chem-bla-ics

CDK Debug classes and fixing the ModelBuilder3D bug

Egon Willighagen

Published December 16, 2005

Citation
Willighagen, E. (2005). CDK Debug classes and fixing the ModelBuilder3D bug. In *chem-bla-ics*. chem-bla-ics. https://doi.org/10.59350/6p49t-sj396

Keywords
Cdk, Cheminf

Abstract
For some weeks now I have been thinking about bug 1309731: “ModelBuilder3D overwrites Atom IDs”. The ModelBuilder3D is a complex piece of source code, reusing many other parts of the CDK, including atom type perception.

For some weeks now I have been thinking about bug 1309731: “ModelBuilder3D overwrites Atom IDs”. The ModelBuilder3D is a complex piece of source code, reusing many other parts of the CDK, including atom type perception.

Somewhere in October, however, I found that Taverna could not create 3D models and convert these into reasonable CML because the Atom ID’s were messed up. So the question is, where did the ModelBuilder3D do this? Did it do this itself, or is it done by one of the other pieces of CDK that it uses? But due to the complex nature of this algorithm, it quickly became clear that looking at the code was not going to solve it; there was too much code to look at.

The solution was clear to me: use the [new data interfaces *](https://chem-bla-ics.linkedchemistry.info/2005/10/25/more-cdkinterfaces-updates.html). To identify where the IDs where messed up, I only needed to write a DebugAtom class with a method that looked like:*

```
public void setID(String identifier) {
  logger.debug("Setting ID: ", identifier);
  super.setID(identifier);
}
```

And I would immediately at what stage the ID was overwritten.

So I started this week to implement the DebugAtom and related classes. By extending `Atom`, I could just add debugging stuff and reuse the code in that class. However, the `DebugAtom` can not extend `DebugAtomType` too then. And this is a pity, because all methods inherited by the `Atom` interface from `AtomType`, `Isotope`, `Element` and `ChemObject` interfaces could not be inherited from the `DebugAtomType` class. Instead, they now have to duplicate those bits of code.

This is not a clean solution, as duplicate code is a known cause of bugs. So, the next step was to write JUnit tests for the new debug classes. And for this I wanted to reuse, i.e. extend, the tests for the default data classes. This required, however, changes to those test classes.

The first thing that needed to be changed was that instantiation of data classes in the tests would now have to depend on the data classes being tested. A simple

```
Atom atom = new Atom("C");
```

only makes sense when a specific `Atom` class was important. Fortunately, the new interfaces provide a solution for this: the `ChemObjectBuilder` implementations. These allow to use the following syntax to replace the hard coded instantiation:

```
Atom atom = builder.newAtom("C");
```

Therefore, I added a protected field to the `AtomTest`, which was instantiated in the `setUp()`:

```
protected ChemObjectBuilder builder;
public void setUp() {
```

```
  builder = DefaultChemObjectBuilder.getInstance();
}
```

and use this builder to instantiate all test objects, as shows for the atom above.

And then I can simply reuse this JUnit test by defining the `DebugAtomTest` like:

```
public class DebugAtomTest extends AtomTest {
  public DebugAtomTest(String name) {
    super(name);
  }

  public void setUp() {
    super.builder = DebugChemObjectBuilder.getInstance();
  }

  public static Test suite() {
    return new TestSuite(DebugAtomTest.class);
  }
}
```

The sources for these debug data classes tests are found in the new `cdk.test.debug` package.

The number of JUnit tests for the CDK jumped from around 1250 to over 1500 tests right now. And if you think these new tests only test old code, because of all the `super.bla()` calls in the debug classes, you're way off. I found bugs in the new debug classes, but **also** many class cast bugs and several other problems in the real data classes!

Anyway. Does this help fix the `ModelBuilder3D` bug? Yes, it does:

```
$ grep "Setting ID" reports/result.modeling.builder3d.ModelBuilder3dTest.txt
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: carbon1
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: oxygen1
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: C
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: HC
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: HC
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: HC
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: O
org.openscience.cdk.debug.DebugAtom DEBUG: Setting ID: HO
```

This shows me where the `Atom` ID is overwritten to be something other than "carbon1"! I can now look at the rest of the `result.modeling.builder3d.ModelBuilder3dTest.txt` file to see what the `ModelBuilder3D` was doing at the time, and which CDK class made the `setID()` call.

I only needed to change this line in the JUnit test for the bug to generate the above debug lines:

chem-bla-ics

```
Molecule methanol = new Molecule();
```

into

```
Molecule methanol = new DebugMolecule();
```