

This is the java code for generating coördinates of molecules numerically using the CDNT framework with real-time input and diagramatic output.

```
/*
 * cdnt.java
 *
 * Created on January 25, 2006, 8:38 AM
 * System worked to read an input file, write an intermediate PDB, read that
 * PDB and display it. February 14, 2005. Note that the desired method
 * of passing the data directly to the viewer without saving a PDB file
 * failed, and a complaint went out to Egon Willighagen [e.willighagen@science.ru.nl]
 * concerning the error.
 *
 *
 *
 *
 * TODO: input interface to allow changes in angles and re-imaging
 *
 * PRODUCTION CHECKLIST, remove assignment of in\_filename's starting directory!
 */

package jmakemol;
import java.awt.Container;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.PrintStream;
import java.text.DecimalFormat;
import java.util.StringTokenizer;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.vecmath.Point3d;
import jmakemol.cdnt.JmolPanel.ApplicationCloser;
import java.lang.String;
```

```

//import jmakemol.cdnt.JmolPanel.ApplicationCloser;

import org.jmol.adapter.cdk.CdkJmolAdapter;
import org.jmol.api.JmolAdapter;
import org.jmol.api.JmolViewer;
import org.openscience.cdk.Atom;
import org.openscience.cdk.Molecule;

/**
 *
 * @author david
 */
public class cdnt extends javax.swing.JFrame {

    /** Creates new form cdnt */
    public cdnt() {

        cdnt2(); //read and display molecule for the first time
        /* this scheme requires that the user create an input file.
         * we therefore need to have a file chooser to locate it.
         * We will assume that this file is over written as the user changes
         * parameters, and that the pdb file will be written in the same directory
         * FOR VERSION ONE
         *
         * CHANGE FROM ABOVE, we've implemented versioned files, so that each
         * change in coordinates is saved in related files (don't do more than
         * 9 changes however, since I've only used one digit integer numbering!
         */
        initComponents();
        this.setVisible(true);
    }

    /*C
    C   READ THE BOND DEFINITION CARD.
    C */
    public void cdnt2() {

        //      DIMENSION DC(3,77),C(3,75),ANGLE(76),TEMP(10),PARM(3),EULER(3),
        //      . X(3),Y(3),Z(3),RDC(3),V1(3),V2(3),R1(3),R2(3),RBOND(2)
        //      DIMENSION R(12)
        double[][] DC = new double[4][80];
        double[][] C = new double[4][80];
        double[] ANGLE = new double[80];
        String[] NAME = new String[80];
    }
}

```

```

int[] BOND = new int[80];
BOND[0] = 76;
int[] REF = new int[80];
int[] ORIGIN = new int[80];
BOND[76] = 77;
REF[76] = 76;
ORIGIN[76]=76;
ANGLE[76] = 90.0;
boolean[] KEY = new boolean[80];
double[] RDC = new double[4];

DC[1][76] = 0.0;
DC[2][76] = 0.0;
DC[3][76] = 1.0;
DC[1][77] = -1.0;
DC[2][77] = 0.0;
DC[3][77] = 0.0;//direction cosines for reference vectors

IBOND = 0;
JUMP = false;

for (int i=1;i<79;i++){
    KEY[i] = false;
}
NGO = NATOM + 1;
if (chooseInputDirectory){
    JFrame frame = new JFrame();
    in = null;
    JFileChooser fc = new JFileChooser(currentDirectoryPath);
    int retval = fc.showOpenDialog(frame);
    if(retval != JFileChooser.APPROVE_OPTION){
        System.exit(1);
    }
    File selfile = fc.getSelectedFile();
    in\_filename = selfile.toString();
    chooseInputDirectory = false;
}

try {
    in = new BufferedReader(new FileReader(in\_filename));
    try {
        //read first line, which has name of atom at origin
        s1 = in.readLine();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

```

```

        }
    } catch (FileNotFoundException ex) {
        ex.printStackTrace();
    }
//      NBOND = 5;//test case for H2O2, need to change
//      NATOM = 6;//test case for H2O2, need to change

NBOND = Integer.parseInt(s1.substring(0,2));
NATOM = Integer.parseInt(s1.substring(2,4));
NNATOM = NATOM;
int iL = s1.length();
NAME[1] = s1.substring(4,iL);
//first card has NBOND, NATOM, and name of atom, assumed at 0,0,0
EditableString = EditableString + s1+"\n";

do{//this is the main loop to be done one atom at a time
    IBOND=IBOND+1;
    if(debug == true) i
System.out.println(" starting over again incementing IBOND = "+IBOND);
    try {
        s1 = in.readLine();
        EditableString = EditableString + s1+"\n";
        StringTokenizer st = new StringTokenizer(s1, ":");
        II = Integer.parseInt(st.nextToken());
        JJ = Integer.parseInt(st.nextToken());
        R = Double.valueOf(st.nextToken()).doubleValue();
        ZETA = Double.valueOf(st.nextToken()).doubleValue();
        ETA = Double.valueOf(st.nextToken()).doubleValue();
        NAME[IBOND+1] = st.nextToken();
        System.out.println(s1);
        System.out.println(" (interpreting check) R =" +R);
        System.out.print(" ZETA =" +ZETA);
        System.out.print(" ETA=" +ETA);
        System.out.print("NAME["+IBOND+"] = " +NAME[IBOND+1]);
        //last character is name
        if(debug == true) System.out.println("R = " +R);
    } catch(IOException ioex) {
        if(debug == true) System.out.println("Input error");
        // System.exit(1);
    } catch(NumberFormatException nfex) {
        if(debug == true) System.out.println("'" +
nfex.getMessage() +
"\' is not numeric");
        // System.exit(1);
    }
}

```

```

        }

//  while (IBOND <= NBOND){

    if(IBOND > NBOND) System.exit(1);
    //134  IF(IBOND.GE.NBOND)  GO TO 158
    /*for test purposes*/

    /*
READ (5,135) IPARM,PARM
135  FORMAT (5I2,3F10.6)
 */
    if(debug == true) System.out.println(" ***STATEMENT 135 ");

    System.out.println("");
    IMAX = Math.max(II,JJ);
    if(debug == true) System.out.println(" IMAX = "+IMAX);
    /*
136  IF(IMAX.GT.NATOM)  GO TO 30
 */
    if(debug == true) System.out.println(" ***STATEMENT 136 ");
    if (IMAX > NATOM) break;

//C
//C  INSERT THE COORDINATES OF THE INITIAL ATOM.
//C
//  if(JUMP)  GO TO 137
    if(JUMP == false){
        KEY[II]= true;
        II = IBOND;
        BOND[II]=76;
        C[1][II]=0.0;
        C[2][II]=0.0;
        C[3][II]=0.0;
        REF[II]=76;
        JUMP=true;
        if(debug == true)
System.out.println("C[1] ["+II+"] = "+C[1][II]+
"; C[2] ["+II+"] = "+C[2][II]+"; C[3] ["+II+"] = "+C[3][II]);
        /*
C
C  "STANDARD" BOND DEFINITION.
C
        */
        /*
137  IF(.NOT.KEY(II).OR.KEY(JJ))  GO TO 40

```

```

IF(RIJ.LT.1.D-09)  GO TO 50 CHECK FOR SMALL BOND LENGTHS??????
*/
}
if(debug == true) System.out.println(" ***STATEMENT 137 ");
if(debug == true) System.out.println("survived 137 choices");
KEY[JJ]= true;
ORIGIN[IBOND]=II;
BOND[JJ]=IBOND;
ANGLE[IBOND]=ZETA;

IREF1=BOND[II];
if(debug == true) System.out.println("IREF1 after 137 = "+IREF1);
IREF=REF[II];
if(debug == true) System.out.println("IREF = "+IREF);
SIN1=Math.sin(ZETA*RADDEG);
COS1=Math.cos(ZETA*RADDEG);
if(debug == true) System.out.println("SIN1 = "+SIN1+"; COS1 = "+COS1);
/*
C
      C  ** TEST "COLLINEARITY" OF THE BOND.
      C

      IF(DABS(SIN1).GT.1.D-08)  GO TO 138
*/
if(Math.abs(SIN1) < 0.00000001){
    SENSE=1.0;
    //IF(COS1.GT.1.D-06)  SENSE=-1.D0
    if(COS1 >= 0.000001)SENSE = -1.0;
    RDCX=SENSE*DC[1][IREF1];
    RDCY=SENSE*DC[2][IREF1];
    RDCZ=SENSE*DC[3][IREF1];
    REF[JJ]=IREF*(int)Math.rint(SENSE);
    if(debug == true)
        System.out.println("RDCX,Y \& Z = "+RDCX+";"+RDCY+";"+RDCZ);
    //
    //          GO TO 141
    /*
    C
    C  ** DETERMINE THE DIRECTIONAL COSINES OF THE BOND VECTOR BY
    C  ** CALLING SUBROUTINE LOCAL.
    C
    */
} else{
    /*
138  AA=SIN1*COS(ETA)
    */
    if(debug == true) System.out.println(" ***STATEMENT 138 ");
}

```

```

AA = SIN1*Math.cos(ETA*RADDEG);
BB=SIN1*Math.sin(ETA*RADDEG);
CC=-COS1;

if(debug == true)
System.out.println("AA =" +AA+"; BB = "+BB+"; CC = "+CC);
if(IREF > 0) {
    ZREF=ANGLE[IREF];
}else{
    IREF=-IREF;
    ZREF=180.0-ANGLE[IREF];
/*
GO TO 140
139 ZREF=ANGLE(IREF)
*/
}
//140
if(debug == true) System.out.println(" ***STATEMENT 140 ");
IATOM=ORIGIN[IREF];
if(debug == true) System.out.println("IATOM at 140 = "+IATOM);

IREF2=BOND[IATOM];
if(debug == true) System.out.println("IREF2 = "+IREF2);
if(debug == true)
System.out.println("at 140, IATOM = "+IATOM+ " and IREF2 = "+IREF2);
double CSC = 1.0/Math.sin(ZREF*RADDEG);
double COT = Math.cos(ZREF*RADDEG)/Math.sin(ZREF*RADDEG);
double ACOT=-COT;
double ACSC=-CSC;
AX=ACOT*DC[1][IREF1]+ACSC*DC[1][IREF2];
if(debug == true) System.out.println("AX = "+AX);
AY=ACOT*DC[2][IREF1]+ACSC*DC[2][IREF2];
if(debug == true) System.out.println("AY = "+AY);
AZ=ACOT*DC[3][IREF1]+ACSC*DC[3][IREF2];
if(debug == true) System.out.println("AZ = "+AZ);
BX=AZ*DC[2][IREF1]-AY*DC[3][IREF1];
if(debug == true) System.out.println("BX = "+BX);
BY=AX*DC[3][IREF1]-AZ*DC[1][IREF1];
if(debug == true) System.out.println("BY = "+BY);
BZ=AY*DC[1][IREF1]-AX*DC[2][IREF1];
if(debug == true) System.out.println("BZ = "+BZ);
CX=DC[1][IREF1];
CY=DC[2][IREF1];
CZ=DC[3][IREF1];

RDCX=AX*AA+BX*BB+CX*CC;

```

```

        RDC[1] = RDCX;
        if(debug == true)
System.out.println("RDCX = "+RDCX+"; RDC[1] = "+RDC[1]);
        RDCY=AY*AA+BY*BB+CY*CC;
        RDC[2] = RDCY;
        if(debug == true)
System.out.println("RDCY = "+RDCY+"; RDC[3] = "+RDC[2]);
        RDCZ=AZ*AA+BZ*BB+CZ*CC;
        RDC[3] = RDCZ;
        if(debug == true)
System.out.println("RDCZ = "+RDCZ+"; RDC[3] = "+RDC[3]);
        REF[JJ]=IBOND;
        /*
        C
        C ** PERFORM CHECK ON BOND LENGTH.
        C
        141 CHECK=RIJ*DSQRT(RDCX**2+RDCY**2+RDCZ**2)
        */
    }
    RIJ = R;
    double CHECK =
RIJ=Math.sqrt(Math.pow(RDCX,2)+Math.pow(RDCY,2)+Math.pow(RDCZ,2));
    if(debug == true) System.out.println("CHECK = "+CHECK);
    /*
    C
    C ** DETERMINE THE COORDINATES FOR THE ATOM BEING DEFINED.
    C
    */
for (int i=1;i<=3;i++){

    //RR=RDC[i];
    switch(i){
        case 1: RR = RDCX;break;
        case 2: RR = RDCY;break;
        case 3: RR = RDCZ;break;
    }
    if(debug == true) System.out.println("RR = "+RR);
    DC[i][IBOND]=RR;
    if(debug == true)
System.out.println("DC["+i+"] ["+IBOND+"] = "+DC[i][IBOND]);

    C[i][JJ]=C[i][II]+RR*RIJ;
    if(debug == true) System.out.println("this is statement 142");
    // 142 CONTINUE
}
/*

```

```

C
C    ** PRINT THE BOND DEFINITION PARAMETERS.
C
      WRITE(6,143) IBOND,NULL,II,JJ,PARM,(C(I,II),I=1,3),
.   (C(J,JJ),J=1,3),CHECK
      143  FORMAT(T4,I2,A2,2I4,3F12.4,2(2X,3F11.4),F11.4)
      GO TO 134
      */
      if(output == true) System.out.println("  IBOND= "+IBOND);
      if(debug == true) System.out.println("II = "+II);
      if(debug == true) System.out.println("JJ = "+JJ);
      if(output == true) System.out.print("C[1]["+II+"] = "+C[1][II]);
      if(output == true) System.out.print("; C[2]["+II+"] = "+C[2][II]);
      if(output == true) System.out.println("; C[3]["+II+"] = "+C[3][II]);
      if(debug == true) System.out.print("C[1]["+JJ+"] = "+C[1][JJ]);
      if(debug == true) System.out.print("; C[2]["+JJ+"] = "+C[2][JJ]);
      if(debug == true) System.out.println("; C[3]["+JJ+"] = "+C[3][JJ]);
      if(debug == true) System.out.println("CHECK = "+CHECK);

}while(IBOND < NBOND);
try {
    in.close();
} catch (IOException ex) {
    ex.printStackTrace();
}
for (int i=1;i<=NATOM;i++){
    for (int ii=1;ii<=3;ii++){
        System.out.print("C["+ii+"]["+ii+"] = "+C[ii][i]+";  ");
    }
    System.out.println();
}

Molecule h2o2 = new Molecule();
Atom atom = new Atom("H");
atom.setPoint3d(new Point3d( C[0][1], C[0][2], C[0][3] ));
h2o2.addAtom(atom);
atom = new Atom("O");
atom.setPoint3d(new Point3d(C[1][1], C[1][2], C[1][3]));
h2o2.addAtom(atom);
atom = new Atom("O");
atom.setPoint3d(new Point3d( C[2][1], C[2][2], C[2][3]));

```

```

h2o2.addAtom(atom);
atom = new Atom("H");
atom.setPoint3d(new Point3d( C[3][1], C[3][2], C[3][3]));
h2o2.addAtom(atom);

FileOutputStream writer;
FileOutputStream out; // declare a file output object
PrintStream p; // declare a print stream object
String outString = "";
String outString2 = "";
String ss1,ss2,ss3;
DecimalFormat myFormatter = new DecimalFormat("0000.00000");
//
//outString ="HEADER
//      COMPND      H2O2
outString = "";
int na = 0;
int atomNumber = 1;
String info = "";
String st = "";
String tab = "\t";
boolean writecharge = false;
boolean writevect = false;

String hetatmRecordName = "HETATM";
String terRecordName = "TER";
//      PrintfFormat serialFormat = new PrintfFormat("%5d");
//      PrintfFormat atomNameFormat = new PrintfFormat("%-4s");
//      PrintfFormat positionFormat = new PrintfFormat("%8.3f");
for (int i=1;i<=NATOM;i++){
    ss1 = myFormatter.format(C[1][i]);
    ss2 = myFormatter.format(C[2][i]);
    ss3 = myFormatter.format(C[3][i]);
    outString = outString + atomNameFormat.printf("HETATM");
    outString = outString+serialFormat.printf(i);
    outString = outString+" "+NAME[i]+           1";
    outString = outString + "   ";
    outString = outString+positionFormat.printf(C[1][i]);
    outString = outString+positionFormat.printf(C[2][i]);
    outString = outString+positionFormat.printf(C[3][i]);
    outString = outString + "  1.00  0.00          \n";
}
outString = outString +
        "TER        4                  1
        "END
System.out.print(outString);

```

```

String filename = in\_filename.substring(0,in\_filename.length()-4)+".pdb";
try {
    // Create a new file output stream
    // connected to "myfile.txt"
    out = new FileOutputStream(filename);
    // Connect print stream to the output stream
    p = new PrintStream( out );
}

p.println(outString);

p.close();
} catch (Exception e) {
    System.err.println("Error writing to file");
}
drawingCount++;
JFrame frame2 = new JFrame("Jmakemol drawing \#" + Integer.toString(drawingCount));
frame2.addWindowListener(new ApplicationCloser());
Container contentPane = frame2.getContentPane();
JmolPanel jmolPanel = new JmolPanel();
contentPane.add(jmolPanel);
frame2.setSize(600, 600);
frame2.setVisible(true);

JmolViewer viewer = jmolPanel.getViewer();
viewer.openFile(filename);
String strError = viewer.getOpenFileDialog();
if (strError != null)
    System.out.println(strError);

viewer.evalString(strScript);
}

static class JmolPanel extends JPanel {
    JmolViewer viewer;
    JmolAdapter adapter;

    JmolPanel() {
        // use CDK IO
        adapter = new CdkJmolAdapter(null);
        viewer = JmolViewer.allocateViewer(this, adapter);
    }

    public JmolViewer getViewer() {
        return viewer;
    }
}

```

```
final Dimension currentSize = new Dimension();
final Rectangle rectClip = new Rectangle();

public void paint(Graphics g) {
    viewer.setScreenDimension(getSize(currentSize));
    g.getClipBounds(rectClip);
    viewer.renderScreenImage(g, currentSize, rectClip);
}

static class ApplicationCloser extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code ">
private void initComponents() {
    jButton1 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    jEditorPane1 = new javax.swing.JEditorPane();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    menuBar = new javax.swing.JMenuBar();
    fileMenu = new javax.swing.JMenu();
    openMenuItem = new javax.swing.JMenuItem();
    saveMenuItem = new javax.swing.JMenuItem();
    saveAsMenuItem = new javax.swing.JMenuItem();
    exitMenuItem = new javax.swing.JMenuItem();
    editMenu = new javax.swing.JMenu();
    cutMenuItem = new javax.swing.JMenuItem();
    copyMenuItem = new javax.swing.JMenuItem();
    pasteMenuItem = new javax.swing.JMenuItem();
    deleteMenuItem = new javax.swing.JMenuItem();
    helpMenu = new javax.swing.JMenu();
    contentsMenuItem = new javax.swing.JMenuItem();
}
```

```

aboutMenuItem = new javax.swing.JMenuItem();

getContentPane().setLayout
(new org.netbeans.lib.awtextra.AbsoluteLayout());

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jButton1.setText("Recompute Coordinates and Re-Display");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

getContentPane().add
(jButton1, new org.netbeans.lib.awtextra.AbsoluteConstraints(190, 280, -1, -1));

jEditorPane1.setFont(new java.awt.Font("Monospaced", 0, 14));
jEditorPane1.setEditable(true);
jEditorPane1.setName("myEditor");
jScrollPane1.setViewportView(jEditorPane1);

getContentPane().add
(jScrollPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 60, 380, 180));

jLabel1.setFont(new java.awt.Font("Monospaced", 0, 14));
jLabel1.setText("two integers, three floating point, one char");
getContentPane().add
(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 30, -1, -1));

jLabel2.setFont(new java.awt.Font("Monospaced", 0, 14));
jLabel2.setText("atom1:atom2:R: theta: chi:Name");
getContentPane().add
(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 10, -1, -1));

jLabel3.setFont(new java.awt.Font("Tahoma", 3, 14));
jLabel3.setForeground(new java.awt.Color(255, 0, 51));
jLabel3.setText("WARNING, increments to new .inp file>");
getContentPane().add
(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(10, 260, -1, -1));

fileMenu.setText("File");
openMenuItem.setText("Open");
fileMenu.add(openMenuItem);

saveMenuItem.setText("Save");
fileMenu.add(saveMenuItem);

```

```
saveAsMenuItem.setText("Save As ...");
fileMenu.add(saveAsMenuItem);

exitMenuItem.setText("Exit");
exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        exitMenuItemActionPerformed(evt);
    }
});

fileMenu.add(exitMenuItem);

menuBar.add(fileMenu);

editMenu.setText("Edit");
cutMenuItem.setText("Cut");
editMenu.add(cutMenuItem);

copyMenuItem.setText("Copy");
editMenu.add(copyMenuItem);

pasteMenuItem.setText("Paste");
editMenu.add(pasteMenuItem);

deleteMenuItem.setText("Delete");
editMenu.add(deleteMenuItem);

menuBar.add(editMenu);

helpMenu.setText("Help");
contentsMenuItem.setText("Contents");
helpMenu.add(contentsMenuItem);

aboutMenuItem.setText("About");
aboutMenuItem.addActionListener
(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aboutMenuItemActionPerformed(evt);
    }
});

helpMenu.add(aboutMenuItem);

menuBar.add(helpMenu);
```

```

        setJMenuBar(menuBar);

        pack();
    } // </editor-fold>

    private void aboutMenuItemActionPerformed
    (java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
        JFrame aboutFrame = new JFrame("About");
        JOptionPane.showMessageDialog(aboutFrame,"JMakeMol, a program" +
            "to create small molecules using bond lengths and bond angles." +
            "\n Carl W. David\n Department of Chemistry \n
                University of Connecticut\n "
            + "Carl.David@uconn.edu\n " +
            "February 2006");
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
        try {
            EditableString = jEditorPane1.getText();
            in\_filename = in\_filename.substring
(0,in\_filename.length()-4)+"_"+Integer.toString(drawingCount)+".inp";
            BufferedWriter out = new BufferedWriter(new FileWriter(in\_filename));
            out.write(EditableString);
            out.close();
        } catch (IOException e) {
        }
        EditableString = "";
        cdnt2();
    }

    private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }

    /**
     * @param args the command line arguments
     */
    /*
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new cdnt().setVisible(true);
            }
        });
    }

```

```
}

/*
// Variables declaration - do not modify
private javax.swing.JMenuItem aboutMenuItem;
private javax.swing.JMenuItem contentsMenuItem;
private javax.swing.JMenuItem copyMenuItem;
private javax.swing.JMenuItem cutMenuItem;
private javax.swing.JMenuItem deleteMenuItem;
private javax.swing.JMenu editMenu;
private javax.swing.JMenuItem exitMenuItem;
private javax.swing.JMenu fileMenu;
private javax.swing.JMenu helpMenu;
private javax.swing.JButton jButton1;
private javax.swing.JEditorPane jEditorPanel1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JMenuBar menuBar;
private javax.swing.JMenuItem openMenuItem;
private javax.swing.JMenuItem pasteMenuItem;
private javax.swing.JMenuItem saveAsMenuItem;
private javax.swing.JMenuItem saveMenuItem;
// End of variables declaration
int NBOND, NATOM, IBOND, II, JJ, KK, LL, MM, IMAX, NGO, IREF1, IREF, IATOM;
int IREF2, NNATOM;
boolean JUMP;
double ZETA, R, COS1, SIN1, SENSE, RDCX, RDCY, RDCZ, AA, BB, CC, ETA, ZREF, RR;
public double AX, BX, CX, AY, BY, CY, AZ, BZ, CZ, RIJ;
String s1,s2;
public double RADDEG=0.01745329251994329;
public double [] RDC;
boolean debug = false;
boolean output = false;
public static String hetatmRecordName = "HETATM";
public static String terRecordName = "TER";
public static PrintFormat serialFormat = new PrintFormat("%5d");
public static PrintFormat atomNameFormat = new PrintFormat("%-4s");
public static PrintFormat positionFormat = new PrintFormat("%8.3f");
final static String strScript = "select *; ball-and-stick on";
public String EditableString = "";
//public File
"C:/Documents and Settings/david/My Documents/
Java Project Folder/JmakeMol/JmakeMol/src/jmakemol/input.dat";
```

```
String in\_filename = "";
public BufferedReader in ;
public int drawingCount = 0;

public boolean chooseInputDirectory = true;
    /*the following must be changed before putting program into production
       by changing to ""
    */
public String currentDirectoryPath =
"C:/Documents and Settings/david/My Documents/
Java Project Folder/JmakeMol/JmakeMol/src/jmakemol/";
//public String currentDirectoryPath = "";
}
```