jAIDA

Contents

1 Include

We use the JAIDA library.

```
import hep.aida.*;
```

Comments: we add all the */lib/*.jar* files to the environment. Outside NETBEANS you should do :

```
$ export CLASSPATH=‘find .../freeHep/jaida-3.3.0-6/lib -name ’*.jar’ -exec echo -n "{}:" \;‘
$ javac Histogram.java
$ java Histogram
```

You will find exemples here.

2 Embedded the plotter

By default JAIDA will build a pop-up windows using the `plotter.show();` call. The following exemple will embed the plotter in a `JPanel` object.

```
package nectar;
import javax.swing.*;
import java.util.logging.Level;
import hep.aida.*;
import hep.aida.ref.plotter.PlotterUtilities;
import java.awt.BorderLayout;
import org.freehep.swing.popup.GlobalMouseListener;
import org.freehep.swing.popup.GlobalPopupListener;

public class GUI_PM extends JPanel {

   UI_PM pm = null;
   static IPlotter plotter = null;

   public GUI_PM()
   {
      super(new BorderLayout());

      plotter = UI_PM.af.createPlotterFactory().create("je veux pas de pop-up");
      plotter.createRegions(1,2,0); // grid for 2 plots
```

```
      plotter.setParameter("plotterWidth","400");
      plotter.setParameter("plotterHeight","600");

      // Now embed the plotter
      add(PlotterUtilities.componentForPlotter(plotter), BorderLayout.CENTER);

      // Enable the context menu like in the popup windows (bug)
      // (http://bugs.freehep.org/browse/JAIDA-165)
      GlobalMouseListener gml = new GlobalMouseListener(this);
      gml.addMouseListener(new GlobalPopupListener());
   }

   void plotLastSample() {
       // plots
       if (pm != null) {
           for (int channel = 0; channel<2; channel++)
               plotter.region(channel).plot(pm.dpsLast[channel], "mode=replace");
       }

       // plots title
       plotter.region(0).setTitle("Event sample (low gain)");
       plotter.region(1).setTitle("Event sample (high gain)");
       // plotter.refresh(); ne sert à rien.
   }

   public static void main(String args[]) {
        UI_Log.allocate();
        UI_Log.logger.log(Level.INFO,"This is the GUI_PM class.");

        GUI_PM gui_pm = new GUI_PM();
        gui_pm.pm = new UI_PM();
        gui_pm.pm.addValuesExemple1();
        gui_pm.plotLastSample();

        Main.jPanel = gui_pm;
        Main.main(null);
    }
}
```

3 2D plot and 1D histogram

3.1 Statics objetcs needed

```
package nectar;
import hep.aida.*;
import java.util.logging.Level;
import java.util.Random;

...
    final static IAnalysisFactory af = IAnalysisFactory.create();
    final static ITree tree = af.createTreeFactory().create();
    final static IHistogramFactory hf = af.createHistogramFactory(tree);
    final static IDataPointSetFactory dpsf = af.createDataPointSetFactory(tree);
    final static IFitFactory fitF = af.createFitFactory ();
```

```
    final static IFunctionFactory funcF = af.createFunctionFactory (tree);
```

3.2 1D histogram

```
public class UI_PM {
    IHistogram1D  1Dhisto = null;

    UI_PM() {
        // binning of 1 and range of [0:maxDataValue]
        1Dhisto = hf.createHistogram1D("Charge Hard (low gain)", maxDataValue, 0, maxDataValue-1);
    }

    public void addCharge(int value) {
        1Dhisto.fill(value);
    }
}
```

3.3 2D plot

```
public class UI_PM {
    IDataPointSet 2Dgraph = null;

    UI_PM() {
        2Dgraph = dpsf.create("sample","Event sample (low gain)",1);
        for ( i = 0; i<windowSize; i++ ) {
            2Dgraph.addPoint();
        }
    }

    public void addSample(int[] sample) {
        int j, chargeSoftValue;

        for ( j=0; j<windowSize; j++ ) {
            2Dgraph.point(j).coordinate(0).setValue(sample[j]);
        }
    }
}
```

3.4 Fit

```
public class UI_PM {
    IFitter fitter = null;
    IFitResult fitted = null;

    UI_PM() {
        fitter = fitF.createFitter("chi2");
    }

    public void fit(){
       int i,j;
/*
       double[] fPars;
       double[] fParErrs;
       String[] fParNames;
*/
       fitted = fitter[i].fit(chargeSoft[i],"g");
/*
       fPars     = fittedSoft[i].fittedParameters();
```

```
        fParErrs  = fittedSoft[i].errors();
        fParNames = fittedSoft[i].fittedParameterNames();

        for(j=0; j< fittedSoft[i].fittedFunction().numberOfParameters(); j++ )
            System.out.println("Soft= " + fParNames[j]+" : "+fPars[j]+" +- "+fParErrs[j]);
*/
   }
}
```