chem-bla-ics

cdk2024 #6: wrapping up already

Published March 30, 2025

Citation

Willighagen, E. (2025, March 30). cdk2024 #6: wrapping up already. *Chem-bla-ics*. https://doi.org/10.59350/143wd-e3m51

Keywords

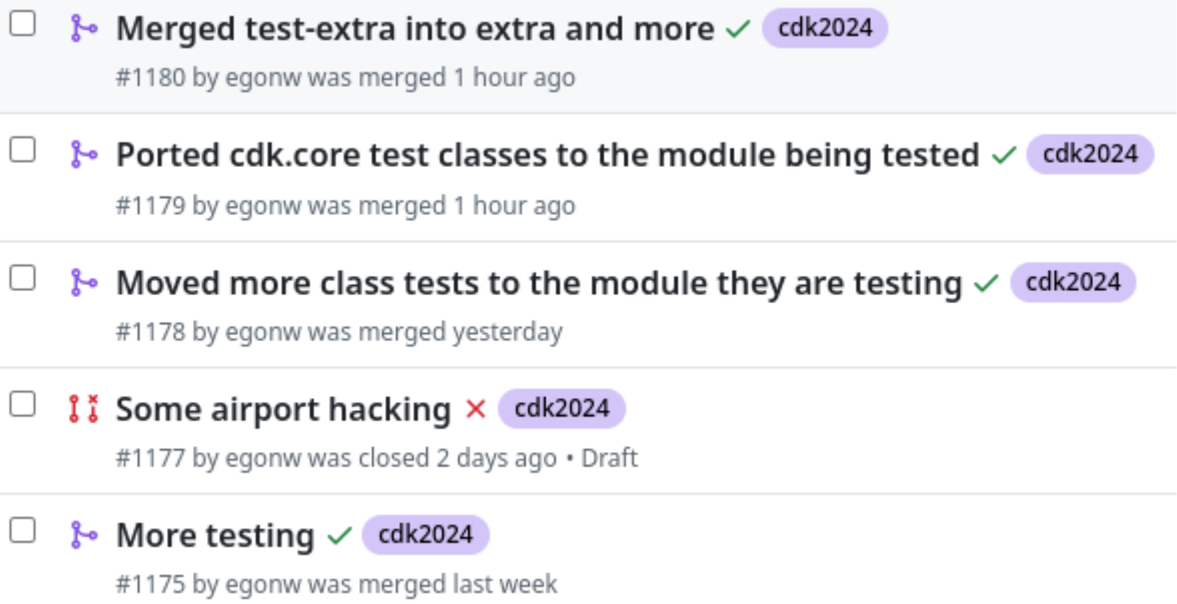
Cdk2024, Cdk, Junit

Abstract

Tomorrow is already the last day of the NWO Open Science grant for the Chemistry Development Kit. We are wrapping up, but I am happy we have a few weeks more to finish up the reporting.

Copyright

Tomorrow is already the last day of the NWO Open Science grant for the Chemistry Development Kit. We are wrapping up, but I am happy we have a few weeks more to finish up the reporting. We held a user group meeting earlier this month (btw, check out the slides by Jonas), and I did a few more JUnit testing updates last week:



□ **Merged test-extra into extra and more** ✓ cdk2024
#1180 by egonw was merged 1 hour ago

□ **Ported cdk.core test classes to the module being tested** ✓ cdk2024
#1179 by egonw was merged 1 hour ago

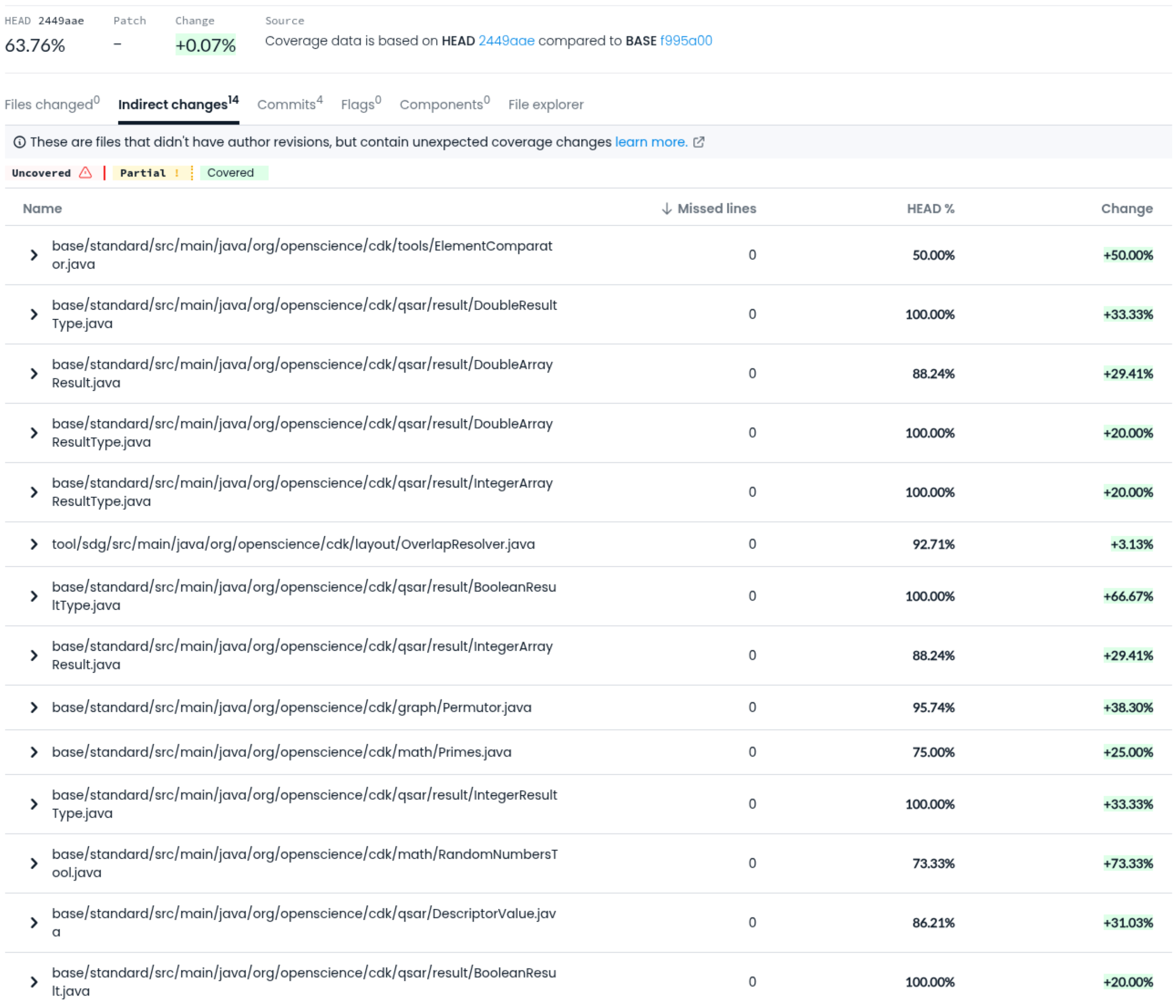
□ **Moved more class tests to the module they are testing** ✓ cdk2024
#1178 by egonw was merged yesterday

□ **Some airport hacking** ✗ cdk2024
#1177 by egonw was closed 2 days ago • Draft

□ **More testing** ✓ cdk2024
#1175 by egonw was merged last week

Actually, you see one pull request here that I closed. I accidentally included a circular dependency. Some core CDK functionality is hard to test with an implementation of the data model, but if that implementation depends on the module you are testing, that won’t work (not in Maven anyway). But the good bits got included in the next pull request. One of the goals was to improve the code covered by the tests.

This *coverage testing* has an imporant code maintenance purpose: it visualizes whih code is not checked. Sometimes code is not tested that under normal conditions should have been (a bug) and sometimes it is handling a rare situation which you want tested too, to make sure that rare case does not get covered by the common code. Thus, the percentage code covered by tests should be as high as is reasonable. The pull requests therefore aim to raise that percentage, such as for this pull request:



HEAD 2449aae 63.76%

Patch –

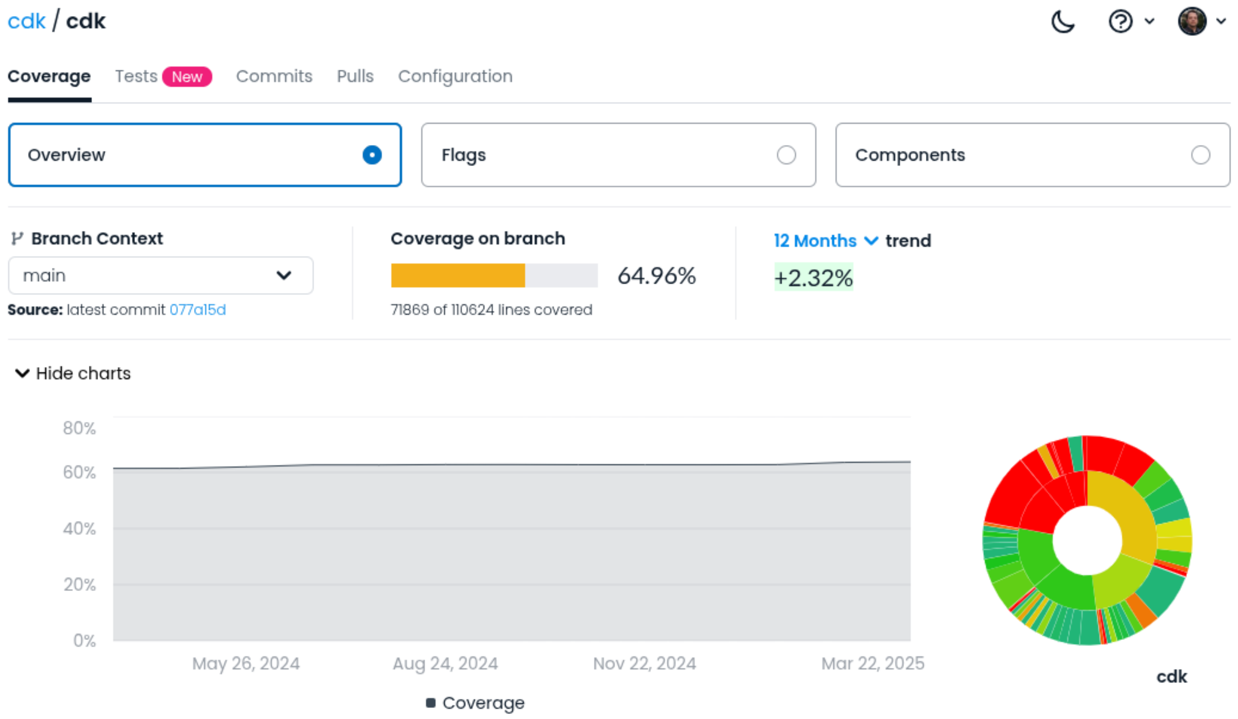
Change +0.07%

Source: Coverage data is based on **HEAD** 2449aae compared to **BASE** f995a00

These are files that didn't have author revisions, but contain unexpected coverage changes learn more.

Name	↓ Missed lines	HEAD %	Change
base/standard/src/main/java/org/openscience/cdk/tools/ElementComparator.java	0	50.00%	+50.00%
base/standard/src/main/java/org/openscience/cdk/qsar/result/DoubleResultType.java	0	100.00%	+33.33%
base/standard/src/main/java/org/openscience/cdk/qsar/result/DoubleArrayResult.java	0	88.24%	+29.41%
base/standard/src/main/java/org/openscience/cdk/qsar/result/DoubleArrayResultType.java	0	100.00%	+20.00%
base/standard/src/main/java/org/openscience/cdk/qsar/result/IntegerArrayResultType.java	0	100.00%	+20.00%
tool/sdg/src/main/java/org/openscience/cdk/layout/OverlapResolver.java	0	92.71%	+3.13%
base/standard/src/main/java/org/openscience/cdk/qsar/result/BooleanResultType.java	0	100.00%	+66.67%
base/standard/src/main/java/org/openscience/cdk/qsar/result/IntegerArrayResult.java	0	88.24%	+29.41%
base/standard/src/main/java/org/openscience/cdk/graph/Permutor.java	0	95.74%	+38.30%
base/standard/src/main/java/org/openscience/cdk/math/Primes.java	0	75.00%	+25.00%
base/standard/src/main/java/org/openscience/cdk/qsar/result/IntegerResultType.java	0	100.00%	+33.33%
base/standard/src/main/java/org/openscience/cdk/math/RandomNumbersTool.java	0	73.33%	+73.33%
base/standard/src/main/java/org/openscience/cdk/qsar/DescriptorValue.java	0	86.21%	+31.03%
base/standard/src/main/java/org/openscience/cdk/qsar/result/BooleanResult.java	0	100.00%	+20.00%

Indeed, over the past 12 months, the coverage did improve, perhaps not as much as we liked, with 2.32 percent point to 64.96 percent:



cdk / cdk

Coverage · Tests New · Commits · Pulls · Configuration

Overview ● · Flags ○ · Components ○

Branch Context: main

Source: latest commit 077a15d

Coverage on branch: 64.96%

71869 of 110624 lines covered

12 Months trend: +2.32%

Hide charts

The CDK started routinely using unit testing somewhere in the zeroes, with home made continous integration as early as 2006, shared with the development community every night, thanks to Rajarshi Guha's effort. We had our first code coverage results in the same year. And at some point we had sufficient coverage that it gave us the opportunity to routinely check the impact of a patch. Of course, this is exactly what many open source projects do with GitHub Actions nowadays.

Unfortunately, we also found that many of the CDK-using tools we worked on in this grant to get updated to use a (more) recent CDK version do not have such solutions in place. That left us in several cases quite in the dark. More about that soon!