jcamp-dx

Stefan Kuhn
Peter Reutemann

September 2, 2014

Contents

Chapter 1

History

The jcamp-dx library (referred to as jcamp-dx here) is a java library for reading and writing the JCAMP-DX format. A rough background on JCAMP-DX is this: "The Joint Committee on Atomic and Molecular Physical Data started as a Task Force on Spectral Data Portability under the direction of Paul A. Wilks, Jr., at the Pittsburgh Conference (Pittcon) of 1983. The scope of JCAMP was originally as follows: "The Joint Committee will generate, collect, evaluate, edit, and approve the publication and encourage the distribution of atomic and molecular physical data in suitable form to serve as references for pure compounds and mixtures" (personal communication from Bob McDonald). JCAMP (the organization) was initially sponsored by: American Chemical Society (ACS), American Physical Society (APS), American Society for Mass Spectrometry (ASMS), American Society for Testing and Materials (ASTM), Optical Society of America (OAS), Society for Applied Spectroscopy (SAS), and Spectroscopy Society of Canada (SSC). The objective of the Task Force was to design a standard file format for exchange of infrared spectra between vendor data systems that used different proprietary file formats. Data exchange capability was in demand by end-users who wished to transfer spectra between different spectrometers in their own and other laboratories.

In 1995 IUPAC took over responsibility for the JCAMP-DX range of scientific standards from the Joint Committee on Atomic and Molecular Physical Data (JCAMP). At that time an IUPAC Working Party had responsibility for the support and development of the JCAMP-DX scientific data standards and this evolved into the Subcommittee for Electronic Data Standards (SEDS). All spectral data is stored as labeled fields of variable length using ASCII characters. The JCAMP-DX file is thus a text file that is human readable and can be edited and annotated using standard text editors.

The JCAMP-DX format has many advantages. It is open-source, using standard terms, so that data from any instrument, or simulated data, can be freely exchanged. Non-proprietary software is available for conversion from other file formats, file compression, internet transmission and visualization of the data. One criticism of the JCAMP-DX standard is that "the specifications are both complicated and incomplete. As the documentation grew to allow other types of instrument data, peak table data, and chemical structures, among other things, programmers had trouble interpreting how the tags should be used to write out their data. This led to the situation where JCAMP files created by a given

software system could not be read by any other system because of inconsistent software implementations. This was exacerbated by the fact that there was (and still is) no way of validating the adherence to the JCAMP standards for files created by different software packages. There is no testing software available, and there is no overseeing body organizing round-robin testing among the vendors and policing their efforts." (An XML-Based File Format for Archival Storage of Analytical Instrument Data[1]) Despite this essentially all spectroscopic instruments currently available have routines for the export of JCAMP-DX files, which work well due to long usage in the practical world and improvements based on that." (quoted from Kuhn et al., Chemical Markup, XML, and the World Wide Web. 7. CMLSpect, an XML vocabulary for spectral data, forthcoming).

This library is intended as a reference implementation of a library for reading of the JCAMP-DX standard. It should work for all JCAMP-DX files up to version 5 (JCAMP-DX 6 is still in draft status).

It was originally written by Thomas Weber from Creon Lab Control and is now maintained as a project on sourceforge. The sourceforge.net/projects/jcamp-dx site should always contain the latest informations.

[1] `http://www.gaml.org/Documentation/XML%20Analytical%20Archive%20Format.pdf`

Chapter 2

Usage

The project can be checked out from CVS at sourceforge.net/projects/jcamp-dx. If a release is published, we also do binary and source file releases. The project has an ant file for compiling and building. The targets are:

- **-clean** – deletes all files created by ant
- **-init** – creates some directories
- **-compile** – comiles the java files to build
- **-dist** – makes a JcampParser.jar in dist. This can be used as library in other projects
- **-javadoc** – creates a javadoc in doc/api
- **-antlr** – this generates lexer and parser files via antlr from .g-files in project root directory

Chapter 3

Code Examples

The main purpose of the library is to read and to write JCAMP-DX files. The classes in org.jcamp.spectrum are the objects read/written. So you will always use these.

3.1 Reading a file

Let's look at a first example: You have a jcamp file and want to read the peaks from it. So firstly you read the file the class to use is JCAMPReader in org.jcamp.parser like this:

```
Spectrum jcampSpectrum = JCAMPReader.getInstance().createSpectrum(jcamp);
NMRSpectrum nmrspectrum = (NMRSpectrum) jcampSpectrum;
```

JCAMPReader is a singleton, it returns one of the subclasses of Spectrum. Here we know the spectrum is an NMR one, so we can directly cast it. Have a look at the api doc to see the different types of spectra and their inheritance relationships. Then we look if the spectrum has got peaks:

```
if (nmrspectrum.hasPeakTable()) {
  Peak[] peaks = nmrspectrum.getPeakTable();
  for (int i = 0; i < peaks.length; i++) {
    ... = (float) peaks[i].getPosition()[0];
    ... = (float) peaks[i].getHeight();
  }
}
```

If it has peaks (remember JCAMP-DX can also contain continuous data), it's an array of Peak objects. Each peak has a got a multi-dimensional position (for 2D etc. spectra) and a height. In case of our nmr spectrum, these would be shift and intensity, so we read them to any variable (... is your code).

A JCAMP-DX file can contain several blocks (a block basically is a spectrum). In such a case, the above code will only return the first child block. In order to access all blocks, you can use JCAMPReader.getInstance().getRootblock()!=null to check if it is a multi block file. If so, use something like this to read them:

```
Enumeration blocks=JCAMPReader.getInstance().getRootblock().getBlocks();
while(blocks.hasMoreElements()){
  JCAMPBlock b = (JCAMPBlock) blocks.nextElement();
  if(b.getID()!=JCAMPReader.getInstance().getIdoffirstspectrum()){
    Spectrum spectrum2=JCAMPReader.getInstance().createSpectrum(JCAMPReader.getInstan
    if(spectrum2.isFullSpectrum())
      cmlSpectrum.addSpectrumData(mapContData((Spectrum1D)spectrum2));
    else
      cmlSpectrum.addPeakList(mapPeaks((Spectrum1D)spectrum2));
    break;
  }
}
```

(idoffirstspectrum contains the id of the spectrum we already get "directly").

3.2 Writing Files

If you want to do it the other way round (i. e. write a file), you first need to build a Spectrum (look at the javadoc to see what parameters mean):

```
double[][] xy = new Double[2][peaknumber];
//this array needs to be filled with x/y values in 0/1 position respectivly
IOrderedDataArray1D x = new OrderedArrayData(xy[0],CommonUnit.mz);
IDataArray1D y = new ArrayData(xy[1], CommonUnit.intensity);
Spectrum1D jdxspectrum = new NMRSpectrum(x, y, nucleus, freq, reference, true);
```

Then we set peaks on it (indeed values are given twice to the writer a bit confusing, admittedly):

```
Peak1D[] jcampPeaks = new Peak1D[5];
Peak1D jcampPeak = new Peak1D(x,y);
jcampPeak.setHeight(y);
jcampPeaks[0] = jcampPeak;
... for all peaks
jdxspectrum.setPeakTable(jcampPeaks);
```

Finally we write it:

```
JCAMPWriter jcamp = JCAMPWriter.getInstance();
String jcampString = jcamp.toJCAMP(jdxspectrum);
```

For specific questions, please consult the javadoc.