CDK-JChemPaint #6: rendering atom numbers

Egon Willighagen

Published June 10, 2010

Citation
Willighagen, E. (n.d.). In *chem-bla-ics*. chem-bla-ics. https://doi.org/10.59350/5cfpy-kym94

Keywords
Cdk, Chemistry

I have made a few new CDK-JChemPaint patches in the past two days, the latest being patch 15. With the help from Gilleain, all rendering parameters are now using the new API, as explained earlier in this series.

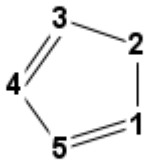
Additionally, the API to work with rendering parameters is now much simpler. The previous posts did not really explain how to tune parameters, so here goes. One important thing to realize, is that a rendering parameter can only be changed if the generator that defines it has been registered. To see what parameters belongs to what generator, for which you can use the script discussed in post #3 .

Atom Numbers

In some situations you like to draw atom numbers. This can be done by replacing the BasicAtomGenerator by an AtomNumberGenerator in the script given in post #1 :

```
List generators = new ArrayList();
generators.add(new BasicSceneGenerator());
generators.add(new BasicBondGenerator());
generators.add(new AtomNumberGenerator());
```

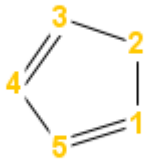
This would result in an image like this:



Now, we also might want to give those numbers a color, to make them stand out a bit. Orange, perhaps :) This is where rendering parameters come in. To the code from post #1 , after the instantiation of the renderer, we add:

```
// tune parameters
model = renderer.getRenderer2DModel();
model.set(
  AtomNumberGenerator.AtomNumberTextColor.class,
  Color.orange
);
```

The output then looks like:



Atom Numbers and Symbols

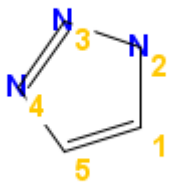
But you can also render both element symbols and numbers. Then, clearly, you just add both generators:

```
List generators = new ArrayList();
generators.add(new BasicSceneGenerator());
generators.add(new BasicAtomGenerator());
generators.add(new BasicBondGenerator());
generators.add(new AtomNumberGenerator());
```

But, in order to have the label and the symbol not overlap, we define an offset (Thanx to Sebastian, of MetFrag fame, for the feature request!):

```
model.set(
  AtomNumberGenerator.Offset.class,
  new javax.vecmath.Vector2d(10,10)
);
```

Then it gets to look like:



This last full example will be available from GitHub shortly.

chem-bla-ics

Update Steffen asked in the comments if it is possible to just color the atoms by element type. CDK-JChemPaint patch 15 does not allow that, but adding that feature is easy enough, and the patch will be part of the next release. Use this configuration:

```
generators.add(new BasicSceneGenerator());
generators.add(new BasicBondGenerator());
generators.add(new AtomNumberGenerator());
```

And these parameter settings:

```
model = renderer.getRenderer2DModel();
model.set(
  AtomNumberGenerator.ColorByType.class,
  true
);
```

This give you for triazole: