chem-bla-ics

Cleaner CDK Code #5: develop against interfaces

Egon Willighagen

Published May 15, 2010

Citation
Willighagen, E. (n.d.). In *chem-bla-ics*. chem-bla-ics. https://doi.org/10.59350/1bedv-rsy83

Keywords
Cdk, Java

Rajarshi is at the EBI (or at least was yesterday), talking about his rcdk package (his excellent slides). One slide is about how to create a new atom; he mentions not to use new Atom() but the DefaultChemObjectBuilder instead (slide 73). I do not entire agree with the message given.

Develop against interfaces

The slide seems to favour the DefaultChemObjectBuilder, but there like are more suitable builders for a particular application. Currently, I am aware of two alternative builders: the *DebugChemObjectBuilder*, and the *NoNotificationChemObjectBuilder*.

The reason the CDK has a builder pattern is the following. It is cleaner to write against interfaces than against implementations, because it allows alternative implementations. I just listed all three provided by the CDK library itself, but other implementations may exists too; they might have a completely different data model, e.g. fully CMLDOM-based, or fully SQL-based. By programming against interfaces, changing the implementation becomes easy.

Now, by using `new Atom()` you choose a particular implementation (in this case, the one around the `DefaultChemObjectBuilder`. However, you like the user to pick the implementation. This is why the CDK library itself uses builders all over the place: it assumes a `IChemObjectBuilder` is predefined and that is used.

For example:

```
IChemObjectBuilder builder = new DefaultChemObjectBuilder();
IMolecule molecule = builder.newMolecule();
IAtom atom = builder.newAtom();
molecule.addAtom(atom);
```

If your method actually has an IChemObject as input, it can retrieve the builder from there:

```
public IMolecule addToMolecule(IAtom atom) {
  IChemObjectBuilder builder = atom.getBuilder();
  IMolecule molecule = builder.newMolecule();
  molecule.addAtom(atom);
}
```

This latter situation is what exists most in the CDK library, actually. In some cases, this is not the case, and sometimes you may need to pass an IChemObjectBuilder to a constructor. This is, for example, the case with the constructor of the `SmilesParser`.

Now, reconsider the first code example. By defining a builder only once, and reusing that builder in the rest of your code, you only have to change one line to use a different implementation. For example, the DebugChemObjectBuilder that sends debug messages for each set and get call to one of the data classes. I used this in the past, and solved several nasty bugs with this; just by seeing in what order data was set and read. And I only needed to change one single line of code for that.