Frequently Asked Questions (FAQ)

Following is a list of common errors encountered by developers which lead to failing tests, with their common solutions:

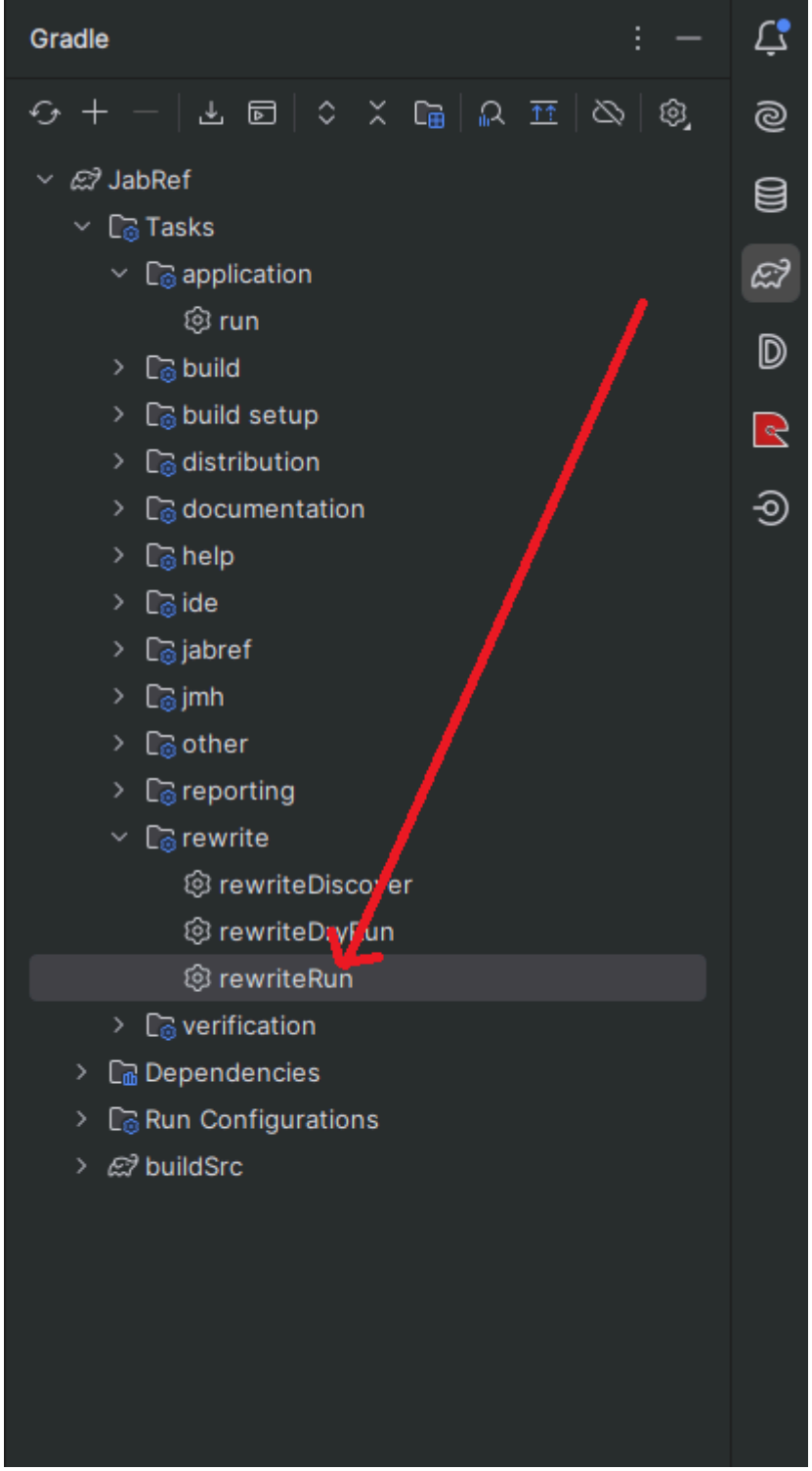
Failing tests

Failing **Checkstyle** tests

JabRef follows a pre-defined style of code for uniformity and maintainability that must be adhered to during development. To set up warnings and auto-fixes conforming to these style rules in your IDE, follow Step 3 of the process to set up a local workspace in the documentation. Ideally, follow all the set up rules in the documentation end-to-end to avoid typical set-up errors.
Note: The steps provided in the documentation are for IntelliJ, which is the preferred IDE for Java development. The `checkstyle.xml` is also available for VSCode, in the same directory as mentioned in the steps.

Failing **OpenRewrite** tests

Execute the Gradle task `rewriteRun` from the `rewrite` group of the Gradle Tool window in IntelliJ to apply the automated refactoring and pass the test:



Background: OpenRewrite is an automated refactoring ecosystem for source code.

`org.jabref.logic.l10n.LocalizationConsistencyTest findMissingLocalizationKeys` FAILED
You have probably used Strings that are visible on the UI (to the user) but not wrapped them using `Localization.lang(...)` and added them to the localization properties file.

Read more about the background and format of localization in JabRef here.

`org.jabref.logic.l10n.LocalizationConsistencyTest findObsoleteLocalizationKeys` FAILED
Navigate to the unused key-value pairs in the file and remove them. You can always click on the details of the failing test to pinpoint which keys are unused.

Background: There are localization keys in the localization properties file that are not used in the code, probably due to the removal of existing code. Read more about the background and

format of localization in JabRef here.

`org.jabref.logic.citationstyle.CitationStyle discoverCitationStyles` ERROR: Could not find any citation style. Tried with /ieee.csl.

Check the directory `src/main/resources/csl-styles`. If it is missing or empty, run `git submodule update`. Now, check inside if `ieee.csl` exists. If it does not, run `git reset --hard` **inside that directory**.

`java.lang.IllegalArgumentException`: Unable to load locale en-US ERROR: Could not generate BibEntry citation. The CSL engine could not create a preview for your item.

Check the directory `src/main/resources/csl-locales`. If it is missing or empty, run `git submodule update`. If still not fixed, run `git reset --hard` **inside that directory**.

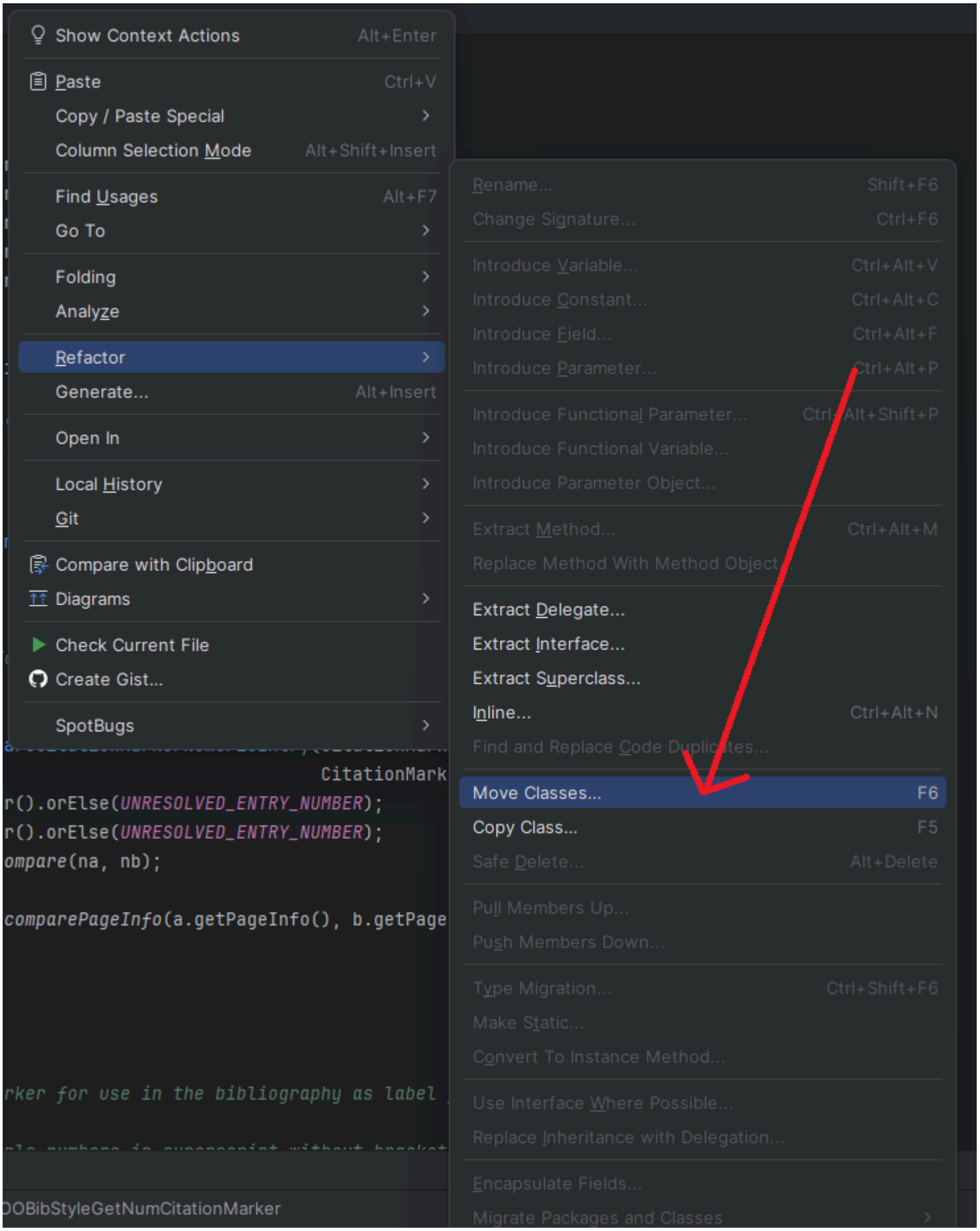
`org.jabref.architecture.MainArchitectureTest restrictStandardStreams` FAILED

Check if you've used `System.out.println(...)` (the standard output stream) to log anything into the console. This is an architectural violation, as you should use the Logger instead for logging. More details on how to log can be found here.

`org.jabref.architecture.MainArchitectureTest doNotUseLogicInModel` FAILED

One common case when this test fails is when you put any class purely containing business logic inside the `model` package (i.e., inside the directory `org/jabref/model/`). To fix this, shift the class to a sub-package within the `logic` package (i.e., the directory `org/jabref/logic/`). An efficient way to do this is to use Intellij's built-in refactoring capabilities - right-click on the file, go to "Refactor" and use "Move Class". The import statement for all the classes using this

class will be automatically adjusted according to the new location.



More information on the architecture can be found at ../getting-into-the-code/high-level-documentation.md.

`Check external href links in the documentation / check-links (push)` FAILED

This test is triggered when any kind of documentation is touched (be it the JabRef docs, or JavaDoc in code). If you changed something in the documentation, and particularly added/changed any links (to external files or websites), check if the links are correct and working. If you didn't change/add any link, or added correct links, the test is most probably failing due to any of the existing links being broken, and thus can be ignored (in the context of your contribution).

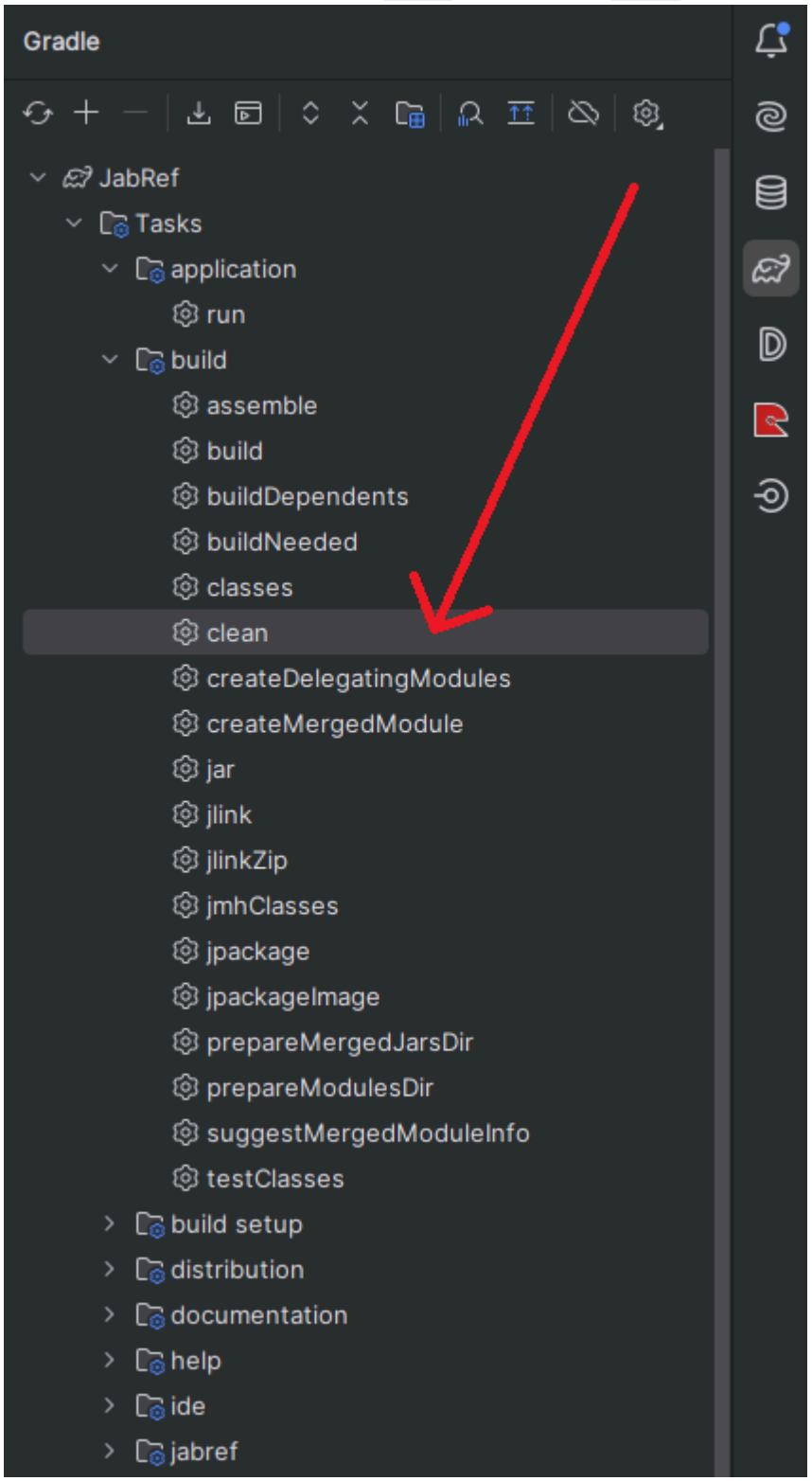
Failing **Fetcher** tests

Fetcher tests are run when any file in the `.../fetcher` directory has been touched. If you have changed any fetcher logic, check if the changes are correct. You can look for more details on how to locally run fetcher tests here. Otherwise, since these tests depend on remote services, their failure can also be caused by the network or an external server, and thus can be ignored in the context of your contribution. For more information, you can look here.

Gradle outputs

```
ANTLR Tool version 4.12.0 used for code generation does not match the current runtime version 4.13.1
```

Execute the Gradle task `clean` from the `build` group of the Gradle Tool Window in IntelliJ:

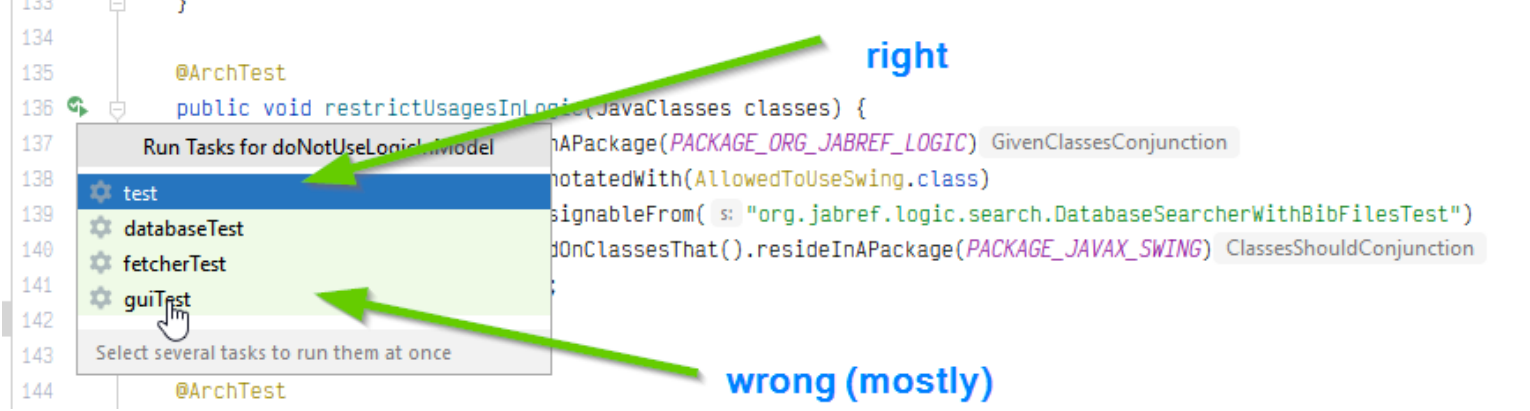


`BstVMVisitor.java:157: error: package BstParser does not exist`

Execute gradle task `clean` from the `build` group of the Gradle Tool Window in IntelliJ.

`No test candidates found`

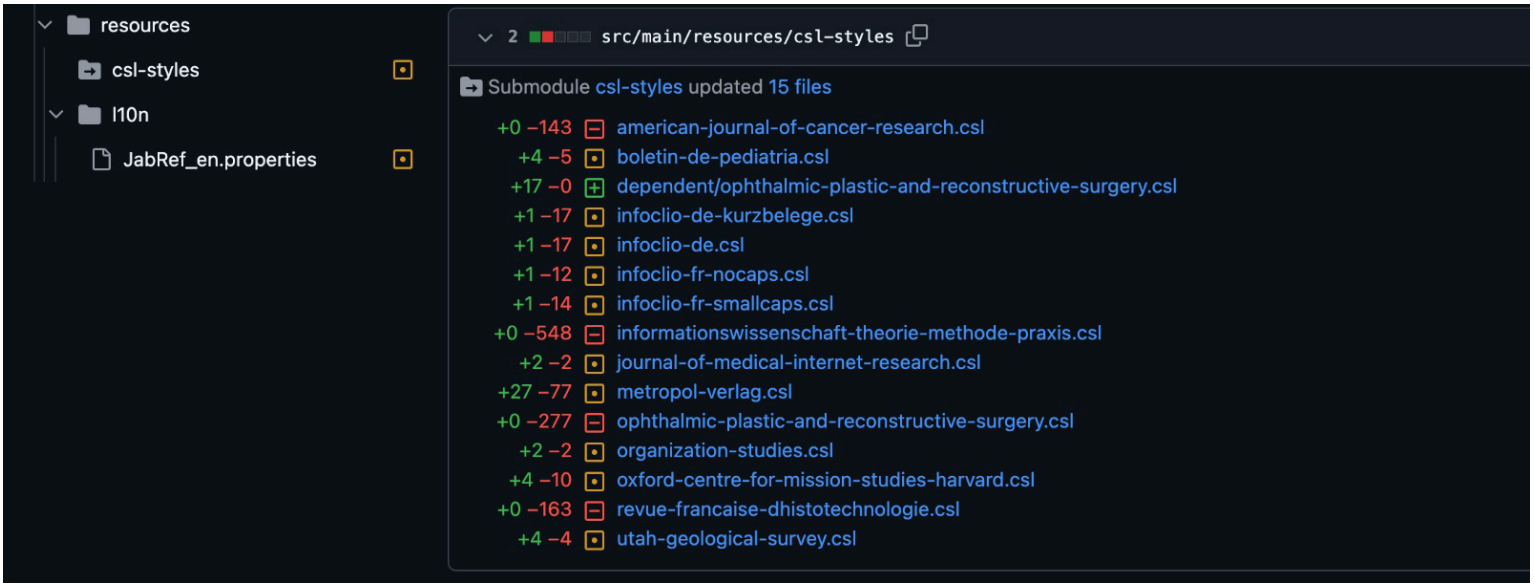
You probably chose the wrong gradle task:



Submodules

The problem

Sometimes, when contributing to JabRef, you may see `abbrv.jabref.org` or `csl-styles` or `csl-locales` among the changed files in your pull request. This means that you have accidentally committed your local submodules into the branch.



Context

JabRef needs external submodules (such as CSL style files) for some of its respective features. These are cloned once when you set up a local development environment, using `--recurse-submodules` (you may have noticed). These submodules, in the main branch, are automatically periodically updated but not fetched into local again when you pull, as they are set to be ignored in `.gitmodules` (this is to avoid merge conflicts). So when remote has updated submodules, and your local has the old ones, when you stage all files, these changes are noticed.

What's strange (mostly an IntelliJ bug): Regardless of CLI or GUI, These changes should ideally not be noticed on staging, as per the `.gitmodules` configuration. However, that is somehow overruled when using IntelliJ's CLI.

Fix

For `csl-styles`:

```
git merge origin/main
git checkout main -- src/main/resources/csl-styles
... git commit ...
git push
```

And similarly for `csl-locales` or `abbrv.jabref.org`.

ALTERNATIVE METHOD (IF THE ABOVE DOESN'T WORK)

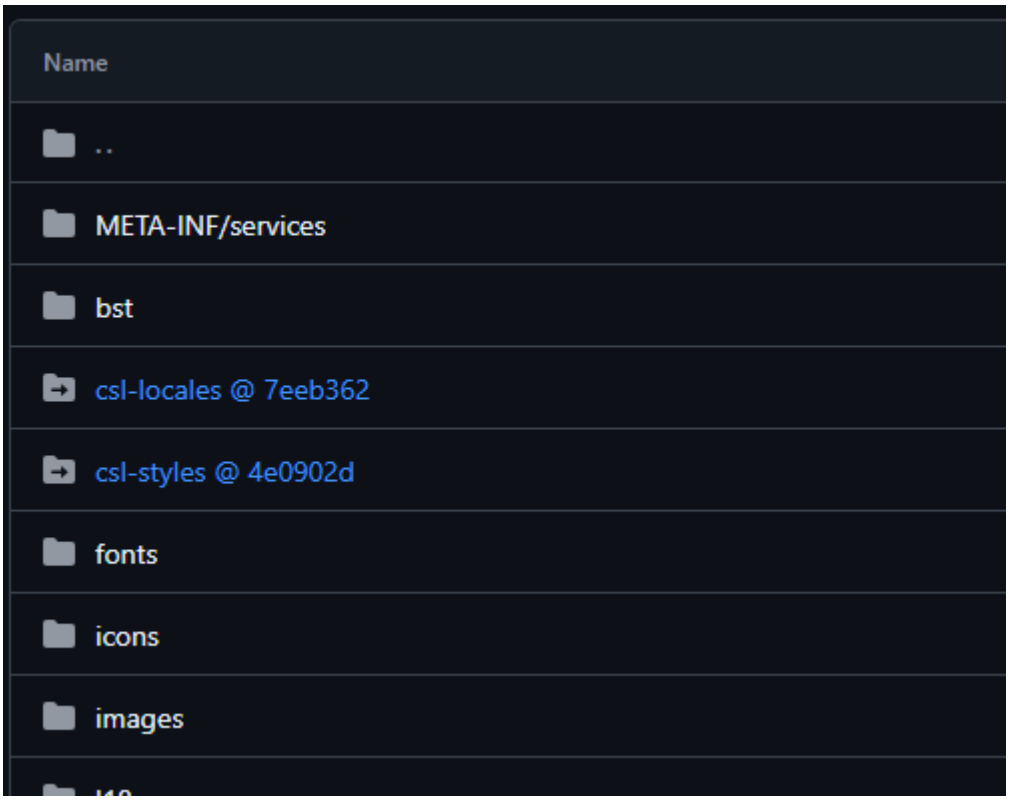
1. Edit `.gitmodules`: comment out `ignore = all` (for the respective submodules you are trying to reset)

   ```
   # ignore = all
   ```

2. `cd` into the changed submodules directory (lets say `csl-styles` was changed):

   ```
   cd src/main/resources/csl-styles
   ```

3. Find the latest submodule commit id from remote (github):

   Here, in the case of `csl-styles`, it is `4e0902d`.

4. Checkout the commit:

   ```
   git checkout 4e0902d
   ```

5. Now, IntelliJ's commit tab will notice that the submodules have been modified. This means we are on the right track.

6. Use IntelliJ’s git manager (commit tab) or `git gui` to commit submodule changes only. Repeat steps 2-5 for other submodules that are shown as modified in the PR. Then, push these changes.
7. Revert the changes in `.gitmodules` (that you made in step 1).

Prevention

To avoid this, avoid staging using `git add .` from CLI. Preferably use a GUI-based git manager, such as the one built in IntelliJ or open git gui from the command line. Even if you accidentally stage them, don’t commit all files, selectively commit the files you touched using the GUI based tool, and push.
