

About JChemPaint's future and today's 2.1.5 release

Egon Willighagen 

Published December 3, 2005

Citation

Willighagen, E. (2005). About JChemPaint's future and today's 2.1.5 release. *Chem-bla-ics*. <https://doi.org/10.59350/v0a2f-hfk94>

Keywords

Jchempaint, Cdk, Bioclipse

Copyright

Copyright © Egon Willighagen 2005. Distributed under the terms of the [Creative Commons Attribution 4.0 International License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

chem-bla-ics

Stefan has done an excellent debugging week on [JChemPaint](#), while I have been late with a 2.1 release. Anyway, I've just uploaded a Java 1.4 compiled JChemPaint 2.1 series release. I was told the (reported) bug count is down to one, so I expect to see the next stable branch to be released soon (2.2 series).

But what after JChemPaint 2.2 gets released? Will a 2.3 developers branch be opened? Or will the JChemPaint application, as we know it, cease to exist, and make place for the [Bioclipse JChemPaint plugin](#), that is being worked on?

It is worth mentioning the pros and cons of JChemPaint. One big pro is the applet version of JChemPaint, though free but closed source alternatives are available (e.g. [MarvinSketch](#)). Another advantage is the great semantics of the chemistry being drawn. For example, when drawing reactions, reactants are really marked as reactants, and are not just molecules left of an arrow. Moreover, JChemPaint is a great platform in which ideas can be tested! One of the key virtues of opensourceness. Cons include the limited amount of templates, print quality graphics, and others. (Comments on JChemPaint most welcomed.)

So what about this Bioclipse then? It is inherently SWT based, but currently the [SWT_AWT](#) bridge is used to embed to current JChemPaint and underlying CDK code as is. Unfortunately, [this bridge is using proprietary code from Sun](#) (`sun.awt` classes), which makes it impossible to use with free virtual machines.

But there is also the option of using the SWT drawing classes. This has the advantage that it can be run with free virtual machines, and that it can even be compiled to native code. It requires serious rewriting of code in the JChemPaint and CDK code base. But, CDK's [Renderer2D](#) needs a rewrite anyway: it does not even use Swing's Java2D efficiently (try to figure out how it transforms atomic 2D coordinates into screen coordinates!). Some efforts have been ongoing, but a rewrite from scratch, with a better, more modular, design cannot hurt at all.

References