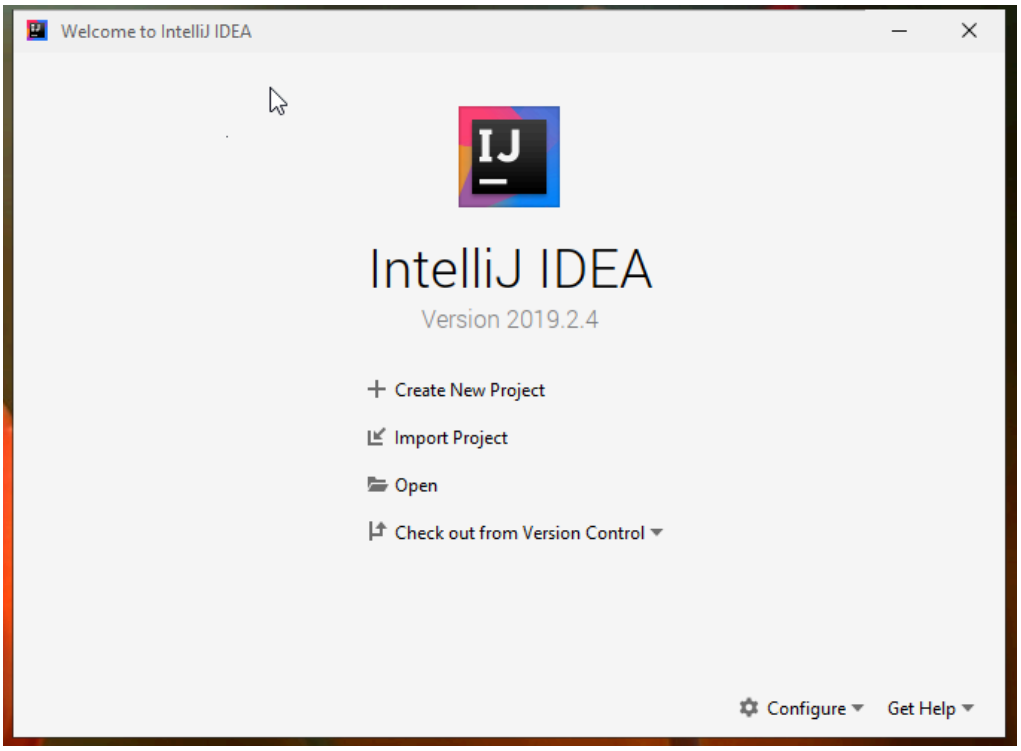
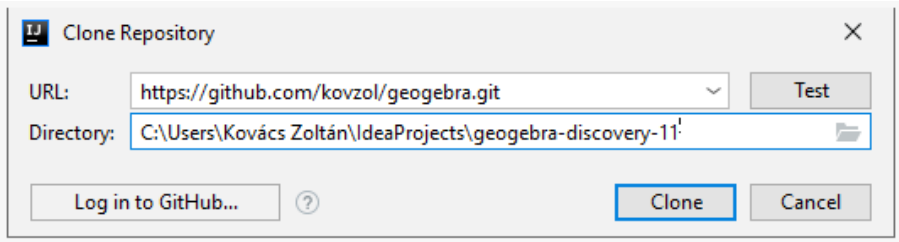
**Step 1: Download and install IntelliJ IDEA**

We will use the Community Version 2019.2.4.

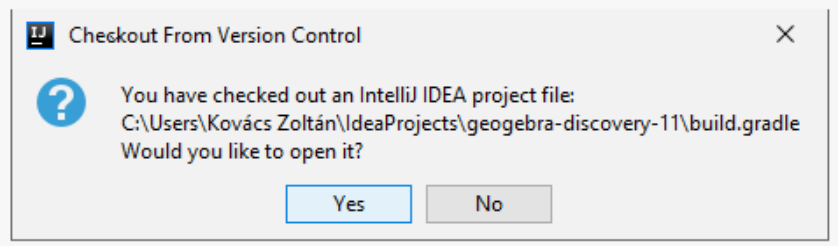
**Step 2: Start IntelliJ IDEA and import an existing repository**



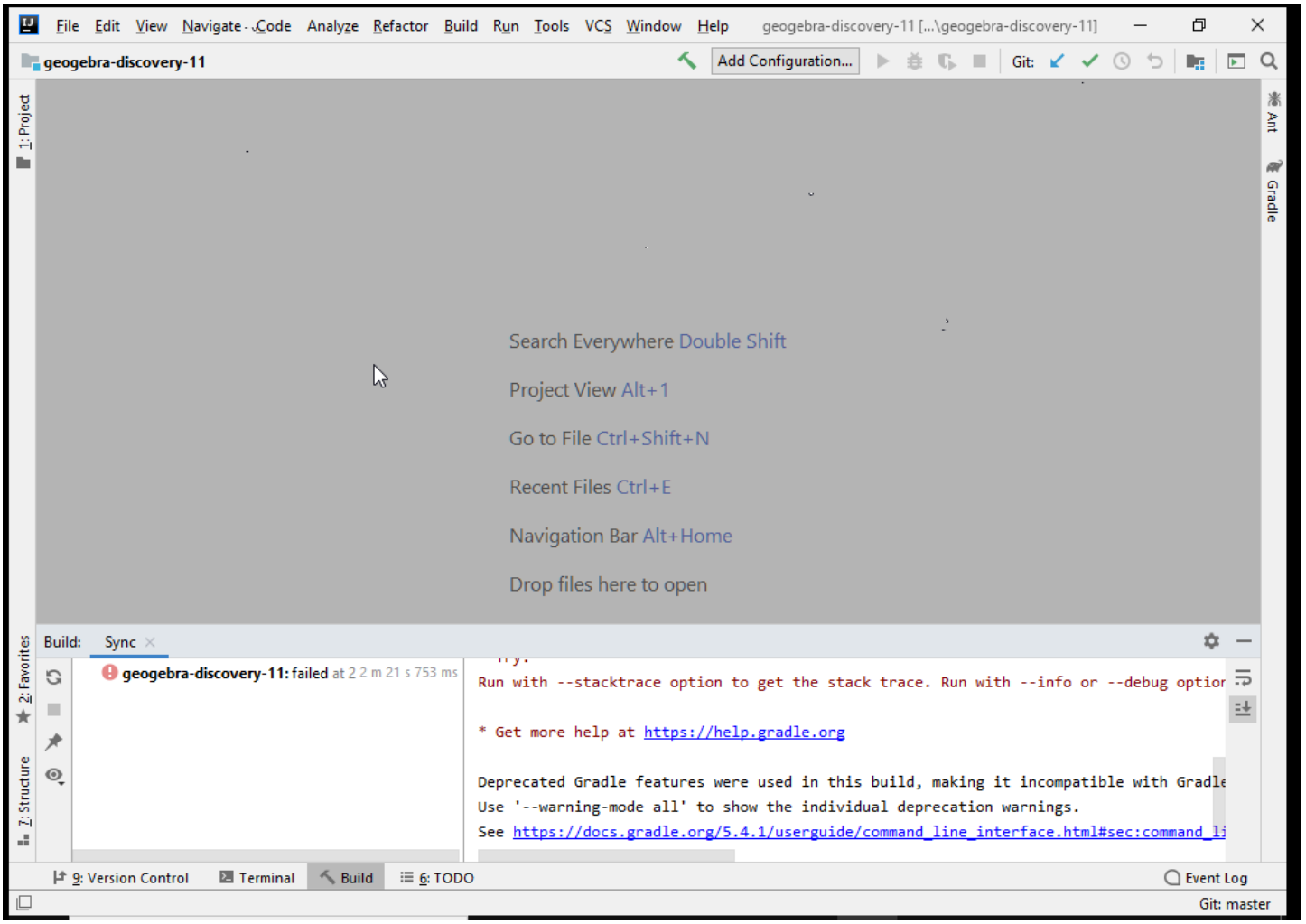

Select **Check out from Version Control**, select **Git** and enter something like the following:



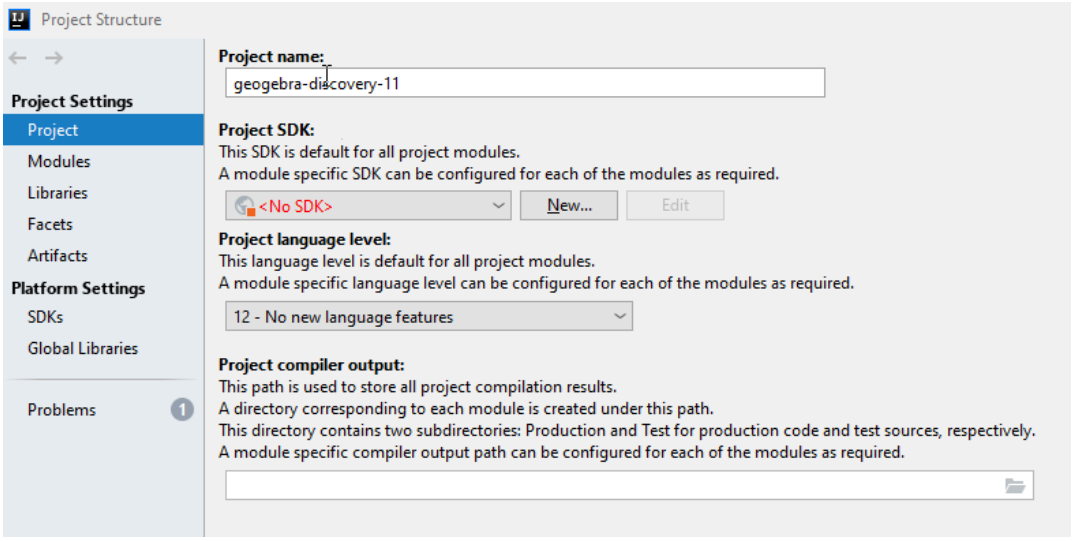

Note that the version you will download is not the official version of GeoGebra. It is an unofficial fork that provides some additional features like automated discovery, real geometry comparison, partial Java 11 support and some bugfixes, but maybe not completely up-to-date to the newest official version of GeoGebra.
Click on **Clone**. The cloning process will take several minutes. After then you will see something like this:



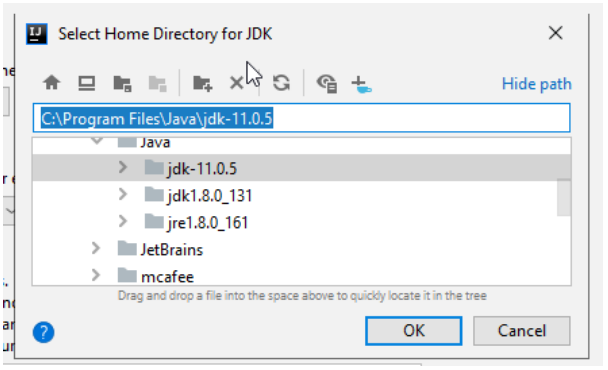

Select **Yes**. Then the following screen will show up:




That is, you get an error message. The reason is that Java 8 (that is installed on the system we are working with in this tutorial) is no longer fully supported in the used fork of GeoGebra, namely, in its Gradle build system. Therefore you need to download a more recent one. Oracle Java 11 SE JDK will be one working option. It is available on Oracle's web site, one may need to register to access to the Download area. After installing it go to **File** > **Project Structure...** > **Project Settings** > **Project** > **Project SDK** > **New...**:

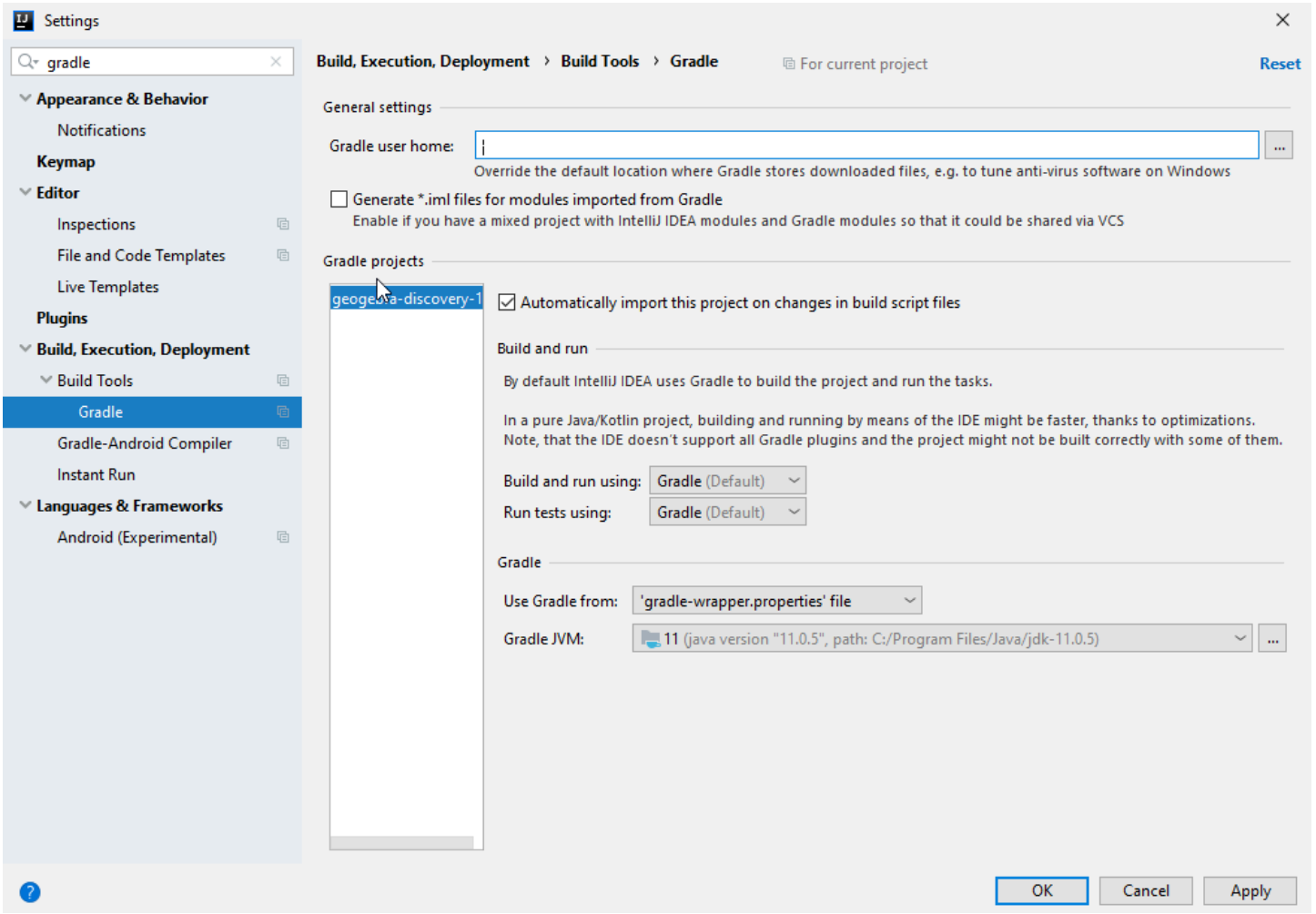



Select **JDK** and click on the freshly installed JDK:

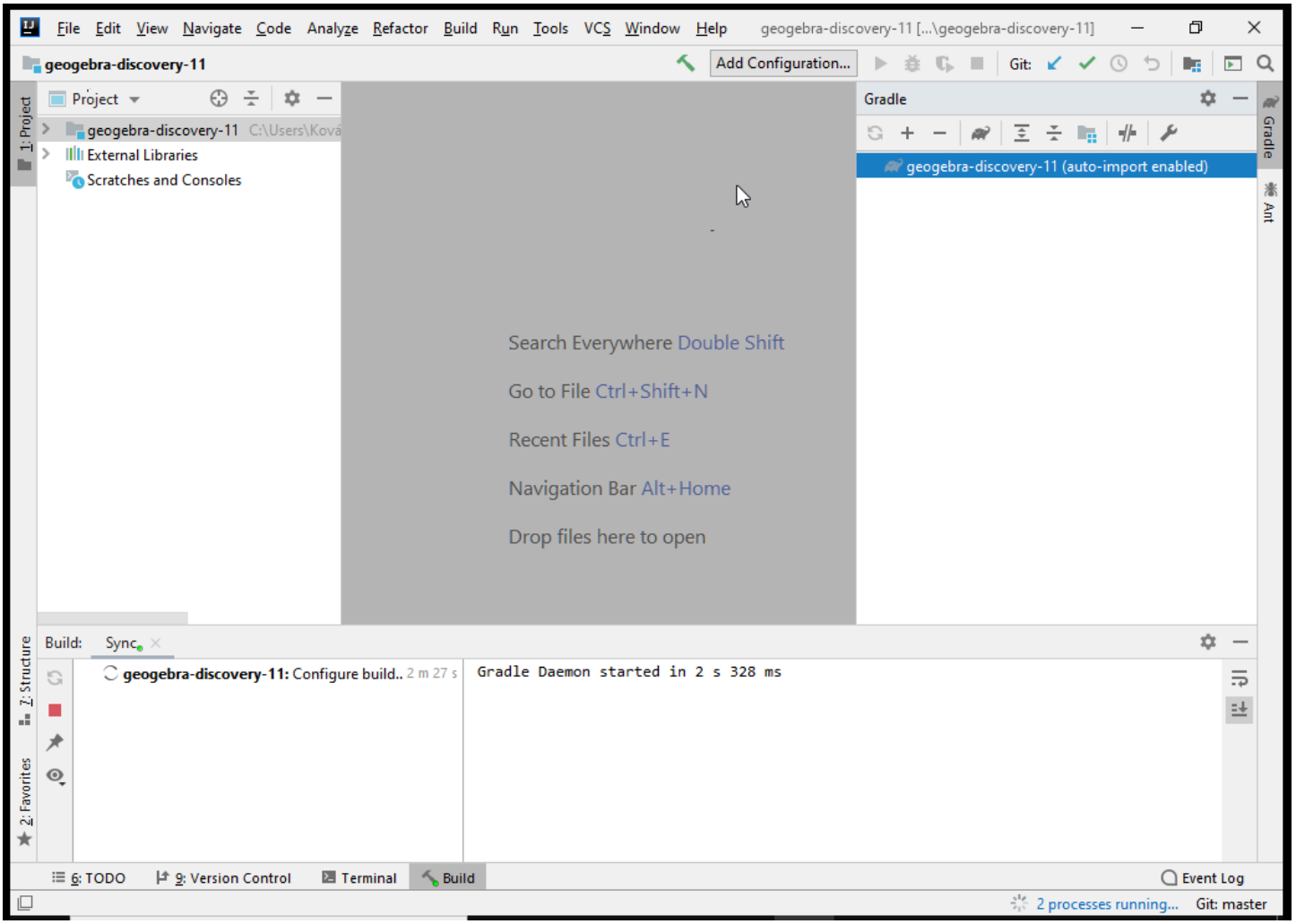



Now click **OK**.

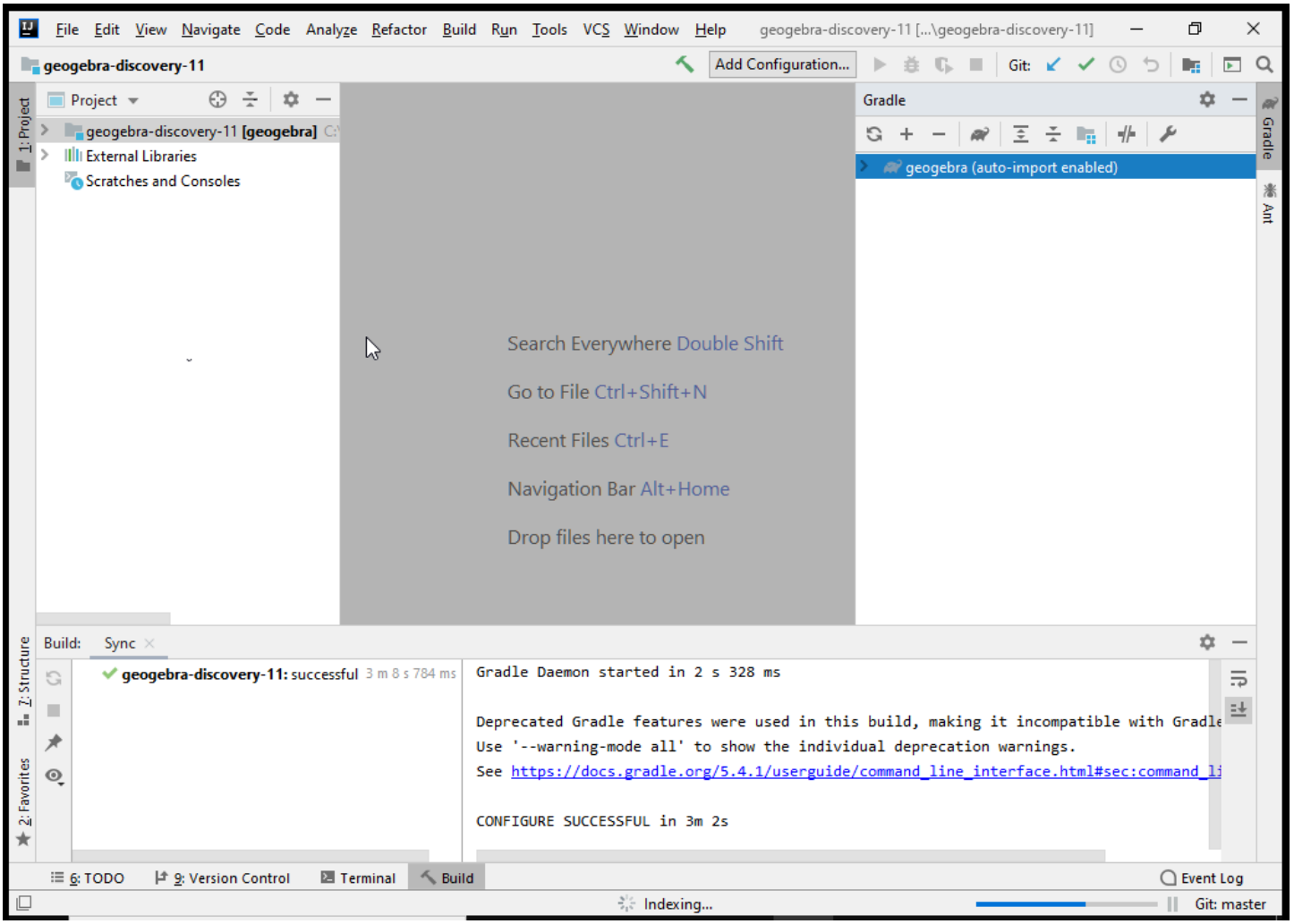
Then go to **File** > **Settings...** and type “Gradle” in the search field on the top-left, then change the **Gradle JVM** to version “11”:



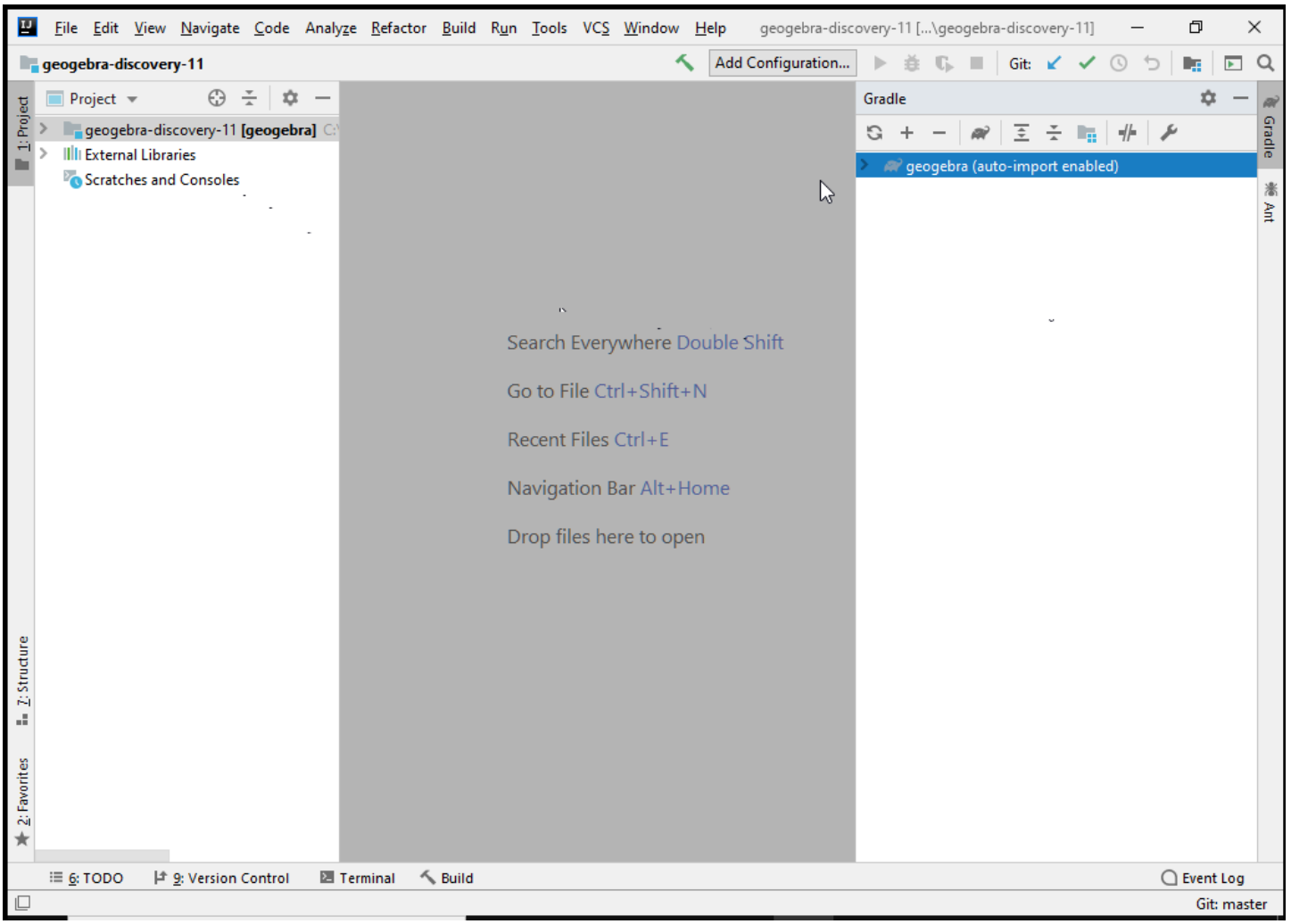

Now press **OK**. Click on the “Project” tab on the left and on the “Gradle” tab on the right to open them and click on the “Reload” icon on the top-left of the Gradle tab on the right:




After several seconds the error message will no longer appear, that is, the configuration will be successful:

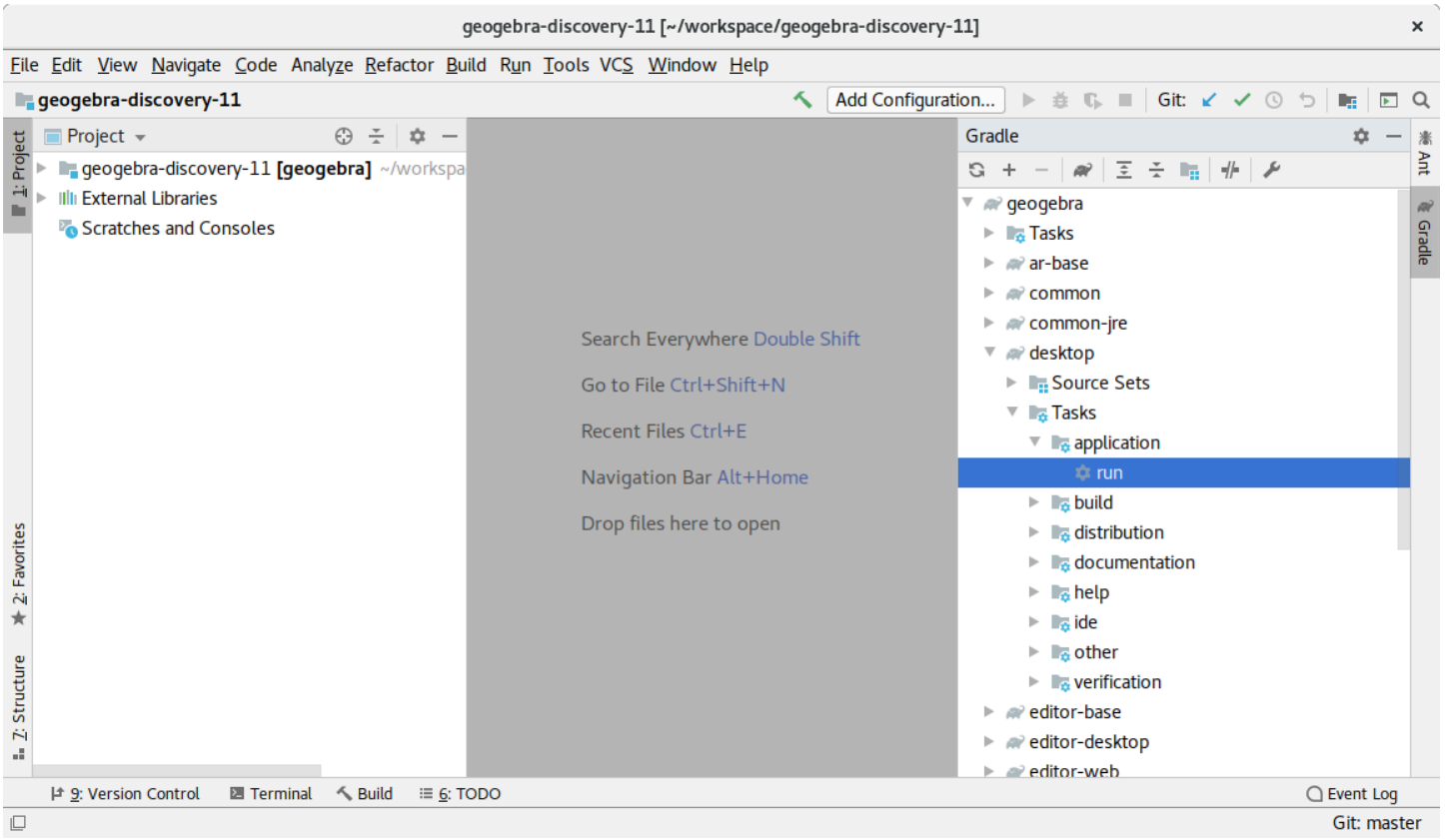



Now click on the Build tab (at the bottom): you will disable it. Then the following screen will be shown (in the meantime the option “enable auto-import” was confirmed, if you do not confirm it, you can do it later, but in that case the “auto-import enabled” option will not be displayed in parentheses):

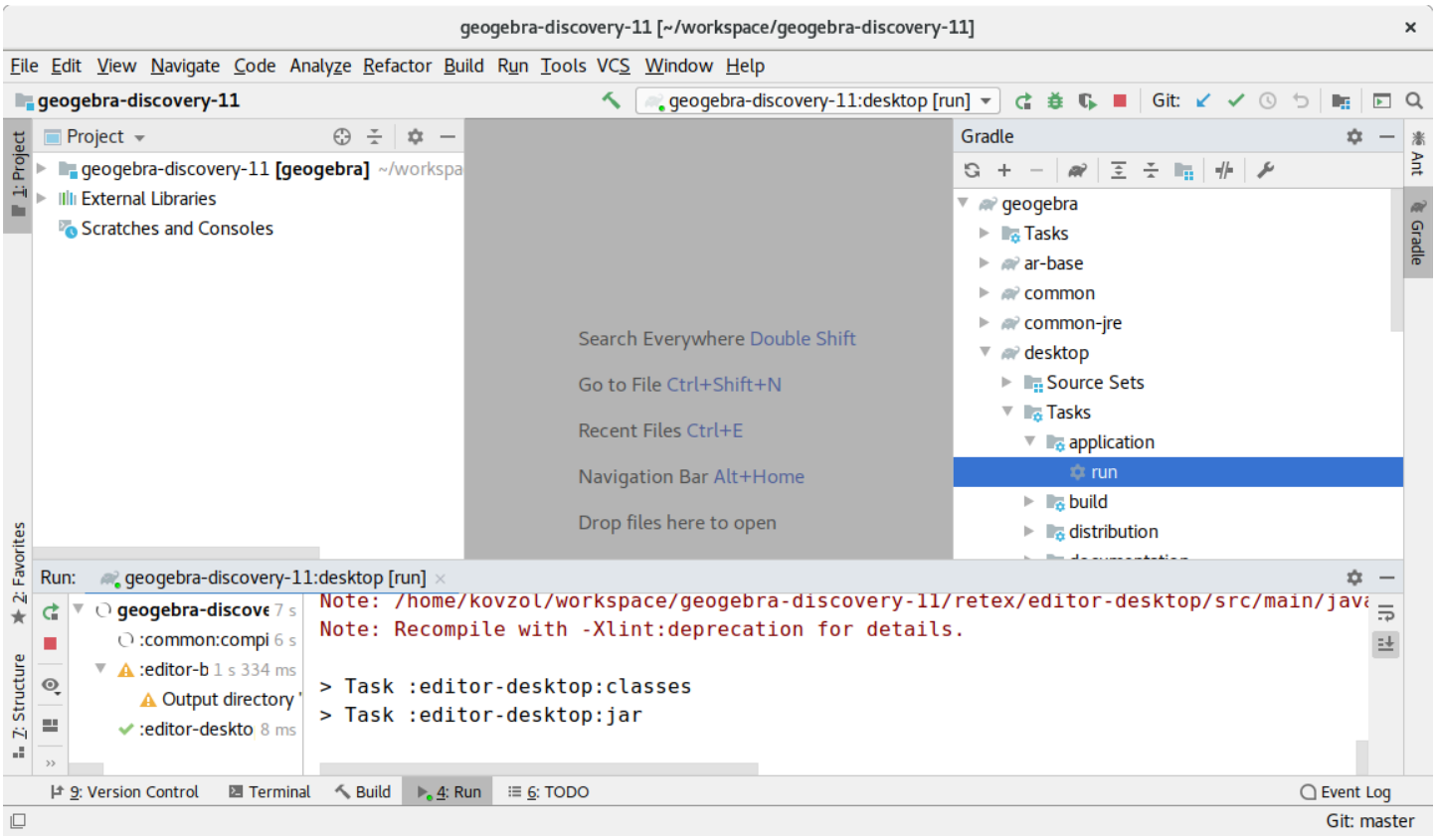



**Step 3: Compile the desktop version of GeoGebra**

(The following screenshots were created under Linux, but the look is mostly the same.)
Open the task hierarchy on the right in Gradle's window:



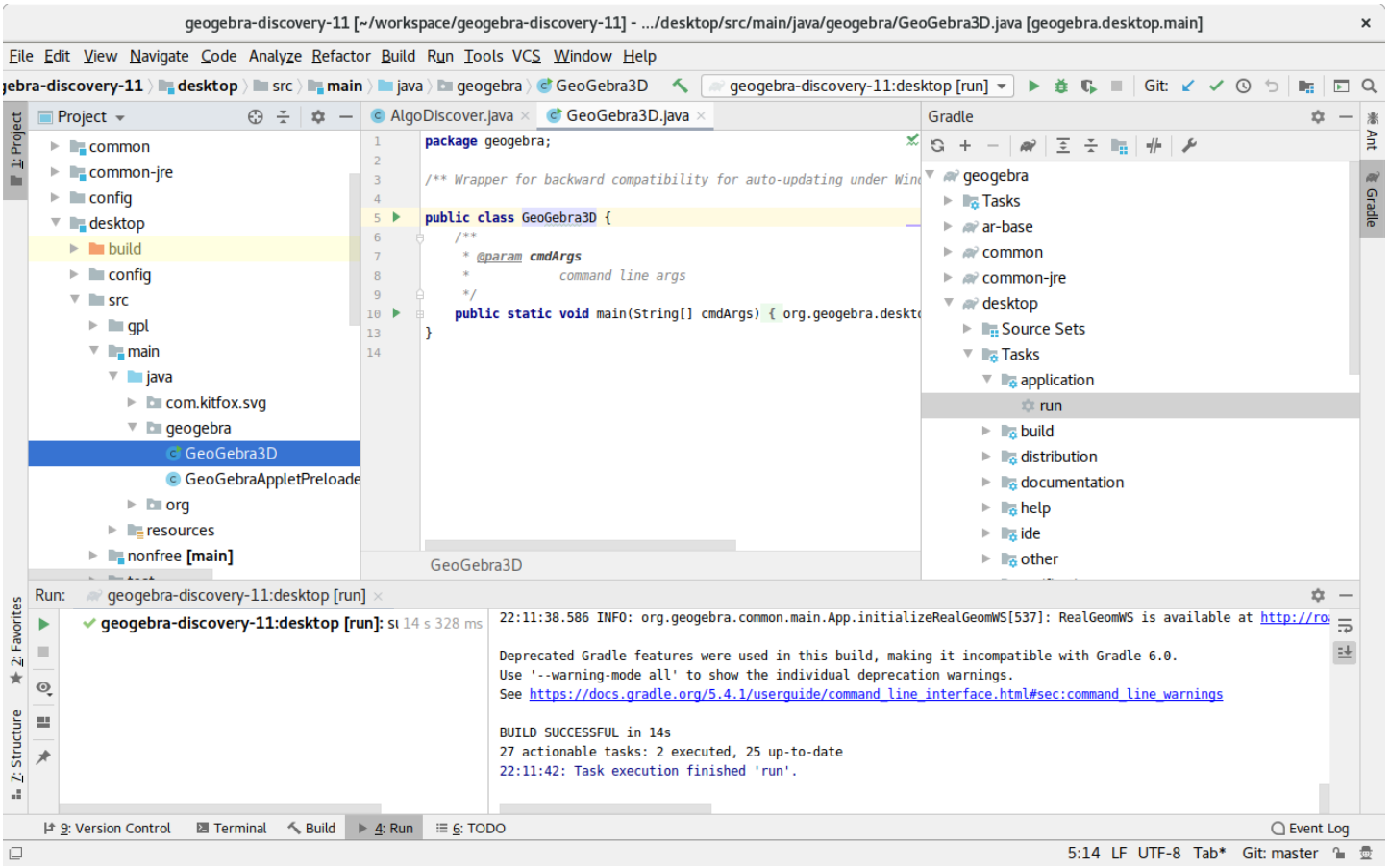

Start the task “run” by double-clicking on it:




After a few seconds the compilation will succeed and GeoGebra starts properly.

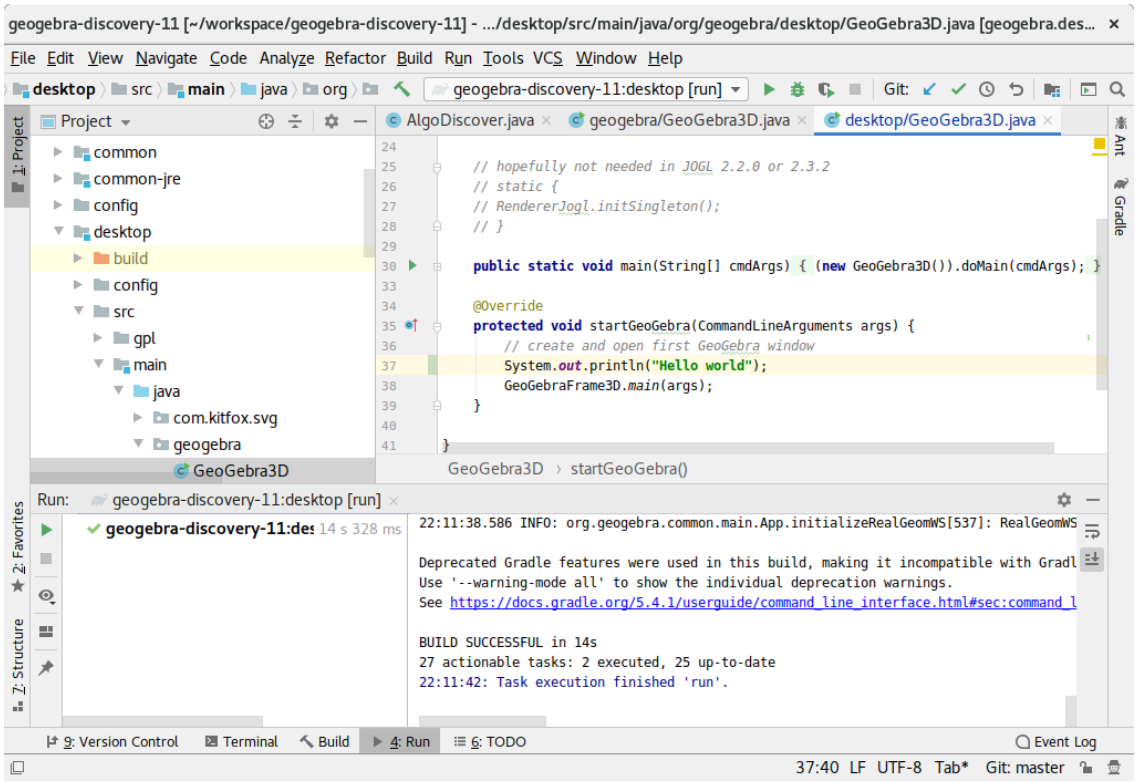
**Step 4: Edit the source code and recompile it**

By using double (left) Shift, search for the file AlgoDiscover.java. Then, by opening the directory structure on the left panel like open GeoGebra3D.java with a double-click:




Now close the Gradle tab on the right side. We prepare more space on the screen for programming with this step.

By using double (left) Shift, type GeoGebra3D and search for another occurrence of GeoGebra3D.java. (This part of GeoGebra 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 5: Debugging**

Click on the inserted line, press Ctrl+F8 to set a breakpoint and click the Debug 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 (Ctrl+F2).

**Further steps**

See an earlier version of this document to learn more (note that some steps may be outdated).