# Virulo 1.0 Code Snapshot
# 18 November 2002

Bart Faulkner
EPA National Risk Management Laboratory
Subsurface Protection and Remediation Division
Ada, Oklahoma

# Contents


1 Virulo.java 1

2 ViruloFrame.java 2

3 Medium.java 14

4 StringAsChars.java 16

5 Normal.java 17

6 MediumXMLWriter.java 18

7 Operandum.java 20

8 OperandumXMLWriter.java 22

9 HspMonteCarlo.java 24

10 Random.java 27

11 GasDev.java 32

12 Mvn.java 34

13 VarCov.java 36

14 Attenuator.java 44

15 AboutFrame.java 45

16 JarLoadable.java 46

17 Gossiper.java 47

18 FlowPanel.java 48

19 FlowComboPanel.java 56

20 SoilStack.java 58

21 NormalF.java 68

22 VirusPanel.java 69

23 VirusComboPanel.java 74

24 VirusStack.java 76

25 OutputPanel.java 83

26 HistoPanel.java 84

27 JpgFilter.java 86

28 DataBuilder.java 87

29 UnsatVirusAttenuator.java 90

30 ImagePanel.java 95

31 Utils.java 96

32 HspBasicMath.java 97

# 1 Virulo.java

```
/**
   <b>Virulo</b> class.
<P>

@author Barton R Faulkner, US EPA National Risk Management Laboratory,
        Ada, Oklahoma, USA.
@version 2 February 2001
*/


public class Virulo {
        public static void main(String[] args) {
                ViruloFrame vmf=new ViruloFrame();
                vmf.show();
        }
}
```

## 2 ViruloFrame.java

```
import com.sun.image.codec.jpeg.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.image.BufferedImage;
import java.text.*;

/**
   <b>ViruloFrame</b> class.
<P>

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 17 October 2002
@see JFrame
*/

public class ViruloFrame extends JFrame {
        Toolkit tk=Toolkit.getDefaultToolkit();
        Dimension d=tk.getScreenSize();
        int screenHeight=d.height;
        int screenWidth =d.width;
        /** Construct a <b>ViruloFrame</b>.
        */
        public ViruloFrame() {
                jar=new JarLoadable();
                gossiper=new Gossiper();
                setTitle("Virulo 1.0");
                setSize(650,620);
                setLocation( (int)(screenWidth/30), (int)(screenHeight/30) );
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
                fileMenu=new JMenu("File");
                openItem=new JMenuItem("Open xml ..");
                saveItem=new JMenuItem("Save as ..");
                chooser=new JFileChooser();
                chooser.setFileFilter(new JpgFilter());
                saveItem.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent se) {
                           chooser.showSaveDialog(null);
                           File file=chooser.getSelectedFile();
```

```
            String fileString=file.getPath();
            if (tp.getSelectedIndex()==3) {
                 try {
                         fileString+=".txt";
                         rof=new RandomAccessFile(
                                 fileString,"rw");
                         rof.writeBytes(outString);
                         rof.close();
                 } catch (IOException ie) {
                 }
            } else if (tp.getSelectedIndex()==2) {
                 try {
                         out=new FileOutputStream(file);
                         JPEGImageEncoder encoder=
                                 JPEGCodec.createJPEGEncoder(out);
                         encoder.encode(panelGout.getImage());
                 } catch (IOException ie) {
                 }
            } else if (tp.getSelectedIndex()==0) {
                 if (fileString.endsWith(".xml")) {
                         fileString=fileString.substring(
                                 0,fileString.lastIndexOf("."));
                 }
                 sd=panelFlow.getData();
                 sdwriter=new MediumXMLWriter(sd,fileString);
                 if (!fileString.endsWith(".xml")) fileString+=".xml";
                 sdwriter.writeXML(fileString);
            } else if (tp.getSelectedIndex()==1) {
                 if (fileString.endsWith(".xml")) {
                         fileString=fileString.substring(
                                 0,fileString.lastIndexOf("."));
                 }
                 vd=panelVirus.getData();
                 vdwriter=new OperandumXMLWriter(vd,fileString);
                 if (!fileString.endsWith(".xml")) fileString+=".xml";
                 vdwriter.writeXML(fileString);
            }
         }
});
saveItem.setMnemonic('S');
fileMenu.add(saveItem);
exitItem=new JMenuItem("Exit");
exitItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ee) {
                System.exit(0);
        }
});
openItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent oe) {
           chooser.showOpenDialog(null);
```

```
            File file=chooser.getSelectedFile();
            String fileString=file.getPath();
            obj=new Object();
            try {
                 if (!fileString.endsWith(".xml")) fileString+=".xml";
                 rof=new RandomAccessFile(fileString,"r");
                 DataBuilder builder=new DataBuilder();
                 obj=builder.getData(rof);
                 if (obj.toString().startsWith("Me"))
                         panelFlow.putData((Medium)obj);
                 if (obj.toString().startsWith("Op"))
                         panelVirus.putData((Operandum)obj);
            } catch (IOException ie) {
            }
         }
});
openItem.setMnemonic('O');
fileMenu.add(openItem);
fileMenu.add(exitItem);
exitItem.setMnemonic('x');
editMenu=new JMenu("Edit");
editMenu.setMnemonic('E');
copyItem=new JMenuItem("Copy to clipboard");
copyItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ce) {
            if (tp.getSelectedIndex()==3) panelTout.copy();
        }
});
copyItem.setMnemonic('C');
editMenu.add(copyItem);
selectItem=new JMenuItem("Select all");
selectItem.setMnemonic('a');
selectItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ae) {
            if (tp.getSelectedIndex()==3) panelTout.selectAll();
        }
});
editMenu.add(selectItem);
runMenu=new JMenu("Run");
runMenu.setMnemonic('R');
goItem=new JMenuItem("Start Simulation",new ImageIcon(
        jar.loadImage("run.gif")));
goItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent ge) {
                stop=false;
                tp.setSelectedIndex(2);
                MloThread=new UpdateThread();
                MloThread.start();
        }
});
```

```
stopItem=new JMenuItem("Stop Simulation",new ImageIcon(
        jar.loadImage("stop.gif")));
stopItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent se) {
                stop=true;
        }
});
runMenu.add(goItem);
runMenu.add(stopItem);
goItem.setMnemonic('S');
helpMenu=new JMenu("Help");
aboutItem=new JMenuItem("About Virulo..");
aboutItem.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent afe) {
                aframe=new AboutFrame();
                aframe.show();
        }
});
aboutItem.setMnemonic('A');
helpMenu.add(aboutItem);
helpMenu.setMnemonic('H');
runButton=new JButton("Start Simulation",
        new ImageIcon(
        jar.loadImage("run.gif")));
sLabel=new JLabel(" Threshold Attenuation (\u03b5): ");
inLabel=new JLabel(" ( -log10 )  ");
breakPointTextField=new JTextField("4",3);
breakPointTextField.setHorizontalAlignment(JTextField.CENTER);
breakPointTextField.setToolTipText("The logarithm, base 10, of "+
        "the target attenuation factor");
runButton.setToolTipText("Start new simulation with current parameters");
runButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                stop=false;
                tp.setSelectedIndex(2);
                MloThread=new UpdateThread();
                MloThread.start();
        }
});
stopButton=new JButton("Stop",
        new ImageIcon(
        jar.loadImage("stop.gif")));
stopButton.setToolTipText("Stop current simulation");
stopButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                stop=true;
        }
});
mb=new JMenuBar();
mb.add(fileMenu);
```

```
        fileMenu.setMnemonic('F');
        mb.add(editMenu);
        mb.add(runMenu);
        mb.add(helpMenu);
        mb.add(runButton);
        mb.add(stopButton);
        mb.add(sLabel);
        mb.add(breakPointTextField);
        mb.add(inLabel);
        setJMenuBar(mb);

        Container contentPane=getContentPane();

        panelFlow=new FlowPanel(gossiper);
        panelVirus=new VirusPanel(gossiper);
        bin=new int[numbins];
        panelGout=new HistoPanel(bin,hpanelwd,hpanelht);
        retainCheckBox=new JCheckBox("Retain and Accumulate");
        panelGout.add(retainCheckBox);
        panelTout=new OutputPanel();
        JScrollPane tsp=new JScrollPane(panelTout);

        tp=new JTabbedPane();
        tp.addTab("Flow Parameters",new ImageIcon(
                jar.loadImage("flow.gif")),
                jar.loadImage("flow.gif")),
                panelFlow,"flow parameters");
        tp.addTab("Virus Parameters",new ImageIcon(
                jar.loadImage("virus.gif")),
                panelVirus,"virus parameters");
        tp.addTab("Histogram",new ImageIcon(
                jar.loadImage("histogram.gif")),
                panelGout,"histogram");
        tp.addTab("Probability",new ImageIcon(
                jar.loadImage("tout.gif")),
                tsp,"text");
        getContentPane().add(tp,"Center");

        sd=panelFlow.getData();
        vd=panelVirus.getData();
        soilData=(Medium)(sd.clone());
        virusData=(Operandum)(vd.clone());
        Mlo=new HspMonteCarlo(soilData,virusData);
}
/** Run the model with the current parameters to be sent to the
<b>Attenuator</b> object.
*/
class UpdateThread extends Thread {
        Runnable update, finish;
        public UpdateThread() {
```

```
if (!retainCheckBox.isSelected()) runcount=0;
update=new Runnable() {
        public void run() {
                panelGout.setBreak(bp);
                panelGout.setHits(hits);
                panelGout.setRuns(runcount);
                panelGout.redraw(bin);
        }
};
finish=new Runnable() {
        public void run() {
                panelGout.setRuns(runcount-1);
                panelGout.setHits(hits);
                panelGout.redraw(bin);
                String[] sLabels=panelFlow.getLabels();
                String[] vLabels=panelVirus.getLabels();
                String[] sUnits=panelFlow.getUnits();
                String[] vUnits=panelVirus.getUnits();
                if (panelFlow.theta_mIsUniform()) {
                        theString=": random between "+
                                sLabels[1]+" and "+sLabels[3];
                } else {
                        theString=": "+
                                soilData.theta_m.mean+
                                "("+soilData.theta_m.sdev+") ";
                }
                outString="Output from Virulo "+
                        "model:\n\n"+
                        "Input parameters used:\n"+
                        "Parameter: Mean Value"+
                        "(Standard Deviation) units:\n"+
                        "Soil Parameters:\n   "+sLabels[0]+
                        ": "+soilData.theta_r.mean+
                        "("+soilData.theta_r.sdev+") "+
                        sUnits[0]+"\n   "+sLabels[1]+
                        theString+
                        ".\n   "+sLabels[2]+
                        ": "+soilData.theta_s.mean+
                        "("+soilData.theta_s.sdev+") "+
                        sUnits[2]+"\n   "+sLabels[3]+
                        ": "+soilData.K0.mean+
                        "("+soilData.K0.sdev+") "+
                        sUnits[3]+"\n   "+sLabels[4]+
                        ": "+soilData.a.mean+
                        "("+soilData.a.sdev+") "+
                        sUnits[4]+"\n   "+sLabels[5]+
                        ": "+soilData.n.mean+
                        "("+soilData.n.sdev+") "+
                        sUnits[5]+"\n   "+sLabels[6]+
                        ": "+soilData.rho.mean+
```

```
                                        "("+soilData.rho.sdev+") "+
                                        sUnits[6]+"\n   "+sLabels[7]+
                                        ": "+soilData.rp.mean+
                                        "("+soilData.rp.sdev+") "+
                                        sUnits[7]+"\n   "+sLabels[8]+
                                        ": "+soilData.alphaz.mean+
                                        "("+soilData.alphaz.sdev+") "+
                                        sUnits[8]+"\n   "+sLabels[9]+
                                        ": "+soilData.t.mean+
                                        "("+soilData.t.sdev+") "+
                                        sUnits[9]+"\n   "+sLabels[10]+
                                        ": "+soilData.z.mean+
                                        "("+soilData.z.sdev+") "+
                                        sUnits[10]+"\n\n"+
                                        "Virus Parameters:\n   "+
                                        vLabels[0]+": "+
                                        virusData.lambda0.mean+"("+
                                        virusData.lambda0.sdev+") "+
                                        vUnits[0]+"\n   "+
                                        vLabels[1]+": "+
                                        virusData.lambda1.mean+"("+
                                        virusData.lambda1.sdev+") "+
                                        vUnits[1]+"\n   "+
                                        vLabels[2]+": "+
                                        virusData.kappa0.mean+"("+
                                        virusData.kappa0.sdev+") "+
                                        vUnits[2]+"\n   "+
                                        vLabels[3]+": "+
                                        virusData.kappa1.mean+"("+
                                        virusData.kappa1.sdev+") "+
                                        vUnits[3]+"\n   "+
                                        vLabels[4]+": "+
                                        virusData.rv.mean+"("+
                                        virusData.rv.sdev+") "+
                                        vUnits[4]+"\n   "+
                                        vLabels[5]+": "+
                                        virusData.Kd.mean+"("+
                                        virusData.Kd.sdev+") "+
                                        vUnits[5]+"\n\n"+
                                        "The probability of failure to achieve "+
                                                bp+"-log attenuation from "+
                                                runcount+
                                                " Monte\n"+"Carlo runs was "+
                                                hits+":"+runcount+".";
                                panelTout.printOutput(outString);
                        }
                };
        }
        public void run() {
                continuing=true;
```

```
bpString=breakPointTextField.getText();
bp=Double.parseDouble(bpString);
sd=panelFlow.getData();
vd=panelVirus.getData();
soilData=(Medium)(sd.clone());
virusData=(Operandum)(vd.clone());

// Now trap invalid user inputs

if (!(bp>0.)) {
        Message="The threshold value you have chosen, \u03b5="+
          bpString+",\n"+
          "imples you expect that the total mass leaching\n"+
          "could be greater than the total mass put in the\n"+
          "soil. Please choose a value greater than zero.";
        JOptionPane.showMessageDialog(null,Message,
          "Illogical choice",JOptionPane.QUESTION_MESSAGE);
        continuing=false;
}
if (continuing && sd.theta_r.mean<Double.MIN_VALUE ||
    sd.theta_r.mean>=1.) {
        tp.setSelectedIndex(0);
        Message="Error, the mean value for this parameter must\n"+
                "be between zero and one. You entered "+
                String.valueOf(sd.theta_r.mean);
        JOptionPane.showMessageDialog(null,Message,
          "Virulo error",JOptionPane.PLAIN_MESSAGE,
                new ImageIcon(jar.loadImage("thetar.gif")));
        panelFlow.requestAttention("theta_r.m");
        continuing=false;
}
if (continuing && !panelFlow.theta_mIsUniform() &&
    (sd.theta_m.mean >=sd.theta_s.mean ||
     sd.theta_m.mean<=sd.theta_r.mean)) {
        tp.setSelectedIndex(0);
        Message="Error, the mean value for this parameter must\n"+
           "be between the means of the residual water content\n"+
           "and the saturated water content. You entered "+
           String.valueOf(sd.theta_m.mean);
        JOptionPane.showMessageDialog(null,Message,
           "Virulo error",JOptionPane.PLAIN_MESSAGE,
                new ImageIcon(jar.loadImage("thetam.gif")));
        panelFlow.requestAttention("theta_m.m");
        continuing=false;
}
if (continuing && sd.theta_s.mean<Double.MIN_VALUE ||
    sd.theta_s.mean>=1.) {
        tp.setSelectedIndex(0);
        Message="Error, the mean value for this parameter must\n"+
                "be between zero and one. You entered "+
```

```
                String.valueOf(sd.theta_s.mean);
        JOptionPane.showMessageDialog(null,Message,
                "Virulo error",JOptionPane.PLAIN_MESSAGE,
                new ImageIcon(jar.loadImage("thetas.gif")));
        panelFlow.requestAttention("theta_s.m");
        continuing=false;
}
if (continuing && sd.rho.mean<0.)
        defaultNegativeNumberError(
                sd.rho.mean,"rho.gif","rho.m",0);
if (continuing && sd.rp.mean<0.)
        defaultNegativeNumberError(sd.rp.mean,"rp.gif","rp.m",0);
if (continuing && sd.alphaz.mean<0.)
        defaultNegativeNumberError(
                sd.alphaz.mean,"alphaz.gif","alphaz.m",0);
if (continuing && sd.z.mean<0.)
        defaultNegativeNumberError(
                sd.z.mean,"l.gif","L.m",0);
if (continuing && vd.rv.mean<0.)
        defaultNegativeNumberError(
                vd.rv.mean,"rv.gif","rv.m",1);
if (continuing && sd.rho.sdev<0.)
        defaultSdevNegativeNumberError(
                sd.rho.sdev,"rho.gif","rho.s",0);
if (continuing && sd.rp.sdev<0.)
        defaultSdevNegativeNumberError(
                sd.rp.sdev,"rp.gif","rp.s",0);
if (continuing && sd.alphaz.sdev<0.)
        defaultSdevNegativeNumberError(
                sd.alphaz.sdev,"alphaz.gif","alphaz.s",0);
if (continuing && sd.t.sdev<0.)
        defaultSdevNegativeNumberError(
                sd.t.sdev,"t.gif","t.s",0);
if (continuing && sd.z.sdev<0.)
        defaultSdevNegativeNumberError(
                sd.z.sdev,"l.gif","z.s",0);
if (continuing && vd.kappa0.sdev<0.)
        defaultSdevNegativeNumberError(
                vd.kappa0.sdev,"kappa.gif","kappa0.s",1);
if (continuing && vd.kappa1.sdev<0.)
        defaultSdevNegativeNumberError(
                vd.kappa1.sdev,"kappadia.gif","kappa1.s",1);
if (continuing && vd.rv.sdev<0.)
        defaultSdevNegativeNumberError(
                vd.rv.sdev,"rv.gif","rv.s",1);
if (continuing && vd.Kd.sdev<0.)
        defaultSdevNegativeNumberError(
                vd.Kd.sdev,"kd.gif","Kd.s",1);

if (continuing) {
```

```
                        ccount++;
                        if (!retainCheckBox.isSelected()) {
                                ccount=1;
                                hits=0;
                                Mlo=new HspMonteCarlo(soilData,virusData);
                                for (i=0; i<numbins; i++) {
                                        bin[i]=0;
                                }
                        }
                        runButton.setEnabled(false);
                        Mlo.setTheta_mUniform(panelFlow.theta_mIsUniform());
                        while (!stop && runcount++<ccount*1000000) {
                                // Note, Mlo.getNext returns simply
                                // the log10(A), thus
                                mf=-Mlo.getNext();
                                if (mf<=300) {
                                        bin[(int)(mf/4)]++;
                                        if (mf<bp) hits++;
                                }
                                SwingUtilities.invokeLater(update);
                        }
                        SwingUtilities.invokeLater(finish);
                        runButton.setEnabled(true);
                }
        }
}
private void defaultNegativeNumberError(
    double bad, String gifString, String attentor, int idx) {
        tp.setSelectedIndex(idx);
        Message="Error, the mean value for this parameter must\n"+
                "be greater than zero. You entered "+
                String.valueOf(bad);
         JOptionPane.showMessageDialog(
                null,Message,"Virulo error",JOptionPane.PLAIN_MESSAGE,
                new ImageIcon(jar.loadImage(gifString)));
        if (idx==0) {
                panelFlow.requestAttention(attentor);
        } else {
                panelVirus.requestAttention(attentor);
        }
        continuing=false;
}
private void defaultSdevNegativeNumberError(
    double bad, String gifString, String attentor, int idx) {
        tp.setSelectedIndex(idx);
        Message="Error, the Std. Deviation value for this parameter must\n"+
                "be greater than zero. You entered "+
                String.valueOf(bad);
         JOptionPane.showMessageDialog(
                null,Message,"Virulo error",JOptionPane.PLAIN_MESSAGE,
```

```
                new ImageIcon(jar.loadImage(gifString)));
        if (idx==0) {
                panelFlow.requestAttention(attentor);
        } else {
                panelVirus.requestAttention(attentor);
        }
        continuing=false;
}
private Object obj;
private String Message, bpString;
private boolean stop=true, continuing;
private UpdateThread MloThread;
private Medium soilData,sd;
private MediumXMLWriter sdwriter;
private Operandum virusData,vd;
private OperandumXMLWriter vdwriter;
private HspMonteCarlo Mlo;
private int hpanelwd=400, hpanelht=460;
private int ccount;
private int[] bin;
private AboutFrame aframe;
private RandomAccessFile rof;
private OutputStream out;
private double mf;
private int i,it,hits,nancount,runcount,numbins=75;
private double bp=4.;
private JarLoadable jar;
private String theString="";
private String outString="";
private JFileChooser chooser;
private JLabel sLabel, inLabel;
private JMenuBar mb;
private JMenu fileMenu;
private JMenu editMenu;
private JMenu runMenu;
private JMenu helpMenu;
private JMenuItem saveItem;
private JMenuItem openItem;
private JMenuItem exitItem;
private JMenuItem copyItem;
private JMenuItem selectItem;
private JMenuItem goItem;
private JMenuItem stopItem;
private JMenuItem aboutItem;
private JButton runButton;
private JButton stopButton;
private JCheckBox retainCheckBox;
private JTextField breakPointTextField;
private JTabbedPane tp;
private Gossiper gossiper;
```

```
        private BufferedImage image;
        private FlowPanel panelFlow;
        private VirusPanel panelVirus;
        private OutputPanel panelTout;
        private HistoPanel panelGout;
}
```

## 3 Medium.java

```
/**
   <b>Medium</b> class.
<P>
Data structure for transport medium. Notice its instance is a
composed object that is cloneable.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 5 March 2001
*/

public class Medium implements Cloneable {

        /** USDA name of the soil stored as a char[].
        */
        public StringAsChars name=new StringAsChars();

        /** Empirical constant van Genucten's alpha (dimensionless).
        */
        public Normal a=new Normal();

        /** Soil water content parameters:
                {Residual water content,
                Water content,
                Saturated water content}
                (dimensionless).
        */
        public Normal theta_s=new Normal();
        public Normal theta_r=new Normal();
        public Normal theta_m=new Normal();

        /** Thickness of the proposed hydrogeologic barrier (meters).
        */
        public Normal z=new Normal();

        /** Vertical dispersivity (meters).
        */
        public Normal alphaz=new Normal();

        /** van Genuchten parameter (dimensionless).
        */
        public Normal n=new Normal();

        /** Temperature for computing molecular diffusivity (degrees Celsius).
        */
        public Normal t=new Normal();
```

```
        /** Soil bulk density (g/m^3).
        */
        public Normal rho=new Normal();

        /** Average radius of soil particles (meter).
        */
        public Normal rp=new Normal();

        /** Saturated hydraulic conductivity (meter/hr).
        */
        public Normal K0=new Normal();

        /** Clone method redefinition
        */
        public Object clone() {
                // Method described by Bruce Eckel p. 555-556.
                Medium sd=null;
                try {
                        sd=(Medium)super.clone();
                } catch (CloneNotSupportedException e) {
                }
                // Clone the Normals too:
                Normal tmp=null;
                sd.name=(StringAsChars)sd.name.clone();
                sd.a=(Normal)sd.a.clone();
                sd.theta_s=(Normal)sd.theta_s.clone();
                sd.theta_r=(Normal)sd.theta_r.clone();
                sd.theta_m=(Normal)sd.theta_m.clone();
                sd.z=(Normal)sd.z.clone();
                sd.alphaz=(Normal)sd.alphaz.clone();
                sd.n=(Normal)sd.n.clone();
                sd.t=(Normal)sd.t.clone();
                sd.rho=(Normal)sd.rho.clone();
                sd.rp=(Normal)sd.rp.clone();
                sd.K0=(Normal)sd.K0.clone();
                return sd;
        }
}
```

# 4 StringAsChars.java

```
/** Data structure for a string in hsp.
@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
*/

public class StringAsChars implements Cloneable {
        public char[] is;
        public Object clone() {
                try {
                        return super.clone();
                } catch (CloneNotSupportedException e) {
                        return null;
                }
        }
}
```

## 5 Normal.java

```
/** Data structure to describe a normally distributed
        random variable.
@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 5 November 2002
*/

public class Normal implements Cloneable {
        public Normal() {
        }
        public Normal(double m, double s) {
                mean=m;
                sdev=s;
        }
        public double mean;
        public double sdev;
        public Object clone() {
                try {
                        return super.clone();
                } catch (CloneNotSupportedException e) {
                        return null;
                }
        }
}
```

## 6 MediumXMLWriter.java

```
import java.io.RandomAccessFile;
import java.io.IOException;
import java.util.StringTokenizer;

public class MediumXMLWriter {
        public MediumXMLWriter(Medium medium, String name) {
                m=medium;
                n=name;
                tokens=new StringTokenizer(n,"/");
        }
        public void writeXML(String fname) {
                for (i=0; i<=tokens.countTokens(); i++) {
                        lastpart=tokens.nextToken();
                }
                lastpart=tokens.nextToken();
                xS="<?xml version=\"1.0\"?>\n\n";
                xS+="<media>\n";
                xS+=" <medium>\n";
                xS+="  <name>"+lastpart+"_xml</name>\n";
                xS+="   <th_r>\n    <mean>"+Double.toString(m.theta_r.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.theta_r.sdev)+"</sdev>\n   </th_r>\n";
                xS+="   <th_m>\n    <mean>"+Double.toString(m.theta_m.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.theta_m.sdev)+"</sdev>\n   </th_m>\n";
                xS+="   <th_s>\n    <mean>"+Double.toString(m.theta_s.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.theta_s.sdev)+"</sdev>\n   </th_s>\n";
                xS+="   <logks>\n    <mean>"+Double.toString(m.K0.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.K0.sdev)+"</sdev>\n   </logks>\n";
                xS+="   <logal>\n    <mean>"+Double.toString(m.a.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.a.sdev)+"</sdev>\n   </logal>\n";
                xS+="   <logn>\n    <mean>"+Double.toString(m.n.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.n.sdev)+"</sdev>\n   </logn>\n";
                xS+="   <rho>\n    <mean>"+Double.toString(m.rho.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.rho.sdev)+"</sdev>\n   </rho>\n";
                xS+="   <rp>\n    <mean>"+Double.toString(m.rp.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.rp.sdev)+"</sdev>\n   </rp>\n";
                xS+="   <alpz>\n    <mean>"+Double.toString(m.alphaz.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.alphaz.sdev)+"</sdev>\n   </alpz>\n";
                xS+="   <temp>\n    <mean>"+Double.toString(m.t.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.t.sdev)+"</sdev>\n   </temp>\n";
                xS+="   <len>\n    <mean>"+Double.toString(m.z.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.z.sdev)+"</sdev>\n   </len>\n";
                xS+=" </medium>\n";
                xS+="</media>\n";

                try {
                        raf=new RandomAccessFile(fname,"rw");
                        raf.setLength(0);
                        raf.writeBytes(xS);
```

```
                raf.close();
            } catch(IOException e) {}
        }
        private RandomAccessFile raf;
        private StringTokenizer tokens;
        private String xS, n, lastpart;
        private Medium m;
        private int i;
}
```

# 7 Operandum.java

```
/**
   <b>Operandum</b> class.
<P>
Data structure for passive participants in transport.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 5 March 2001
*/

public class Operandum implements Cloneable {

        /** The name of this operandum.
        */
        public StringAsChars name=new StringAsChars();

        /** Inactivation rate coefficients {liquid, solid-sorbed, air-sorbed}
                (log[per hour]).
        */
        public Normal lambda0=new Normal();
        public Normal lambda1=new Normal();

        /** Mass transfer coefficients {liquid-liq/solid, liquid-liq/air}
                (per hour).
        */
        public Normal kappa0=new Normal();
        public Normal kappa1=new Normal();

        /** Radius (m)
        */
        public Normal rv=new Normal();
        /** Initial concentration or mass (any units).
        */
        public Normal Kd=new Normal();
        /** Clone method redefinition
        */
        public Object clone() {
                Operandum op=null;
                try {
                        op=(Operandum)super.clone();
                } catch (CloneNotSupportedException e) {
                }
                // clone the handles too.
                op.name=(StringAsChars)op.name.clone();
                op.lambda0=(Normal)op.lambda0.clone();
                op.lambda1=(Normal)op.lambda1.clone();
```

```
                op.kappa0=(Normal)op.kappa0.clone();
                op.kappa1=(Normal)op.kappa1.clone();
                op.rv=(Normal)op.rv.clone();
                op.Kd=(Normal)op.Kd.clone();
                return op;
        }
}
```

## 8 OperandumXMLWriter.java

```
import java.io.RandomAccessFile;
import java.io.IOException;
import java.util.StringTokenizer;

// This class is a surrogate for a forthcoming bona fide XMLWriter.
// It'll do for now.

public class OperandumXMLWriter {
        public OperandumXMLWriter(Operandum operandum, String name) {
                m=operandum;
                n=name;
                tokens=new StringTokenizer(n,"/");
        }
        public void writeXML(String fname) {
                for (i=0; i<=tokens.countTokens(); i++) {
                        lastpart=tokens.nextToken();
                }
                lastpart=tokens.nextToken();
                xS="<?xml version=\"1.0\"?>\n\n";
                xS+="<operandi>\n";
                xS+=" <operandum>\n";
                xS+="  <name>"+lastpart+"_xml</name>\n";
                xS+="   <lam0>\n    <mean>"+Double.toString(m.lambda0.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.lambda0.sdev)+"</sdev>\n   </lam0>\n";
                xS+="   <lam1>\n    <mean>"+Double.toString(m.lambda1.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.lambda1.sdev)+"</sdev>\n   </lam1>\n";
                xS+="   <kap0>\n    <mean>"+Double.toString(m.kappa0.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.kappa0.sdev)+"</sdev>\n   </kap0>\n";
                xS+="   <kap1>\n    <mean>"+Double.toString(m.kappa1.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.kappa1.sdev)+"</sdev>\n   </kap1>\n";
                xS+="   <radv>\n    <mean>"+Double.toString(m.rv.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.rv.sdev)+"</sdev>\n   </radv>\n";
                xS+="   <kd>\n    <mean>"+Double.toString(m.Kd.mean)+"</mean>\n";
                xS+= "    <sdev>"+Double.toString(m.Kd.sdev)+"</sdev>\n   </kd>\n";
                xS+=" </operandum>\n";
                xS+="</operandi>\n";

                try {
                        raf=new RandomAccessFile(fname,"rw");
                        raf.setLength(0);
                        raf.writeBytes(xS);
                        raf.close();
                } catch(IOException e) {}
        }
        private RandomAccessFile raf;
        private StringTokenizer tokens;
        private String xS, n, lastpart;
        private Operandum m;
```

```
        private int i;
}
```

# 9 HspMonteCarlo.java

```
import javax.swing.*;
import java.awt.*;
import java.util.*;
import Jama.*;

/**
Class for generating Monte Carlo simulations.

@see Mvn
@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
*/

public class HspMonteCarlo {
        public HspMonteCarlo(Medium soildata,Operandum virusdata) {
                sd=soildata;
                vd=virusdata;
                // and make extra copies that can be perturbed:

                sdp=(Medium)sd.clone();
                vdp=(Operandum)vd.clone();
                att=new UnsatVirusAttenuator(sd,vd);
                maxit=100000;
                gen=new Random(0.,1.);
                gasdev=new GasDev();
                varcov=new VarCov();
                Matrix Cov=varcov.getVarCov(String.valueOf(sd.name.is));
                double[] means={sd.theta_r.mean,
                                sd.theta_s.mean,
                                sd.a.mean,
                                sd.n.mean,
                                sd.K0.mean
                        };
                mvn=new Mvn(Cov,means);
                getMvn();
        }
        private double[][] getMvn() {
                bb=mvn.perturb();
                return bb;
        }
        public double getNext() {
                if (++i>=maxit) {
                        getMvn();
                        i=0;
                }
//              varcov=new VarCov();
```

```
        sdp.a.mean=bb[i][2];
        sdp.theta_s.mean=bb[i][1];
        sdp.theta_r.mean=bb[i][0];
        if (theta_mIsUniform) {
                unv=(sdp.theta_s.mean-sdp.theta_r.mean)*gen.nextDouble();
                sdp.theta_m.mean=sdp.theta_r.mean+unv;
        } else {
                sdp.theta_m.mean=perturb(sd.theta_m.mean,sd.theta_m.sdev);
        }
        sdp.z.mean=perturb(sd.z.mean,sd.z.sdev);
        sdp.alphaz.mean=perturb(sd.alphaz.mean,
                sd.alphaz.sdev);
        sdp.n.mean=bb[i][3];
        sdp.t.mean=perturb(sd.t.mean,sd.t.sdev);
        sdp.rho.mean=perturb(sd.rho.mean,sd.rho.sdev);
        sdp.rp.mean=perturb(sd.rp.mean,sd.rp.sdev);
        sdp.K0.mean=bb[i][4];
        vdp.lambda0.mean=perturb(vd.lambda0.mean,
                vd.lambda0.sdev);
        vdp.lambda1.mean=perturb(vd.lambda1.mean,
                vd.lambda1.sdev);
        vdp.kappa0.mean=perturb(vd.kappa0.mean,
                vd.kappa0.sdev);
        vdp.kappa1.mean=perturb(vd.kappa1.mean,
                vd.kappa1.sdev);
        vdp.rv.mean=perturb(vd.rv.mean,vd.rv.sdev);
        vdp.Kd.mean=perturb(vd.Kd.mean,vd.Kd.sdev);

        att.replace(sdp,vdp);
        mr=att.getAttenuation();
        return mr;
}
private double perturb(double avg,double std) {
        double pnum;
        pnum=avg+std*gasdev.nextDeviate();
        return pnum;
}
public void setTheta_mUniform(boolean bu) {
        theta_mIsUniform=bu;
}
private Random gen;
private GasDev gasdev;
private Mvn mvn;
private double[][] bb;
private VarCov varcov;
private Attenuator att;
private double mr;
private double ave, std, unv;
private int maxit,i;
private double gset, ret;
```

```
        private boolean iset=false, theta_mIsUniform=true;
        private Medium sd, sdp;
        private Operandum vd, vdp;
        private static final double TINY=1.0e-15;
}
```

## 10 Random.java

```
/* Note: This class was written by researchers for the Java Numerical
   Toolkit. http://math.nist.gov/jnt/
*/
/* Random.java based on Java Numerical Toolkit (JNT) Random.UniformSequence
        class. We do not use Java's own java.util.Random so that we can
        compare results with equivalent C and Fortran coces.
*/
public class Random {

/* ------------------------------------------------------------------------
                              CLASS VARIABLES
   ------------------------------------------------------------------------ */

  int seed = 0;

  private int m[];
  private int i = 4;
  private int j = 16;

  private final int mdig = 32;
  private final int one = 1;
  private final int m1 = (one << mdig-2) + ((one << mdig-2)-one);
  private final int m2 = one << mdig/2;

  /* For mdig = 32 : m1 =          2147483647, m2 =      65536
     For mdig = 64 : m1 = 9223372036854775807, m2 = 4294967296
  */
  private double dm1 = 1.0 / (double) m1;
  private boolean haveRange = false;
  private double left  = 0.0;
  private double right = 1.0;
  private double width = 1.0;

/* ------------------------------------------------------------------------
                              CONSTRUCTORS
   ------------------------------------------------------------------------ */

/**
   Initializes a sequence of uniformly distributed quasi random numbers with a
   seed based on the system clock.
*/
  public Random () {
    initialize( (int) System.currentTimeMillis());
  }

/**
   Initializes a sequence of uniformly distributed quasi random numbers on a
   given half-open interval [left,right) with a seed based on the system
```

```
   clock.
@param <B>left</B> (double)<BR>
       The left endpoint of the half-open interval [left,right).
@param <B>right</B> (double)<BR>
       The right endpoint of the half-open interval [left,right).
*/
  public Random ( double left, double right) {
    initialize( (int) System.currentTimeMillis() );
    this.left = left;
    this.right = right;
    width = right - left;
    haveRange = true;
  }

/**
   Initializes a sequence of uniformly distributed quasi random numbers with a
   given seed.

@param <B>seed</B> (int)<BR>

       The seed of the random number generator.  Two sequences with the same
       seed will be identical.
*/
  public Random (int seed) {
    initialize( seed);
  }
/**
   Initializes a sequence of uniformly distributed quasi random numbers
   with a given seed on a given half-open interval [left,right).
@param <B>seed</B> (int)<BR>
       The seed of the random number generator.  Two sequences with the same
       seed will be identical.
@param <B>left</B> (double)<BR>
       The left endpoint of the half-open interval [left,right).
@param <B>right</B> (double)<BR>
       The right endpoint of the half-open interval [left,right).
*/
  public Random (int seed, double left, double right) {
    initialize( seed);
    this.left = left;
    this.right = right;
    width = right - left;
    haveRange = true;
  }

/* ------------------------------------------------------------------------
                              PUBLIC METHODS
   ------------------------------------------------------------------------ */

/**
```

```
   Returns the next random number in the sequence.
*/
  public final synchronized double nextDouble () {
    int k;
    double nextValue;
    k = m[i] - m[j];
    if (k < 0) k += m1;
    m[j] = k;
    if (i == 0)
                i = 16;
        else i--;
    if (j == 0)
                j = 16 ;
        else j--;
    if (haveRange)
                return  left +  dm1 * (double) k * width;
        else
                return dm1 * (double) k;
  }
/**
   Returns the next N random numbers in the sequence, as
   a vector.
*/
  public final synchronized void nextDoubles (double x[])
  {
        int N = x.length;
        int remainder = N & 3;          // N mod 4
        if (haveRange)
        {
                for (int count=0; count<N; count++)
                {
                int k = m[i] - m[j];
                if (i == 0) i = 16;
                m[j] = k;
                if (j == 0) j = 16;
                                else j--;
                x[count] = left + dm1 * (double) k * width;
                }
        }
        else
        {
                for (int count=0; count<remainder; count++)
                {
                int k = m[i] - m[j];

                if (i == 0) i = 16;
                                else i--;
                if (k < 0) k += m1;
                m[j] = k;
                if (j == 0) j = 16;
```

```
                    else j--;
                x[count] = dm1 * (double) k;
                }
                for (int count=remainder; count<N; count+=4)
                {
                int k = m[i] - m[j];
                if (i == 0) i = 16;
                    else i--;
                if (k < 0) k += m1;
                m[j] = k;
                if (j == 0) j = 16;
                    else j--;
                x[count] = dm1 * (double) k;
                k = m[i] - m[j];
                if (i == 0) i = 16;
                    else i--;
                if (k < 0) k += m1;
                m[j] = k;
                if (j == 0) j = 16;
                    else j--;
                x[count+1] = dm1 * (double) k;
                k = m[i] - m[j];
                if (i == 0) i = 16;
                    else i--;
                if (k < 0) k += m1;
                m[j] = k;
                if (j == 0) j = 16;
                    else j--;
                x[count+2] = dm1 * (double) k;
                k = m[i] - m[j];
                if (i == 0) i = 16;
                    else i--;
                if (k < 0) k += m1;
                m[j] = k;
                if (j == 0) j = 16;
                x[count+3] = dm1 * (double) k;
                }
        }
  }
/*---------------------------------------------------------------------
                        PRIVATE METHODS
  --------------------------------------------------------------------- */
  private void initialize (int seed) {
    int jseed, k0, k1, j0, j1, iloop;
    this.seed = seed;
    m = new int[17];
    jseed = Math.min(Math.abs(seed),m1);
    if (jseed % 2 == 0) --jseed;
    k0 = 9069 % m2;
    k1 = 9069 / m2;
```

```
    j0 = jseed % m2;
    j1 = jseed / m2;
    for (iloop = 0; iloop < 17; ++iloop)
        {
                jseed = j0 * k0;
                j1 = (jseed / m2 + j0 * k1 + j1 * k0) % (m2 / 2);
                j0 = jseed % m2;
                m[iloop] = j0 + m2 * j1;
    }
    i = 4;
    j = 16;
  }
}
```

## 11 GasDev.java

```
/**
Merely a Java translation of the popular C program <tt>gasdev.c</tt>
(Press et al. 1988) for generating normally distributed deviates.
<hr>
<b>Reference</b>:
<p>
Press, WH, Teukolsky, SA, Vetterling, WT, Flannery, BP, 1992, <i>Numerical
        Recipes in C</i>, Cambridge University Press, Cambridge, UK, 994 p.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 18 July 2001
*/

public class GasDev {
        /** Create one.
        */
        public GasDev() {
                gen=new Random(ll,ul);
        }
        /** Obtain the next random deviate ~ N(ll,ul), where by default
                ll=0, ul=1000.
        @return the deviate
        */
        public double nextDeviate() {
                // From Numerical Recipes in C, pp.289-290.
                // This is the Box-Muller method. It returns
                // Gaussian distributed deviates with zero
                // mean and variance of 1.
                if (!iset) {
                        do {
                                v1=2.*gen.nextDouble()/(ul-ll)-1.;
                                v2=2.*gen.nextDouble()/(ul-ll)-1.;
                                rsq=v1*v1+v2*v2;
                        } while (rsq>=1. || rsq==0.);
                        fac=Math.sqrt(-2.*Math.log(rsq)/rsq);
                        gset=v1*fac;
                        iset=true;
                        ret=v2*fac;
                } else {
                        ret=gset;
                        iset=false;
                }
                return ret;
        }
        /** Change the range for the deviates from the default of
```

```
                [0., 10000].
        @param lower Lower limit of the range;
        @param upper Upper limit of the range;
        */
        public void changeRange(double lower, double upper) {
                ll=lower;
                ul=upper;
        }
        private Random gen;
        private double v1, v2, rsq, gset, ret, fac;
        private double ll=0.;
        private double ul=10000.;
        private boolean iset=false;
}
```

## 12 Mvn.java

```
import Jama.*;

/**
This class is for generating multvariate normal deviates.
<hr>
References:
<p>
Kitanidis, P.K. 1997. <i>Introduction to Geostatistics,
Applications in Hydrogeology.</i> Cambridge University Press,
Cambridge, UK. Appendix C.3.
<p>
Knuth, D.E. 1998. <i>The Art of Computer Programming,
Volume 2. Seminumerical Algorithms, Third Edition.</i>
Addision-Wesley, Reading Mass. p. 586.
<p>
The <a href="http://math.nist.gov/javanumerics/jama/">
<i>Jama</i> matrix package</a> for Java.<br>

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
*/

public class Mvn {
        /** Create one.
        @param V Variance-Covariance Matrix of the random variables
        @param means Vector of means of the random variables
        */
        public Mvn(Matrix V, double[] means) {
                m=means;
                S=V;
                gasdev=new GasDev();
                q=100000; // Must use chunks of arrays no bigger than this else out-of-mem.
                aa=new double[q][m.length];
                cholesky=S.chol();
                L=cholesky.getL();
                L_=L.transpose();
        }
        /** Generate the Matrix of Multivariate Normal Deviates where each row
                corresponds to an event for each random variable.
        */
        public double[][] perturb() {
                for (i=0; i<q; i++) {
                        for (j=0; j<m.length; j++) {
                                aa[i][j]=gasdev.nextDeviate();
                        }
                }
```

```
                A=Matrix.constructWithCopy(aa);
                B=A.times(L_);
                bb=B.getArrayCopy();
                for (i=0; i<q; i++) {
                        for (j=0; j<m.length; j++) {
                                bb[i][j]+=m[j];
                        }
                }
                return bb;
        }
        private double[] m;
        private double[][] aa, bb;
        private Matrix S, B, A, L, L_;
        private GasDev gasdev;
        private CholeskyDecomposition cholesky;
        private int i, j, q;
}
```

## 13 VarCov.java

```
import java.util.*;
import java.awt.*;
import javax.swing.JOptionPane;
import Jama.*;

public class VarCov {
        /* Construct one.
        */
        public VarCov() {
                double[][] clay=new double[5][5];
                double[][] clayloam=new double[5][5];
                double[][] loam=new double[5][5];
                double[][] loamysand=new double[5][5];
                double[][] sand=new double[5][5];
                double[][] sandyclayloam=new double[5][5];
                double[][] sandyloam=new double[5][5];
                double[][] silt=new double[5][5];
                double[][] siltloam=new double[5][5];
                double[][] siltyclay=new double[5][5];
                double[][] siltyclayloam=new double[5][5];
                double[][] everything=new double[5][5];
                clay[0][0]=0.000114665;
                clay[0][1]=0.000901357;
                clay[0][2]=0.001102455;
                clay[0][3]=-5.76439E-05;
                clay[0][4]=0.004693541;
                clay[1][0]=0.000901357;
                clay[1][1]=0.007269272;
                clay[1][2]=0.008709913;
                clay[1][3]=-0.000384136;
                clay[1][4]=0.038629384;
                clay[2][0]=0.001102455;
                clay[2][1]=0.008709913;
                clay[2][2]=0.016763594;
                clay[2][3]=-0.001521695;
                clay[2][4]=0.047965491;
                clay[3][0]=-5.76439E-05;
                clay[3][1]=-0.000384136;
                clay[3][2]=-0.001521695;
                clay[3][3]=0.000231081;
                clay[3][4]=-0.00179356;
                clay[4][0]=0.004693541;
                clay[4][1]=0.038629384;
                clay[4][2]=0.047965491;
                clay[4][3]=-0.00179356;
                clay[4][4]=0.22575866;
                clayloam[0][0]=0.000113318;
                clayloam[0][1]=0.000885529;
```

```
clayloam[0][2]=-0.000603475;
clayloam[0][3]=0.000202533;
clayloam[0][4]=0.006349192;
clayloam[1][0]=0.000885529;
clayloam[1][1]=0.007803567;
clayloam[1][2]=-0.004461905;
clayloam[1][3]=0.001616667;
clayloam[1][4]=0.056786252;
clayloam[2][0]=-0.000603475;
clayloam[2][1]=-0.004461905;
clayloam[2][2]=0.011621638;
clayloam[2][3]=-0.002717;
clayloam[2][4]=-0.031312243;
clayloam[3][0]=0.000202533;
clayloam[3][1]=0.001616667;
clayloam[3][2]=-0.002717;
clayloam[3][3]=0.000777043;
clayloam[3][4]=0.012314215;
clayloam[4][0]=0.006349192;
clayloam[4][1]=0.056786252;
clayloam[4][2]=-0.031312243;
clayloam[4][3]=0.012314215;
clayloam[4][4]=0.423609991;
loam[0][0]=0.000194643;
loam[0][1]=0.000890578;
loam[0][2]=-0.001387056;
loam[0][3]=0.000197588;
loam[0][4]=0.004000768;
loam[1][0]=0.000890578;
loam[1][1]=0.004973962;
loam[1][2]=-0.007217154;
loam[1][3]=0.001172822;
loam[1][4]=0.028475434;
loam[2][0]=-0.001387056;
loam[2][1]=-0.007217154;
loam[2][2]=0.026840993;
loam[2][3]=-0.004216122;
loam[2][4]=-0.043381828;
loam[3][0]=0.000197588;
loam[3][1]=0.001172822;
loam[3][2]=-0.004216122;
loam[3][3]=0.000732296;
loam[3][4]=0.00811325;
loam[4][0]=0.004000768;
loam[4][1]=0.028475434;
loam[4][2]=-0.043381828;
loam[4][3]=0.00811325;
loam[4][4]=0.201946096;
loamysand[0][0]=4.64758E-05;
loamysand[0][1]=0.000135559;
```

```
loamysand[0][2]=-0.000390292;
loamysand[0][3]=9.67775E-05;
loamysand[0][4]=0.000597495;
loamysand[1][0]=0.000135559;
loamysand[1][1]=0.001831864;
loamysand[1][2]=-0.000918065;
loamysand[1][3]=-0.000110459;
loamysand[1][4]=0.007524966;
loamysand[2][0]=-0.000390292;
loamysand[2][1]=-0.000918065;
loamysand[2][2]=0.003626052;
loamysand[2][3]=-0.000962766;
loamysand[2][4]=-0.004508382;
loamysand[3][0]=9.67775E-05;
loamysand[3][1]=-0.000110459;
loamysand[3][2]=-0.000962766;
loamysand[3][3]=0.001947419;
loamysand[3][4]=0.005086197;
loamysand[4][0]=0.000597495;
loamysand[4][1]=0.007524966;
loamysand[4][2]=-0.004508382;
loamysand[4][3]=0.005086197;
loamysand[4][4]=0.049878335;
sand[0][0]=9.40003E-06;
sand[0][1]=3.02921E-05;
sand[0][2]=-8.823E-05;
sand[0][3]=0.000122302;
sand[0][4]=0.000417934;
sand[1][0]=3.02921E-05;
sand[1][1]=0.00103147;
sand[1][2]=0.00020647;
sand[1][3]=-0.000384913;
sand[1][4]=0.001905256;
sand[2][0]=-8.823E-05;
sand[2][1]=0.00020647;
sand[2][2]=0.001134166;
sand[2][3]=-0.001846692;
sand[2][4]=-0.004462323;
sand[3][0]=0.000122302;
sand[3][1]=-0.000384913;
sand[3][2]=-0.001846692;
sand[3][3]=0.00592846;
sand[3][4]=0.015056131;
sand[4][0]=0.000417934;
sand[4][1]=0.001905256;
sand[4][2]=-0.004462323;
sand[4][3]=0.015056131;
sand[4][4]=0.047309545;
sandyclayloam[0][0]=6.08414E-05;
sandyclayloam[0][1]=0.000285711;
```

```
sandyclayloam[0][2]=-0.000522571;
sandyclayloam[0][3]=0.000119065;
sandyclayloam[0][4]=0.001610565;
sandyclayloam[1][0]=0.000285711;
sandyclayloam[1][1]=0.001917612;
sandyclayloam[1][2]=-0.00339727;
sandyclayloam[1][3]=0.000985868;
sandyclayloam[1][4]=0.012877837;
sandyclayloam[2][0]=-0.000522571;
sandyclayloam[2][1]=-0.00339727;
sandyclayloam[2][2]=0.008996074;
sandyclayloam[2][3]=-0.002083578;
sandyclayloam[2][4]=-0.017899802;
sandyclayloam[3][0]=0.000119065;
sandyclayloam[3][1]=0.000985868;
sandyclayloam[3][2]=-0.002083578;
sandyclayloam[3][3]=0.000667732;
sandyclayloam[3][4]=0.007143408;
sandyclayloam[4][0]=0.001610565;
sandyclayloam[4][1]=0.012877837;
sandyclayloam[4][2]=-0.017899802;
sandyclayloam[4][3]=0.007143408;
sandyclayloam[4][4]=0.108472576;
sandyloam[0][0]=6.31743E-05;
sandyloam[0][1]=0.000203048;
sandyloam[0][2]=-0.000596761;
sandyloam[0][3]=5.27283E-05;
sandyloam[0][4]=0.000400306;
sandyloam[1][0]=0.000203048;
sandyloam[1][1]=0.001728818;
sandyloam[1][2]=-0.003305176;
sandyloam[1][3]=0.000735009;
sandyloam[1][4]=0.010876746;
sandyloam[2][0]=-0.000596761;
sandyloam[2][1]=-0.003305176;
sandyloam[2][2]=0.015921726;
sandyloam[2][3]=-0.00034528;
sandyloam[2][4]=-0.006570103;
sandyloam[3][0]=5.27283E-05;
sandyloam[3][1]=0.000735009;
sandyloam[3][2]=-0.00034528;
sandyloam[3][3]=0.000883369;
sandyloam[3][4]=0.008737094;
sandyloam[4][0]=0.000400306;
sandyloam[4][1]=0.010876746;
sandyloam[4][2]=-0.006570103;
sandyloam[4][3]=0.008737094;
sandyloam[4][4]=0.114850179;
silt[0][0]=5.52609E-05;
silt[0][1]=0.000196064;
```

```
silt[0][2]=5.47382E-05;
silt[0][3]=-1.69118E-05;
silt[0][4]=-0.000762317;
silt[1][0]=0.000196064;
silt[1][1]=0.000695672;
silt[1][2]=0.000198589;
silt[1][3]=-6.06626E-05;
silt[1][4]=-0.002715093;
silt[2][0]=5.47382E-05;
silt[2][1]=0.000198589;
silt[2][2]=0.000522595;
silt[2][3]=-8.73345E-05;
silt[2][4]=-0.001868726;
silt[3][0]=-1.69118E-05;
silt[3][1]=-6.06626E-05;
silt[3][2]=-8.73345E-05;
silt[3][3]=1.58122E-05;
silt[3][4]=0.000401115;
silt[4][0]=-0.000762317;
silt[4][1]=-0.002715093;
silt[4][2]=-0.001868726;
silt[4][3]=0.000401115;
silt[4][4]=0.013163826;
siltloam[0][0]=0.000159577;
siltloam[0][1]=0.00049185;
siltloam[0][2]=-0.00014516;
siltloam[0][3]=1.25546E-07;
siltloam[0][4]=-0.000498166;
siltloam[1][0]=0.00049185;
siltloam[1][1]=0.002510005;
siltloam[1][2]=-0.001462379;
siltloam[1][3]=0.000302755;
siltloam[1][4]=0.010171358;
siltloam[2][0]=-0.00014516;
siltloam[2][1]=-0.001462379;
siltloam[2][2]=0.00560484;
siltloam[2][3]=-0.001141615;
siltloam[2][4]=-0.015060119;
siltloam[3][0]=1.25546E-07;
siltloam[3][1]=0.000302755;
siltloam[3][2]=-0.001141615;
siltloam[3][3]=0.000258054;
siltloam[3][4]=0.004248505;
siltloam[4][0]=-0.000498166;
siltloam[4][1]=0.010171358;
siltloam[4][2]=-0.015060119;
siltloam[4][3]=0.004248505;
siltloam[4][4]=0.147437463;
siltyclay[0][0]=4.15543E-05;
siltyclay[0][1]=0.00031489;
```

```
siltyclay[0][2]=0.000216435;
siltyclay[0][3]=1.44621E-05;
siltyclay[0][4]=0.002506275;
siltyclay[1][0]=0.00031489;
siltyclay[1][1]=0.002643139;
siltyclay[1][2]=0.001429338;
siltyclay[1][3]=0.000226114;
siltyclay[1][4]=0.022642563;
siltyclay[2][0]=0.000216435;
siltyclay[2][1]=0.001429338;
siltyclay[2][2]=0.003040552;
siltyclay[2][3]=-0.000463895;
siltyclay[2][4]=0.009611306;
siltyclay[3][0]=1.44621E-05;
siltyclay[3][1]=0.000226114;
siltyclay[3][2]=-0.000463895;
siltyclay[3][3]=0.00017523;
siltyclay[3][4]=0.002736838;
siltyclay[4][0]=0.002506275;
siltyclay[4][1]=0.022642563;
siltyclay[4][2]=0.009611306;
siltyclay[4][3]=0.002736838;
siltyclay[4][4]=0.205535904;
siltyclayloam[0][0]=0.00014543;
siltyclayloam[0][1]=0.001202472;
siltyclayloam[0][2]=0.001092085;
siltyclayloam[0][3]=-9.41236E-05;
siltyclayloam[0][4]=0.007325819;
siltyclayloam[1][0]=0.001202472;
siltyclayloam[1][1]=0.010491368;
siltyclayloam[1][2]=0.009480995;
siltyclayloam[1][3]=-0.000818748;
siltyclayloam[1][4]=0.064185163;
siltyclayloam[2][0]=0.001092085;
siltyclayloam[2][1]=0.009480995;
siltyclayloam[2][2]=0.011802254;
siltyclayloam[2][3]=-0.001471059;
siltyclayloam[2][4]=0.055203009;
siltyclayloam[3][0]=-9.41236E-05;
siltyclayloam[3][1]=-0.000818748;
siltyclayloam[3][2]=-0.001471059;
siltyclayloam[3][3]=0.000238661;
siltyclayloam[3][4]=-0.004225565;
siltyclayloam[4][0]=0.007325819;
siltyclayloam[4][1]=0.064185163;
siltyclayloam[4][2]=0.055203009;
siltyclayloam[4][3]=-0.004225565;
siltyclayloam[4][4]=0.402058219;
everything[0][0]=0.0003416344;
everything[0][1]=0.0010310062;
```

```
        everything[0][2]=-0.002623562;
        everything[0][3]=-0.000987818;
        everything[0][4]=-0.004302541;
        everything[1][0]=0.0010310062;
        everything[1][1]=0.0046931158;
        everything[1][2]=-0.007182752;
        everything[1][3]=-0.002927961;
        everything[1][4]=0.0001602851;
        everything[2][0]=-0.002623562;
        everything[2][1]=-0.007182752;
        everything[2][2]=0.0946687612;
        everything[2][3]=0.0177579194;
        everything[2][4]=0.1202738411;
        everything[3][0]=-0.000987818;
        everything[3][1]=-0.002927961;
        everything[3][2]=0.0177579194;
        everything[3][3]=0.0203543467;
        everything[3][4]=0.0873275195;
        everything[4][0]=-0.004302541;
        everything[4][1]=0.0001602851;
        everything[4][2]=0.1202738411;
        everything[4][3]=0.0873275195;
        everything[4][4]=0.5202631192;
        Clay=new Matrix(clay);
        Clayloam=new Matrix(clayloam);
        Loam=new Matrix(loam);
        Loamysand=new Matrix(loamysand);
        Sand=new Matrix(sand);
        Sandyclayloam=new Matrix(sandyclayloam);
        Sandyloam=new Matrix(sandyloam);
        Silt=new Matrix(silt);
        Siltloam=new Matrix(siltloam);
        Siltyclay=new Matrix(siltyclay);
        Siltyclayloam=new Matrix(siltyclayloam);
        Everything=new Matrix(everything);
}
public Matrix getVarCov(String soilname) {
        VC=null;
        if (soilname.equals("clay")) VC=Clay;
        else if (soilname.equals("clayloam")) VC=Clayloam;
        else if (soilname.equals("loam")) VC=Loam;
        else if (soilname.equals("loamysand")) VC=Loamysand;
        else if (soilname.equals("sand")) VC=Sand;
        else if (soilname.equals("sandyclayloam")) VC=Sandyclayloam;
        else if (soilname.equals("sandyloam")) VC=Sandyloam;
        else if (soilname.equals("silt")) VC=Silt;
        else if (soilname.equals("siltloam")) VC=Siltloam;
        else if (soilname.equals("siltyclay")) VC=Siltyclay;
        else if (soilname.equals("siltyclayloam")) VC=Siltyclayloam;
        else if (soilname.equals("everything")) VC=Everything;
```

```
                else {
                        String Message="Virulo needs a variance-covariance\n"+
                                        "matrix to use to generate multi-\n"+
                                        "variate normal deviates. You must\n"+
                                        "choose a soil from one of the 12 USDA\n"+
                                        "types. It is important that you under-\n"+
                                        "stand the assumptions you are making\n"+
                                        "if you are using your own mean values\n"+
                                        "for the 5 correlated flow parameters.";
                        Object[] selectionValues={"clay","clayloam","loam","loamysand",
                                "sand","sandyclayloam","sandyloam","silt","siltloam",
                                "siltyclay","siltyclayloam","everything"};
                        String s=(String)JOptionPane.showInputDialog(null,Message,
                                  "Variance-covariance matrix needed",
                                        JOptionPane.QUESTION_MESSAGE,
                                        null,
                                        selectionValues,
                                        selectionValues[11]);
                        VC=getVarCov(s);
                }

                return VC;
        }
        private int i, j;
        private double[][] clay, clayloam, loam, loamysand, sand,
                sandyclayloam, sandyloam, silt, siltloam,
                siltyclay, siltyclayloam, everything;
        private Matrix Clay, Clayloam, Loam, Loamysand, Sand, Sandyclayloam,
                Sandyloam, Silt, Siltloam, Siltyclay, Siltyclayloam, Everything, VC;
}
```

## 14 Attenuator.java

```
/** This is the interface that generalizes attenuation of an <i>Operandum</i>
    (e.g., a solute, colloid, heat) in a <i>Medium</i> (e.g., a lake, stream,
    porous medium, soil).

@author B.R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 18 July 2000
@see Medium
@see Operandum
*/

public interface Attenuator {
        Medium getMedium();
        Operandum getOperandum();

        /** A replace is used to replace the data for purposes of a
            time-series or a Monte Carlo implementation.
        */
        void replace(Medium medium,Operandum operandum);

        /** This method returns the mean advection rate supplied
            by the Medium.
        */
        double getAdvection();

        /** This method returns the attenuation factor which is by
            definition the quantity of Operandum remaining
            divided by the initial input quantity of Operandum.
        */
        double getAttenuation();
}
```

## 15 AboutFrame.java

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class AboutFrame extends JFrame {
        public AboutFrame() {
                super("About Virulo");
                setSize(600,230);
                setLocation(400,300);
                setVisible(true);
                ImagePanel iP=new ImagePanel();
                getContentPane().add(iP);
        }
}
```

## 16 JarLoadable.java

```
import java.net.*;
import java.awt.Image;
import java.awt.Toolkit;
import java.io.InputStream;
import javax.swing.*;
/** Allows images files to be loaded from a Java Archive (jar) file.
Adapted from examples posted in the UC Berkeley Advanced Java Archives<br>
<code>https://lists.xcf.berkeley.edu/lists/advanced-java/</code><br>
lists.
@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 17 October 2002
*/

public class JarLoadable {
        protected Image loadImage(String name){
                Image image=null;
                byte[] bite=null;
                Toolkit toolkit=Toolkit.getDefaultToolkit();
                InputStream in=getClass().getResourceAsStream(name);
                try {
                        int length=in.available();
                        bite=new byte[length];
                        in.read(bite);
                        image=toolkit.createImage(bite);
                } catch(Exception exc) {
                        return null;
                }
                return image;
        }
}
```

## 17 Gossiper.java

```
import java.util.Observable;

/**
<P>
Subclass of <tt>java.util.Observable</tt> that acts as the
"subject" of the <i>Observer</i> design pattern (Gamma et al. 1995)
by using the java class written for this purpose.
<hr>
<b>Reference:</b>
<p>
Gamma, E., Helm, R., Johnson, R., Vlissides, J. 1995. <i>Design Patterns,
Elements of Reusable Object-Oriented Software</i>. Addision-Wesley.

@see Observable
@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 6 February 2001
*/

public class Gossiper extends Observable {
        /** Creat a new <b>Gossiper</b> object.
        */
        public Gossiper() {
        }
        /** Inform <b>Gossiper</b> object of a change in the
            concrete subjects so that it will notify the
            designated observers.
        */
        public void tell(Object o) {
                setChanged();
                notifyObservers(o);
        }
}
```

## 18 FlowPanel.java

```
import java.util.Observable;
import java.util.Observer;
import java.awt.*;
import java.awt.event.*;
import java.text.NumberFormat;
import javax.swing.*;
import javax.swing.event.*;

/**
   <b>FlowPanel</b> class.
<P>
Displays the text fields and labels for the parameters of the Virus
transport model that are specific to the soil data. Instances of this
class are concrete observers that respond to updates of the <b>Medium</b>
data structure.

@author Barton R. Faulkner,
        U.S. EPA Office of Research and Development
        National Risk Management Research Laboratory
        Ada, Oklahoma, USA.
@version 18 October 2002

@see Medium
@see Gossiper
*/

public class FlowPanel extends JPanel implements Observer {

/* ------------------------
   Constructor
 * ------------------------ */
        /** Construct an instance of FlowPanel.
        @param gossiper A <b>Gossiper</b> object that makes this
                <b>FlowPanel</b> aware of changes.
        @see Gossiper
        */
        public FlowPanel(Gossiper gossiper) {
                JarLoadable jar=new JarLoadable();
                gossiper.addObserver(this);
                JLabel h1L=new JLabel("Parameter");
                JLabel h2L=new JLabel("Mean"); //mean
                JLabel h3L=new JLabel("Std. Deviation"); //standard deviation
                aL=new JLabel(new ImageIcon(
                        jar.loadImage("a.gif")));
                aL.setToolTipText("Log van Genuchten's alpha");
                theta_rL=new JLabel(new ImageIcon(
                        jar.loadImage("thetar.gif")));
                theta_rL.setToolTipText("Residual water content");
```

```
theta_mL=new JLabel(new ImageIcon(
        jar.loadImage("thetam.gif")));
theta_mL.setToolTipText("Water content");
theta_sL=new JLabel(new ImageIcon(
        jar.loadImage("thetas.gif")));
theta_sL.setToolTipText("Saturated water content");
zL=new JLabel(new ImageIcon(
        jar.loadImage("l.gif")));
zL.setToolTipText("Thickness of proposed barrier");
alphazL=new JLabel(new ImageIcon(
        jar.loadImage("alphaz.gif")));
alphazL.setToolTipText("Hydrodynamic dispersivity");
nL=new JLabel(new ImageIcon(
        jar.loadImage("n.gif")));
nL.setToolTipText("Log van Genuchten's n");
tL=new JLabel(new ImageIcon(
        jar.loadImage("t.gif")));
tL.setToolTipText(
        "Temperature of soil (for computing molecular diffusivity)");
rhoL=new JLabel(new ImageIcon(
        jar.loadImage("rho.gif")));
rhoL.setToolTipText("Soil bulk density");
rpL=new JLabel(new ImageIcon(
        jar.loadImage("rp.gif")));
rpL.setToolTipText("Mean soil particle radius");
K0L=new JLabel(new ImageIcon(
        jar.loadImage("k0.gif")));
K0L.setToolTipText("Log Saturated hydraulic conductivity");
String noNo="Modifying this mean value is not recommended."+
        "See model documentation.";
aF=new NormalF();
aF.m.setToolTipText(noNo);
aF.m.setForeground(Color.red);
aF.s.setEditable(false);

theta_sF=new NormalF();
theta_sF.m.setToolTipText(noNo);
theta_sF.m.setForeground(Color.red);
theta_sF.s.setEditable(false);

theta_rF=new NormalF();
theta_rF.m.setToolTipText(noNo);
theta_rF.m.setForeground(Color.red);
theta_rF.s.setEditable(false);

theta_mF=new NormalF();
theta_mF.m.setEditable(false);
theta_mF.s.setEditable(false);
zF=new NormalF();
```

```
alphazF=new NormalF();

nF=new NormalF();
nF.m.setToolTipText(noNo);
nF.m.setForeground(Color.red);
nF.s.setEditable(false);
tF=new NormalF();
rhoF=new NormalF();
rpF=new NormalF();

K0F=new NormalF();
K0F.m.setToolTipText(noNo);
K0F.m.setForeground(Color.red);
K0F.s.setEditable(false);

units=new String[13];
units[0]=units[1]=units[2]="  m\u00b3 m-\u00b3";
units[3]=" log10( m h -\u00b9 )";
units[4]=" log10( m -\u00b9 )";
units[5]=" log10( . )";
units[6]=" g m -\u00b3 ";
units[7]=units[8]=" m";
units[9]=" Celsius";
units[10]=units[11]=" m";

GridLayout gl=new GridLayout(14,4);
FillPanel=new JPanel();
editCheckBox=new JCheckBox("Uniformly Random",true);
editCheckBox.setForeground(Color.blue);
editCheckBox.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                theta_mF.m.setEditable(!editCheckBox.isSelected());
                theta_mF.s.setEditable(!editCheckBox.isSelected());
                if (editCheckBox.isSelected()) {
                        editCheckBox.setForeground(Color.blue);
                } else {
                        editCheckBox.setForeground(Color.gray);
                }
        }
});
setLayout(gl);
Fcp=new FlowComboPanel(gossiper);

 add(h1L);
  add(h2L);
   add(h3L);
    add(new JLabel("Units"));
 add(theta_rL);
  add(theta_rF.m);
   add(theta_rF.s);
```

```
            add(new JLabel(units[0]));
        add(theta_mL);
         add(theta_mF.m);
          add(theta_mF.s);
           add(editCheckBox);
        add(theta_sL);
         add(theta_sF.m);
          add(theta_sF.s);
           add(new JLabel(units[2]));
        add(K0L);
         add(K0F.m);
          add(K0F.s);
           add(new JLabel(units[3]));
        add(aL);
         add(aF.m);
          add(aF.s);
           add(new JLabel(units[4]));
        add(nL);
         add(nF.m);
          add(nF.s);
           add(new JLabel(units[5]));
        add(rhoL);
         add(rhoF.m);
          add(rhoF.s);
           add(new JLabel(units[6]));
        add(rpL);
         add(rpF.m);
          add(rpF.s);
           add(new JLabel(units[7]));
        add(alphazL);
         add(alphazF.m);
          add(alphazF.s);
           add(new JLabel(units[8]));
        add(tL);
         add(tF.m);
          add(tF.s);
           add(new JLabel(units[9]));
        add(zL);
         add(zF.m);
          add(zF.s);
           add(new JLabel(units[10]));
        add(FillPanel);
         add(Fcp);
          add(FillPanel);
           add(FillPanel);
        add(FillPanel);
         add(FillPanel);
          add(FillPanel);
           add(FillPanel);
}
```

```
/** This method is called by the <b>Gossiper</b> (or other subclass
    of <b>Observable</b>) and updates the <b>Medium</b> instance.
@see Gossiper
*/
public void update(Observable oble, Object o) {
        String name=o.getClass().getName();
        if (name.equals("Medium")) {
                Medium d=(Medium)(o);
                NumberFormat nf=NumberFormat.getNumberInstance();

                nf.setMaximumFractionDigits(2);

                aF.m.setText(nf.format(d.a.mean));
                aF.s.setText(nf.format(d.a.sdev));

                theta_rF.m.setText(nf.format(d.theta_r.mean));
                theta_rF.s.setText(nf.format(d.theta_r.sdev));

                theta_mF.m.setText(nf.format(d.theta_m.mean));
                theta_mF.s.setText(nf.format(d.theta_m.sdev));

                theta_sF.m.setText(nf.format(d.theta_s.mean));
                theta_sF.s.setText(nf.format(d.theta_s.sdev));

                zF.m.setText(nf.format(d.z.mean));
                zF.s.setText(nf.format(d.z.sdev));

                alphazF.m.setText(Double.toString(d.alphaz.mean));
                alphazF.s.setText(Double.toString(d.alphaz.sdev));

                nF.m.setText(nf.format(d.n.mean));
                nF.s.setText(nf.format(d.n.sdev));

                tF.m.setText(nf.format(d.t.mean));
                tF.s.setText(nf.format(d.t.sdev));

                rpF.m.setText(Double.toString(d.rp.mean));
                rpF.s.setText(Double.toString(d.rp.sdev));

                K0F.m.setText(nf.format(d.K0.mean));
                K0F.s.setText(nf.format(d.K0.sdev));

                rhoF.m.setText(Double.toString(d.rho.mean));
                rhoF.s.setText(Double.toString(d.rho.sdev));

                repaint();
        }
}
/** Obtain the current data.
@return Medium
```

```
*/
public Medium getData() {
        Medium sd=new Medium();
        //get the name from the FlowComboPanel:
        String Name=Fcp.getSelection();
        Name=Name.toLowerCase();
        sd.name.is=Name.toCharArray();

        Double D;
        D=new Double(aF.m.getText());
        sd.a.mean=D.doubleValue();
        D=new Double(aF.s.getText());
        sd.a.sdev=D.doubleValue();

        D=new Double(theta_rF.m.getText());
        sd.theta_r.mean=D.doubleValue();
        D=new Double(theta_rF.s.getText());
        sd.theta_r.sdev=D.doubleValue();

        D=new Double(theta_mF.m.getText());
        sd.theta_m.mean=D.doubleValue();
        D=new Double(theta_mF.s.getText());
        sd.theta_m.sdev=D.doubleValue();

        D=new Double(theta_sF.m.getText());
        sd.theta_s.mean=D.doubleValue();
        D=new Double(theta_sF.s.getText());
        sd.theta_s.sdev=D.doubleValue();

        D=new Double(zF.m.getText());
        sd.z.mean=D.doubleValue();
        D=new Double(zF.s.getText());
        sd.z.sdev=D.doubleValue();

        D=new Double(alphazF.m.getText());
        sd.alphaz.mean=D.doubleValue();
        D=new Double(alphazF.s.getText());
        sd.alphaz.sdev=D.doubleValue();

        D=new Double(nF.m.getText());
        sd.n.mean=D.doubleValue();
        D=new Double(nF.s.getText());
        sd.n.sdev=D.doubleValue();

        D=new Double(tF.m.getText());
        sd.t.mean=D.doubleValue();
        D=new Double(tF.s.getText());
        sd.t.sdev=D.doubleValue();

        D=new Double(rhoF.m.getText());
```

```
        sd.rho.mean=D.doubleValue();
        D=new Double(rhoF.s.getText());
        sd.rho.sdev=D.doubleValue();

        D=new Double(rpF.m.getText());
        sd.rp.mean=D.doubleValue();
        D=new Double(rpF.s.getText());
        sd.rp.sdev=D.doubleValue();

        D=new Double(K0F.m.getText());
        sd.K0.mean=D.doubleValue();
        D=new Double(K0F.s.getText());
        sd.K0.sdev=D.doubleValue();

        return sd;
}
public void putData(Medium newdata) {
        Fcp.addSoil(newdata, new String(newdata.name.is));
}
/** Obtain the labels for the parameter fields as a
<b>String</b> array.
*/
public String[] getLabels() {
        String[] labels={
                theta_rL.getToolTipText(),
                theta_mL.getToolTipText(),
                theta_sL.getToolTipText(),
                K0L.getToolTipText(),
                aL.getToolTipText(),
                nL.getToolTipText(),
                rhoL.getToolTipText(),
                rpL.getToolTipText(),
                alphazL.getToolTipText(),
                tL.getToolTipText(),
                zL.getToolTipText()
        };
        return labels;
}
public boolean theta_mIsUniform() {
        return editCheckBox.isSelected();
}
public String[] getUnits() {
        return units;
}
public void requestAttention(String sF) {
        if (sF.equalsIgnoreCase("theta_r.m")) theta_rF.m.requestFocus();
        if (sF.equalsIgnoreCase("theta_m.m")) theta_mF.m.requestFocus();
        if (sF.equalsIgnoreCase("theta_m.s")) theta_mF.s.requestFocus();
        if (sF.equalsIgnoreCase("theta_s.m")) theta_sF.m.requestFocus();
        if (sF.equalsIgnoreCase("rho.m")) rhoF.m.requestFocus();
```

```
                if (sF.equalsIgnoreCase("rho.s")) rhoF.s.requestFocus();
                if (sF.equalsIgnoreCase("rp.m")) rpF.m.requestFocus();
                if (sF.equalsIgnoreCase("rp.s")) rpF.s.requestFocus();
                if (sF.equalsIgnoreCase("alphaz.m")) alphazF.m.requestFocus();
                if (sF.equalsIgnoreCase("alphaz.s")) alphazF.s.requestFocus();
                if (sF.equalsIgnoreCase("z.m")) zF.m.requestFocus();
                if (sF.equalsIgnoreCase("z.s")) zF.s.requestFocus();
        }
        private JLabel blockP;
        private String[] units;
        private JPanel FillPanel;
        private FlowComboPanel Fcp;
        private JCheckBox editCheckBox;
        private JLabel h1L;
        private JLabel h2L;
        private JLabel aL;
        private JLabel theta_rL;
        private JLabel theta_mL;
        private JLabel theta_sL;
        private JLabel zL;
        private JLabel alphazL;
        private JLabel nL;
        private JLabel tL;
        private JLabel rhoL;
        private JLabel rpL;
        private JLabel K0L;
//      private JLabel h0L;
//      private JLabel bL;
        private NormalF aF;
        private NormalF theta_rF;
        private NormalF theta_mF;
        private NormalF theta_sF;
        private NormalF zF;
        private NormalF alphazF;
        private NormalF nF;
        private NormalF tF;
        private NormalF rhoF;
        private NormalF rpF;
        private NormalF K0F;
//      private NormalF h0F;
//      private NormalF bF;
}
```

# 19 FlowComboPanel.java

```
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
   <b>FlowComboPanel</b> class.
<P>
This is a combo panel that the user uses to select a soil type.
It has a listener that retrieves the appropriate data indexed by
the <b>String</b> which is the title of the soil type.

@author Barton R. Faulkner,
        U.S. EPA Office of Research and Development
        National Risk Management Research Laboratory
        Ada, Oklahoma, USA.
@version 11 November 2002
*/

public class FlowComboPanel extends JPanel {

/* ------------------------
   Constructor
 * ------------------------ */

        /** Construct a <b>FlowComboPanel</b> instance.
        @param gossiper A <b>Gossiper</b> for this object.
        @see Gossiper
        @see Medium
        @see SoilStack
        */

        public FlowComboPanel(Gossiper gossiper) {
                gpr=gossiper;
                soilStack=new SoilStack();
                newdata=new Medium();
                comboBox=new JComboBox();
                comboBox.addItem("clay");
                comboBox.addItem("clayloam");
                comboBox.addItem("loam");
                comboBox.addItem("loamysand");
                comboBox.addItem("sand");
                comboBox.addItem("sandyclayloam");
                comboBox.addItem("sandyloam");
                comboBox.addItem("silt");
                comboBox.addItem("siltloam");
                comboBox.addItem("siltyclay");
                comboBox.addItem("siltyclayloam");
```

```
                comboBox.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                newdata=soilStack.getData(
                                        (String)(comboBox.getSelectedItem())
                                );
                                tell(newdata);
                        }
                });

                GridLayout gl=new GridLayout(1,2);
                setLayout(gl);
                add(comboBox);
                newdata=soilStack.getData(
                        (String)(comboBox.getSelectedItem())
                );
                tell(newdata);
        }
        public String getSelection() {
                return (String)(comboBox.getSelectedItem());
        }
        public void addSoil(Medium soil, String name) {
                comboBox.addItem(name);
                soilStack.addSoil(soil);
        }
        private void tell(Medium nd) {
                gpr.tell(nd);
        }
        private Gossiper gpr;
        private SoilStack soilStack;
        private Medium newdata;
        private JComboBox comboBox;
        private int i;
}
```

## 20 SoilStack.java

```
import java.util.*;

/**
   <b>SoilStack</b> class.
<P>
Stack for Medium.

@author Barton R Faulkner, US EPA National Risk Management Laboratory,
        Ada, Oklahoma, USA.
@version 11 November 2002
@see Medium
*/

public class SoilStack {
        /** Retrieve data by soil type descriptor.
        @param soil Soil type descriptor (e.g., "sand").
        @return The Medium object requested.
        */
        public Medium getData(String soil){
                Medium sd=new Medium();
                if (soil=="clay") {
                        String Name=new String("clay");
                        sd.name.is=Name.toCharArray();

                        sd.a.mean=0.276202682;
                        sd.a.sdev=0.129474299;

                        sd.theta_r.mean=0.100993333;
                        sd.theta_r.sdev=0.010708176;

                        sd.theta_m.mean=0;
                        sd.theta_m.sdev=0;

                        sd.theta_s.mean=0.515332222;
                        sd.theta_s.sdev=0.085260028;

                        sd.z.mean=0.5;
                        sd.z.sdev=0.1;

                        sd.alphaz.mean=8.75e-5;
                        sd.alphaz.sdev=0.0001;

                        sd.n.mean=0.113751456;
                        sd.n.sdev=0.015201357;

                        sd.t.mean=11.7;
                        sd.t.sdev=7.38;
```

```
                        sd.rho.mean=1.29E+06;
                        sd.rho.sdev=1.68E+05;

                        sd.rp.mean=9.95E-05;
                        sd.rp.sdev=6.15E-05;

                        sd.K0.mean=-2.085670553;
                        sd.K0.sdev=0.475140674;

//                      sd.h0.mean=0.3730;
//                      sd.h0.sdev=0.89855;

//                      sd.b.mean=0.165;
//                      sd.b.sdev=0.128;
                } else if (soil=="clayloam") {
                        String Name=new String("clayloam");
                        sd.name.is=Name.toCharArray();

                        sd.a.mean=0.131688015;
                        sd.a.sdev=0.107803701;

                        sd.theta_r.mean=0.083457308;
                        sd.theta_r.sdev=0.010645087;

                        sd.theta_m.mean=0;
                        sd.theta_m.sdev=0;

                        sd.theta_s.mean=0.455308077;
                        sd.theta_s.sdev=0.088337801;

                        sd.z.mean=0.5;
                        sd.z.sdev=0.1;

                        sd.alphaz.mean=8.75e-5;
                        sd.alphaz.sdev=0.0001;

                        sd.n.mean=0.140818958;
                        sd.n.sdev=0.027875485;

                        sd.t.mean=11.7;
                        sd.t.sdev=7.38;

                        sd.rho.mean=1.41E+06;
                        sd.rho.sdev=2.39E+05;

                        sd.rp.mean=1.68E-04;
                        sd.rp.sdev=3.72E-05;

                        sd.K0.mean=-2.273130814;
                        sd.K0.sdev=0.650853279;
```

```
//                      sd.h0.mean=0.2589;
//                      sd.h0.sdev=0.5495;

//                      sd.b.mean=0.242;
//                      sd.b.sdev=0.172;
                } else if (soil=="loam") {
                        String Name=new String("loam");
                        sd.name.is=Name.toCharArray();

                        sd.a.mean=0.041061472;
                        sd.a.sdev=0.163832209;

                        sd.theta_r.mean=0.061114828;
                        sd.theta_r.sdev=0.013951437;

                        sd.theta_m.mean=0;
                        sd.theta_m.sdev=0;

                        sd.theta_s.mean=0.421203103;
                        sd.theta_s.sdev=0.070526324;

                        sd.z.mean=0.5;
                        sd.z.sdev=0.1;

                        sd.alphaz.mean=8.75e-5;
                        sd.alphaz.sdev=0.0001;

                        sd.n.mean=0.171560556;
                        sd.n.sdev=0.027060968;

                        sd.t.mean=11.7;
                        sd.t.sdev=7.38;

                        sd.rho.mean=1.34E+06;
                        sd.rho.sdev=2.75E+05;

                        sd.rp.mean=2.15E-04;
                        sd.rp.sdev=3.38E-05;

                        sd.K0.mean=-1.988473636;
                        sd.K0.sdev=0.44938413;

//                      sd.h0.mean=0.1115;
//                      sd.h0.sdev=0.37385;

//                      sd.b.mean=0.252;
//                      sd.b.sdev=0.166;
                } else if (soil=="loamysand") {
                        String Name=new String("loamysand");
```

```
                        sd.name.is=Name.toCharArray();

                        sd.a.mean=0.574047568;
                        sd.a.sdev=0.060216706;

                        sd.theta_r.mean=0.047099;
                        sd.theta_r.sdev=0.006817318;

                        sd.theta_m.mean=0;
                        sd.theta_m.sdev=0;

                        sd.theta_s.mean=0.3885762;
                        sd.theta_s.sdev=0.042800282;

                        sd.z.mean=0.5;
                        sd.z.sdev=0.1;

                        sd.alphaz.mean=5.59e-3;
                        sd.alphaz.sdev=0.0001;

                        sd.n.mean=0.280855311;
                        sd.n.sdev=0.04412957;

                        sd.t.mean=11.7;
                        sd.t.sdev=7.38;

                        sd.rho.mean=1.50E+06;
                        sd.rho.sdev=1.67E+05;

                        sd.rp.mean=4.20E-04;
                        sd.rp.sdev=2.20E-05;

                        sd.K0.mean=-1.198044153;
                        sd.K0.sdev=0.223334581;

//                      sd.h0.mean=0.0869;
//                      sd.h0.sdev=0.20025;

//                      sd.b.mean=0.553;
//                      sd.b.sdev=0.319;
                } else if (soil=="sand") {
                        String Name=new String("sand");
                        sd.name.is=Name.toCharArray();

                        sd.a.mean=0.53060969;
                        sd.a.sdev=0.033677377;

                        sd.theta_r.mean=0.049638516;
                        sd.theta_r.sdev=0.003065946;
```

```
        sd.theta_m.mean=0;
        sd.theta_m.sdev=0;

        sd.theta_s.mean=0.366745391;
        sd.theta_s.sdev=0.032116508;

        sd.z.mean=0.5;
        sd.z.sdev=0.1;

        sd.alphaz.mean=5.59e-3;
        sd.alphaz.sdev=0.0001;

        sd.n.mean=0.482299856;
        sd.n.sdev=0.076996491;

        sd.t.mean=11.7;
        sd.t.sdev=7.38;

        sd.rho.mean=1.58E+06;
        sd.rho.sdev=1.42E+05;

        sd.rp.mean=4.71E-04;
        sd.rp.sdev=1.60E-05;

        sd.K0.mean=-0.691792728;
        sd.K0.sdev=0.033615473;

        //sd.h0.mean=0.0726;
        //sd.h0.sdev=0.1872;

        //sd.b.mean=0.694;
        //sd.b.sdev=0.296;
} else if (soil=="sandyclayloam") {
        String Name=new String("sandyclayloam");
        sd.name.is=Name.toCharArray();

        sd.a.mean=0.344005176;
        sd.a.sdev=0.094847636;

        sd.theta_r.mean=0.064246923;
        sd.theta_r.sdev=0.00780009;

        sd.theta_m.mean=0;
        sd.theta_m.sdev=0;

        sd.theta_s.mean=0.39204;
        sd.theta_s.sdev=0.043790548;

        sd.z.mean=0.5;
        sd.z.sdev=0.1;
```

```
        sd.alphaz.mean=8.75e-5;
        sd.alphaz.sdev=0.0001;

        sd.n.mean=0.120380137;
        sd.n.sdev=0.025840507;

        sd.t.mean=11.7;
        sd.t.sdev=7.38;

        sd.rho.mean=1.50E+06;
        sd.rho.sdev=2.24E+05;

        sd.rp.mean=3.08E-04;
        sd.rp.sdev=4.20E-05;

        sd.K0.mean=-2.271251064;
        sd.K0.sdev=0.329351751;

        //sd.h0.mean=0.2808;
        //sd.h0.sdev=0.67965;

        //sd.b.mean=0.319;
        //sd.b.sdev=0.24;
} else if (soil=="sandyloam") {
        String Name=new String("sandyloam");
        sd.name.is=Name.toCharArray();

        sd.a.mean=0.485840457;
        sd.a.sdev=0.126181322;

        sd.theta_r.mean=0.044244026;
        sd.theta_r.sdev=0.007948225;

        sd.theta_m.mean=0;
        sd.theta_m.sdev=0;

        sd.theta_s.mean=0.369563636;
        sd.theta_s.sdev=0.041579061;

        sd.z.mean=0.5;
        sd.z.sdev=0.1;

        sd.alphaz.mean=8.75e-5;
        sd.alphaz.sdev=0.0001;

        sd.n.mean=0.154301527;
        sd.n.sdev=0.029721521;

        sd.t.mean=11.7;
```

```
        sd.t.sdev=7.38;

        sd.rho.mean=1.53E+06;
        sd.rho.sdev=1.69E+05;

        sd.rp.mean=3.35E-04;
        sd.rp.sdev=4.59E-05;

        sd.K0.mean=-1.867049932;
        sd.K0.sdev=0.338895528;

        //sd.h0.mean=0.1466;
        //sd.h0.sdev=0.29395;

        //sd.b.mean=0.378;
        //sd.b.sdev=0.238;
} else if (soil=="silt") {
        String Name=new String("silt");
        sd.name.is=Name.toCharArray();

        sd.a.mean=-0.244777645;
        sd.a.sdev=0.022860344;

        sd.theta_r.mean=0.059473333;
        sd.theta_r.sdev=0.00743377;

        sd.theta_m.mean=0;
        sd.theta_m.sdev=0;

        sd.theta_s.mean=0.433063333;
        sd.theta_s.sdev=0.026375595;

        sd.z.mean=0.5;
        sd.z.sdev=0.1;

        sd.alphaz.mean=8.75e-5;
        sd.alphaz.sdev=0.0001;

        sd.n.mean=0.221016764;
        sd.n.sdev=0.003976454;

        sd.t.mean=11.7;
        sd.t.sdev=7.38;

        sd.rho.mean=1.39E+06;
        sd.rho.sdev=2.90E+04;

        sd.rp.mean=4.53E-05;
        sd.rp.sdev=2.36E-05;
```

```
        sd.K0.mean=-1.838232525;
        sd.K0.sdev=0.114733717;

        //sd.h0.mean=0;
        //sd.h0.sdev=9999;

        //sd.b.mean=0;
        //sd.b.sdev=9999;
} else if (soil=="siltloam") {
        String Name=new String("siltloam");
        sd.name.is=Name.toCharArray();

        sd.a.mean=-0.207340275;
        sd.a.sdev=0.07486548;

        sd.theta_r.mean=0.063289709;
        sd.theta_r.sdev=0.012632368;

        sd.theta_m.mean=0;
        sd.theta_m.sdev=0;

        sd.theta_s.mean=0.406203592;
        sd.theta_s.sdev=0.050099953;

        sd.z.mean=0.5;
        sd.z.sdev=0.1;

        sd.alphaz.mean=8.75e-5;
        sd.alphaz.sdev=0.0001;

        sd.n.mean=0.206442995;
        sd.n.sdev=0.016064068;

        sd.t.mean=11.7;
        sd.t.sdev=7.38;

        sd.rho.mean=1.43E+06;
        sd.rho.sdev=1.48E+05;

        sd.rp.mean=1.18E-04;
        sd.rp.sdev=5.50E-05;

        sd.K0.mean=-2.160084702;
        sd.K0.sdev=0.383975863;

        //sd.h0.mean=0.2076;
        //sd.h0.sdev=0.5841;

        //sd.b.mean=0.234;
        //sd.b.sdev=0.129;
```

```
} else if (soil=="siltyclay") {
        String Name=new String("siltyclay");
        sd.name.is=Name.toCharArray();

        sd.a.mean=0.090811457;
        sd.a.sdev=0.055141199;

        sd.theta_r.mean=0.094913636;
        sd.theta_r.sdev=0.006446264;

        sd.theta_m.mean=0;
        sd.theta_m.sdev=0;

        sd.theta_s.mean=0.47609;
        sd.theta_s.sdev=0.051411467;

        sd.z.mean=0.5;
        sd.z.sdev=0.1;

        sd.alphaz.mean=8.75e-5;
        sd.alphaz.sdev=0.0001;

        sd.n.mean=0.139975077;
        sd.n.sdev=0.013237462;

        sd.t.mean=11.7;
        sd.t.sdev=7.38;

        sd.rho.mean=1.20E+06;
        sd.rho.sdev=2.57E+05;

        sd.rp.mean=4.07E-05;
        sd.rp.sdev=2.45E-05;

        sd.K0.mean=-2.434048919;
        sd.K0.sdev=0.453360678;

        //sd.h0.mean=0.3419;
        //sd.h0.sdev=0.7958;

        //sd.b.mean=0.150;
        //sd.b.sdev=0.11;
} else if (soil=="siltyclayloam") {
        String Name=new String("siltyclayloam");
        sd.name.is=Name.toCharArray();

        sd.a.mean=0.027733578;
        sd.a.sdev=0.108638178;

        sd.theta_r.mean=0.092101429;
```

```
                        sd.theta_r.sdev=0.012059445;

                        sd.theta_m.mean=0;
                        sd.theta_m.sdev=0;

                        sd.theta_s.mean=0.499877619;
                        sd.theta_s.sdev=0.102427379;

                        sd.z.mean=0.5;
                        sd.z.sdev=0.1;

                        sd.alphaz.mean=8.75e-5;
                        sd.alphaz.sdev=0.0001;

                        sd.n.mean=0.157510945;
                        sd.n.sdev=0.01544867;

                        sd.t.mean=11.7;
                        sd.t.sdev=7.38;

                        sd.rho.mean=1.29E+06;
                        sd.rho.sdev=2.22E+05;

                        sd.rp.mean=5.78E-05;
                        sd.rp.sdev=2.95E-05;

                        sd.K0.mean=-2.20762404;
                        sd.K0.sdev=0.634080609;

                        //sd.h0.mean=0.3256;
                        //sd.h0.sdev=0.7601;

                        //sd.b.mean=0.177;
                        //sd.b.sdev=0.138;
                } else {
                        for (i=0; i<xmlsoils.size(); i++) {
                                if (String.copyValueOf(
                                   ((Medium)xmlsoils.get(i)).name.is).equalsIgnoreCase(
                                        soil)
                                ) sd=(Medium)xmlsoils.get(i);
                        }
                }
                return sd;
        }
        public void addSoil(Medium soil) {
                xmlsoils.add(soil);
        }
        private int i;
        private Vector xmlsoils=new Vector();
}
```

## 21 NormalF.java

```
import javax.swing.JTextField;

public class NormalF {
        public JTextField m=new JTextField("0.001");
        public JTextField s=new JTextField("0.001");
}
```

## 22 VirusPanel.java

```
import java.util.Observable;
import java.util.Observer;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
   <b>VirusPanel</b> class.
<P>
Panel to display the virus parameter fields to the user.

@author Barton R. Faulkner, US EPA National Risk Management Laboratory,
        Ada, Oklahoma, USA.
@version 18 October 2002
@see Operandum
*/


public class VirusPanel extends JPanel implements Observer {
        /** Construct a <b>VirusPanel</b>.
        @param gossiper A <b>Gossiper</b> object that makes this
        object aware of changes.
        */
        public VirusPanel(Gossiper gossiper) {
                JarLoadable jar=new JarLoadable();
                gossiper.addObserver(this);
                JLabel h1L=new JLabel("Parameter");
                JLabel h2L=new JLabel("Mean"); //mean
                JLabel h3L=new JLabel("Std. Deviation"); //standard deviation
                JLabel h4L=new JLabel("Units");
                JLabel[] lamL={ new JLabel(new ImageIcon(
                        jar.loadImage("lambda.gif"))),
                                new JLabel(new ImageIcon(
                        jar.loadImage("lambdastar.gif")))};
                lamL[0].setToolTipText("Log Mobile Virus Inactivation Rate");
                lamL[1].setToolTipText(
                "Log Solid-sorbed Virus Inactivation Rate");
                lambdaL=lamL;
                JLabel[] kapL={new JLabel(new ImageIcon(
                        jar.loadImage("kappa.gif"))),
                        new JLabel(new ImageIcon(
                        jar.loadImage("kappadia.gif")))};
                kapL[0].setToolTipText(
                "Mobile to solid-sorbed mass transfer coeff.");
                kapL[1].setToolTipText(
                "Mobile to air-sorbed mass transfer coeff.");
                kappaL=kapL;
```

```
rvL=new JLabel(new ImageIcon(jar.loadImage("rv.gif")));
rvL.setToolTipText("Radius of virus");
kdL=new JLabel(new ImageIcon(jar.loadImage("kd.gif")));
kdL.setToolTipText(
"Mobile to solid-sorbed equilibrium partition coeff.");

lambdaF=new NormalF[3];
lambdaF[0]=new NormalF();
lambdaF[1]=new NormalF();
kappaF=new NormalF[2];
kappaF[0]=new NormalF();
kappaF[1]=new NormalF();
rvF=new NormalF();
kdF=new NormalF();
units=new String[8];
units[0]="  h-\u00b9";
units[1]=units[2]=" log10( h-\u00b9 )";
units[3]=units[4]=" m h -\u00b9 ";
units[5]=" m";
units[6]=" any";
units[7]=" m\u00b3 g -\u00b9";

// I'm mystified what makes this layout work versus what
// causes it to fail. My duct-tape and bailing wire fix
// is to use this blank JPanel to fill in gaps.
fP=new JPanel();
GridLayout gl=new GridLayout(8,4);
setLayout(gl);
Vcp=new VirusComboPanel(gossiper);

add(h1L);
 add(h2L);
  add(h3L);
   add(h4L);
add(lambdaL[0]);
 add(lambdaF[0].m);
  add(lambdaF[0].s);
   add(new JLabel(units[1]));
add(lambdaL[1]);
 add(lambdaF[1].m);
  add(lambdaF[1].s);
   add(new JLabel(units[2]));
add(kappaL[0]);
 add(kappaF[0].m);
  add(kappaF[0].s);
   add(new JLabel(units[3]));
add(kappaL[1]);
 add(kappaF[1].m);
  add(kappaF[1].s);
   add(new JLabel(units[4]));
```

```
        add(rvL);
         add(rvF.m);
          add(rvF.s);
           add(new JLabel(units[5]));
        add(kdL);
         add(kdF.m);
          add(kdF.s);
           add(new JLabel(units[7]));
        add(fP);
         add(Vcp);
          add(fP);
           add(fP);
}
/** This method is called by the <b>Gossiper</b> (or other subclass
    of <b>Observable</b>) and updates the <b>Operandum</b> instance.
@see Gossiper
*/
public void update(Observable oble, Object o) {
        String name=o.getClass().getName();
        if (name.equals("Operandum")) {
                Operandum d=(Operandum)(o);

                lambdaF[0].m.setText(Double.toString(d.lambda0.mean));
                lambdaF[0].s.setText(Double.toString(d.lambda0.sdev));

                lambdaF[1].m.setText(Double.toString(d.lambda1.mean));
                lambdaF[1].s.setText(Double.toString(d.lambda1.sdev));

                kappaF[0].m.setText(Double.toString(d.kappa0.mean));
                kappaF[0].s.setText(Double.toString(d.kappa0.sdev));

                kappaF[1].m.setText(Double.toString(d.kappa1.mean));
                kappaF[1].s.setText(Double.toString(d.kappa1.sdev));

                rvF.m.setText(Double.toString(d.rv.mean));
                rvF.s.setText(Double.toString(d.rv.sdev));

                kdF.m.setText(Double.toString(d.Kd.mean));
                kdF.s.setText(Double.toString(d.Kd.sdev));

                repaint();
        }
}
/** Retrieve the virus data the user has selected.
@return The virus data.
*/
public Operandum getData() {
        Operandum vd=new Operandum();
        Double D;
```

```
        D=new Double(lambdaF[0].m.getText());
        vd.lambda0.mean=D.doubleValue();

        D=new Double(lambdaF[1].m.getText());
        vd.lambda1.mean=D.doubleValue();
        D=new Double(lambdaF[1].s.getText());
        vd.lambda1.sdev=D.doubleValue();

        D=new Double(kappaF[0].m.getText());
        vd.kappa0.mean=D.doubleValue();
        D=new Double(kappaF[0].s.getText());
        vd.kappa0.sdev=D.doubleValue();

        D=new Double(kappaF[1].m.getText());
        vd.kappa1.mean=D.doubleValue();
        D=new Double(kappaF[1].s.getText());
        vd.kappa1.sdev=D.doubleValue();

        D=new Double(rvF.m.getText());
        vd.rv.mean=D.doubleValue();
        D=new Double(rvF.s.getText());
        vd.rv.sdev=D.doubleValue();

        D=new Double(kdF.m.getText());
        vd.Kd.mean=D.doubleValue();
        D=new Double(kdF.s.getText());
        vd.Kd.sdev=D.doubleValue();

        return vd;
}
public void putData(Operandum newdata) {
        Vcp.addVirus(newdata, new String(newdata.name.is));
}
/** Retrieve the virus labels as a array of Strings.
@return The labels.
*/
public String[] getLabels() {
        String[] labels={
                lambdaL[0].getToolTipText(),
                lambdaL[1].getToolTipText(),
                kappaL[0].getToolTipText(),
                kappaL[1].getToolTipText(),
                rvL.getToolTipText(),
                kdL.getToolTipText()
        };
        return labels;
}
public String[] getUnits() {
        return units;
}
```

```
        public void requestAttention(String sF) {
                if (sF.equalsIgnoreCase("rv.m")) rvF.m.requestFocus();
                if (sF.equalsIgnoreCase("rv.s")) rvF.s.requestFocus();
                if (sF.equalsIgnoreCase("kappa0.s")) kappaF[0].s.requestFocus();
                if (sF.equalsIgnoreCase("kappa1.s")) kappaF[1].s.requestFocus();
                if (sF.equalsIgnoreCase("Kd.s")) kdF.s.requestFocus();
        }
        private VirusComboPanel Vcp;
        private JPanel fP;
        private JLabel[] lambdaL;
        private JLabel[] kappaL;
        private JLabel rvL;
        private JLabel kdL;
        private String[] units;
        private NormalF[] lambdaF;
        private NormalF[] kappaF;
        private NormalF rvF;
        private NormalF kdF;
}
```

## 23 VirusComboPanel.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
   <b>VirusComboPanel</b> class.
<P>
Combo panel of viruses.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 18 July 2000
@see Operandum
@see VirusStack
*/

public class VirusComboPanel extends JPanel {
        /** Construct a VirusComboPanel.
        @param gossiper A <b>Gossiper</b> object to make this object
        a concrete subject.
        */
        public VirusComboPanel(Gossiper gossiper) {
                gpr=gossiper;
                virusStack=new VirusStack();
                newdata=new Operandum();

                comboBox=new JComboBox();
                comboBox.addItem("polio-clay");
                comboBox.addItem("polio-silt");
                comboBox.addItem("polio-sand");
                comboBox.addItem("hepatitis A-clay");
                comboBox.addItem("hepatitis A-sand");
                comboBox.addItem("reovirus 3-clay");
                comboBox.addItem("reovirus 3-silt");
                comboBox.addItem("reovirus 3-sand");
                comboBox.addItem("coxsackievirus-clay");
                comboBox.addItem("coxsackievirus-sand");
                comboBox.addItem("echovirus-clay");
                comboBox.addItem("echovirus-silt");
                comboBox.addItem("echovirus-sand");
                comboBox.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                newdata=virusStack.getData(
                                        (String)(comboBox.getSelectedItem())
                                );
                                tell(newdata);
```

```
                }
        });

        GridLayout gl=new GridLayout(1,2);
        setLayout(gl);

        add(comboBox);
        newdata=virusStack.getData(
                (String)(comboBox.getSelectedItem())
        );
        tell(newdata);
    }
    public void addVirus(Operandum virus, String name) {
        comboBox.addItem(name);
        virusStack.addVirus(virus);
    }
    private void tell(Operandum nd) {
        gpr.tell(nd);
    }
    private Gossiper gpr;
    private VirusStack virusStack;
    private Operandum newdata;
    private JComboBox comboBox;
}
```

## 24 VirusStack.java

```
import java.util.*;

/**
   <b>VirusStack</b> class.
<P>
Stack for the virus data.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 18 July 2000
@see Operandum
*/

public class VirusStack {
        /** Retrieve the data entered by the user.
        @return the virus data.
        */
        public Operandum getData(String virus){
                Operandum vd=new Operandum();
                if (virus.equalsIgnoreCase("polio-sand")) {
                        String Name=new String("polio");
                        vd.name.is=Name.toCharArray();

                        vd.lambda0.mean=0.605; // log10(per hr)
                        vd.lambda0.sdev=0.608;

                        vd.lambda1.mean=0.304; // log10(per hr)
                        vd.lambda1.sdev=0.608;

                        vd.kappa0.mean=0.00134; // m/hr
                        vd.kappa0.sdev=0.00180;

                        vd.kappa1.mean=0.00927; // m/hr
                        vd.kappa1.sdev=0.00180;

                        vd.rv.mean=1.375e-8;
                        vd.rv.sdev=1.250e-9;

                        vd.Kd.mean=2.43e-4; // m^3/g
                        vd.Kd.sdev=5.66e-4;
                } else if (virus.equalsIgnoreCase("polio-silt")) {
                        String Name=new String("polio");
                        vd.name.is=Name.toCharArray();

                        vd.lambda0.mean=0.605; // log10(per hr)
                        vd.lambda0.sdev=0.608;
```

```
        vd.lambda1.mean=0.304; // log10(per hr)
        vd.lambda1.sdev=0.608;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-8;
        vd.rv.sdev=1.250e-9;

        vd.Kd.mean=3.77e-4; // m^3/g
        vd.Kd.sdev=7.16e-4;
} else if (virus.equalsIgnoreCase("polio-clay")) {
        String Name=new String("polio");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=0.605; // log10(per hr)
        vd.lambda0.sdev=0.608;

        vd.lambda1.mean=0.304; // log10(per hr)
        vd.lambda1.sdev=0.608;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-8;
        vd.rv.sdev=1.250e-9;

        vd.Kd.mean=7.20e-4; // m^3/g
        vd.Kd.sdev=9.74e-4;
} else if (virus.equalsIgnoreCase("hepatitis A-clay")) {
        String Name=new String("hepatitis");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-3.941; // log10(per hr)
        vd.lambda0.sdev= 0.782;

        vd.lambda1.mean=-3.446; // log10(per hr)
        vd.lambda1.sdev= .782;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
```

```
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.4e-8;
        vd.rv.sdev=1.250e-9;

        vd.Kd.mean=1.900e-3; // m^3/g
        vd.Kd.sdev=4.059e-6;
} else if (virus.equalsIgnoreCase("hepatitis A-sand")) {
        String Name=new String("hepatitis");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-3.941; // log10(per hr)
        vd.lambda0.sdev= 0.782;

        vd.lambda1.mean=-3.446; // log10(per hr)
        vd.lambda1.sdev= .782;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.4e-8;
        vd.rv.sdev=1.250e-9;

        vd.Kd.mean=4.68e-6; // m^3/g
        vd.Kd.sdev=4.06e-6;
} else if (virus.equalsIgnoreCase("reovirus 3-clay")) {
        String Name=new String("reovirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-3.941; // log10(per hr)
        vd.lambda0.sdev= 0.782;

        vd.lambda1.mean=-3.446; // log10(per hr)
        vd.lambda1.sdev= .782;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=3.5e-8;
        vd.rv.sdev=5.0e-9;

        vd.Kd.mean=1.203e-3; // m^3/g
        vd.Kd.sdev=3.188e-3;
} else if (virus.equalsIgnoreCase("reovirus 3-silt")) {
```

```
        String Name=new String("reovirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-3.941; // log10(per hr)
        vd.lambda0.sdev= 0.782;

        vd.lambda1.mean=-3.446; // log10(per hr)
        vd.lambda1.sdev= .782;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=3.5e-8;
        vd.rv.sdev=5.0e-9;

        vd.Kd.mean=2.113e-3; // m^3/g
        vd.Kd.sdev=1.648e-3;
} else if (virus.equalsIgnoreCase("reovirus 3-sand")) {
        String Name=new String("reovirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-3.941; // log10(per hr)
        vd.lambda0.sdev= 0.782;

        vd.lambda1.mean=-3.446; // log10(per hr)
        vd.lambda1.sdev= .782;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=3.5e-8;
        vd.rv.sdev=5.0e-9;

        vd.Kd.mean=3.000e-3; // m^3/g
        vd.Kd.sdev=9.068e-3;
} else if (virus.equalsIgnoreCase("coxsackievirus-clay")) {
        String Name=new String("coxsackievirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-3.941; // log10(per hr)
        vd.lambda0.sdev= 0.782;

        vd.lambda1.mean=-3.446; // log10(per hr)
        vd.lambda1.sdev= .782;
```

```
        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-10;
        vd.rv.sdev=1.e-11;

        vd.Kd.mean=8.657e-5; // m^3/g
        vd.Kd.sdev=1.956e-4;
} else if (virus.equalsIgnoreCase("coxsackievirus-sand")) {
        String Name=new String("coxsackievirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-2.515; // log10(per hr)
        vd.lambda0.sdev= 0.212;

        vd.lambda1.mean=-2.774; // log10(per hr)
        vd.lambda1.sdev=  .212;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-10;
        vd.rv.sdev=1.e-11;

        vd.Kd.mean=6.150e-4; // m^3/g
        vd.Kd.sdev=2.350e-3;
} else if (virus.equalsIgnoreCase("echovirus-clay")) {
        String Name=new String("echovirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-2.406; // log10(per hr)
        vd.lambda0.sdev= 0.162;

        vd.lambda1.mean=-2.684; // log10(per hr)
        vd.lambda1.sdev=  .162;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-10;
```

```
        vd.rv.sdev=1.e-11;

        vd.Kd.mean=4.535e-4; // m^3/g
        vd.Kd.sdev=7.655e-4;
} else if (virus.equalsIgnoreCase("echovirus-silt")) {
        String Name=new String("echovirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-2.406; // log10(per hr)
        vd.lambda0.sdev= 0.162;

        vd.lambda1.mean=-2.684; // log10(per hr)
        vd.lambda1.sdev=  .162;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-10;
        vd.rv.sdev=1.e-11;

        vd.Kd.mean=4.42e-4; // m^3/g
        vd.Kd.sdev=2.76e-3;
} else if (virus.equalsIgnoreCase("echovirus-sand")) {
        String Name=new String("echovirus");
        vd.name.is=Name.toCharArray();

        vd.lambda0.mean=-2.406; // log10(per hr)
        vd.lambda0.sdev= 0.162;

        vd.lambda1.mean=-2.684; // log10(per hr)
        vd.lambda1.sdev=  .162;

        vd.kappa0.mean=0.00134; // m/hr
        vd.kappa0.sdev=0.00180;

        vd.kappa1.mean=0.00927; // m/hr
        vd.kappa1.sdev=0.00180;

        vd.rv.mean=1.375e-10;
        vd.rv.sdev=1.e-11;

        vd.Kd.mean=7.44e-4; // m^3/g
        vd.Kd.sdev=2.525e-3;
} else {
        for (i=0; i<xmlviruses.size(); i++) {
                if (String.copyValueOf(
                        ((Operandum)xmlviruses.get(
```

```
                                        i)).name.is).equalsIgnoreCase(virus)
                                ) vd=(Operandum)xmlviruses.get(i);
                        }
                }
                return vd;
        }
        public void addVirus(Operandum virus) {
                xmlviruses.add(virus);
        }
        private int i;
        private Vector xmlviruses=new Vector();
}
```

## 25 OutputPanel.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

/**
   <b>OutputPanel</b> class.
<P>
Prints a string. Inherits <b>JTextPane</b>.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 18 January 2001
*/

public class OutputPanel extends JTextPane {
        /** Construct a <b>OutputPanel</b> object.
        */
        public OutputPanel() {
        }
        /** Print the <i>outString</i>.
        @param outString The string.
        */
        public void printOutput(String outString) {
                setText(outString);
        }
}
```

## 26 HistoPanel.java

```
import java.awt.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import javax.swing.*;

public class HistoPanel extends JPanel {
        public HistoPanel(int[] bins, int wd, int ht) {
                w=wd;
                h=ht;
                bin=bins;
                JarLoadable jar=new JarLoadable();
                String sb=new String();
        }
        public void redraw(int[] bins) {
                bin=bins;
                repaint();
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                g.setColor(Color.red);
                g.drawString("Exceedances",margin,h-15);
                g.drawString(Hits,margin,h);
                g.drawRect(margin+(int)(bp),margin+20,0,320);
                if (bp<1.) {
                        g.drawString("Note: resolution will not allow "+
                                "display of vertical marker bar.",
                                margin+bin.length/2,margin-20);
                }
                g.setColor(Color.black);
                g.drawRect(margin,margin+20,0,320);
                g.drawString("Right Truncated Histogram",
                                margin+bin.length/2,margin);
                g.drawString("-log A",margin+4*bin.length/2-8,h-30);
                g.drawString("0",58,h-45);
                g.drawString("150",64+4*bin.length/2,h-45);
                g.drawString("300",50+4*bin.length,h-45);
                g.drawString(":   "+Runs,margin+85,h);
                g.drawString(":   Runs",margin+85,h-15);
                for (i=0; i<bin.length; i++) {
                        g.drawRect(margin+i*4, h-bin[i]-margin ,2, bin[i] );
                }
                for (i=0; i<320; i+=20) {
                        g.drawRect(margin-5,h-margin-i,4,0);
                        g.drawString(sb.valueOf(i),margin-30,h-margin-i+5);
                }
                g.drawString("C",15,h-margin-210);
                g.drawString("O",15,h-margin-190);
                g.drawString("U",15,h-margin-170);
```

```
                g.drawString("N",15,h-margin-150);
                g.drawString("T",15,h-margin-130);
        }
        public void setBreak(double brk) {
                bp=brk;
        }
        public void setHits(int hits) {
                Hits=Integer.toString(hits);
        }
        public void setRuns(int runs) {
                Runs=Integer.toString(runs);
        }
        public BufferedImage getImage() {
                BufferedImage image=new BufferedImage(w,h,BufferedImage.TYPE_INT_RGB);
                return image;
        }
        private String sb;
        private JarLoadable jar;
        private int[] bin;
        private String Hits="0";
        private String Runs="0";
        private String is;
        private int h, w, i, margin=60;
        private double bp=4;
}
```

## 27 JpgFilter.java

```
import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;

public class JpgFilter extends FileFilter {
    public boolean accept(File f) {
        if (f.isDirectory()) {
            return true;
        }
        String extension = Utils.getExtension(f);
        if (extension != null) {
            if (extension.equals(Utils.jpg) ||
                extension.equals(Utils.txt) ||
                extension.equals(Utils.xml)) {
                return true;
            } else {
                return false;
            }
        }
        return false;
    }
    // The description of this filter
    public String getDescription() {
        return "*.jpg, *.txt, *.xml";
    }
}
```

## 28 DataBuilder.java

```
import java.util.*;
import java.io.*;

/** This class is a surrogate for a forthcoming <i>bona fide</i> XML
parser which I have not yet written. It simply takes an entire
<b>RandomAccessFile</b> and tokenizes it to build either new
<b>Operandum</b> or new <b>Medium</b> data sets created by the user.
In the future, <i>Virulo</i> should also be able to employ a <i>bona
fide</i> persistence model so user's won't have to keep opening up
their XML files every time.
<P>
@author Barton R. Faulkner<br>
U.S. EPA Office of Research and Development<br>
National Risk Management Research Laboratory<br>
Ada, Oklahoma, USA<br>
@version 31 October 2002
*/

public class DataBuilder {
        public DataBuilder() {
                me=new Medium();
                op=new Operandum();
                obj=new Object();
                element=new String();
        }
        public Object getData(RandomAccessFile xmlfile) {
                fstring="";
                // Try to read the whole xml file in to one long String
                // for tokenizing:
                try {
                        while((xmlline=xmlfile.readLine()) != null) {
                                fstring+=xmlline;
                        }
                        xmlfile.close();
                } catch (Exception e) {
                        System.out.println(e);
                }
                return parse(fstring);
        }
        private Object parse(String str) {
                Obviously=null;
                Name=null;
                c=str.toCharArray();
                for (i=0; i<c.length; i++) {
                        // There are 4 possible places I can be,
                        // 1. at the start of an element:
                        if (c[i]=='<') {
                                offset=i;
```

```
                element="";
                ImInAnElement=true;
            // 2. at the end of an element's tag, in which case
            // I'm entering the place where I can find data:
            } else if (c[i]=='>') {
                element=new String(c,offset+1,count);
                element=element.trim();
                if (element.equalsIgnoreCase("operandum"))
                        Obviously="operandum";
                if (element.equalsIgnoreCase("medium"))
                        Obviously="medium";
                if (element.equalsIgnoreCase("name")) {
                        j=0;
                        while (c[i+j]!='<') j++;
                        Name=new String(c,i+1,j-1);
                }
                ImInAnElement=false;
                if (gotMean && gotSdev) {
                        element=element.substring(1);
                        element=element.trim();
                        buildUp(Obviously, Name, element, Mean, Sdev);
                        gotMean=false;
                        gotSdev=false;
                } else if (element.equalsIgnoreCase("mean")) {
                        j=0;
                        while (c[i+j]!='<') j++;
                        Mean=new String(c,i+1,j-1);
                        gotMean=true;
                        k=0;
                        while (c[i+j+k]!='>') k++;
                        i=i+j+k;
                } else if (element.equalsIgnoreCase("sdev")) {
                        j=0;
                        while (c[i+j]!='<') j++;
                        Sdev=new String(c,i+1,j-1);
                        gotSdev=true;
                        k=0;
                        while (c[i+j+k]!='>') k++;
                        i=i+j+k;
                }
                count=0;
            // 3. inside, traversing the element:
            } else {
                if (ImInAnElement) count++;
            }
            // 4. waiting until I find something I care about...
    }
    if (Obviously.equalsIgnoreCase("medium")) {
            obj=(Object)me;
    } else if (Obviously.equalsIgnoreCase("operandum")) {
```

```
                obj=(Object)op;
        }
        return obj;
    }
    private void buildUp(String object, String name, String key, String m, String s) {
        if (object.equalsIgnoreCase("operandum")) {
                n=new Normal(Double.parseDouble(m), Double.parseDouble(s));
                StringAsChars sac=new StringAsChars();
                sac.is=name.toCharArray();
                op.name=sac;
                if (key.equals("lam0")) op.lambda0=n;
                if (key.equals("lam1")) op.lambda1=n;
                if (key.equals("kap0")) op.kappa0=n;
                if (key.equals("kap1")) op.kappa1=n;
                if (key.equals("radv"))  op.rv=n;
                if (key.equals("kd"))    op.Kd=n;
        } else if (object.equalsIgnoreCase("medium")) {
                n=new Normal(Double.parseDouble(m), Double.parseDouble(s));
                StringAsChars sac=new StringAsChars();
                sac.is=name.toCharArray();
                me.name=sac;
                if (key.equals("th_r")) me.theta_r=n;
                if (key.equals("th_m")) me.theta_m=n;
                if (key.equals("th_s")) me.theta_s=n;
                if (key.equals("len"))  me.z=n;
                if (key.equals("alpz")) me.alphaz=n;
                if (key.equals("logn")) me.n=n;
                if (key.equals("temp")) me.t=n;
                if (key.equals("rho"))  me.rho=n;
                if (key.equals("rp"))   me.rp=n;
                if (key.equals("logks")) me.K0=n;
                if (key.equals("logal")) me.a=n;
        }
    }
    private Operandum op;
    private Medium me;
    private Object obj;
    private char[] c;
    private String xmlline, fstring, element, Mean, Sdev, Name, Obviously, name;
    private Normal n;
    private int i, j, k, offset, count;
    private boolean ImInAnElement, gotMean, gotSdev;
}
```

## 29 UnsatVirusAttenuator.java

```
/**
   <b>UnsatVirusAttenuator</b> class.
<P>
Computes the unsaturated gravity flow water flux and attenuation
factor (<i>A=Mf/cmax</i>) for the given hydrogeologic layer.
Fairly single-minded purpose for existing.
<hr>
<b>References</b>
<p>
Durner,W. Priesack, E., Vogel, H., Zurmuhl, T. 1997,
Determination of Parameters for Flexible
Hydraulic Functions by Inverse Modeling (In) <i>Characterization
and Measurement of the Hydraulic Properties of
Unsaturated Porous Media</i>. pp. 817-829:
<p>
Rose, W., Bruce, W.A. 1949. Evaluation of capillary character in
petroleum reservoir rock. <i>Transactions of the American Institute
of Mining and Metallurgical Engineers 186</i>:127-142.
<p>
Schaefer, C.E., Arands, R.R., van der Sloot, H.A. Kosson, D.S.
Prediction and experimental validation of liquid-phase diffusion
resistance in unsaturated soils. <i>Journal of Contaminant Hydrology
20</i>:145-166.
<p>
Skopp, J., 1985. Oxygen uptake and transport in soils: analysis of
the airwater interfacial area.
<i>Soil Sci. Soc. Am. J. 49</i>(6):1327-1331.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version 12 February 2002
@see Medium
@see Operandum
*/
public class UnsatVirusAttenuator implements Attenuator {

/* ------------------------
   Constructor
 * ------------------------ */

        /** Construct one.
        */
        public UnsatVirusAttenuator(Medium soildata, Operandum virusdata) {
                sd=soildata;
                vd=virusdata;
                random=new Random(300.,1000);
```

```
        }
/* ------------------------
   Public Methods
 * ------------------------ */

        /** Compute the gravity flow water flux.
        @return qf      The flux rate.
        */
        public double getAdvection() {
                qf=getKh(sd.theta_m.mean)*(0.1-geth(sd.theta_m.mean))/sd.z.mean+
                        getKh(sd.theta_m.mean)*sd.z.mean;
                return qf;
        }
        public double getKh(double wc) {
                Se=(wc-sd.theta_r.mean)/
                        (sd.theta_s.mean-sd.theta_r.mean);
                innermost=Math.pow( Se , alog_n/(alog_n-1) );
                double kh=alog_K0*Math.sqrt(Se)*
                   Math.pow(1- Math.pow(1-innermost, 1-1/alog_n) ,2);
                return kh;
        }
        public double geth(double wc) {
                Se=(wc-sd.theta_r.mean)/
                        (sd.theta_s.mean-sd.theta_r.mean);
                innermost=Math.pow(Se, alog_n/(1-alog_n));
                return -(1/ALPHA)*Math.pow(innermost-1,1/alog_n);
        }
        /** Compute the attenuation factor.
        @param q        The flux rate.
        @return Mf/Cmax The attenuation factor.
        */
        public double getAttenuation() {
                // Adjust the variables stored as logarithms:
                alog_a=Math.pow(10,sd.a.mean);
                alog_n=Math.pow(10,sd.n.mean);
                alog_K0=Math.pow(10,sd.K0.mean);
                alog_lambda0=Math.pow(10,vd.lambda0.mean);
                alog_lambda1=Math.pow(10,vd.lambda1.mean);
                T=sd.t.mean+273.16;
                q=getAdvection();

                at=3.*(1-sd.theta_s.mean)/sd.rp.mean;
                k=vd.kappa0.mean*at;

                arg=Math.pow(Se , alog_n/(1.-alog_n));

//  This is Rose-Bruce (1949):
                atdia=RHOW*GRAVITY*sd.theta_m.mean*Math.pow(
                        arg-1,1/alog_n)/(SIGMA*alog_a);
/*
```

```
 *  This is Skopp (1985):
 *              atdia=1.2*RHOW*GRAVITY*sd.theta_m.mean*Math.pow(
 *                      arg-1,1/alog_n)/(SIGMA*alog_a);
 */

/*
 *  This is Cary:
 *              r0=2*SIGMA/(RHOW*GRAVITY*sd.h0.mean);
 *              arg1=-sd.b.mean/(Math.pow(sd.theta_s.mean,-sd.b.mean)-
 *                      Math.pow(LESSTINY,-sd.b.mean));
 *              arg2=(Math.pow(sd.theta_s.mean,-sd.b.mean+1)-Math.pow(
 *                      LESSTINY,-sd.b.mean+1))/
 *                              (-sd.b.mean+1);
 *              eta=at*r0/(2*LESSTINY*Math.pow(sd.theta_s.mean,sd.b.mean))*
 *                      arg1-arg2*arg1/LESSTINY;
 *              atdia=2*Math.pow(sd.theta_s.mean,sd.b.mean)/r0*(
 *                                      eta*sd.theta_r.mean/arg1+arg2);
 */
                kdia=vd.kappa1.mean*atdia;
                gamma=alog_lambda0
                                +alog_lambda1*sd.rho.mean*k
                                /(k*sd.theta_m.mean/vd.Kd.mean+alog_lambda1*sd.rho.mean)
                                +kdia;
                V=q/sd.theta_m.mean;
                // Use Schaefer et al (1995):
                if (sd.theta_m.mean<=0.2) {
                        tau=Math.pow(sd.theta_s.mean,2)/
                                Math.pow(sd.theta_m.mean,11/5);
                } else {
                        tau=Math.pow(sd.theta_s.mean,2)/
                                Math.pow(sd.theta_m.mean,7/3);
                }
                D=BOLTZMANN*T/(6*Math.PI*VISCOSITY*vd.rv.mean);
                De=D*tau;
                Dz=sd.alphaz.mean*V+De;
                Gamma=(V-Math.sqrt(V*V+4*Dz*gamma))/(2*Dz);
                A=Math.exp(Gamma*sd.z.mean);
                // Cannot create viruses and cannot have less than zero, hence
                if (A>1.) A=1.;
                if (A<=INFINITESIMAL) {
                        pA=-random.nextDouble();
                } else {
                        pA=HspBasicMath.log10(A);
                }
                return pA;
        }
        public Medium getMedium() {
                return sd;
        }
        public Operandum getOperandum() {
```

```
        return vd;
}
public void replace(Medium sdat, Operandum vdat) {
        sd=sdat;
        vd=vdat;

        /*
         *  Perform some checks on the physical reasonableness
         *  of the perturbed parameters, since they were
         *  generated randomly, and condition them as necessary:
         *
         *  First, all normally distributed parameters
         *  must be > zero:
         */

        if (sd.theta_s.mean<=0.) sd.theta_s.mean=INFINITESIMAL;
        if (sd.theta_r.mean<=0.) sd.theta_r.mean=INFINITESIMAL;
        if (sd.theta_m.mean<=0.) sd.theta_m.mean=INFINITESIMAL;
        if (sd.z.mean<=0.) sd.z.mean=INFINITESIMAL;

        /*
         *  We use n<10 (or log10(n)<1) (Durner et al. 1997)
         */

        if (sd.n.mean>1.) sd.n.mean=1.-INFINITESIMAL;
        if (sd.rho.mean<=0.) sd.rho.mean=INFINITESIMAL;
        if (sd.rp.mean<=0.) sd.rp.mean=INFINITESIMAL;
        if (vd.kappa0.mean<0.) vd.kappa0.mean=INFINITESIMAL;
        if (vd.kappa1.mean<=0.) vd.kappa1.mean=INFINITESIMAL;
        if (vd.Kd.mean<=0.) vd.Kd.mean=INFINITESIMAL;

        /*
         *  Next, all water contents must also be less than 1:
         */

        if (sd.theta_s.mean>=1) sd.theta_s.mean=1-INFINITESIMAL;
        if (sd.theta_r.mean>=1) sd.theta_r.mean=1-INFINITESIMAL;
        if (sd.theta_m.mean>=1) sd.theta_m.mean=1-INFINITESIMAL;

        /*
         *  Next, the residual moisture must be less than
         *  the moisture, and we place more trust in the
         *  in the moisture value:
         */

        if (sd.theta_r.mean>=sd.theta_m.mean) {
                sd.theta_r.mean=sd.theta_m.mean-LESSTINY;
        }

        /*
```

```
                 *  Next, the saturated moisture must be greater
                 *  than the soil moisture content:
                 */

                if (sd.theta_s.mean<=sd.theta_m.mean) {
                        sd.theta_s.mean=sd.theta_m.mean+LESSTINY;
                }
        }
        private Random random;
        private Medium sd;
        private Operandum vd;
        private double alog_a, alog_n, alog_K0, alog_lambda0, alog_lambda1;
        private double q, innermost, qf, T;
        private double at, k, atdia, arg;
        private double kdia, gamma, V, tau, D, De;
        private double Se, Dz, Gamma, A, pA;
        private static final double ALPHA=1.0;
        private static final double SIGMA=962020800;            // g*m/hr^2
        private static final double RHOW=1.e6;                  // g/m^3
        private static final double GRAVITY=127008000.;         // m/hr^2
        private static final double BOLTZMANN=1.789332768e-13;  //(g*m)/hr^2 (CRC)
        private static final double VISCOSITY=4.705200006e15;   // g/(m*hr) (CRC)
        private static final double INFINITESIMAL=1e-300;
        private static final double TINY=1e-15;                 // A small number,
                                                                // but not the smallest.
        private static final double LESSTINY=1e-6;              // Another small number.
}
```

## 30 ImagePanel.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
This code is by
Horstmann and Cornell, p. 310.
<p>
Reference:<br>
Horstmann, Cay S., and Gary Cornell. 1999. <i>Core Java 2, Volume 1--Fundamentals</i>
The Sun Microsystems Press, Palo Alto. 742 p.
*/

public class ImagePanel extends JPanel {
        public ImagePanel() {
                Toolkit tkt=Toolkit.getDefaultToolkit();
                image=tkt.createImage("logo.gif");
                MediaTracker tracker=new MediaTracker(this);
                tracker.addImage(image,0);
                try {
                        tracker.waitForID(0);
                } catch(InterruptedException e){}
        }
        public void paintComponent(Graphics g) {
                super.paintComponent(g);
                Dimension d=getSize();
                int clientWidth=d.width;
                int clientHeight=d.height;
                int imageWidth=image.getWidth(this);
                int imageHeight=image.getHeight(this);
                g.drawImage(image,10,10,this);
        }
        public Image image;
}
```

## 31 Utils.java

```
import java.io.File;

//http://java.sun.com/docs/books/tutorial/uiswing/components/example-swing/Utils.java

public class Utils {

    public final static String jpg = "jpg";
    public final static String txt = "txt";
    public final static String xml = "xml";

    /*
     * Get the extension of a file.
     */
    public static String getExtension(File f) {
        String ext = null;
        String s = f.getName();
        int i = s.lastIndexOf('.');

        if (i > 0 &&  i < s.length() - 1) {
            ext=s.substring(i+1).toLowerCase();
        }
        return ext;
    }
}
```

## 32 HspBasicMath.java

```
import java.lang.reflect.*;
/**
   <b>HspBasicMath</b> class.
<P>
Repository for static methods useful to the Hsp program.

@author Barton R. Faulkner<br>
        U.S. EPA Office of Research and Development<br>
        National Risk Management Research Laboratory<br>
        Ada, Oklahoma, USA<br>
@version March 2000
*/

public class HspBasicMath {

/* ------------------------
   Public Methods
 * ------------------------ */
        /** Grow any type of native vector by one element. Based on
            Horstmann & Cornell, "Core Java 2, Vol. 1"
        @param a The <b>Object</b> which is an array underneath.
        @return newVector The <b>Object</b> which is the input array
        grown by one element.
        */ static Object vectorGrow(Object a) {
                // grows any type of vector by one element.
                // based on Horstmann & Cornell, "Core Java2, Vol. 1"
                Class cl=a.getClass();
                if (!cl.isArray()) return null;
                Class componentType=a.getClass().getComponentType();
                int length=Array.getLength(a);
                int newLength=length+1;
                Object newVector=Array.newInstance(componentType,newLength);
                System.arraycopy(a,0,newVector,0,length);
                return newVector;
        }
        /** A log-base-10 function.
        */
        static double log10(double x) {
                return Math.log(x)/Math.log(10.0);
        }
}
```