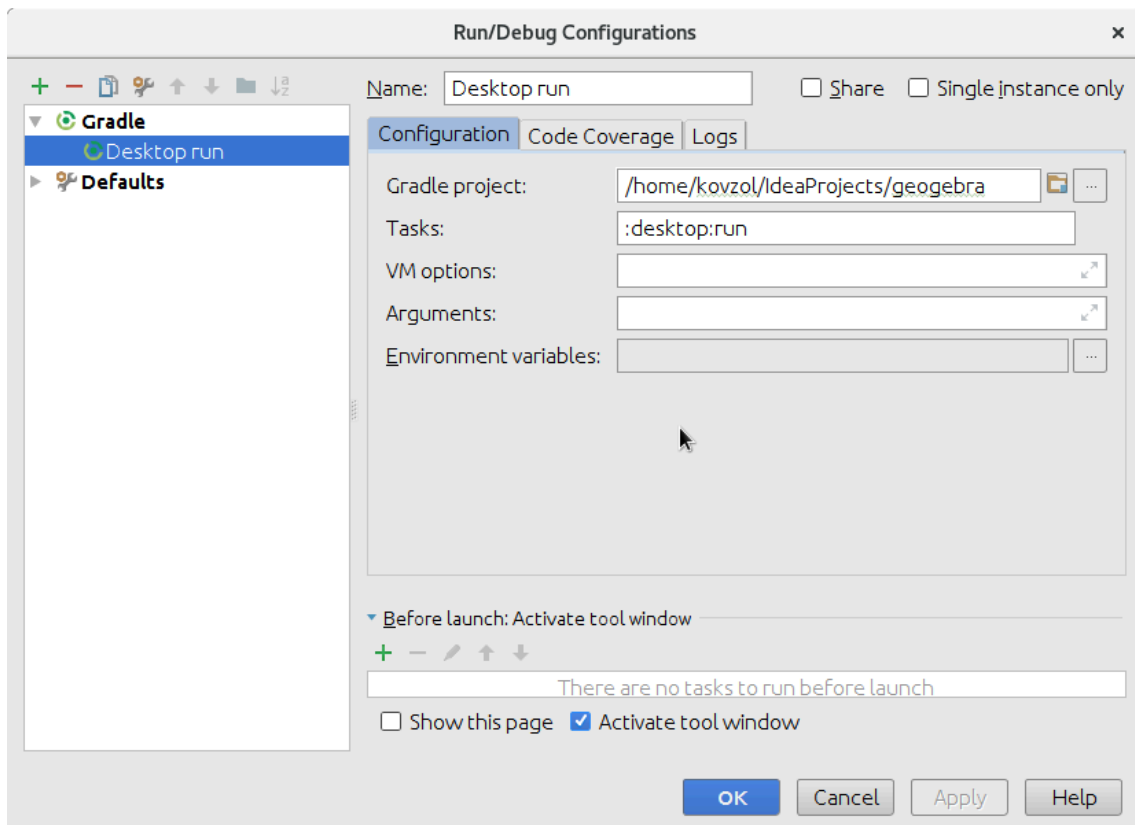
Finally you will see something like this:



If IntelliJ IDEA asks for “Gradle projects need to be imported”, answer by selecting “Enable Auto-Import”. This can be set later as well.

Step 4: Compile the desktop version of GeoGebra

To the right to the Build Project icon , in the top-down list, select **Edit configurations**. Add the following configuration (by using the + icon):



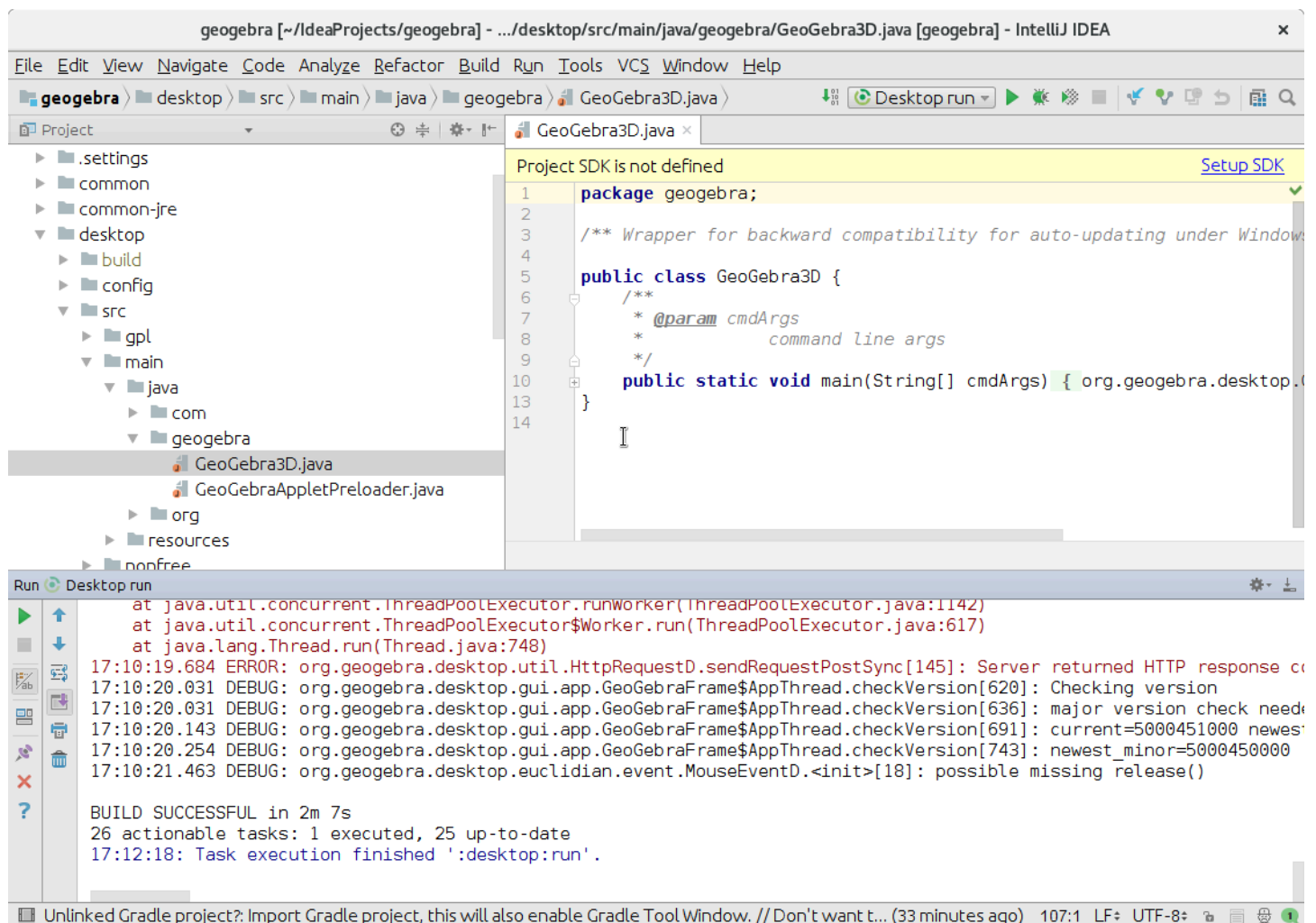
It is very important to set the **Gradle project** folder properly. It must be the same folder that contains the main build.gradle file. Note that the Name on the top is set to “Desktop run”, however, you can set another name if you like. (But we assume from now on that you call this configuration the same, too.)

Press **OK**, then select the “Desktop run” configuration and press the green “Play” button. During operation there will be several messages printed in the bottom part of the main window in IntelliJ IDEA. After several minutes you should see the GeoGebra logo and the application will start.

If you use a Java version different from the one by Oracle, you may have difficulties on compiling and running GeoGebra. So you really should use Oracle’s Java here.

Step 5: Edit the source code and recompile it

Open the directory structure on the left panel like this and open the file GeoGebra3D.java with double-click:

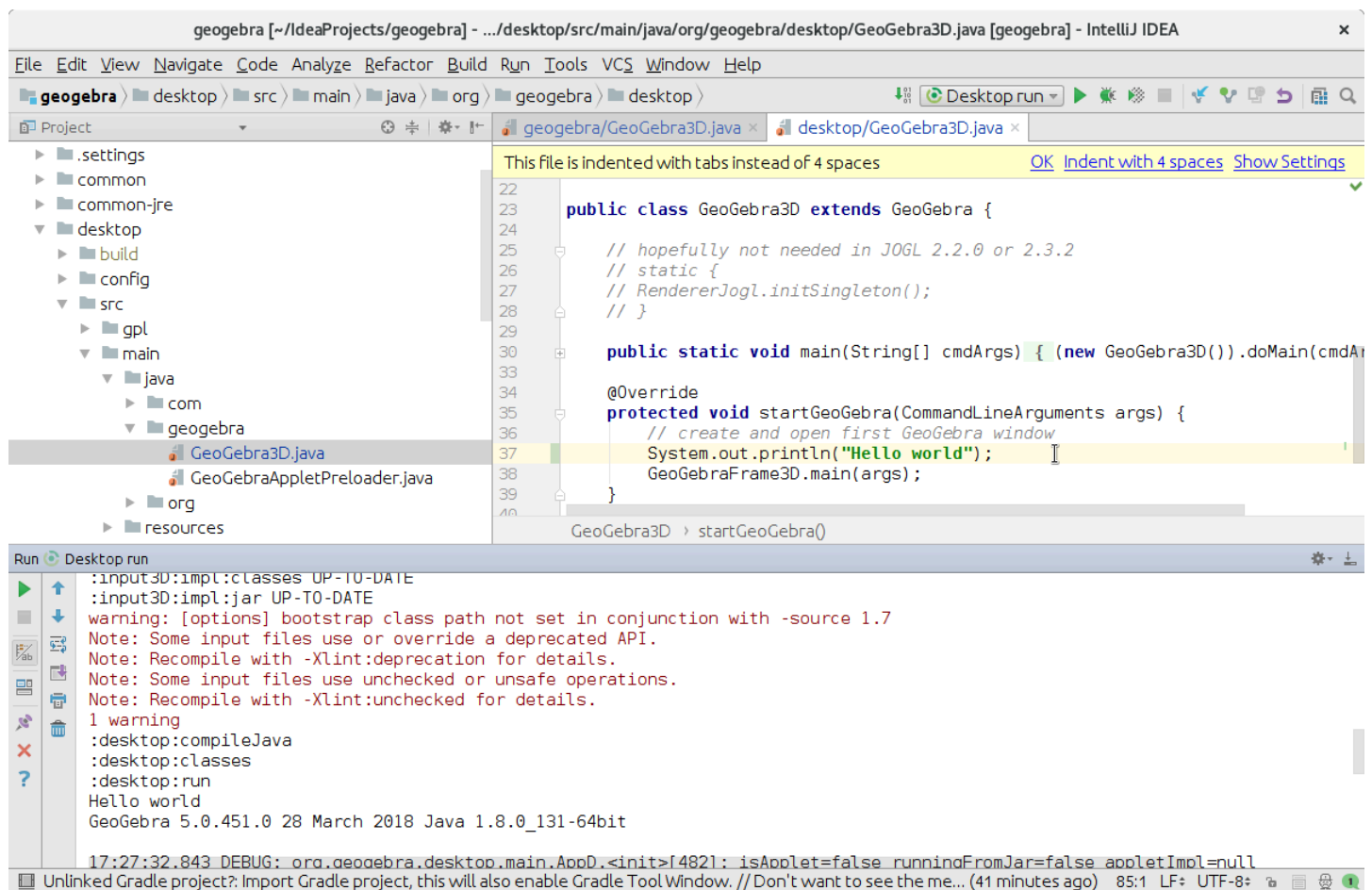


IntelliJ IDEA will try to automatically set up the Java SDK, but it may be incorrect. To check this, try to build the project first by pressing Ctrl-F9. If you get an error “Error:java: Fatal Error: Unable to find package java.lang in classpath or bootclasspath” at the bottom of the window (in Messages: Build), [you may need to remove the autodetected SDK and insert the correct one](#). Here is [how you can manually set the SDK](#).

Other errors like “Error:(3, 47) java: package com.himamis.retex.editor.share.io.latex does not exist” can be ignored.

If you don’t set the SDK properly, you will not have full Java support in IntelliJ IDEA. It will compile the source code, but you will not get code hints, full syntax highlighting and other useful kinds of support.

By using double (left) Shift, type GeoGebra3D and search for another occurrence of GeoGebra3D.java. (This part will be used only when running GeoGebra from command line, from a .jar file.) It should be placed as shown in the next figure. Go ahead and do a minor change on the code, then run it by clicking on the green Play button, and scroll in the bottom part of the IntelliJ IDEA window to search for your newly added code:



Step 6: Debugging

Click on the inserted line, press **Ctrl+F8** to set a breakpoint and click the Debug 'Desktop run' icon (**Shift+F9**) to start debugging. Now you can execute the statements line by line when pressing **F7** consecutively. Learn how to control the debugging flow on the left and bottom part of the IntelliJ IDEA main window. You will want to Resume Program (**F9**) and Stop 'Desktop run' (**Ctrl+F2**).

Step 7: Testing the web version

Compiling the web version takes more time and requires more resources. Create a new configuration and do the same as for the "Desktop run" configuration, but use the tasks `:"web:clean :web:draftRun"` instead.

Step 8: Using the superdev mode to debug the web version

Create a new configuration and do the same as for the "Desktop run" configuration, but use the tasks `:"web:superDev"` instead. Start it by the Play button, then, after a while a window titled "GWT Development Mode" will be shown. Select, for example, `app.html` as the Startup URL and click on "Launch Default Browser". A message will be shown "Compiling web3d" for several seconds. On how to continue, you may want to search for "GWT superdev mode" and "GWT superdev mode IDEA" on Google.

Step 9: Fine tuning

If you plan to be a real GeoGebra developer and commit some code someday, you need to change some settings in IntelliJ IDEA to have your code look the same as the GeoGebra codebase. In order to do that, consider changing the following:

1. Set the import order as Eclipse does by following <https://stackoverflow.com/a/17194980>.
2. Set automatic import by following <https://stackoverflow.com/a/11704462>.

3. If you see a warning that "This file is indented with tabs instead of 4 spaces.", just press OK.
4. You can pass command line arguments to GeoGebra with the Gradle option `--args`. See https://docs.gradle.org/current/userguide/application_plugin.html#sec:application_usage more on this. Unfortunately this still does not seem to work in IntelliJ IDEA 2019.x, only with a workaround. See <https://stackoverflow.com/questions/52267489/how-do-i-use-gradle-bootrun-with-args-in-intellij> for more details. There is another workaround: Use the terminal tab in IntelliJ and enter the Gradle command manually: **`gradlew.bat :desktop:run --args="..."`** (on Windows) or **`./gradlew :desktop:run --args="..."`** (on Mac/Linux).