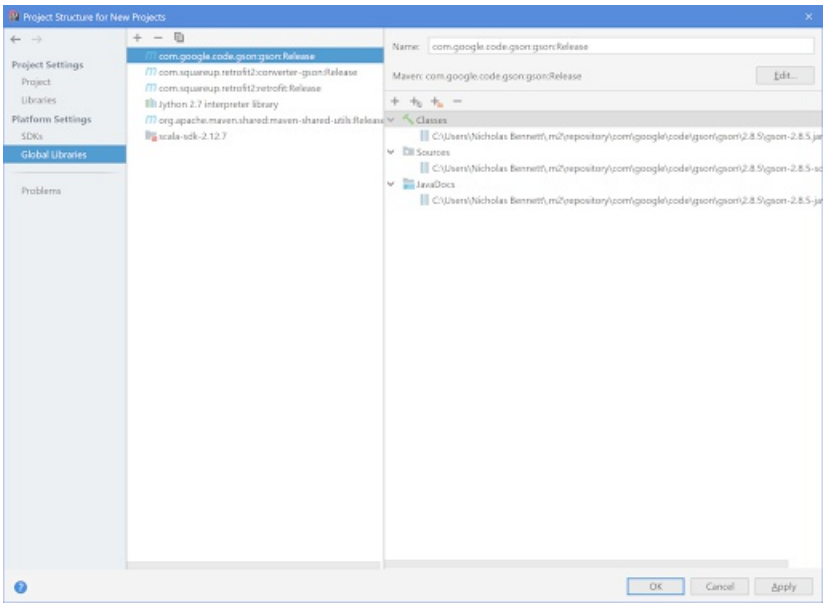
Incorporating JUnit5 into Your IntelliJ IDEA Project

Add the core JUnit5 library to IntelliJ IDEA as a global library

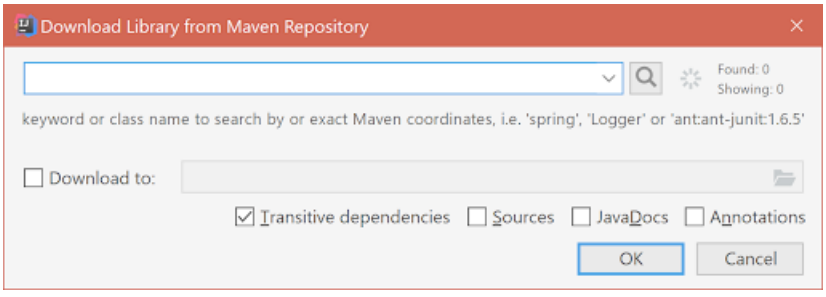
1. From the main IntelliJ IDEA workspace, select the **File/Project Structure** menu option. (From the Welcome to IntelliJ IDEA screen, select **Configure/Project Defaults/Project Structure**.)

2. In the left sidebar of the **Project Structure** window, select **Global Libraries** (in the **Platform Settings** section).

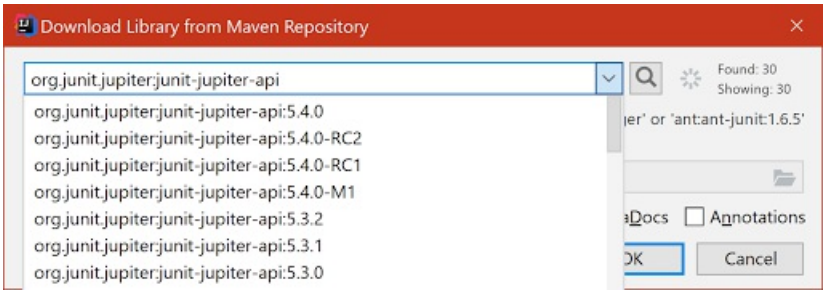


3. If you already have some 5.x version of `org.junit.jupiter:junit-jupiter-api` listed, then you already have the core JUnit5 library installed as a global library; proceed directly to "Add JUnit5 to an IntelliJ IDEA project".

4. Click the plus (+) sign above the list of libraries, and select **From Maven** from the menu that appears; the **Download Library from Maven Repository** dialog will appear.

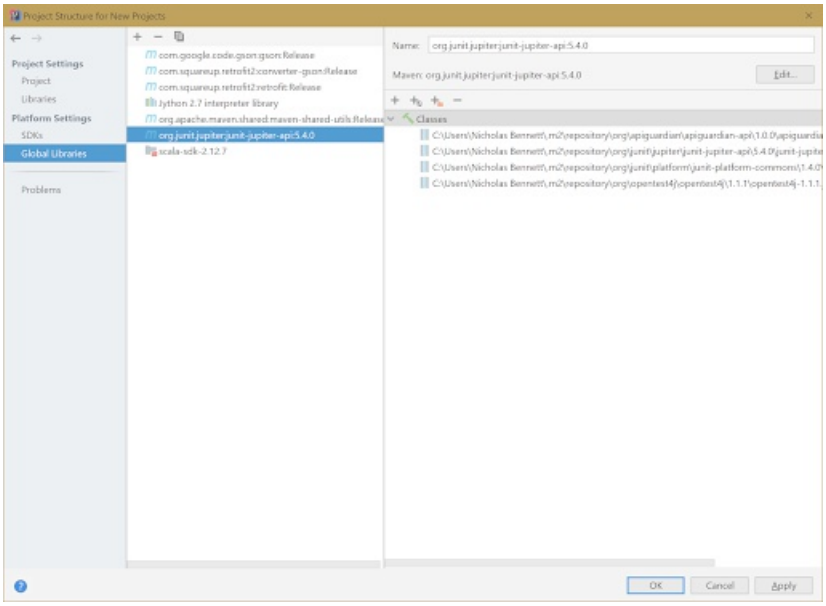


5. In the input field, type `org.junit.jupiter:junit-jupiter-api`, and click the search (magnifying glass) icon. After a few seconds, you should see the list of all available versions of the `junit-jupier-api` artifact in the `org.junit.jupiter` group.



6. From the list that appears, select the highest-numbered version that is not a release candidate (RC) or milestone (M) build. (In the screen capture above, this would be 5.4.0.)

7. Make sure that the **Transitive dependencies** option is checked, and click the **OK** button. All of the component files will be downloaded, and the selected library will appear in the list of global libraries.



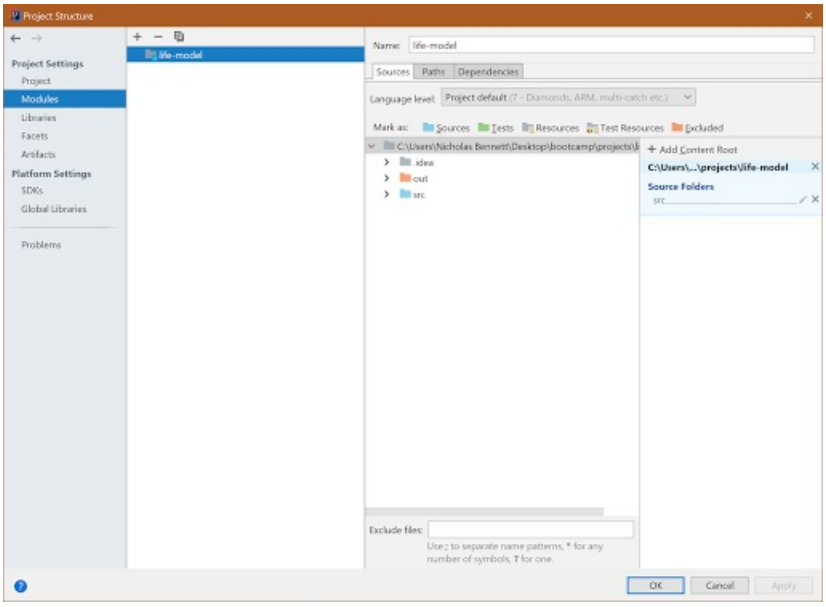
8. If you are going to use parameterized tests, repeat steps 4–7, using `junit-jupiter-params`, in place of `junit-jupiter-api`.

9. **Important:** Click the **Apply** or **OK** button to finish adding the library to IntelliJ as a global library.

Add JUnit5 to an IntelliJ IDEA project

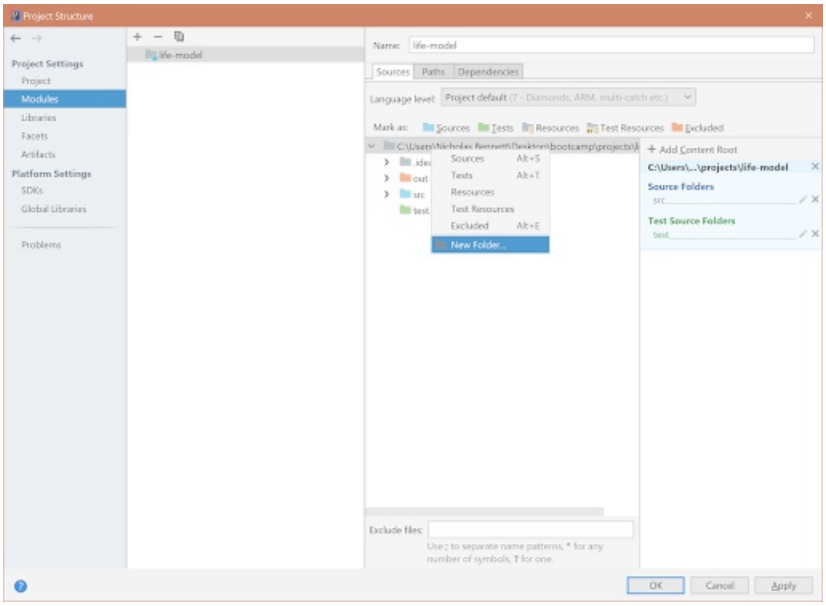
1. With your project open in IntelliJ IDEA, select the **File/Project Structure** menu option.

2. In the left sidebar, select **Modules** in the **Project Settings** section, select the module containing the code you want to test in the next panel, and click the **Sources** tab in the right panel.

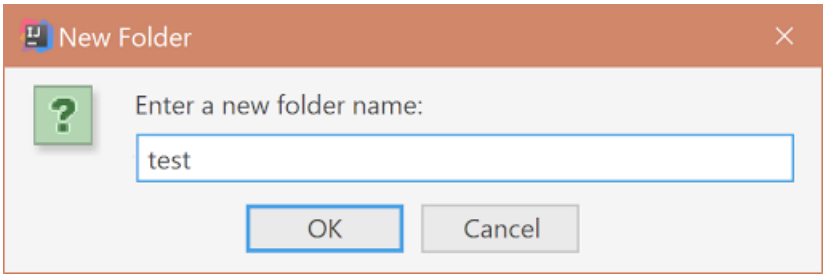


3. If there is already a directory marked as a test source folder, it will appear in green in the folder tree, and will be listed under **Test Source Folders**. If that is the case, proceed directly to step 7.

4. Add a new folder to the project by right-clicking on the root of the folder tree, and selecting **New Folder** from the context menu.

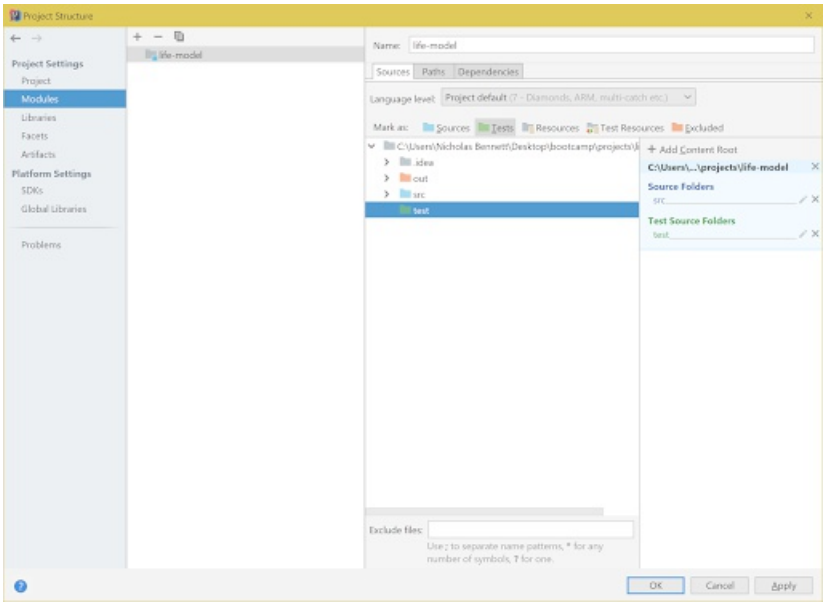


5. In the New Folder dialog that appears, specify an appropriate name (e.g. “test”, “tests”).

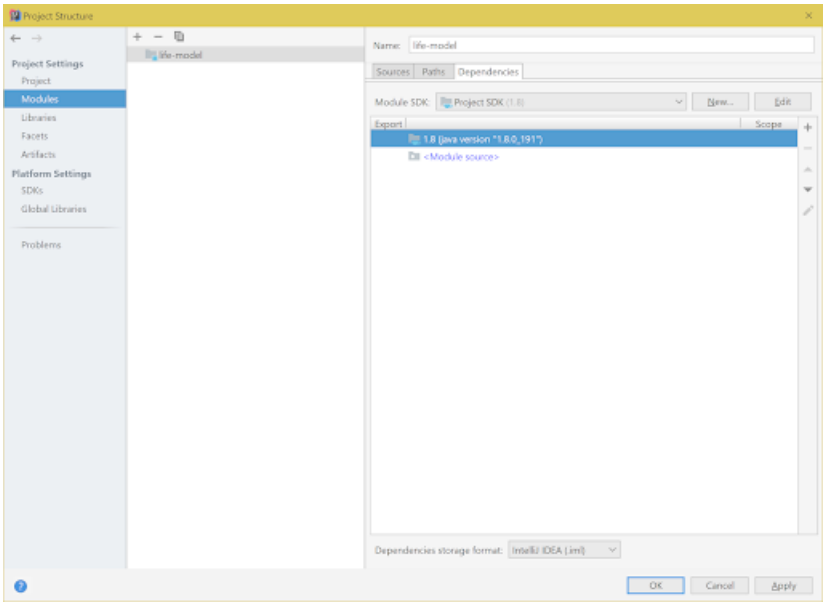


6. Select the new folder in the folder tree, and click the **Tests** button above the folder tree; this

designates that folder as a test source folder. It should now be displayed in green in the folder tree, and be listed under **Test Source Folders** in the content root list on the right.

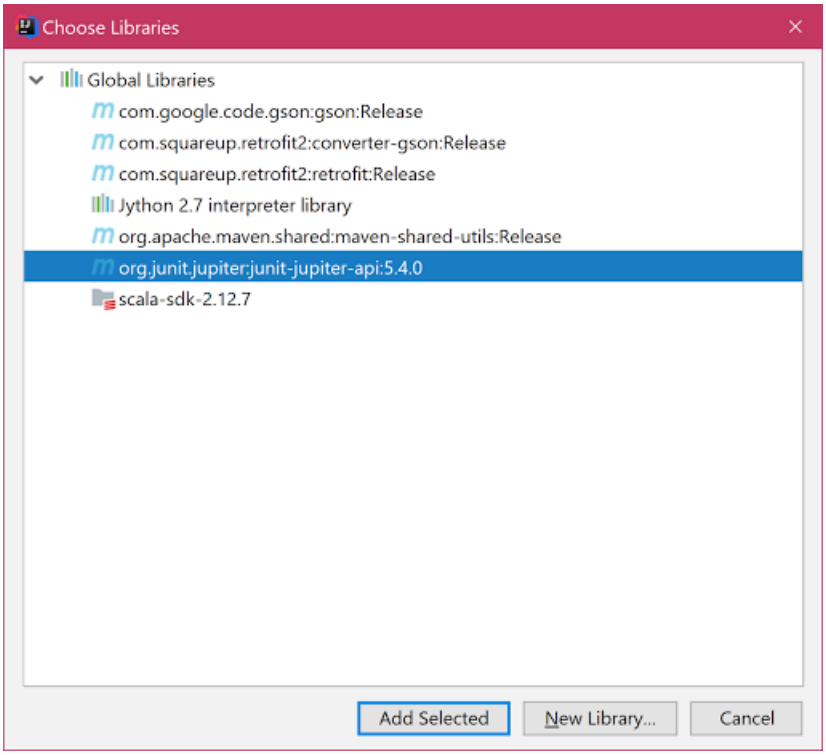


7. Click the **Dependencies** tab in the right panel. A list including the JDK and any external libraries used by the module will be displayed.

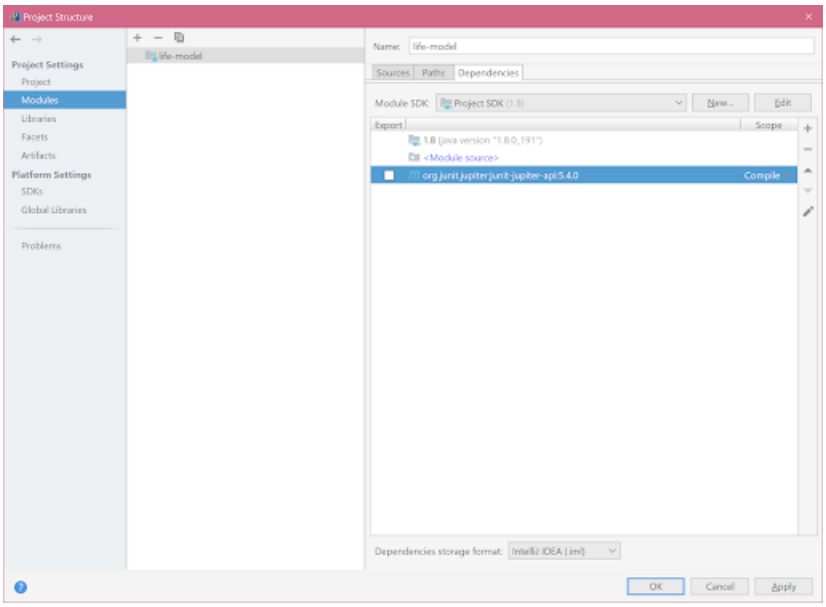


8. Click the plus (+) sign in the right margin (it may appear in the bottom margin in OS X and some Linux versions of IntelliJ IDEA), and select **Library** from the pop-up menu.

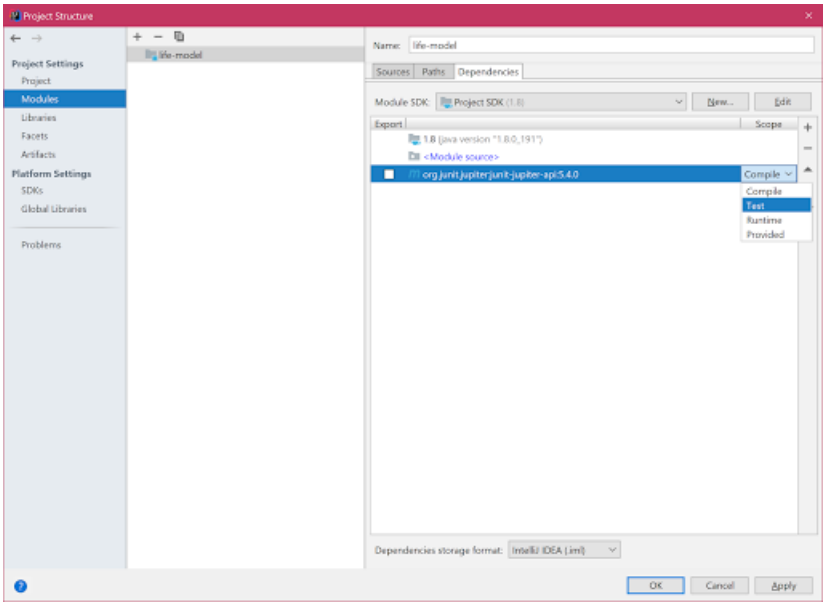
9. In the **Choose Libraries** dialog that appears, select the JUnit5 API library (e.g. `org.junit.jupiter:junit-jupiter-api:5.4.0`) from the list.



10. Click the **Add Selected** button. The library now appears as a module dependency.



11. **Important:** If **Compile** appears in the **Scope** column of the dependencies list for the JUnit5 library, click on the pull-down control next to **Compile**, and select **Test** from the menu.

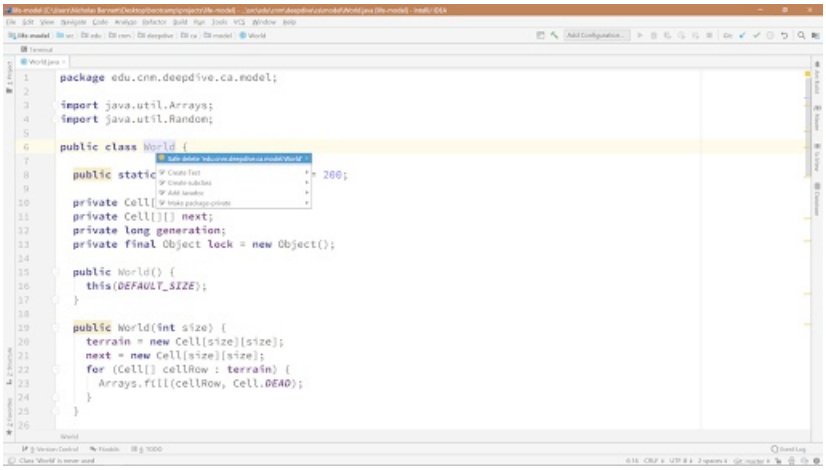


12. If your unit tests will include parameterized tests, repeat steps 8–11, but this time selecting the JUnit5 Params library (e.g. `org.junit.jupiter:junit-jupiter-params:5.4.0`).

13. **Important:** Click the **OK** or **Apply** button to finish configuring JUnit5 for use in your project.

Creating a test class

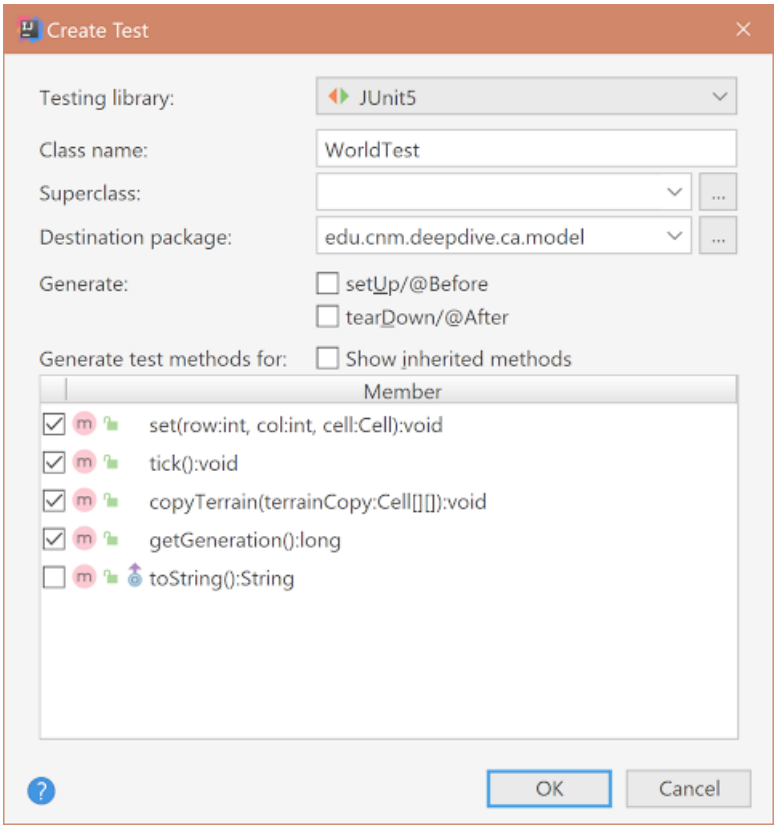
1. In the source file for the class for which you want to create one or more JUnit5 tests, click on the class name in the class declaration, and press [*Alt*]-[*Enter*].



2. From the menu that appears, select **Create Test**.

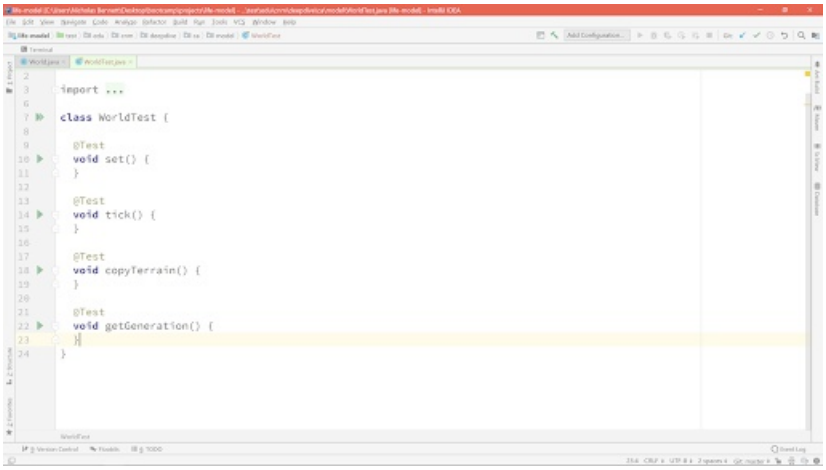
3. In the **Create Test** dialog, select JUnit5 from the **Testing library** pull-down. (If you see a message indicating that JUnit5 is not a module dependency, along with a **Fix** button, please review the steps in "Add the core JUnit5 library to IntelliJ IDEA as a global library" and "Add JUnit5 to an IntelliJ IDEA project", above.)

4. If you want to create test methods corresponding to specific methods in the class you're testing, select the checkboxes for any such methods in **Generate test methods for** list.



5. Click the **OK** button to create the test class.

6. Note that in the test class, test methods (whether created manually, or generated automatically in steps 3-5, above) must be annotated with `@Test` (or one of the other JUnit5 annotation for special-purpose test methods). Methods annotated with `@Test` must have the `void` return type, and must have no parameters declared. However, while it is common to name test methods with names matching the methods in the class being tested, this is not required.



7. Use the appropriate `static` methods of the `org.junit.jupiter.api.Assertions` class to write your tests.