Java source code

GammaCell.java

```
   /* written by Brian Chow
    * created February 27, 1998
    * last modified March 23, 1999
    */

   import java.awt.*;
   import java.awt.event.*;

   /**
    *
    */
   public class GammaCell extends Frame implements ActionListener {
      private static final int numOptions = 5;
      private static final String[] optionTitles = {"Units", "Experiment
Date", "Source Bundle", "Source Fixture", "Target"};
      private static final String[] optionButtonLabels = {"Set Units", "Set
Date", "Select Bundle", "Configure Fixtures", "Configure Target"};
      private Day experimentDate;
      private Units experimentUnits;
      private BundleList sources;
      private Fixture experimentFixture;
      private Target experimentTarget;
      private Font titleFont = new Font("Serif", Font.BOLD, 14);
      private Font labelFont = new Font("San Serif", Font.PLAIN, 12);
      private String[] optionLabels = new String[numOptions];
      private Panel optionsPanel, actionButtonsPanel;
      private Panel[] optionPanelsArray = new Panel[numOptions];
      private Label[] optionTitlesArray = new Label[numOptions];
      private Label[] optionLabelsArray = new Label[numOptions];
      private Button aboutButton, gradientButton, runButton, quitButton;
      private Button[] optionButtonsArray = new Button[numOptions];
   /**
   *
   */
   public GammaCell() {
      experimentUnits = new Units();
      experimentDate = new Day();
      sources = new BundleList();
      sources.add(new RadBundle("1994", new Day(1994, 3, 11), 10794, 12));
      sources.add(new RadBundle("1979", new Day(1979, 9, 19), 9950, 12));
      sources.add(new RadBundle("1963", new Day(1963, 1, 14), 10600, 20));
      sources.setDate(experimentDate);
      sources.setExpBundle("1979");
      experimentFixture = new AnnularFixture(new Coordinate(),
   sources.getExpBundle(), 1, 6);
      // experimentTarget = new TargetLine(new Coordinate(-1,0,0),
      // new Coordinate(1,0,0),11);
      experimentTarget = new TargetRect(new Coordinate(10, 0, 0), 6, 3, 3);
      setOptionLabels();
      setTitle("Gamma Cell Dosage");
      setLayout(new BorderLayout());
      optionsPanel = new Panel();
      constructOptionsPanel(optionsPanel);
      add(optionsPanel, "Center");
      actionButtonsPanel = new Panel();
      aboutButton = new Button("About");
      aboutButton.addActionListener(this);
      actionButtonsPanel.add(aboutButton);
      gradientButton = new Button("Gradient Plot");
```

```
      gradientButton.addActionListener(this);
      actionButtonsPanel.add(gradientButton);
      runButton = new Button("Calculate");
      runButton.addActionListener(this);
      actionButtonsPanel.add(runButton);
      quitButton = new Button("Quit");
      quitButton.addActionListener(this);
      actionButtonsPanel.add(quitButton);
      add(actionButtonsPanel, "South");
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            quitProgram();
         }
      });
      pack();
   }
   public void actionPerformed(ActionEvent evt) {
      if (ClickChecker.isDouble()) {
         return;
      }
      String arg = evt.getActionCommand();
      if (arg.equals(optionButtonLabels[0])) {
         UnitDialog experimentUnitDialog;
         experimentUnitDialog = new UnitDialog(this);
         experimentUnitDialog.showDialog();
      }
      else
         if (arg.equals(optionButtonLabels[1])) {
            (new DateDialog(this, "Change Experiment Date",
   experimentDate)).setVisible(true);
            sources.setDate(experimentDate);
         }
         else
            if (arg.equals(optionButtonLabels[2])) {
               (new BundleDialog(this, sources,
   experimentDate)).setVisible(true);
               experimentFixture.setSource(sources.getExpBundle());
            }
            else
               if (arg.equals(optionButtonLabels[3])) {
                  FixtureDialog selectFixtureDialog;
                  selectFixtureDialog = new FixtureDialog(this,
   experimentFixture, sources.getExpBundle());
                  experimentFixture = selectFixtureDialog.getFixture();
               }
               else
                  if (arg.equals(optionButtonLabels[4])) {
                     TargetDialog selectTargetDialog;
                     selectTargetDialog = new TargetDialog(this,
   experimentTarget);
                     experimentTarget = selectTargetDialog.getTarget();
                  }
                  else
                     if (arg.equals("About")) {
                        Dialog d = new AboutDialog(this);
                        d.show();
                     }
                     else
                        if (arg.equals("Gradient Plot")) {
                           GradientDialog gd = new GradientDialog(this);
                           gd.setFixture(experimentFixture);
                           gd.setVisible(true);
                        }
                        else
```

```
                           if (arg.equals("Calculate")) {
                              experimentTarget.calculate(experimentFixture);
                           }
                           else
                              if (arg.equals("Quit")) {
                                 quitProgram();
                              }
      setOptionLabels();
   }
   private void constructOptionsPanel(Panel p) {
      p.setLayout(new GridBagLayout());
      GridBagConstraints gbc = new GridBagConstraints();
      gbc.weighty = 100;
      gbc.gridwidth = 1;
      gbc.gridheight = 1;
      for (int i = 0; i < numOptions; i++) {
         gbc.gridy = i;
         optionPanelsArray[i] = new Panel(new GridLayout(3, 1));
         optionTitlesArray[i] = new Label(optionTitles[i]);
         optionTitlesArray[i].setFont(titleFont);
         optionPanelsArray[i].add(optionTitlesArray[i]);
         optionLabelsArray[i].setFont(labelFont);
         optionPanelsArray[i].add(optionLabelsArray[i]);
         Label tempLabel = new Label(" ");
         tempLabel.setFont(labelFont);
         optionPanelsArray[i].add(tempLabel);
         gbc.weightx = 100;
         gbc.fill = GridBagConstraints.BOTH;
         gbc.gridx = 1;
         p.add(optionPanelsArray[i], gbc);
         optionButtonsArray[i] = new Button(optionButtonLabels[i]);
         optionButtonsArray[i].addActionListener(this);
         gbc.weightx = 0;
         gbc.fill = GridBagConstraints.HORIZONTAL;
         gbc.gridx = 2;
         p.add(optionButtonsArray[i], gbc);
      }
   }
   /**
   *
   */
   public static void main(String[] args) {
      Frame f = new GammaCell();
      f.show();
   }
   private void quitProgram() {
      setVisible(false);
      if (System.getSecurityManager() == null) {
         System.exit(0);
      }
   }
   private void setOptionLabels() {
      optionLabels[0] = experimentUnits.toString();
      optionLabels[1] = experimentDate.toString();
      optionLabels[2] = sources.getExpBundle().toString();
      optionLabels[3] = experimentFixture.toString();
      optionLabels[4] = experimentTarget.toString();
      for (int i = 0; i < numOptions; i++) {
         if (optionLabelsArray[i] == null) {
            optionLabelsArray[i] = new Label(optionLabels[i]);
         }
         else {
            optionLabelsArray[i].setText(optionLabels[i]);
         }
```

```
      }
   }
   }
```

GammaCellApplet.java

```
   /* written by Brian Chow
    * created February 28, 1998
    * last modified March 23, 1999
    */

   import java.awt.*;
   import java.applet.*;
   public class GammaCellApplet extends Applet {
      private Frame f;
      private String messageString;
   public void paint(Graphics g) {
      g.drawString(messageString, 10, 25);
   }
   public void start() {
      if (f == null) {
         try {
            messageString = "Loading ...";
            repaint();
            f = new GammaCell();
         }
         catch (SecurityException e) {
            messageString = "GammaCell could not be loaded";
            repaint();
         }
      }
      messageString = "GammaCell loaded";
      repaint();
      f.setVisible(true);
   }
   public void stop() {
      f.setVisible(false);
      f.dispose();
      messageString = "GammaCell stopped";
      repaint();
   }
   }
```

AboutDialog.java

```
   /* written by Brian Chow
    * created February 28, 1998
    * last modified May 2, 1999
    */

   import java.awt.*;
   import java.awt.event.*;

   class AboutDialog extends Dialog
   {
      public AboutDialog(Frame parent)
      {  super(parent,"About Gamma Cell",true);
        setLayout(new BorderLayout());

        Panel p = new Panel();
        String[] text = {"Gamma Cell Irradiation Dose Program / Applet",
         " ",
         "  by Brian Y. Chow",
      "  advisor:",
      "    Jean B. Hunter",
      "      Department of Agricultural and Biological Engineering",
      "  special thanks:",
      "    Scott Lassell, Martin Moravek, & Samuel J. DiPasquale",
      "      Ward Center for Nuclear Sciences",
      "    Doug Caveney",
      "      Department of Agricultural and Biological Engineering",
      " ",
      "  Cornell University, Ithaca NY"};
     GridLayout gl = new GridLayout(text.length,1);
     gl.setVgap(-5);
     p.setLayout(gl);
     Label l;
     for (int i = 0;i < text.length;i++)
     {  l = new Label(text[i]);
       p.add(l);
     }
     add(p,"Center");

     Panel p2 = new Panel();
     p2.setLayout(new FlowLayout());
     Button okButton = new Button("OK");
     okButton.addActionListener(new ActionListener()
     {  public void actionPerformed(ActionEvent e)
       {  setVisible(false);
       }
     });
     p2.add(okButton);
     add(p2,"South");

     addWindowListener(new WindowAdapter()
     {  public void windowClosing(WindowEvent e)
       {  setVisible(false);
       }
     });
     pack();
   }
}
```

CalculationProgress.java

```
/* written by Brian Chow
 * created April 24, 1998
 * last modified March 23, 1999
 */

import java.awt.*;
import java.awt.event.*;

/**
 * Calculation progress window.
 */
public class CalculationProgress extends Frame {
   /**
   * String of spaces for initialization of the output area.
   */
   private static final String spaces = "
";
   /**
   * Text for the output.
   */
   private Label outputArea;
   /**
```

```
   * Font used for the output area.
   */
   private Font outputFont = new Font("Monospaced", Font.PLAIN, 12);
   /**
   * Time when the output text has last been updated.
   */
   private long textLastUpdated;
/**
* Constructs new window with a text for the output.
*/
public CalculationProgress() {
   // Set up the window with title and output area.
   setTitle("Calculating...");
   setLayout(new GridLayout(2, 1));
   Label waitLabel = new Label("Please Wait");
   add(waitLabel);
   outputArea = new Label(spaces);
   outputArea.setFont(outputFont);
   add(outputArea);
   setSize(350, 100);

   // Center the window in the screen.
   Dimension screenSize = getToolkit().getScreenSize();
   Dimension windowSize = getSize();
   setLocation((screenSize.width - windowSize.width) / 2,
(screenSize.height - windowSize.height) / 2);
   setEnabled(false);
}
/**
* Clear the text window.
*/
public void clear() {
   outputArea.setText(spaces);
}
/**
* Change the output string value.
*/
public void print(String value) {
   // Update output field only if more than 100 milliseconds elapsed
   // since last update.  (Prevents lengthy calculations from being
   // bogged down by screen updates)
   final long nowTime = System.currentTimeMillis();
   if (nowTime - textLastUpdated > 100) {
      textLastUpdated = nowTime;
      outputArea.setText(value);
   }
}
}
```

DoubleTextFieldPanel.java

```
/* written by Brian Chow
 * created March 23, 1998
 * last modified March 23, 1999
 */

import java.awt.*;
import java.awt.event.*;

/**
 *
 */
public class DoubleTextFieldPanel extends Panel {
   private TextField tValue;
```

```
   private double value;
   private TextField ivjDoubleTextfield = null;
   private Label ivjLabel = null;
/**
 * Constructor
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public DoubleTextFieldPanel() {
   super();
   initialize();
}
public DoubleTextFieldPanel(TextListener parent, String labelText, String
textFieldValue, int columnWidth) {
   getLabel().setText(labelText);
   getDoubleTextfield().setText(textFieldValue);
   getDoubleTextfield().setColumns(columnWidth);
   getDoubleTextfield().addTextListener(parent);
   initialize();
}
/**
 * Return the textfield0 property value.
 * @return java.awt.TextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private TextField getDoubleTextfield() {
   if (ivjDoubleTextfield == null) {
      try {
         ivjDoubleTextfield = new java.awt.TextField();
         ivjDoubleTextfield.setName("DoubleTextfield");
         ivjDoubleTextfield.setText("0.0");
         ivjDoubleTextfield.setColumns(10);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjDoubleTextfield;
}
/**
 * Return the label0 property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Label getLabel() {
   if (ivjLabel == null) {
      try {
         ivjLabel = new java.awt.Label();
         ivjLabel.setName("Label");
         ivjLabel.setText("Label");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjLabel;
}
public double getValue() {
   if (textValid()) {
```

```
      value = (new Double(getDoubleTextfield().getText())).doubleValue();
   }
   return value;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout
*/
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   // exception.printStackTrace(System.out);
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   // user code begin {1}
   // user code end
   java.awt.GridBagConstraints constraintsLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsDoubleTextfield = new
java.awt.GridBagConstraints();
   setName("DoubleTextFieldPanel");
   setLayout(new java.awt.GridBagLayout());
   setSize(426, 240);

   constraintsLabel.gridx = 1; constraintsLabel.gridy = 1;
   constraintsLabel.gridwidth = 1; constraintsLabel.gridheight = 1;
   constraintsLabel.anchor = java.awt.GridBagConstraints.WEST;
   constraintsLabel.weightx = 0.0;
   constraintsLabel.weighty = 0.0;
   constraintsLabel.insets = new java.awt.Insets(2, 2, 2, 2);
   add(getLabel(), constraintsLabel);

   constraintsDoubleTextfield.gridx = 2; constraintsDoubleTextfield.gridy =
1;
   constraintsDoubleTextfield.gridwidth = 1;
constraintsDoubleTextfield.gridheight = 1;
   constraintsDoubleTextfield.anchor = java.awt.GridBagConstraints.WEST;
   constraintsDoubleTextfield.weightx = 0.0;
   constraintsDoubleTextfield.weighty = 0.0;
   constraintsDoubleTextfield.insets = new java.awt.Insets(2, 2, 2, 2);
   add(getDoubleTextfield(), constraintsDoubleTextfield);
   // user code begin {2}
   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      Frame frame;
      try {
         Class aFrameClass =
Class.forName("com.ibm.uvm.abt.edit.TestFrame");
         frame = (Frame)aFrameClass.newInstance();
      } catch (java.lang.Throwable ivjExc) {
         frame = new Frame();
      }
      DoubleTextFieldPanel aDoubleTextFieldPanel;
      aDoubleTextFieldPanel = new DoubleTextFieldPanel();
      frame.add("Center", aDoubleTextFieldPanel);
      frame.setSize(aDoubleTextFieldPanel.getSize());
      frame.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception occurred in main() of java.awt.Panel");
      exception.printStackTrace(System.out);
   }
}
public void setValue(double newValue) {
   getDoubleTextfield().setText("" + newValue);
}
public boolean textValid() {
   Double tempVal;
   try {
      tempVal = new Double(getDoubleTextfield().getText());
   }
   catch (NumberFormatException e) {
      tempVal = null;
   }
   return (tempVal != null);
}
}
```

UnitLength.java

```
/* written by Brian Chow
 * created March 17, 1998
 * last modified March 23, 1999
 */
/**
 *
 */
public class UnitLength implements Cloneable
{  /**
   *
   */
   private boolean cm;
   private int accuracy;
   public UnitLength(String units,int accuracy)
   {  setUnits(units);
     this.accuracy = accuracy;
   }
   public Object clone()
   {  try
     {  return super.clone();
     } catch (CloneNotSupportedException e)
     {  // this shouldn't happen, since we are Cloneable
       return null;
     }
   }
   public boolean equals(String units)
   {  return this.toString().equals(units);
   }
   public int getAccuracy()
   {  return accuracy;
   }
   public String getUnits()
   {  if (cm)
     {  return "Centimeters";
     }
     else
     {  return "Inches";
```

```
     }
   }
   public double inUnits(double value)
   {  if (cm)
     {  value /= 2.54;
     }
     return value;
   }
   public double outUnits(double value)
   {  if (cm)
     {  value *= 2.54;
     }
     // Rounds value to the nearest accuracy unit.  Note:  the floor
     // function was necessary to avoid an implementation error in the
     // Metrowerks VM for Macintosh dealing with negative values
     // otherwise, the round function could have been used.
     value = (Math.floor(value * accuracy + 0.5)) / (double)accuracy;
     return value;
   }
   public void setAccuracy(int accuracy)
   {  this.accuracy = accuracy;
   }
   public void setUnits(String units)
   {  if (units.equals("Centimeters"))
     {  cm = true;
     }
     else if (units.equals("Inches"))
     {  cm = false;
     }
     else
     {  System.err.println("Length unit error");
       cm = false;
     }
   }
   public String toString()
   {  return getUnits();
   }
}
```

Coordinate.java

```
/* written by Brian Chow
 * created March 7, 1998
 * last modified March 23, 1999
 */

/**
 * Data structure that holds three double precision numbers used to
 * represent a point in rectangular coordinates.
 */

class Coordinate implements Cloneable
{  /**
   * Initializes coordinate to origin for no arguments.
   */
   /**
   * Double precision values for rectangular coordinates.
   */
   double x,y,z;
   public Coordinate()
   {  x = 0.0;
     y = 0.0;
     z = 0.0;
   }
```

```
   /**
   * Initializes coordinate to specified Cartesian coordinates.
   */
   public Coordinate(double x,double y,double z)
   {  this.x = x;
     this.y = y;
     this.z = z;
   }
/**
 * Adds p1 to p2.
 * @return Coordinate
 * @param p1 Coordinate
 * @param p2 Coordinate
 */
public static Coordinate add(Coordinate p1, Coordinate p2) {
   return new Coordinate(p1.x + p2.x, p1.y + p2.y, p1.z + p2.z);
}
   /**
   * Makes a bitwise copy of this point.
   */
   public Object clone()
   {  try
     {  return super.clone();
     } catch (CloneNotSupportedException e)
     {  return null;
     }
   }
/**
 * Evenly distributes points between the endpoints.
 */
public static Coordinate[] constructGrid(Coordinate p1, Coordinate p2,
Coordinate p3, int hNumPoints, int vNumPoints) {
   Coordinate[] points = new Coordinate[hNumPoints * vNumPoints];
   // Calculate the left most vertical column of points.
   Coordinate[] v1Points = constructLine(p1, p3, vNumPoints);
   // Calculate the difference between the first and the last point on
   // a horizontal row.
   Coordinate hDifference = Coordinate.subtract(p2, p1);
   for (int i = 0; i < vNumPoints; i++) {
      Coordinate leftPoint = v1Points[i];
      Coordinate rightPoint = Coordinate.add(leftPoint, hDifference);
      Coordinate[] hPoints = constructLine(leftPoint, rightPoint,
         hNumPoints);
      System.arraycopy(hPoints, 0, points, i * hNumPoints, hNumPoints);
   }
   return points;
}
/**
 * Evenly distributes points between the endpoints.
 */
public static Coordinate[] constructLine(Coordinate p1, Coordinate p2,
   int numPoints) {
   double dx, dy, dz;
   dx = (p2.x - p1.x) / (double) (numPoints - 1);
   dy = (p2.y - p1.y) / (double) (numPoints - 1);
   dz = (p2.z - p1.z) / (double) (numPoints - 1);
   Coordinate[] points = new Coordinate[numPoints];
   for (int i = 0; i < numPoints; i++) {
      points[i] = new Coordinate(p1.x + dx * i, p1.y + dy * i,
         p1.z + dz * i);
   }
   return points;
}
   /**
```

```
   * Calculates the distance between points.
   */
   public double distance(Coordinate c2)
   {  return Math.sqrt(Math.pow(x-c2.x,2) + Math.pow(y-c2.y,2) +
      Math.pow(z-c2.z,2));
   }
   /**
   * Calculates the distance between points projected onto the z plane.
   */
   public double distanceZ(Coordinate c2)
   {  return Math.sqrt(Math.pow(x-c2.x,2) + Math.pow(y-c2.y,2));
   }
   /**
   * Check equality of two points by checking each of the fields.
   */
   public boolean equals(Coordinate p)
   {  return (this.x == p.x) && (this.y == p.y) && (this.z == p.z);
   }
   /**
   * Rotates point in the XY plane by degree angle about point (x,y).
   */
   public void rotateXY(double theta,double aboutX,double aboutY)
   {  double tempX = x - aboutX;
     double tempY = y - aboutY;
     tempX = tempX * Math.cos(theta * Math.PI / 180) -
      tempY * Math.sin(theta * Math.PI / 180);
     tempY = tempX * Math.sin(theta * Math.PI / 180) +
      tempY * Math.cos(theta * Math.PI / 180);
     x = tempX + aboutX;
     y = tempY + aboutY;
   }
/**
 * Subtracts p2 from p1.
 * @return Coordinate
 * @param p1 Coordinate
 * @param p2 Coordinate
 */
public static Coordinate subtract(Coordinate p1, Coordinate p2) {
   return new Coordinate(p1.x - p2.x, p1.y - p2.y, p1.z - p2.z);
}
   /**
   * Converts coordinate to (x,y,z) string.
   */
   public String toString()
   {  return ("(" + x + ", " + y + ", " + z + ")");
   }
   /**
   * Shifts point according to specified X, Y, and Z distances.
   */
   public void translate(double deltaX,double deltaY,double deltaZ)
   {  x += deltaX;
     y += deltaY;
     z += deltaZ;
   }
}
```

TargetPoints.java

```
/* written by Brian Chow
 * created March 20, 1998
 * last modified March 23, 1999
 */

import java.util.*;
```

```
/**
 * Target consisting of a set of points.
 */
class TargetPoints extends Target {
/**
* Sets default point target with one point at the origin.
*/
public TargetPoints() {
   points = new Coordinate[1];
   points[0] = new Coordinate();
}
/**
* Sets point target to the array of points specified.
*/
public TargetPoints(Coordinate[] points) {
   this.points = points;
}
/**
* Sets point target to the Vector of points specified.
*/
public TargetPoints(Vector points) {
   this.points = new Coordinate[points.size()];
   points.copyInto(this.points);
}
/**
* Calculates the dosage at each point in this target and outputs the
* data to a calculation output window.
*/
public void calculate(Fixture currentFixture) { // Do calculations.
   super.calculate(currentFixture);

   // Output to user.
   CalculationOutput out = new CalculationOutput(25, 76);
   out.setTitle("Calculation Results");
   out.println("Calculation date: " + new Day());
   out.println("Fixture " + currentFixture.toString());
   out.println("Target " + this.toString());
   out.println();
   Vector lineToPrint = getCalcColumnHeadings();
   for (int i = 0; i < lineToPrint.size(); i++) {
      out.print((String) lineToPrint.elementAt(i), 25);
   }
   out.println();
   for (int i = 0; i < dose.length; i++) {
      lineToPrint = getCalcResultsRow(i);
      for (int j = 0; j < lineToPrint.size(); j++) {
         out.print((String) lineToPrint.elementAt(j), 25);
      }
      out.println();
   }
   out.println();
   out.println("Calculation time: " + (calculationTime / 1000.0) + " s");
   out.setVisible(true);
}
/**
* Clones this point target.
*/
public Object clone() {
   return super.clone();
}
/**
 * Return a vector of string headings to use to identify
 * calculation results.
```

```
 * @return java.util.Vector
 */
protected Vector getCalcColumnHeadings() {
   Vector returnVal = new Vector(2);
   returnVal.addElement("Coordinate (" + Units.getLengthUnits() + ")");
   returnVal.addElement("Dose (" + Units.getDoseUnits() + "/hour)");
   return returnVal;
}
/**
 * Return a vector of strings for one row of
 * calculation results.
 * @return java.util.Vector
 */
protected Vector getCalcResultsRow(int row) {
   Vector returnVal = new Vector(2);
   returnVal.addElement(Units.outLength(points[row]).toString());
   returnVal.addElement("" + Units.outDose(dose[row]));
   return returnVal;
}
/**
* Returns array of points in this point target.
*/
public Coordinate[] getPoints() {
   return points;
}
/**
* Sets points to the specified array of points.
*/
public void setPoints(Coordinate[] points) {
   this.points = points;
}
/**
* Sets points to the specified Vector of points.
*/
public void setPoints(Vector points) {
   this.points = new Coordinate[points.size()];
   points.copyInto(this.points);
}
/**
* String representation of this point target.
*/
public String toString() {
   return "User specified points";
}
}
```

TargetLine.java

```
/* written by Brian Chow
 * created March 20, 1998
 * last modified March 23, 1999
 */

/**
 * Line target represented by a specified number of points evenly spaced
 * between two endpoints.
 */
class TargetLine extends TargetPoints
{  /**
   * Sets the default line target with two points.  One at the origin
   * and the other at (2,0,0).
   */
   /**
   * Endpoints of the line target.
```

```
   */
   private Coordinate p1,p2;
   /**
   * Number of points for the line target.
   */
   private int numPoints;
   public TargetLine()
   {  numPoints = 2;
     p1 = new Coordinate();
     p2 = new Coordinate(2,0,0);
     points = new Coordinate[numPoints];
     points[0] = p1;
     points[1] = p2;
   }
   /**
   * Sets the line target with the specified endpoints and number of
   * points.
   */
   public TargetLine(Coordinate p1,Coordinate p2,int numPoints)
   {  this.numPoints = numPoints;
     this.p1 = p1;
     this.p2 = p2;
     constructLine();
   }
   /**
   * Clones this line target.
   */
   public Object clone()
   {  return new TargetLine((Coordinate)p1.clone(),
        (Coordinate)p2.clone(),numPoints);
   }
/*
   public static void main(String[] args)
   {  Target t = new TargetLine(new Coordinate(1.5,1.5,1.5),
        new Coordinate(2.5,2.5,2.5),11);
     Units u = new Units();
     RadBundle b = new RadBundle("1979",new Day(1979,9,19),9950,12);
     b.setDate(new Day(1990,10,19));
     b.recalculate();
     Fixture f = new AnnularFixture(new Coordinate(),b,1,12);
     t.calculate(u,f);
   }
*/

   /**
   * Evenly distributes target points between the endpoints.
   */
   private void constructLine()
   {  points = Coordinate.constructLine(p1,p2,numPoints);
   }
   /**
   * Returns the number of points.
   */
   public int getNumPoints()
   {  return numPoints;
   }
   /**
   * Returns the first endpoint.
   */
   public Coordinate getPoint1()
   {  return p1;
   }
   /**
   * Returns the second endpoint.
```

```
   */
   public Coordinate getPoint2()
   {  return p2;
   }
   /**
   * Sets the line target to the specified endpoints and number of
   * points.
   */
   public void setLine(Coordinate point1,Coordinate point2,int numP)
   {  p1 = point1;
     p2 = point2;
     numPoints = numP;
     constructLine();
   }
   /**
   * Returns the string representation of the line target.
   */
   public String toString()
   {  if (numPoints == 1)
     {  return "Line: One point from " + Units.outLength(p1).toString()
          + " to " + Units.outLength(p2).toString();
     }
     else
     {  return "Line: " + numPoints + " points from " +
          Units.outLength(p1).toString() + " to " + Units.outLength(p2).
          toString();
     }
   }
}
```

TargetGradient.java

```
/* written by Brian Chow
 * created December 11, 1998
 * last modified May 19, 1999
 */

import java.lang.reflect.*;

/**
 *
 */
public class TargetGradient extends Target {
   public static final int XY = 8;
   public static final int YZ = 16;
   public static final int ZX = 32;
   protected GradientPlotCalculator gpc = new GradientPlotCalculator();
   protected int plane;
   protected Coordinate p1, p2, p3, center;
   protected double height, width;
   protected int hPoints,vPoints;
   protected int numIndices = 16;
/**
 * This method was created in VisualAge.
 */
public TargetGradient() {
   plane = XY;
   width = 1;
   height = 1;
   hPoints = 51;
   vPoints = 51;
   center = new Coordinate();
}
/**
```

```
* Calculates the dosage at each point in this target and outputs the
* data to a calculation output window.
*/
public void calculate(Fixture currentFixture) {
   // Make sure the points have been calculated
   if (points == null) {
      calculatePoints();
   }

   // Do calculations.
   super.calculate(currentFixture);

   // Output to user.
   gpc.setNumIndices(numIndices);
   gpc.setMaxMin(new double[] {0, 0});
   gpc.setWidth(hPoints);
   gpc.setHeight(vPoints);
   gpc.setData(dose);
   GradientPlot gp = new GradientPlot(gpc);
   gp.setVisible(true);
}
/**
 * This method was created in VisualAge.
 */
private void calculatePoints() {
   points = Coordinate.constructGrid(p1, p2, p3, hPoints, vPoints);
}
/**
 * This method was created in VisualAge.
 * @param plane int
 * @param center Coordinate
 * @param width double
 * @param height double
 */
public void setDimensions(int plane, Coordinate center, double width,
double height) {
   points = null;
   this.plane = plane;
   this.center = center;
   this.width = width;
   this.height = height;
   this.hPoints = 82;
   this.vPoints = 82;
   if (plane == XY) {
      p1 = (Coordinate)center.clone();
      p1.translate(-width / 2.0, height / 2.0, 0.0);
      p2 = (Coordinate)p1.clone();
      p2.translate(width, 0.0, 0.0);
      p3 = (Coordinate)p1.clone();
      p3.translate(0.0, -height, 0.0);
   }
   else if (plane == YZ) {
      p1 = (Coordinate)center.clone();
      p1.translate(0.0, -width / 2.0, height / 2.0);
      p2 = (Coordinate)p1.clone();
      p2.translate(0.0, width, 0.0);
      p3 = (Coordinate)p1.clone();
      p3.translate(0.0, 0.0, -height);
   }
   else if (plane == ZX) {
      p1 = (Coordinate)center.clone();
      p1.translate(height / 2.0, 0, -width / 2.0);
      p2 = (Coordinate)p1.clone();
      p2.translate(0.0, 0.0, width);
```

```
      p3 = (Coordinate)p1.clone();
      p3.translate(-height, 0.0, 0.0);
   }
   else {
      throw new IllegalArgumentException("Invalid plane designation");
   }
}
/**
 * This method was created in VisualAge.
 * @param plane String
 * @param center Coordinate
 * @param width double
 * @param height double
 */
public void setDimensions(String plane, Coordinate center, double width,
double height) {
   int intPlane;
   try {
      intPlane = this.getClass().getField(plane).getInt(this);
   }
   catch (Exception e) {
      intPlane = XY;
   }
   setDimensions(intPlane, center, width, height);
}
/**
 * Returns a String that represents the value of this object.
 * @return a string representation of the receiver
 */
public String toString() {
   // Insert code to print the receiver here.
   // This implementation forwards the message to super. You may replace or
supplement this.
   return super.toString();
}
}
```

GradientPlot.java

```
/* written by Brian Chow
 * created November 14, 1998
 * last modified February 12, 1999
 */

import java.awt.*;
import java.awt.event.*;
/**
 *
 */
public class GradientPlot extends Frame implements WindowListener,
java.beans.PropertyChangeListener {
   private Panel ivjContentsPane = null;
   private GradientPlotLegendCanvas ivjLegendCanvas = null;
   private Label ivjLegendLabel = null;
   private GradientPlotFieldCanvas ivjPlotCanvas = null;
   private GradientPlotCalculator ivjGradientPlotCalculator = null;
   private Label ivjPlotLabel = null;
   private TextField ivjHorizontalCoordinateTextField = null;
   private Panel ivjPlotCoordinatePanel = null;
   private TextField ivjVerticalCoordinateTextField = null;
/**
 * Constructor
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
public GradientPlot() {
   super();
   initialize();
}
/**
 * This method was created in VisualAge.
 * @param gpc GradientPlotCalculator
 */
public GradientPlot(GradientPlotCalculator gpc) {
   ivjGradientPlotCalculator = gpc;
   initialize();
}
/**
 * connEtoC1:
(GradientPlot.window.windowClosing(java.awt.event.WindowEvent) -->
GradientPlot.dispose()V)
 * @param arg1 java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoC1(WindowEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.dispose();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connPtoP1SetTarget:  (LegendCanvas.this <-->
GradientPlotCalculator.legendCanvas)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP1SetTarget() {
   /* Set the target from the source */
   try {
      getGradientPlotCalculator().setLegendCanvas(getLegendCanvas());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connPtoP2SetTarget:  (PlotCanvas.this <-->
GradientPlotCalculator.fieldCanvas)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP2SetTarget() {
   /* Set the target from the source */
   try {
      getGradientPlotCalculator().setFieldCanvas(getPlotCanvas());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
```

```
   }
}
/**
 * connPtoP3SetTarget:  (PlotCanvas.horizontalCoordinate <-->
HorizontalCoordinateTextField.text)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP3SetTarget() {
   /* Set the target from the source */
   try {

   getHorizontalCoordinateTextField().setText(getPlotCanvas().getHorizontal
Coordinate());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connPtoP4SetTarget:  (PlotCanvas.verticalCoordinate <-->
VerticalCoordinateTextField.text)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP4SetTarget() {
   /* Set the target from the source */
   try {

   getVerticalCoordinateTextField().setText(getPlotCanvas().getVerticalCoor
dinate());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * Return the ContentsPane property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Panel getContentsPane() {
   java.awt.GridBagConstraints constraintsPlotCanvas = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsLegendCanvas = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsLegendLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsPlotLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsPlotCoordinatePanel = new
java.awt.GridBagConstraints();
   if (ivjContentsPane == null) {
      try {
         ivjContentsPane = new java.awt.Panel();
         ivjContentsPane.setName("ContentsPane");
         ivjContentsPane.setLayout(new java.awt.GridBagLayout());

         constraintsPlotCanvas.gridx = 0; constraintsPlotCanvas.gridy = 1;
         constraintsPlotCanvas.gridwidth = 1;
constraintsPlotCanvas.gridheight = 1;
         constraintsPlotCanvas.fill = java.awt.GridBagConstraints.BOTH;
         constraintsPlotCanvas.anchor = java.awt.GridBagConstraints.CENTER;
         constraintsPlotCanvas.weightx = 75.0;
         constraintsPlotCanvas.weighty = 100.0;
         constraintsPlotCanvas.insets = new java.awt.Insets(5, 5, 5, 5);
         getContentsPane().add(getPlotCanvas(), constraintsPlotCanvas);

         constraintsLegendCanvas.gridx = 1; constraintsLegendCanvas.gridy =
1;
         constraintsLegendCanvas.gridwidth = 0;
constraintsLegendCanvas.gridheight = 1;
         constraintsLegendCanvas.fill = java.awt.GridBagConstraints.BOTH;
         constraintsLegendCanvas.anchor =
java.awt.GridBagConstraints.NORTH;
         constraintsLegendCanvas.weightx = 0.0;
         constraintsLegendCanvas.weighty = 100.0;
         constraintsLegendCanvas.insets = new java.awt.Insets(5, 5, 5, 5);
         getContentsPane().add(getLegendCanvas(), constraintsLegendCanvas);

         constraintsLegendLabel.gridx = 1; constraintsLegendLabel.gridy =
0;
         constraintsLegendLabel.gridwidth = 1;
constraintsLegendLabel.gridheight = 1;
         constraintsLegendLabel.fill = java.awt.GridBagConstraints.BOTH;
         constraintsLegendLabel.anchor = java.awt.GridBagConstraints.WEST;
         constraintsLegendLabel.weightx = 0.0;
         constraintsLegendLabel.weighty = 0.0;
         constraintsLegendLabel.insets = new java.awt.Insets(5, 5, 5, 5);
         getContentsPane().add(getLegendLabel(), constraintsLegendLabel);

         constraintsPlotLabel.gridx = 0; constraintsPlotLabel.gridy = 0;
         constraintsPlotLabel.gridwidth = 1;
constraintsPlotLabel.gridheight = 1;
         constraintsPlotLabel.anchor = java.awt.GridBagConstraints.CENTER;
         constraintsPlotLabel.weightx = 0.0;
         constraintsPlotLabel.weighty = 0.0;
         constraintsPlotLabel.insets = new java.awt.Insets(5, 5, 5, 5);
         getContentsPane().add(getPlotLabel(), constraintsPlotLabel);

         constraintsPlotCoordinatePanel.gridx = 0;
constraintsPlotCoordinatePanel.gridy = 2;
         constraintsPlotCoordinatePanel.gridwidth = 1;
constraintsPlotCoordinatePanel.gridheight = 0;
         constraintsPlotCoordinatePanel.anchor =
java.awt.GridBagConstraints.WEST;
         constraintsPlotCoordinatePanel.weightx = 0.0;
         constraintsPlotCoordinatePanel.weighty = 0.0;
         getContentsPane().add(getPlotCoordinatePanel(),
constraintsPlotCoordinatePanel);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjContentsPane;
}
/**
 * Return the GradientPlotCalculator property value.
 * @return GradientPlotCalculator
```

```
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private GradientPlotCalculator getGradientPlotCalculator() {
   if (ivjGradientPlotCalculator == null) {
      try {
         ivjGradientPlotCalculator = new GradientPlotCalculator();
         ivjGradientPlotCalculator.setHeight(4);
         ivjGradientPlotCalculator.setNumIndices(10);
         double ivjLocal0maxMin [] = {
            110.0,
            10.0};
         ivjGradientPlotCalculator.setMaxMin(ivjLocal0maxMin);
         ivjGradientPlotCalculator.setWidth(4);
         double ivjLocal0data [] = {
            10.0,
            20.0,
            30.0,
            40.0,
            50.0,
            40.0,
            30.0,
            20.0,
            10.0,
            100.0,
            90.0,
            80.0,
            70.0,
            60.0,
            50.0,
            40.0};
         ivjGradientPlotCalculator.setData(ivjLocal0data);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjGradientPlotCalculator;
}
/**
 * Return the HorizontalCoordinateTextField property value.
 * @return java.awt.TextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private TextField getHorizontalCoordinateTextField() {
   if (ivjHorizontalCoordinateTextField == null) {
      try {
         ivjHorizontalCoordinateTextField = new java.awt.TextField();

   ivjHorizontalCoordinateTextField.setName("HorizontalCoordinateTextField"
);
         ivjHorizontalCoordinateTextField.setEditable(false);
         ivjHorizontalCoordinateTextField.setEnabled(false);
         ivjHorizontalCoordinateTextField.setColumns(10);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
```

```
   return ivjHorizontalCoordinateTextField;
}
/**
 * Return the LegendCanvas property value.
 * @return GradientPlotLegendCanvas
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private GradientPlotLegendCanvas getLegendCanvas() {
   if (ivjLegendCanvas == null) {
      try {
         ivjLegendCanvas = new GradientPlotLegendCanvas();
         ivjLegendCanvas.setName("LegendCanvas");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjLegendCanvas;
}
/**
 * Return the LegendLabel property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Label getLegendLabel() {
   if (ivjLegendLabel == null) {
      try {
         ivjLegendLabel = new java.awt.Label();
         ivjLegendLabel.setName("LegendLabel");
         ivjLegendLabel.setText("Legend");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjLegendLabel;
}
/**
 * Return the PlotCanvas property value.
 * @return GradientPlotFieldCanvas
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private GradientPlotFieldCanvas getPlotCanvas() {
   if (ivjPlotCanvas == null) {
      try {
         ivjPlotCanvas = new GradientPlotFieldCanvas();
         ivjPlotCanvas.setName("PlotCanvas");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlotCanvas;
}
/**
```

```
 * Return the PlotCoordinatePanel property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Panel getPlotCoordinatePanel() {
   java.awt.GridBagConstraints constraintsHorizontalCoordinateTextField =
new java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsVerticalCoordinateTextField = new
java.awt.GridBagConstraints();
   if (ivjPlotCoordinatePanel == null) {
      try {
         ivjPlotCoordinatePanel = new java.awt.Panel();
         ivjPlotCoordinatePanel.setName("PlotCoordinatePanel");
         ivjPlotCoordinatePanel.setLayout(new java.awt.GridBagLayout());

         constraintsHorizontalCoordinateTextField.gridx = 1;
constraintsHorizontalCoordinateTextField.gridy = 1;
         constraintsHorizontalCoordinateTextField.gridwidth = 1;
constraintsHorizontalCoordinateTextField.gridheight = 1;
         constraintsHorizontalCoordinateTextField.anchor =
java.awt.GridBagConstraints.CENTER;
         constraintsHorizontalCoordinateTextField.weightx = 0.0;
         constraintsHorizontalCoordinateTextField.weighty = 0.0;
         constraintsHorizontalCoordinateTextField.insets = new
java.awt.Insets(5, 5, 5, 2);
         getPlotCoordinatePanel().add(getHorizontalCoordinateTextField(),
constraintsHorizontalCoordinateTextField);

         constraintsVerticalCoordinateTextField.gridx = 2;
constraintsVerticalCoordinateTextField.gridy = 1;
         constraintsVerticalCoordinateTextField.gridwidth = 1;
constraintsVerticalCoordinateTextField.gridheight = 1;
         constraintsVerticalCoordinateTextField.anchor =
java.awt.GridBagConstraints.CENTER;
         constraintsVerticalCoordinateTextField.weightx = 0.0;
         constraintsVerticalCoordinateTextField.weighty = 0.0;
         constraintsVerticalCoordinateTextField.insets = new
java.awt.Insets(5, 3, 5, 5);
         getPlotCoordinatePanel().add(getVerticalCoordinateTextField(),
constraintsVerticalCoordinateTextField);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlotCoordinatePanel;
}
/**
 * Return the PlotLabel property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Label getPlotLabel() {
   if (ivjPlotLabel == null) {
      try {
         ivjPlotLabel = new java.awt.Label();
         ivjPlotLabel.setName("PlotLabel");
         ivjPlotLabel.setText("");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlotLabel;
}
/**
 * Method generated to support the promotion of the plotLabelText
attribute.
 * @return java.lang.String
 */
public String getPlotLabelText() {
   return getPlotLabel().getText();
}
/**
 * Return the VerticalCoordinateTextField property value.
 * @return java.awt.TextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private TextField getVerticalCoordinateTextField() {
   if (ivjVerticalCoordinateTextField == null) {
      try {
         ivjVerticalCoordinateTextField = new java.awt.TextField();

   ivjVerticalCoordinateTextField.setName("VerticalCoordinateTextField");
         ivjVerticalCoordinateTextField.setEditable(false);
         ivjVerticalCoordinateTextField.setEnabled(false);
         ivjVerticalCoordinateTextField.setColumns(10);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjVerticalCoordinateTextField;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout
*/
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   exception.printStackTrace(System.out);
}
/**
 * Initializes connections
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initConnections() {
   // user code begin {1}
   // user code end
   this.addWindowListener(this);
   getPlotCanvas().addPropertyChangeListener(this);
   connPtoP1SetTarget();
   connPtoP2SetTarget();
   connPtoP3SetTarget();
   connPtoP4SetTarget();
}
```

```
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   // user code begin {1}
   // user code end
   setName("GradientPlot");
   setLayout(new java.awt.BorderLayout());
   setSize(555, 447);
   setTitle("Gradient Plot");
   add(getContentsPane(), "Center");
   initConnections();
   // user code begin {2}
   pack();
   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   try {
      GradientPlot aGradientPlot;
      aGradientPlot = new GradientPlot();
      try {
         Class aCloserClass =
Class.forName("com.ibm.uvm.abt.edit.WindowCloser");
         Class parmTypes[] = {java.awt.Window.class};
         Object parms[] = {aGradientPlot};
         java.lang.reflect.Constructor aCtor =
aCloserClass.getConstructor(parmTypes);
         aCtor.newInstance(parms);
      }
      catch (java.lang.Throwable exc) {
      };
      aGradientPlot.setVisible(true);
   }
   catch (Throwable exception) {
      System.err.println("Exception occurred in main() of java.awt.Frame");
      exception.printStackTrace(System.out);
   }
}
/**
 * Method to handle events for the PropertyChangeListener interface.
 * @param evt java.beans.PropertyChangeEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void propertyChange(java.beans.PropertyChangeEvent evt) {
   // user code begin {1}
   // user code end
   if ((evt.getSource() == getPlotCanvas()) &&
(evt.getPropertyName().equals("horizontalCoordinate"))) {
      connPtoP3SetTarget();
   }
   if ((evt.getSource() == getPlotCanvas()) &&
(evt.getPropertyName().equals("verticalCoordinate"))) {
      connPtoP4SetTarget();
   }
   // user code begin {2}
   // user code end
}
/**
```

```
 * Method generated to support the promotion of the plotLabelText
attribute.
 * @param arg1 java.lang.String
 */
public void setPlotLabelText(String arg1) {
   getPlotLabel().setText(arg1);
}
/**
 * windowActivated method comment.
 */
public void windowActivated(WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * windowClosed method comment.
 */
public void windowClosed(WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * windowClosing method comment.
 */
public void windowClosing(WindowEvent e) {
   // user code begin {1}
   // user code end
   if ((e.getSource() == this) ) {
      connEtoC1(e);
   }
   // user code begin {2}
   // user code end
}
/**
 * windowDeactivated method comment.
 */
public void windowDeactivated(WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * windowDeiconified method comment.
 */
public void windowDeiconified(WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * windowIconified method comment.
 */
public void windowIconified(WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
```

```
/**
 * windowOpened method comment.
 */
public void windowOpened(WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
}
```

GradientPlotFieldCanvas.java

```
/* written by Brian Chow
 * created November 15, 1998
 * last modified February 12, 1999
 */

import java.awt.*;
import java.awt.event.*;
/**
 * This type was created in VisualAge.
 */
public class GradientPlotFieldCanvas extends GradientPlotCanvas implements
MouseListener {
   protected static final int MINPIXELWIDTH = 5;
   protected static final int MINPIXELHEIGHT = 5;
   protected static final int PREFERREDWIDTH = 450;
   protected static final int PREFERREDHEIGHT = 450;
   protected int[] data;
   protected int width,height;
   protected FontMetrics fm;
   private int textOffset;
   private int pixelWidth, pixelHeight;
   private String fieldHorizontalCoordinate = new String();
   private String fieldVerticalCoordinate = new String();
   protected transient java.beans.PropertyChangeSupport propertyChange;
/**
 * This method was created in VisualAge.
 */
public GradientPlotFieldCanvas() {
   super();
   setFont(new Font("SanSerif", Font.PLAIN, 12));
   fm = getToolkit().getFontMetrics(getFont());
   final int height = fm.getHeight();
   textOffset = height;
   addMouseListener(this);
}
/**
 * The addPropertyChangeListener method was generated to support the
propertyChange field.
 */
public synchronized void
addPropertyChangeListener(java.beans.PropertyChangeListener listener) {
   getPropertyChange().addPropertyChangeListener(listener);
}
/**
 * This method was created in VisualAge.
 */
protected void checkDimensions() {
   if (data != null && data.length == width * height) {
      setMinimumSize();
   }
   else {
      setMinimumSize(0, 0);
   }
}
/**
 * The firePropertyChange method was generated to support the
propertyChange field.
 */
public void firePropertyChange(String propertyName, Object oldValue, Object
newValue) {
   getPropertyChange().firePropertyChange(propertyName, oldValue,
newValue);
}
/**
 * This method was created in VisualAge.
 * @return int[]
 */
public int[] getData() {
   return data;
}
/**
 * Returns height in terms of the number of data points.
 * @return int
 */
public int getHeight() {
   return height;
}
/**
 * Gets the horizontalCoordinate property (java.lang.String) value.
 * @return The horizontalCoordinate property value.
 * @see #setHorizontalCoordinate
 */
public String getHorizontalCoordinate() {
   return fieldHorizontalCoordinate;
}
/**
 * Accessor for the propertyChange field.
 */
protected java.beans.PropertyChangeSupport getPropertyChange() {
   if (propertyChange == null) {
      propertyChange = new java.beans.PropertyChangeSupport(this);
   };
   return propertyChange;
}
/**
 * Gets the verticalCoordinate property (java.lang.String) value.
 * @return The verticalCoordinate property value.
 * @see #setVerticalCoordinate
 */
public String getVerticalCoordinate() {
   return fieldVerticalCoordinate;
}
/**
 * Returns width in terms of the number of data points.
 * @return int
 */
public int getWidth() {
   return width;
}
/**
 * This method was created in VisualAge.
 * @param e java.awt.event.MouseEvent
 */
public void mouseClicked(MouseEvent e) {
}
```

```
/**
 * This method was created in VisualAge.
 * @param e java.awt.event.MouseEvent
 */
public void mouseEntered(MouseEvent e) {
}
/**
 * This method was created in VisualAge.
 * @param e java.awt.event.MouseEvent
 */
public void mouseExited(MouseEvent e) {
}
/**
 * This method was created in VisualAge.
 * @param e java.awt.event.MouseEvent
 */
public void mousePressed(MouseEvent e) {
   if (colors == null || data == null || pixelWidth == 0 || pixelHeight ==
0) {
      return;
   }
   // Get the clicked point
   Point p = e.getPoint();
   setHorizontalCoordinate("" + p.x / pixelWidth);
   setVerticalCoordinate("" + p.y / pixelHeight);
}
/**
 * This method was created in VisualAge.
 * @param e java.awt.event.MouseEvent
 */
public void mouseReleased(MouseEvent e) {
}
/**
 * This method was created in VisualAge.
 * @param g java.awt.Graphics
 */
public void paint(Graphics g) {
   if (colors == null || data == null) {
      return;
   }
   final Dimension size = getSize();
   pixelWidth = size.width / width;
   pixelHeight = size.height / height;
   int pixelSize;
   if (pixelWidth < pixelHeight) {
      pixelSize = pixelWidth;
   }
   else {
      pixelSize = pixelHeight;
   }
   for (int h = 0; h < height; h++) {
      for (int w = 0; w < width; w++) {
         g.setColor(colors[data[h * width + w]]);
         g.fillRect(pixelSize * w, pixelSize * h, pixelSize, pixelSize);
      }
   }
   g.dispose();
}
/**
 * The removePropertyChangeListener method was generated to support the
propertyChange field.
 */
public synchronized void
removePropertyChangeListener(java.beans.PropertyChangeListener listener) {
```

```
   getPropertyChange().removePropertyChangeListener(listener);
}
/**
 * This method was created in VisualAge.
 * @param data int[]
 */
public void setData(int[] data) {
   this.data = data;
   checkDimensions();
}
/**
 * This method was created in VisualAge.
 * @param height int
 */
public void setHeight(int height) {
   this.height = height;
   checkDimensions();
}
/**
 * Sets the horizontalCoordinate property (java.lang.String) value.
 * @param horizontalCoordinate The new value for the property.
 * @see #getHorizontalCoordinate
 */
protected void setHorizontalCoordinate(String horizontalCoordinate) {
   String oldValue = fieldHorizontalCoordinate;
   fieldHorizontalCoordinate = horizontalCoordinate;
   firePropertyChange("horizontalCoordinate", oldValue,
horizontalCoordinate);
}
/**
 * This method was created in VisualAge.
 */
protected void setMinimumSize() {
   int minWidth = width * MINPIXELWIDTH;
   int minHeight = height * MINPIXELHEIGHT;
   minWidth = minWidth < PREFERREDWIDTH ? PREFERREDWIDTH : minWidth;
   minHeight = minHeight < PREFERREDHEIGHT ? PREFERREDHEIGHT : minHeight;
   setMinimumSize(minWidth, minHeight);
}
/**
 * Sets the verticalCoordinate property (java.lang.String) value.
 * @param verticalCoordinate The new value for the property.
 * @see #getVerticalCoordinate
 */
protected void setVerticalCoordinate(String verticalCoordinate) {
   String oldValue = fieldVerticalCoordinate;
   fieldVerticalCoordinate = verticalCoordinate;
   firePropertyChange("verticalCoordinate", oldValue, verticalCoordinate);
}
/**
 * This method was created in VisualAge.
 * @param width int
 */
public void setWidth(int width) {
   this.width = width;
   checkDimensions();
}
}
```

GradientDialog.java

```
/* written by Brian Chow
 * created November 21, 1998
 * last modified March 23, 1999
```

```
 */
/**
 *
 */
public class GradientDialog extends java.awt.Dialog implements
java.awt.event.ActionListener, java.awt.event.WindowListener {
   private java.awt.Panel ivjContentsPane = null;
   private java.awt.CheckboxGroup ivjPlaneCheckboxGroup = null;
   private java.awt.Label ivjPlaneLabel = null;
   private java.awt.Panel ivjPlanePanel = null;
   private java.awt.Checkbox ivjXYCheckbox = null;
   private java.awt.Checkbox ivjYZCheckbox = null;
   private java.awt.Checkbox ivjZXCheckbox = null;
   private java.awt.Label ivjCenterLabel = null;
   private java.awt.Panel ivjCenterPanel = null;
   private java.awt.FlowLayout ivjPlanePanelFlowLayout = null;
   private java.awt.Label ivjSizeLabel = null;
   private java.awt.TextField ivjSizeTextField = null;
   private java.awt.Panel ivjButtonPanel = null;
   private java.awt.Button ivjPlotButton = null;
   private java.awt.Button ivjCancelButton = null;
   private TargetGradient ivjTargetGradient = null;
   private CoordinateTextFieldPanel ivjCenterTextFieldPanel = null;
   private Fixture fieldFixture = new Fixture();
   private java.awt.Checkbox ivjPlaneSelectedCheckbox = null;
   private java.awt.Label ivjSizeUnitsLabel = null;
   private Units ivjUnits = null;
   private java.awt.Panel ivjSizePanel = null;
   private Double ivjConvertedSizeDouble = null;
   private java.awt.FlowLayout ivjSizePanelFlowLayout = null;
/**
 * Constructor
 * @param parent Symbol
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public GradientDialog(java.awt.Frame parent) {
   super(parent);
   initialize();
}
/**
 * GradientDialog constructor comment.
 * @param parent java.awt.Frame
 * @param title java.lang.String
 */
public GradientDialog(java.awt.Frame parent, String title) {
   super(parent, title);
}
/**
 * GradientDialog constructor comment.
 * @param parent java.awt.Frame
 * @param title java.lang.String
 * @param modal boolean
 */
public GradientDialog(java.awt.Frame parent, String title, boolean modal) {
   super(parent, title, modal);
}
/**
 * GradientDialog constructor comment.
 * @param parent java.awt.Frame
 * @param modal boolean
 */
public GradientDialog(java.awt.Frame parent, boolean modal) {
   super(parent, modal);
}
```

```
/**
 * Method to handle events for the ActionListener interface.
 * @param e java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void actionPerformed(java.awt.event.ActionEvent e) {
   // user code begin {1}
   // user code end
   if ((e.getSource() == getCancelButton()) ) {
      connEtoM1(e);
   }
   if ((e.getSource() == getPlotButton()) ) {
      connEtoM6(e);
   }
   if ((e.getSource() == getPlotButton()) ) {
      connEtoM9(e);
   }
   if ((e.getSource() == getPlotButton()) ) {
      connEtoM4(e);
   }
   if ((e.getSource() == getPlotButton()) ) {
      connEtoM2(e);
   }
   if ((e.getSource() == getPlotButton()) ) {
      connEtoM3(e);
   }
   // user code begin {2}
   // user code end
}
/**
 * connEtoC1:
(GradientDialog.window.windowClosing(java.awt.event.WindowEvent) -->
GradientDialog.dispose()V)
 * @param arg1 java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoC1(java.awt.event.WindowEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.dispose();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM1:  (Button2.action.actionPerformed(java.awt.event.ActionEvent)
--> GradientDialog.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM1(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.dispose();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
```

```
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM10:  (
(PlotButton,action.actionPerformed(java.awt.event.ActionEvent) -->
Units,outLength(D)D).normalResult --> ConvertedSizeDouble.this)
 * @param result double
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM10(double result) {
   try {
      // user code begin {1}
      // user code end
      setConvertedSizeDouble(new java.lang.Double(result));
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM11:  (GradientDialog.initialize() --> Units.outLength(D)D)
 * @return double
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private double connEtoM11() {
   double connEtoM11Result = 0;
   try {
      // user code begin {1}
      // user code end
      connEtoM11Result = Units.outLength(5.0);
      connEtoM12(connEtoM11Result);
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
   return connEtoM11Result;
}
/**
 * connEtoM12:  ( (GradientDialog,initialize() -->
Units,outLength(D)D).normalResult --> SizeTextField.text)
 * @param result double
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM12(double result) {
   try {
      // user code begin {1}
      // user code end
      getSizeTextField().setText(String.valueOf(result));
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
```

```
/**
 * connEtoM2:
(PlotButton.action.actionPerformed(java.awt.event.ActionEvent) -->
TargetGradient.setDimensions(Ljava.lang.String;LCoordinate;DD)V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM2(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end

   getTargetGradient().setDimensions(getPlaneSelectedCheckbox().getLabel(),
getCenterTextFieldPanel().getCoordinate(),
(getConvertedSizeDouble()).doubleValue(),
(getConvertedSizeDouble()).doubleValue());
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM3:
(PlotButton.action.actionPerformed(java.awt.event.ActionEvent) -->
TargetGradient.calculate(LFixture;)V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM3(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      getTargetGradient().calculate(this.getFixture());
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM4:
(PlotButton.action.actionPerformed(java.awt.event.ActionEvent) -->
PlaneCheckboxGroup.getSelectedCheckbox()Ljava.awt.Checkbox;)
 * @return java.awt.Checkbox
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Checkbox connEtoM4(java.awt.event.ActionEvent arg1) {
   java.awt.Checkbox connEtoM4Result = null;
   try {
      // user code begin {1}
      // user code end
      connEtoM4Result = getPlaneCheckboxGroup().getSelectedCheckbox();
      connEtoM5(connEtoM4Result);
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
```

```
      handleException(ivjExc);
   }
   return connEtoM4Result;
}
/**
 * connEtoM5:  (
(PlotButton,action.actionPerformed(java.awt.event.ActionEvent) -->
PlaneCheckboxGroup,getSelectedCheckbox()Ljava.awt.Checkbox;).normalResult -
-> PlaneSelectedCheckbox.this)
 * @param result java.awt.Checkbox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM5(java.awt.Checkbox result) {
   try {
      // user code begin {1}
      // user code end
      setPlaneSelectedCheckbox(result);
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM6:
(PlotButton.action.actionPerformed(java.awt.event.ActionEvent) -->
GradientDialog.dispose()V)
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM6(java.awt.event.ActionEvent arg1) {
   try {
      // user code begin {1}
      // user code end
      this.dispose();
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM7:  (GradientDialog.initialize() -->
Units.getLengthUnits()LUnitLength;)
 * @return UnitLength
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private UnitLength connEtoM7() {
   UnitLength connEtoM7Result = null;
   try {
      // user code begin {1}
      // user code end
      connEtoM7Result = Units.getLengthUnits();
      connEtoM8(connEtoM7Result);
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
   return connEtoM7Result;
}
/**
 * connEtoM8:  ( (GradientDialog,initialize() -->
Units,getLengthUnits()LUnitLength;).normalResult -->
SizeUnitsLabelString.this)
 * @param result UnitLength
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connEtoM8(UnitLength result) {
   try {
      // user code begin {1}
      // user code end
      getSizeUnitsLabel().setText(String.valueOf(result));
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connEtoM9:
(PlotButton.action.actionPerformed(java.awt.event.ActionEvent) -->
Units.outLength(D)D)
 * @return double
 * @param arg1 java.awt.event.ActionEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private double connEtoM9(java.awt.event.ActionEvent arg1) {
   double connEtoM9Result = 0;
   try {
      // user code begin {1}
      // user code end
      connEtoM9Result = Units.inLength(new
Double(getSizeTextField().getText()).doubleValue());
      connEtoM10(connEtoM9Result);
      // user code begin {2}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
   return connEtoM9Result;
}
/**
 * connPtoP1SetTarget:  (PlaneCheckboxGroup.this <-->
XYCheckbox.checkboxGroup)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP1SetTarget() {
   /* Set the target from the source */
   try {
      getXYCheckbox().setCheckboxGroup(getPlaneCheckboxGroup());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
```

```
}
/**
 * connPtoP2SetTarget:  (PlaneCheckboxGroup.this <-->
YZCheckbox.checkboxGroup)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP2SetTarget() {
   /* Set the target from the source */
   try {
      getYZCheckbox().setCheckboxGroup(getPlaneCheckboxGroup());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connPtoP3SetTarget:  (PlaneCheckboxGroup.this <-->
ZXCheckbox.checkboxGroup)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP3SetTarget() {
   /* Set the target from the source */
   try {
      getZXCheckbox().setCheckboxGroup(getPlaneCheckboxGroup());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * connPtoP4SetTarget:  (XYCheckbox.this <-->
PlaneCheckboxGroup.selectedCheckbox)
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void connPtoP4SetTarget() {
   /* Set the target from the source */
   try {
      getPlaneCheckboxGroup().setSelectedCheckbox(getXYCheckbox());
      // user code begin {1}
      // user code end
   } catch (java.lang.Throwable ivjExc) {
      // user code begin {3}
      // user code end
      handleException(ivjExc);
   }
}
/**
 * Return the ButtonPanel property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Panel getButtonPanel() {
   if (ivjButtonPanel == null) {
      try {
         ivjButtonPanel = new java.awt.Panel();
         ivjButtonPanel.setName("ButtonPanel");
         ivjButtonPanel.setLayout(new java.awt.FlowLayout());
         ivjButtonPanel.add(getPlotButton());
```

```
         getButtonPanel().add(getCancelButton(),
getCancelButton().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjButtonPanel;
}
/**
 * Return the Button2 property value.
 * @return java.awt.Button
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Button getCancelButton() {
   if (ivjCancelButton == null) {
      try {
         ivjCancelButton = new java.awt.Button();
         ivjCancelButton.setName("CancelButton");
         ivjCancelButton.setLabel("Cancel");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjCancelButton;
}
/**
 * Return the CenterLabel property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Label getCenterLabel() {
   if (ivjCenterLabel == null) {
      try {
         ivjCenterLabel = new java.awt.Label();
         ivjCenterLabel.setName("CenterLabel");
         ivjCenterLabel.setText("Center");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjCenterLabel;
}
/**
 * Return the CenterPanel property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Panel getCenterPanel() {
   if (ivjCenterPanel == null) {
      try {
         ivjCenterPanel = new java.awt.Panel();
         ivjCenterPanel.setName("CenterPanel");
```

```
         ivjCenterPanel.setLayout(null);
         getCenterPanel().add(getCenterTextFieldPanel(),
getCenterTextFieldPanel().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjCenterPanel;
}
/**
 * Return the CenterTextFieldPanel property value.
 * @return CoordinateTextFieldPanel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private CoordinateTextFieldPanel getCenterTextFieldPanel() {
   if (ivjCenterTextFieldPanel == null) {
      try {
         ivjCenterTextFieldPanel = new CoordinateTextFieldPanel();
         ivjCenterTextFieldPanel.setName("CenterTextFieldPanel");
         ivjCenterTextFieldPanel.setBounds(0, 0, 250, 100);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjCenterTextFieldPanel;
}
/**
 * Return the ContentsPane property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Panel getContentsPane() {
   java.awt.GridBagConstraints constraintsPlaneLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsPlanePanel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsSizeLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsSizePanel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsCenterLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsCenterPanel = new
java.awt.GridBagConstraints();
   if (ivjContentsPane == null) {
      try {
         ivjContentsPane = new java.awt.Panel();
         ivjContentsPane.setName("ContentsPane");
         ivjContentsPane.setLayout(new java.awt.GridBagLayout());

         constraintsPlaneLabel.gridx = 0; constraintsPlaneLabel.gridy = 0;
         constraintsPlaneLabel.gridwidth = 1;
constraintsPlaneLabel.gridheight = 1;
         constraintsPlaneLabel.anchor =
java.awt.GridBagConstraints.NORTHWEST;
         constraintsPlaneLabel.weightx = 0.0;
         constraintsPlaneLabel.weighty = 0.0;
         getContentsPane().add(getPlaneLabel(), constraintsPlaneLabel);

         constraintsPlanePanel.gridx = 1; constraintsPlanePanel.gridy = 0;
         constraintsPlanePanel.gridwidth = 1;
constraintsPlanePanel.gridheight = 1;
         constraintsPlanePanel.anchor =
java.awt.GridBagConstraints.NORTHWEST;
         constraintsPlanePanel.weightx = 100.0;
         constraintsPlanePanel.weighty = 0.0;
         getContentsPane().add(getPlanePanel(), constraintsPlanePanel);

         constraintsSizeLabel.gridx = 0; constraintsSizeLabel.gridy = 1;
         constraintsSizeLabel.gridwidth = 1;
constraintsSizeLabel.gridheight = 1;
         constraintsSizeLabel.anchor =
java.awt.GridBagConstraints.NORTHWEST;
         constraintsSizeLabel.weightx = 0.0;
         constraintsSizeLabel.weighty = 0.0;
         getContentsPane().add(getSizeLabel(), constraintsSizeLabel);

         constraintsSizePanel.gridx = 1; constraintsSizePanel.gridy = 1;
         constraintsSizePanel.gridwidth = 1;
constraintsSizePanel.gridheight = 1;
         constraintsSizePanel.fill = java.awt.GridBagConstraints.BOTH;
         constraintsSizePanel.anchor =
java.awt.GridBagConstraints.NORTHWEST;
         constraintsSizePanel.weightx = 50.0;
         constraintsSizePanel.weighty = 0.0;
         getContentsPane().add(getSizePanel(), constraintsSizePanel);

         constraintsCenterLabel.gridx = 0; constraintsCenterLabel.gridy =
2;
         constraintsCenterLabel.gridwidth = 1;
constraintsCenterLabel.gridheight = 1;
         constraintsCenterLabel.anchor =
java.awt.GridBagConstraints.NORTHWEST;
         constraintsCenterLabel.weightx = 0.0;
         constraintsCenterLabel.weighty = 0.0;
         getContentsPane().add(getCenterLabel(), constraintsCenterLabel);

         constraintsCenterPanel.gridx = 1; constraintsCenterPanel.gridy =
2;
         constraintsCenterPanel.gridwidth = 1;
constraintsCenterPanel.gridheight = 1;
         constraintsCenterPanel.fill = java.awt.GridBagConstraints.BOTH;
         constraintsCenterPanel.anchor =
java.awt.GridBagConstraints.NORTHWEST;
         constraintsCenterPanel.weightx = 100.0;
         constraintsCenterPanel.weighty = 75.0;
         getContentsPane().add(getCenterPanel(), constraintsCenterPanel);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjContentsPane;
}
/**
 * Return the ConvertedSizeString property value.
 * @return java.lang.Double
```

```
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Double getConvertedSizeDouble() {
   // user code begin {1}
   // user code end
   return ivjConvertedSizeDouble;
}
/**
 * Gets the fixture property (Fixture) value.
 * @return The fixture property value.
 * @see #setFixture
 */
public Fixture getFixture() {
   return fieldFixture;
}
/**
 * Return the PlaneCheckboxGroup property value.
 * @return java.awt.CheckboxGroup
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.CheckboxGroup getPlaneCheckboxGroup() {
   if (ivjPlaneCheckboxGroup == null) {
      try {
         ivjPlaneCheckboxGroup = new java.awt.CheckboxGroup();
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlaneCheckboxGroup;
}
/**
 * Return the PlaneLabel property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Label getPlaneLabel() {
   if (ivjPlaneLabel == null) {
      try {
         ivjPlaneLabel = new java.awt.Label();
         ivjPlaneLabel.setName("PlaneLabel");
         ivjPlaneLabel.setText("Plane");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlaneLabel;
}
/**
 * Return the PlanePanel property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Panel getPlanePanel() {
   if (ivjPlanePanel == null) {
      try {
         ivjPlanePanel = new java.awt.Panel();
```

```
         ivjPlanePanel.setName("PlanePanel");
         ivjPlanePanel.setLayout(getPlanePanelFlowLayout());
         getPlanePanel().add(getXYCheckbox(), getXYCheckbox().getName());
         getPlanePanel().add(getYZCheckbox(), getYZCheckbox().getName());
         getPlanePanel().add(getZXCheckbox(), getZXCheckbox().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlanePanel;
}
/**
 * Return the PlanePanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getPlanePanelFlowLayout() {
   java.awt.FlowLayout ivjPlanePanelFlowLayout = null;
   try {
      /* Create part */
      ivjPlanePanelFlowLayout = new java.awt.FlowLayout();
      ivjPlanePanelFlowLayout.setAlignment(java.awt.FlowLayout.LEFT);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   };
   return ivjPlanePanelFlowLayout;
}
/**
 * Return the PlaneSelectedCheckbox property value.
 * @return java.awt.Checkbox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Checkbox getPlaneSelectedCheckbox() {
   // user code begin {1}
   // user code end
   return ivjPlaneSelectedCheckbox;
}
/**
 * Return the PlotButton property value.
 * @return java.awt.Button
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Button getPlotButton() {
   if (ivjPlotButton == null) {
      try {
         ivjPlotButton = new java.awt.Button();
         ivjPlotButton.setName("PlotButton");
         ivjPlotButton.setLabel("Plot");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjPlotButton;
}
/**
 * Return the SizeLabel property value.
```

```
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Label getSizeLabel() {
   if (ivjSizeLabel == null) {
      try {
         ivjSizeLabel = new java.awt.Label();
         ivjSizeLabel.setName("SizeLabel");
         ivjSizeLabel.setText("Size");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjSizeLabel;
}
/**
 * Return the SizePanel property value.
 * @return java.awt.Panel
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Panel getSizePanel() {
   if (ivjSizePanel == null) {
      try {
         ivjSizePanel = new java.awt.Panel();
         ivjSizePanel.setName("SizePanel");
         ivjSizePanel.setLayout(getSizePanelFlowLayout());
         getSizePanel().add(getSizeTextField(),
getSizeTextField().getName());
         getSizePanel().add(getSizeUnitsLabel(),
getSizeUnitsLabel().getName());
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjSizePanel;
}
/**
 * Return the SizePanelFlowLayout property value.
 * @return java.awt.FlowLayout
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.FlowLayout getSizePanelFlowLayout() {
   java.awt.FlowLayout ivjSizePanelFlowLayout = null;
   try {
      /* Create part */
      ivjSizePanelFlowLayout = new java.awt.FlowLayout();
      ivjSizePanelFlowLayout.setAlignment(java.awt.FlowLayout.LEFT);
   } catch (java.lang.Throwable ivjExc) {
      handleException(ivjExc);
   };
   return ivjSizePanelFlowLayout;
}
/**
 * Return the SizeTextField property value.
 * @return java.awt.TextField
 */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.TextField getSizeTextField() {
   if (ivjSizeTextField == null) {
      try {
         ivjSizeTextField = new java.awt.TextField();
         ivjSizeTextField.setName("SizeTextField");
         ivjSizeTextField.setText("5");
         ivjSizeTextField.setColumns(10);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjSizeTextField;
}
/**
 * Return the SizeUnitsLabel property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Label getSizeUnitsLabel() {
   if (ivjSizeUnitsLabel == null) {
      try {
         ivjSizeUnitsLabel = new java.awt.Label();
         ivjSizeUnitsLabel.setName("SizeUnitsLabel");
         ivjSizeUnitsLabel.setText("");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjSizeUnitsLabel;
}
/**
 * Return the TargetGradient property value.
 * @return TargetGradient
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private TargetGradient getTargetGradient() {
   if (ivjTargetGradient == null) {
      try {
         ivjTargetGradient = new TargetGradient();
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjTargetGradient;
}
/**
 * Return the Units property value.
 * @return Units
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private Units getUnits() {
```

```
   if (ivjUnits == null) {
      try {
         ivjUnits = new Units();
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjUnits;
}
/**
 * Return the XYCheckbox property value.
 * @return java.awt.Checkbox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Checkbox getXYCheckbox() {
   if (ivjXYCheckbox == null) {
      try {
         ivjXYCheckbox = new java.awt.Checkbox();
         ivjXYCheckbox.setName("XYCheckbox");
         ivjXYCheckbox.setLabel("XY");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjXYCheckbox;
}
/**
 * Return the YZCheckbox property value.
 * @return java.awt.Checkbox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Checkbox getYZCheckbox() {
   if (ivjYZCheckbox == null) {
      try {
         ivjYZCheckbox = new java.awt.Checkbox();
         ivjYZCheckbox.setName("YZCheckbox");
         ivjYZCheckbox.setLabel("YZ");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjYZCheckbox;
}
/**
 * Return the ZXCheckbox property value.
 * @return java.awt.Checkbox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Checkbox getZXCheckbox() {
   if (ivjZXCheckbox == null) {
      try {
         ivjZXCheckbox = new java.awt.Checkbox();
```

```
         ivjZXCheckbox.setName("ZXCheckbox");
         ivjZXCheckbox.setLabel("ZX");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjZXCheckbox;
}
/**
 * Called whenever the part throws an exception.
 * @param exception java.lang.Throwable
 */
private void handleException(Throwable exception) {

   /* Uncomment the following lines to print uncaught exceptions to stdout
*/
   // System.out.println("--------- UNCAUGHT EXCEPTION ---------");
   exception.printStackTrace(System.out);
}
/**
 * Initializes connections
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initConnections() {
   // user code begin {1}
   // user code end
   this.addWindowListener(this);
   getCancelButton().addActionListener(this);
   getPlotButton().addActionListener(this);
   connPtoP1SetTarget();
   connPtoP2SetTarget();
   connPtoP3SetTarget();
   connPtoP4SetTarget();
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   // user code begin {1}
   // user code end
   setName("GradientDialog");
   setLayout(new java.awt.BorderLayout());
   setSize(334, 282);
   setModal(true);
   setTitle("Gradient Plot");
   add(getContentsPane(), "Center");
   add(getButtonPanel(), "South");
   initConnections();
   connEtoM7();
   connEtoM11();
   // user code begin {2}
   // user code end
}
/**
 * main entrypoint - starts the part when it is run as an application
 * @param args java.lang.String[]
 */
public static void main(java.lang.String[] args) {
   Units experimentUnits = new Units();
```

```
   Day experimentDate = new Day();
   BundleList sources = new BundleList();
   sources.add(new RadBundle("1994", new Day(1994, 3, 11), 10794, 12));
   sources.add(new RadBundle("1979", new Day(1979, 9, 19), 9950, 12));
   sources.add(new RadBundle("1963", new Day(1963, 1, 14), 10600, 20));
   sources.setDate(experimentDate);
   sources.setExpBundle("1979");
   Fixture currentFixture = new AnnularFixture(new Coordinate(),
sources.getExpBundle(), 1, 6);
   try {
      GradientDialog aGradientDialog = new GradientDialog(new
java.awt.Frame());
      aGradientDialog.setModal(true);
      try {
         Class aCloserClass =
Class.forName("com.ibm.uvm.abt.edit.WindowCloser");
         Class parmTypes[] = {java.awt.Window.class};
         Object parms[] = {aGradientDialog};
         java.lang.reflect.Constructor aCtor =
aCloserClass.getConstructor(parmTypes);
         aCtor.newInstance(parms);
      }
      catch (java.lang.Throwable exc) {
      };
      aGradientDialog.setFixture(currentFixture);
      aGradientDialog.setVisible(true);
   }
   catch (Throwable exception) {
      System.err.println("Exception occurred in main() of
java.awt.Dialog");
      exception.printStackTrace(System.out);
   }
}
/**
 * Set the ConvertedSizeDouble to a new value.
 * @param newValue java.lang.Double
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void setConvertedSizeDouble(Double newValue) {
   if (ivjConvertedSizeDouble != newValue) {
      try {
         ivjConvertedSizeDouble = newValue;
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
}
/**
 * Sets the fixture property (Fixture) value.
 * @param fixture The new value for the property.
 * @see #getFixture
 */
public void setFixture(Fixture fixture) {
   fieldFixture = fixture;
}
/**
 * Set the PlaneSelectedCheckbox to a new value.
 * @param newValue java.awt.Checkbox
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
```

```
private void setPlaneSelectedCheckbox(java.awt.Checkbox newValue) {
   if (ivjPlaneSelectedCheckbox != newValue) {
      try {
         ivjPlaneSelectedCheckbox = newValue;
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowActivated(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowClosed(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowClosing(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   if ((e.getSource() == this) ) {
      connEtoC1(e);
   }
   // user code begin {2}
   // user code end
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowDeactivated(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
```

```
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowDeiconified(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowIconified(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
/**
 * Method to handle events for the WindowListener interface.
 * @param e java.awt.event.WindowEvent
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
public void windowOpened(java.awt.event.WindowEvent e) {
   // user code begin {1}
   // user code end
   // user code begin {2}
   // user code end
}
}
```

Dose.java

```
/* written by Brian Chow
 * created March 18, 1998
 * last modified February 12, 1999
 */

import java.util.*;

/**
 * Dose calculation method(s).
 */
class Dose
{  /**
   * Dose calculation of an array of target points using point sources
   * to approximate a fixture with vertically arranged cylindrical
   * pencils.
   */
   /**
   * Number of points used in approximating a single pencil.
   */
   private static int numPoints = 12;
   private static double waterDoseFactor = 0.96;
   /**
   * Output window for displaying calculation progress to user.
   */
   private static CalculationProgress output = new CalculationProgress();
   public static double[] pointSource(Fixture f,Coordinate[] c)
   {  // Setup window for user to see calculation progress.
     output.clear();
     output.show();

     double[] returnDoseArray = new double[c.length];
```

```
     double returnDose;
     Enumeration fEnum;
     Pencil p;
     // Calculate dose for each target point.
     for (int i = 0;i < c.length;i++)
     {  returnDose = 0;
       fEnum = f.elements();
       // Sum up dose for a single target point due to one pencil.
       while (fEnum.hasMoreElements())
       {  p = (Pencil)fEnum.nextElement();
         returnDose += pointSourceApprox(p,f,c[i]);
       }
       returnDoseArray[i] = returnDose * waterDoseFactor;
//     output.print("" + Units.outLength(c[i]) + " " +
//       Units.outDose(returnDose));
     }

     // Close the progress window.
     output.dispose();

     return returnDoseArray;
   }
   /**
   * Calculates dose at a single target point due to one pencil using a
   * point source approximation.
   */
   private static double pointSourceApprox(Pencil p,Fixture f,
    Coordinate c)
   {  double dose = 0;
     // Radiation of one point source.
     final double chunkRad = p.getSource().getRad() / numPoints;
     // Distance between each point source.
     final double chunkSize = p.length / numPoints;
     final Coordinate pencilCenter = p.getCenter();
     final double z = pencilCenter.z;
     // Current point source location.
     Coordinate point = new Coordinate(pencilCenter.x,pencilCenter.y,z);
     for (int i = -numPoints + 1;i <= numPoints - 1;i += 2)
     {  // Move point source along z axis.
       point.z = z + (chunkSize * i / 2.0);
/**       double attFactor = 0;
       Enumeration fEnum = f.elements();
       // Calculate attenuation factor from the source to the target.
       while (fEnum.hasMoreElements())
       {  Pencil tempP = (Pencil)fEnum.nextElement();
         attFactor += tempP.intersectDistance(point,c) *
           tempP.getAttCoefficient();
       }
*/     dose += chunkRad / Math.pow(point.distance(c),2)/** *
         Math.exp(attFactor)*/;
     }
     return dose;
   }
}
```