Step 3: Set up JabRef's code style

Contributions to JabRef's source code need to have a code formatting that is consistent with existing source code. For that purpose, JabRef provides code-style and check-style definitions.

Install the CheckStyle-IDEA plugin, it can be found via the plug-in repository: Navigate to **File > Settings... > Plugins"**. On the top, click on "Marketplace". Then, search for "Checkstyle". Click on "Install" choose "CheckStyle-IDEA".
Note: In some MacBooks, `Settings` can be found at the "IntelliJ" button of the app menu instead of at "File".

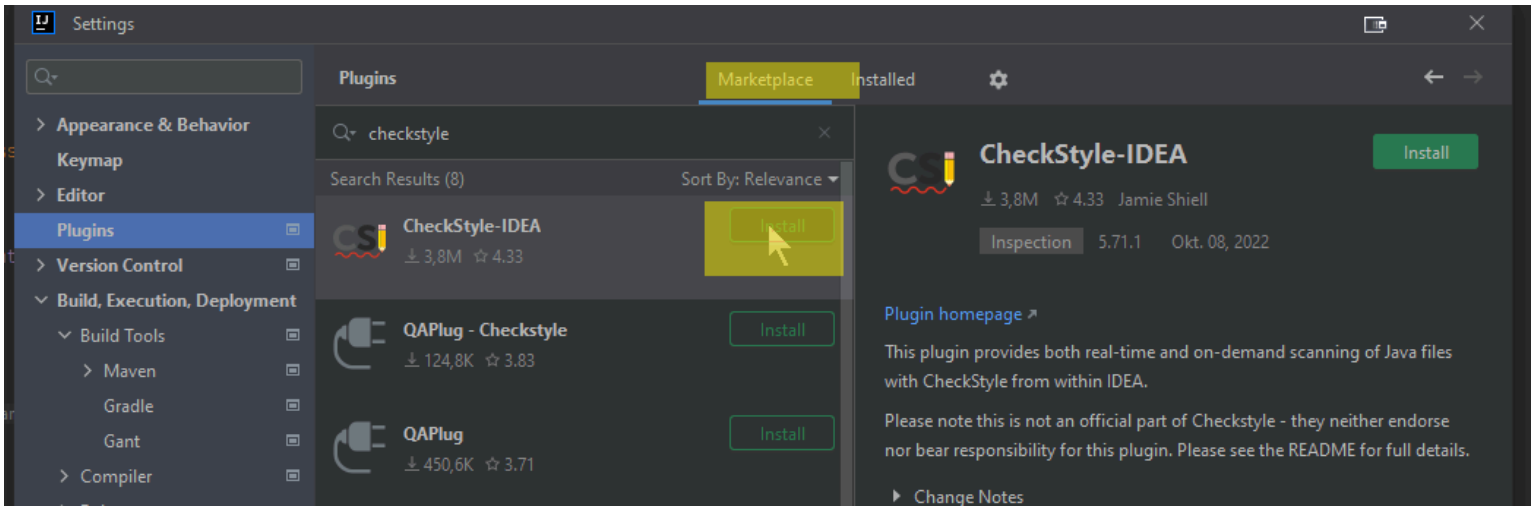

Figure: Install CheckStyle

After clicking, IntelliJ asks for confirmation:

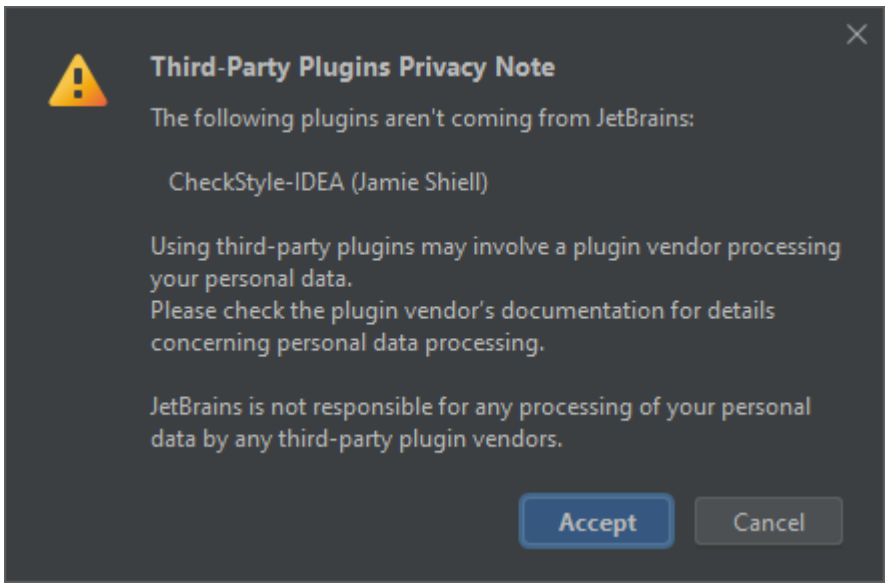

Figure: Third Party Plugin Privacy Notice

If you agree, click on "Agree" and you can continue.

Afterwards, use the "Restart IDE" button to restart IntelliJ.

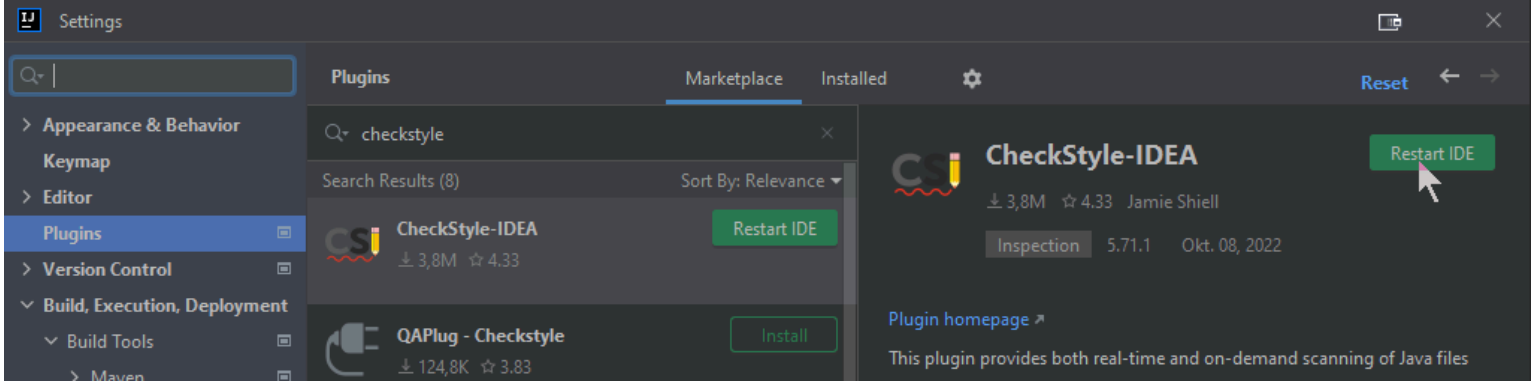

Figure: IntelliJ restart IDE

Click on “Restart” to finally restart.

Wait for IntelliJ coming up again.

Go to **File > Settings… > Editor > Code Style**

Click on the settings wheel (next to the scheme chooser), then click “Import Scheme >”, then click “IntelliJ IDEA code style XML”

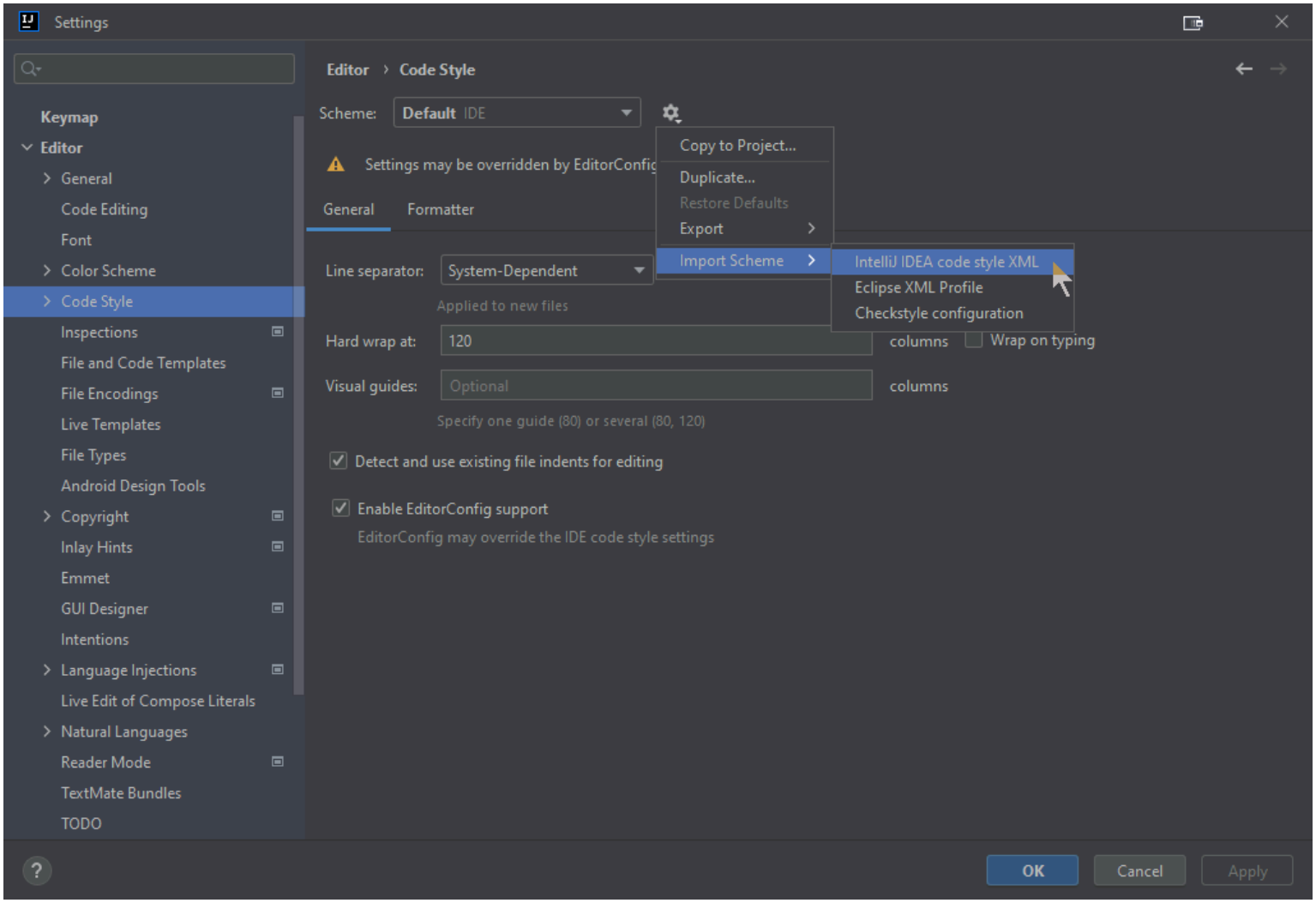

Figure: Location of “Import Scheme > IntelliJ IDEA code style XML”

You have to browse for the directory `config` in JabRef’s code. There is an `IntelliJ Code Style.xml`.

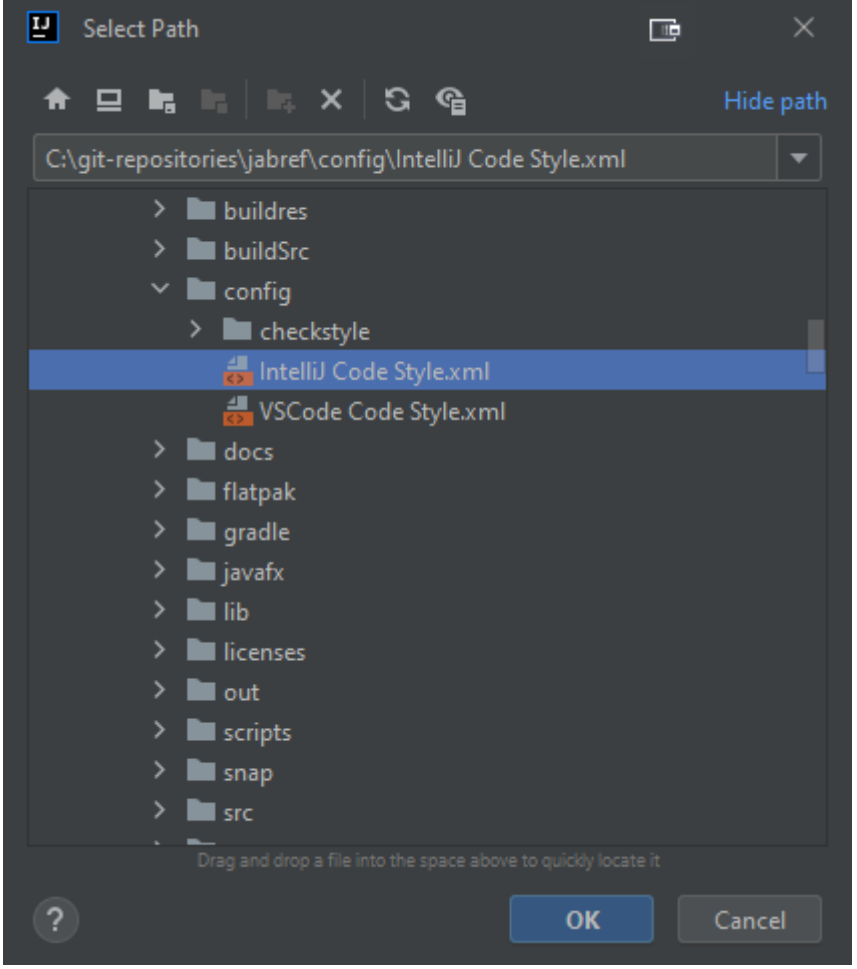

Figure: Browsing for `config/IntelliJ Code Style.xml`

Click “OK”.

At following dialog is “Import Scheme”. Click there “OK”, too.

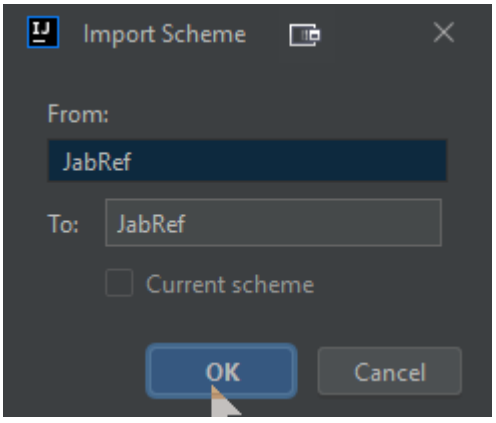

Figure: Import to JabRef

Click on “Apply” to store the preferences.

Put JabRef’s checkstyle configuration in place

Now, put the checkstyle configuration file is in place:

Go to **File > Settings… > Tools > Checkstyle > Configuration File**

Trigger the import dialog of a CheckStyle style by clicking the [+] button:

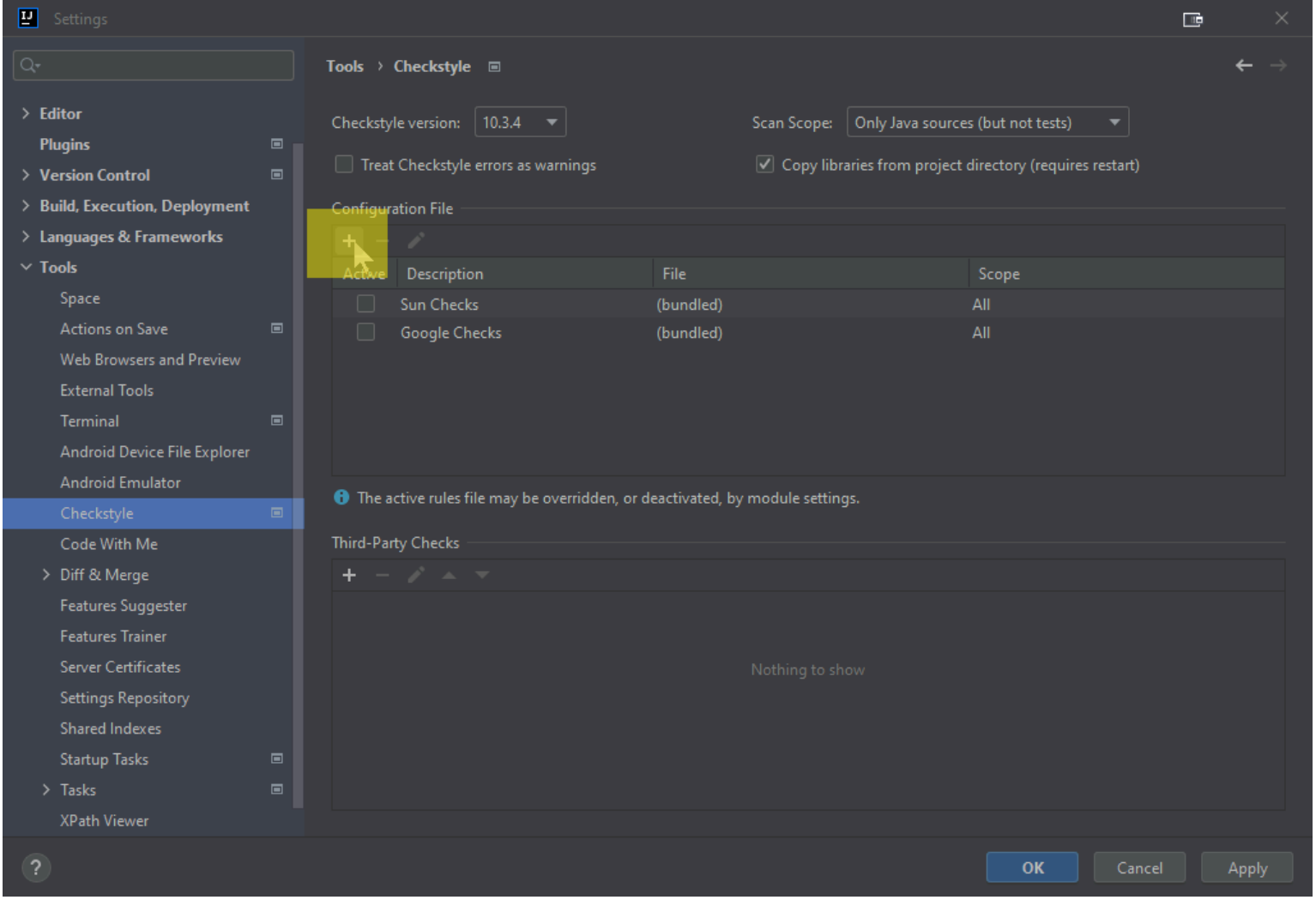

Figure: Trigger the rule import dialog

Then:

- Put “JabRef” as description.
- Browse for `config/checkstyle/checkstyle.xml`
- Tick “Store relative to project location”
- Click “Next”

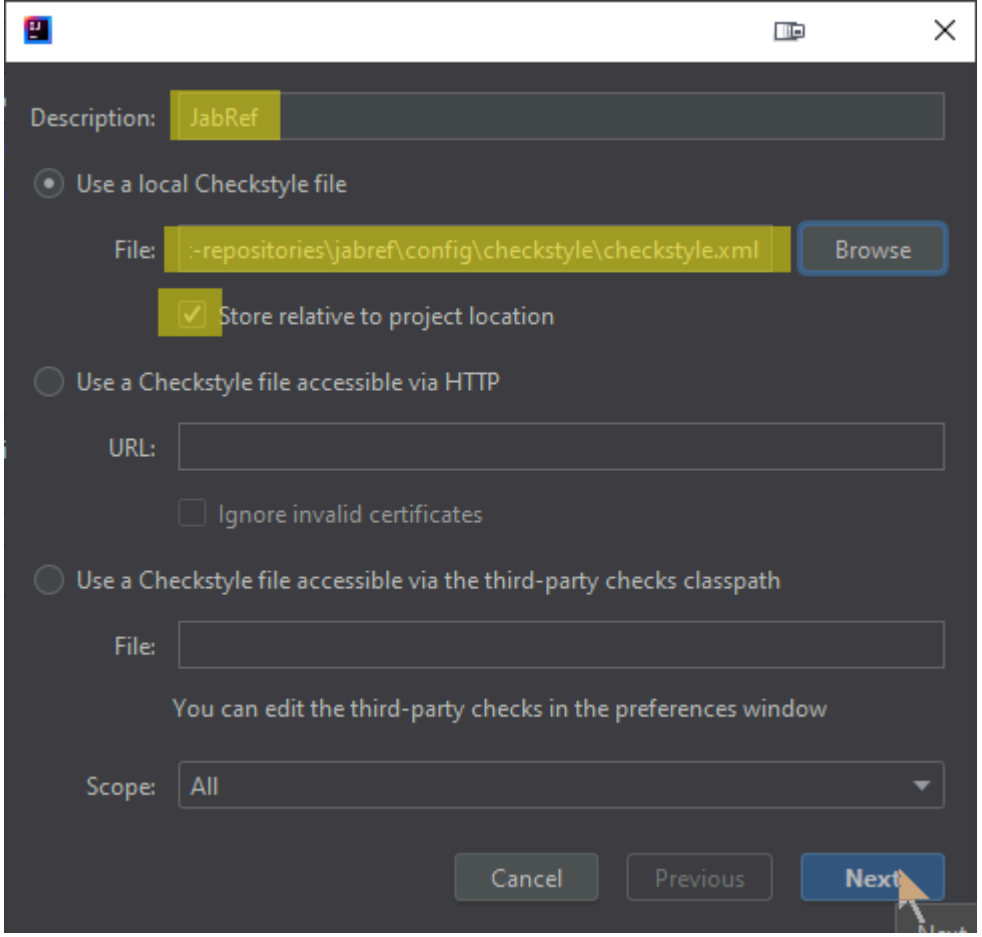

Figure: Filled Rule Import Dialog

Click on “Finish”

Activate the CheckStyle configuration file by ticking it in the list

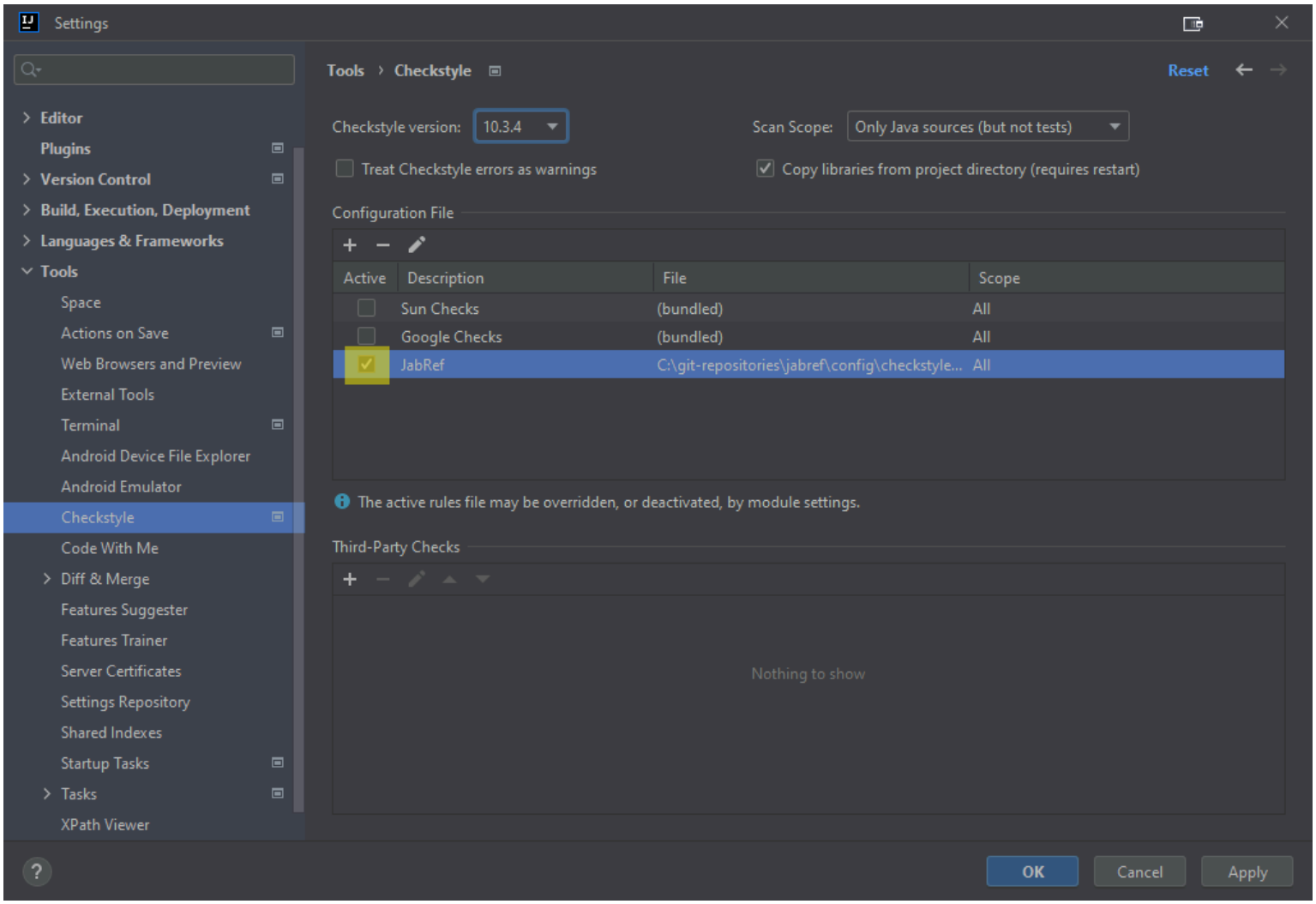

Figure: JabRef's checkstyle config is activated

Ensure that the latest CheckStyle version is selected (10.21.0 or higher). Also, set the "Scan Scope" to "Only Java sources (including tests)".

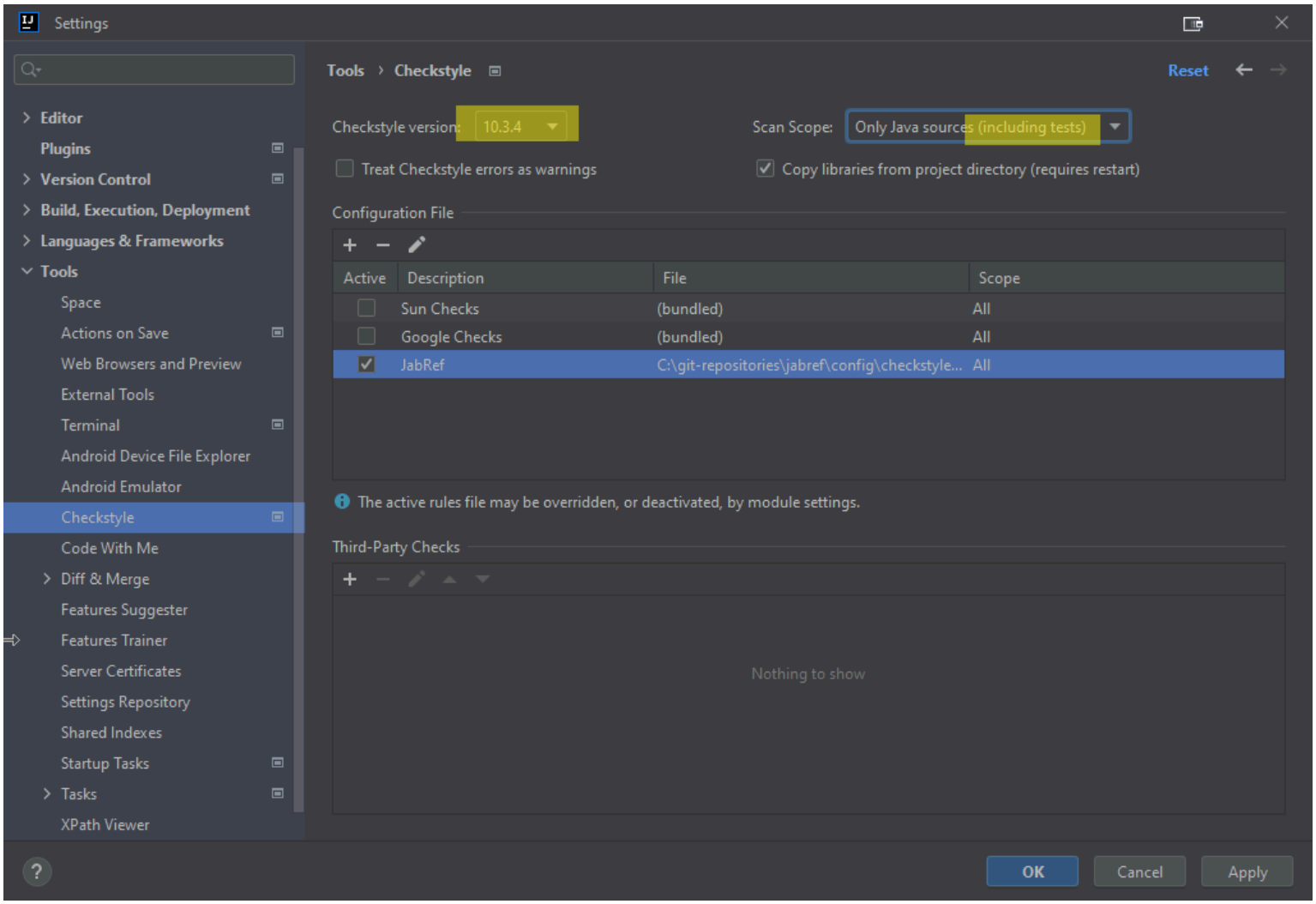

Figure: Checkstyle is the highest version - and tests are also scanned

Save settings by clicking "Apply" and then "OK"

Run checkstyle

In the lower part of IntelliJ's window, click on "Checkstyle". In "Rules", change to "JabRef". Then, you can run a check on all modified files.

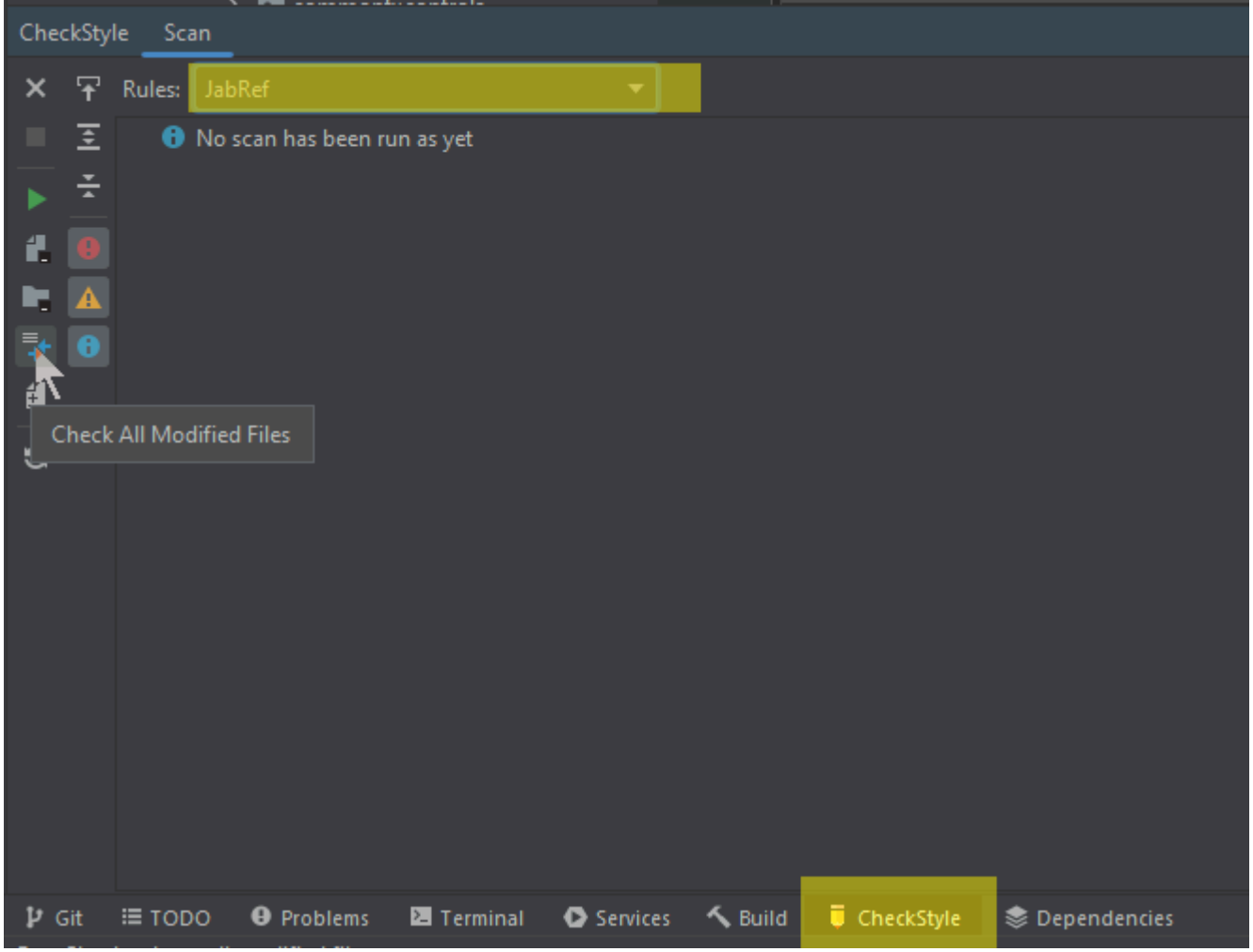

Figure: JabRef's style is active - and we are ready to run a check on all modified files

Have auto format working properly in JavaDoc

To have auto format working properly in the context of JavaDoc and line wrapping, "Wrap at right margin" has to be disabled. Details are found in IntelliJ issue 240517.

Go to **File > Settings… > Editor > Code Style > Java > JavaDoc**.

At "Other", disable "Wrap at right margin"

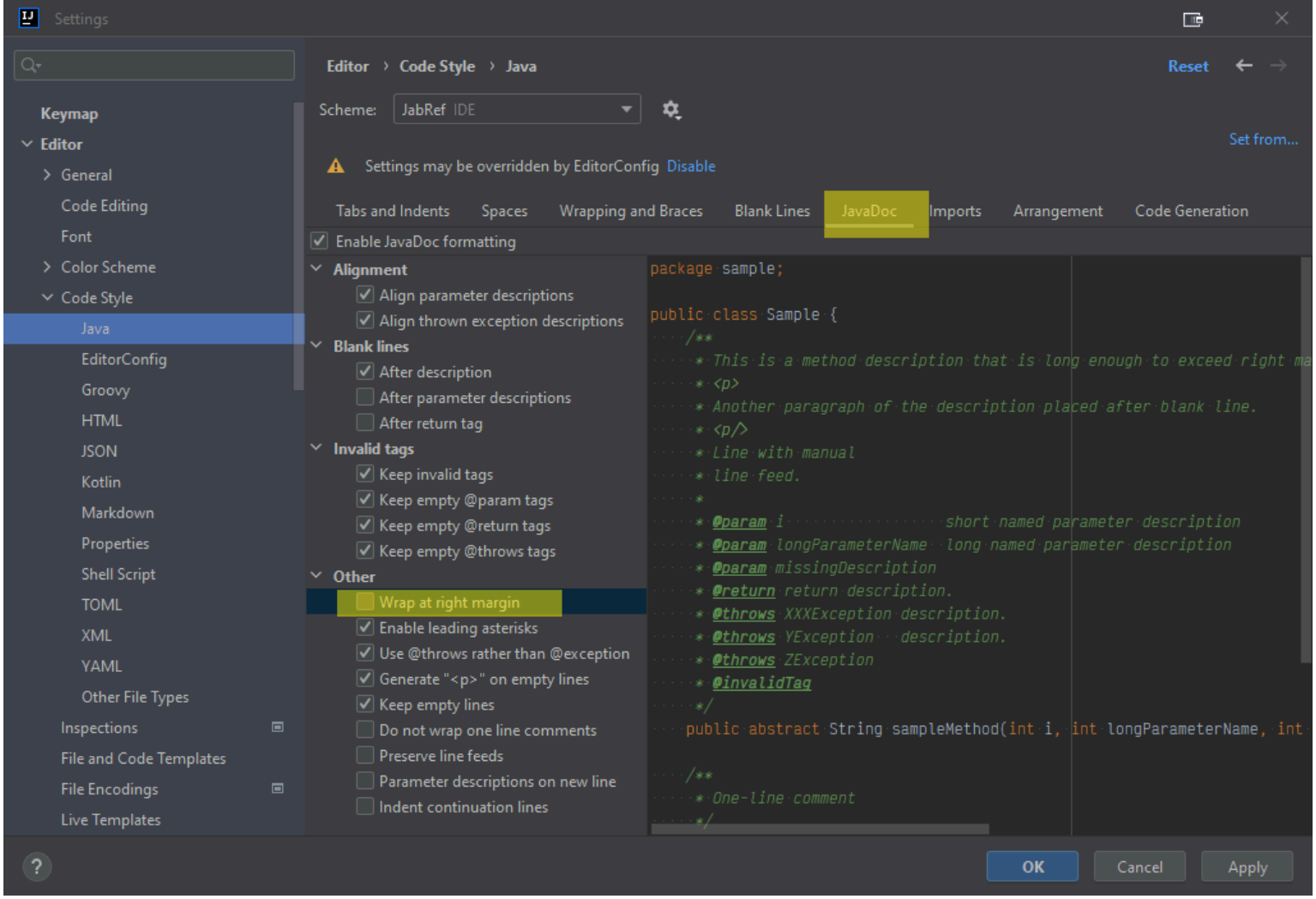

Figure: ”Wrap at right margin” disabled

Enable proper import cleanup

To enable “magic” creation and auto cleanup of imports, go to **File > Settings… > Editor > General > Auto Import**. There, enable both “Add unambiguous imports on the fly” and “Optimize imports on the fly” (Source: JetBrains help).

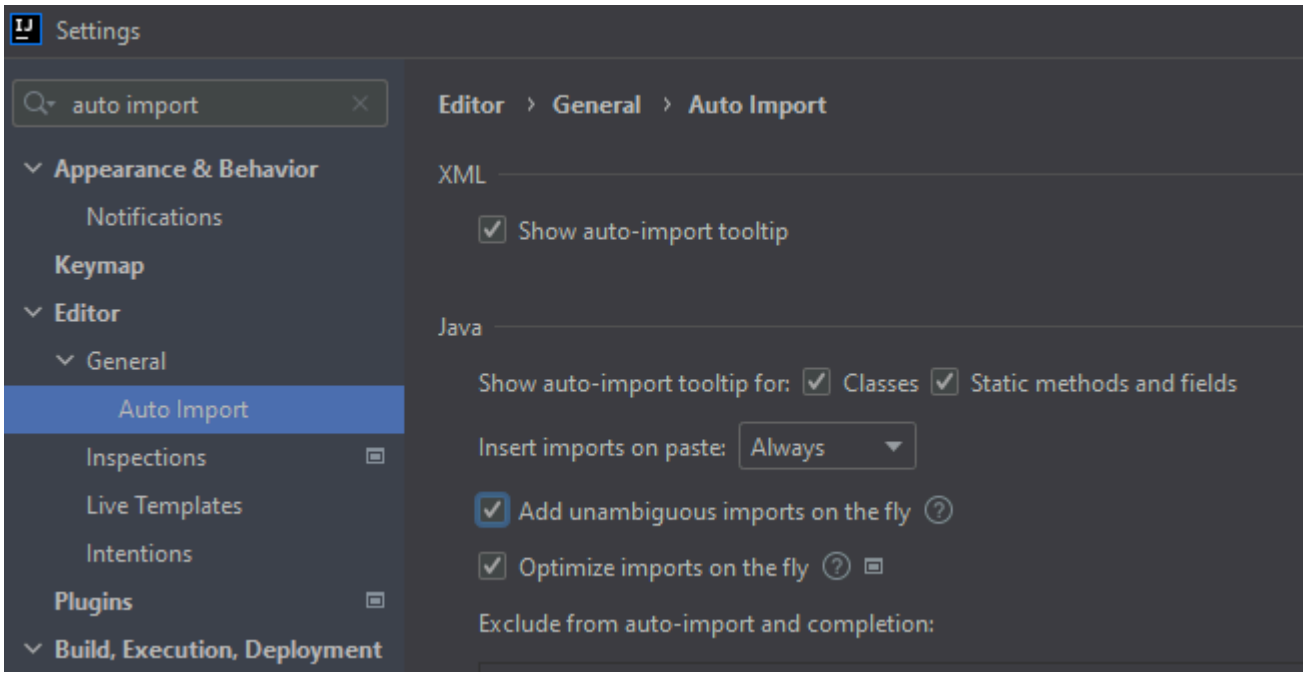

Figure: Auto import enabled

Press “OK”.

Disable too advanced code folding

Go to **File > Settings… > Editor > General > Code Folding**. At section “General”, disable “File header” and “Imports”. At section “Java”, disable “One-line methods”.

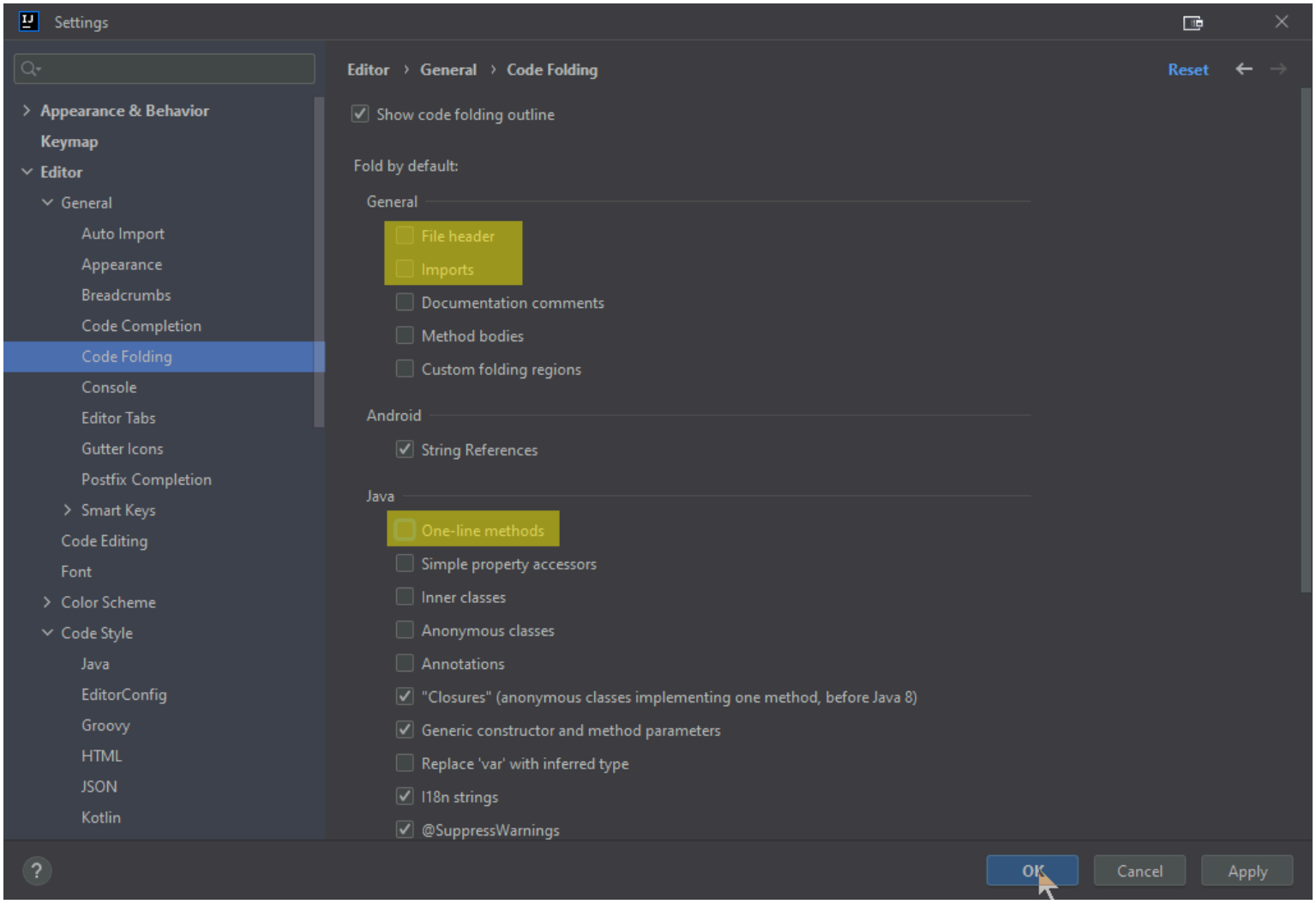

Figure: Code foldings disabled

Press “OK”.

Final comments

SUMMARY

Now you have configured IntelliJ completely. You can run the main application using Gradle and the test cases using IntelliJ. The code formatting rules are imported - and the most common styling issue at imports is automatically resolved by IntelliJ. Finally, you have Checkstyle running locally so that you can check for styling errors before submitting the pull request.

Got it running? GREAT! You are ready to lurk the code and contribute to JabRef. Please make sure to also read our contribution guide.