Java source code

GammaCell.java

```
/* written by Brian Chow
 * created February 27, 1998
 * last modified October 11, 1998
 */

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, 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);
   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("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]);
      }
   }
}
}
```

TargetPanelPoints.java

```
/* written by Brian Chow
 * created March 21, 1998
 * last modified October 11, 1998
```

```
 */

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

/**
 *
 */
class TargetPanelPoints extends TargetPanel implements ItemListener,
ActionListener {
   private TargetPoints currentPointsTarget;
   private Vector points;
   private GridBagConstraints gbc = new GridBagConstraints();
   private List pointsList = new List(15, false);
   private CoordinateTextFieldPanel pointPanel = new
CoordinateTextFieldPanel();
   private Button addButton, deleteButton, changeButton;
/**
*
*/
public TargetPanelPoints() {
   currentPointsTarget = new TargetPoints();
   points = new Vector(5, 5);
   points.addElement(currentPointsTarget.getPoints()[0]);
   pointsList.add((Units.outLength(currentPointsTarget.getPoints()[0])).toS
tring());
   constructPanel();
}
public TargetPanelPoints(Target currentPointsTarget) {
   this.currentPointsTarget = (TargetPoints) currentPointsTarget;
   Coordinate[] tempPoints = this.currentPointsTarget.getPoints();
   points = new Vector(tempPoints.length, 5);
   for (int i = 0; i < tempPoints.length; i++) {
      points.addElement(tempPoints[i]);
      pointsList.add((Units.outLength(tempPoints[i])).toString());
   }
   constructPanel();
}
public void actionPerformed(ActionEvent evt) {
   if (ClickChecker.isDouble()) {
      return;
   }
   String arg = evt.getActionCommand();
   if (arg.equals("Add")) {
      addPoint();
   }
   else
      if (arg.equals("Change")) {
         changePoint();
      }
      else
         if (arg.equals("Delete")) {
            deletePoint();
         }
}
private void add(Component c, int width, int height, int x, int y) {
   gbc.gridwidth = width;
   gbc.gridheight = height;
   gbc.gridx = x;
   gbc.gridy = y;
   add(c, gbc);
}
private void addPoint() {
   Coordinate changedPoint = pointPanel.getCoordinate();
   int indexToAdd = findPoint(changedPoint);
   if (indexToAdd < 0) {
      points.addElement(changedPoint);
      pointsList.add((Units.outLength(changedPoint)).toString());
      pointsList.select(pointsList.getItemCount() - 1);
   }
   else {
      pointsList.select(indexToAdd);
   }
   pointPanel.setCoordinate(changedPoint);
   updateButtons();
}
private void changePoint() {
   int indexToChange = pointsList.getSelectedIndex();
   Coordinate changedPoint = pointPanel.getCoordinate();
   int indexFound = findPoint(changedPoint);
   if (indexToChange == indexFound) {
      pointsList.select(indexToChange);
   }
   else
      if (indexFound < 0) {
         points.setElementAt(changedPoint, indexToChange);
         pointsList.replaceItem((Units.outLength(changedPoint)).toString(),
indexToChange);
         pointsList.select(indexToChange);
      }
      else {
         pointsList.select(indexFound);
         points.removeElementAt(indexToChange);
         pointsList.remove(indexToChange);
      }
   pointPanel.setCoordinate(changedPoint);
   updateButtons();
}
private void constructPanel() {
   setLayout(new GridBagLayout());
   gbc.weightx = 100;
   gbc.weighty = 100;
   gbc.insets = new Insets(10, 10, 10, 10);
   gbc.fill = GridBagConstraints.NONE;
   pointsList.addItemListener(this);
   pointsList.select(0);
   gbc.anchor = GridBagConstraints.NORTHWEST;
   add(pointsList, 1, 2, 1, 1);
   pointPanel.setCoordinate((Coordinate) points.elementAt(0));
   pointPanel.addTextListener(this);
   add(pointPanel, 3, 1, 2, 1);
   addButton = new Button("Add");
   addButton.addActionListener(this);
   deleteButton = new Button("Delete");
   deleteButton.setEnabled(pointsList.getItemCount() > 1);
   deleteButton.addActionListener(this);
   changeButton = new Button("Change");
   changeButton.addActionListener(this);
   gbc.anchor = GridBagConstraints.CENTER;
   add(addButton, 1, 1, 2, 2);
   add(deleteButton, 1, 1, 3, 2);
   add(changeButton, 1, 1, 4, 2);
}
private void deletePoint() { // Stop if only one point is left.
   if (points.size() <= 1) {
      return;
   }
```

```
   int indexToDelete = pointsList.getSelectedIndex();
   points.removeElementAt(indexToDelete);
   pointsList.remove(indexToDelete);
   int numItems = pointsList.getItemCount();
   if (indexToDelete >= numItems) {
      pointsList.select(numItems - 1);
   }
   else {
      pointsList.select(indexToDelete);
   }
   pointPanel.setCoordinate((Coordinate)
points.elementAt(pointsList.getSelectedIndex()));
   updateButtons();
}
private int findPoint(Coordinate p) {
   int i = 0;
   boolean found = false;
   while (i < points.size() && !found) {
      found = ((Coordinate) points.elementAt(i)).equals(p);
      i++;
   }
   if (found) {
      return i - 1;
   }
   else {
      return -1;
   }
}
public Target getTarget() {
   return new TargetPoints(points);
}
public void itemStateChanged(ItemEvent evt) {
   if (evt.getStateChange() == ItemEvent.DESELECTED) {
      return;
   }
   Coordinate changedCoordinate = (Coordinate)
points.elementAt(pointsList.getSelectedIndex());
   pointPanel.setCoordinate(changedCoordinate);
}
public boolean textValid() {
   return !points.isEmpty();
}
public void textValueChanged(TextEvent evt) {
   boolean pointOK = pointPanel.textValid();
   addButton.setEnabled(pointOK);
   changeButton.setEnabled(pointOK);
}
private void updateButtons() {
   boolean pointOK = pointPanel.textValid();
   boolean pointSelected = pointsList.getSelectedItem() != null;
   deleteButton.setEnabled(pointsList.getItemCount() > 1 && pointSelected);
   addButton.setEnabled(pointOK);
   changeButton.setEnabled(pointOK && pointSelected);
}
}
```

CalculationProgress.java

```
/* written by Brian Chow
 * created April 24, 1998
 * last modified December 11, 1998
 */

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);
/**
* 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) {
   outputArea.setText(value);
}
}
```

Units.java

```
/* written by Brian Chow
 * created March 20, 1998
 * last modified November 21, 1998
 */

/**
 *
 */
```

```
class Units {
   /**
   *
   */
   private static UnitLength lengthUnits;
   private static UnitDose doseUnits;
   private static Units self;
public Units() {
   self = this;
   lengthUnits = new UnitLength("Inches", 100);
   doseUnits = new UnitDose("Rads");
}
public Units(UnitLength lengthUnits, UnitDose doseUnits) {
   self = this;
   this.lengthUnits = lengthUnits;
   this.doseUnits = doseUnits;
}
/**
 *
 */
private static void checkInit() {
   if (self == null)
   {  self = new Units();
   }
}
public static UnitDose getDoseUnits() {
   checkInit();
   return doseUnits;
}
public static UnitLength getLengthUnits() {
   checkInit();
   return lengthUnits;
}
public static double inLength(double value) {
   checkInit();
   return lengthUnits.inUnits(value);
}
public static double outDose(double value) {
   checkInit();
   return doseUnits.outUnits(value);
}
public static double outLength(double value) {
   checkInit();
   return lengthUnits.outUnits(value);
}
public static Coordinate outLength(Coordinate point) {
   checkInit();
   double x, y, z;
   x = lengthUnits.outUnits(point.x);
   y = lengthUnits.outUnits(point.y);
   z = lengthUnits.outUnits(point.z);
   return new Coordinate(x, y, z);
}
public static void setDoseUnits(UnitDose u) {
   checkInit();
   doseUnits = u;
}
public static void setLengthUnits(UnitLength u) {
   checkInit();
   lengthUnits = u;
}
public String toString() {
   return "Dose: " + doseUnits + "  Accuracy: 1/" +
lengthUnits.getAccuracy() + " " + lengthUnits;
}
}
```

Pencil.java

```
/* written by Brian Chow
 * created March 10, 1998
 * last modified December 11, 1998
 */
class Pencil
{  //
   public final double radius;
   public final double length,halfLength;
   private static final double attCoefficient = 0;//-0.128 * 2.54;
   private Coordinate center;
   private Coordinate min = new Coordinate();
   private Coordinate max = new Coordinate();
   private RadBundle source;
   public Pencil(Coordinate center,RadBundle source)
   {  this.center = center;
     radius = 0.375 / 2;
     length = 7.375;
     halfLength = length / 2;
     setMaxMin();
     this.source = source;
   }
   //
   public Pencil(Coordinate center,RadBundle source,double diameter,
    double length)
   {  this.center = center;
     radius = diameter / 2;
     this.length = length;
     halfLength = length / 2;
     setMaxMin();
     this.source = source;
   }
   public double getAttCoefficient()
   {  return attCoefficient;
   }
   public Coordinate getCenter()
   {  return center;
   }
   public RadBundle getSource()
   {  return source;
   }
   /**
   * Given two endpoints of a line segment, return the intersection
   * distance with this pencil.
   * Reference: http://www.mhri.edu.au/~pdb/geometry/sphereline/
   */
   public double intersectDistance(Coordinate p1,Coordinate p2)
   {  if (p1.x <= min.x && p2.x <= min.x ||
        p1.y <= min.y && p2.z <= min.y ||
        p1.z <= min.z && p2.z <= min.z ||
        p1.x >= max.x && p2.x >= max.x ||
        p1.y >= max.y && p2.z >= max.y ||
        p1.z >= max.z && p2.z >= max.z)
     {  return 0.0;
     }
     final double dX,dY,dZ;
     dX = p2.x - p1.x;
     dY = p2.y - p1.y;
     dZ = p2.z - p1.z;
     final double uA;
```

```
     uA = ((center.x - p1.x) * dX + (center.y - p1.x) * dY) /
      (Math.pow(dX,2) + Math.pow(dY,2));
     if (uA < 0 || uA > 1)
     {  return 0.0;
     }
     Coordinate perpIntersect = new Coordinate();
     perpIntersect.x = p1.x + uA * dX;
     perpIntersect.y = p1.y + uA * dY;
     if (center.distanceZ(perpIntersect) >= radius)
     {  return 0.0;
     }
     else
     {  final boolean p1Inside,p2Inside;
       p1Inside = (center.distanceZ(p1) <= radius &&
         Math.abs(center.z - p1.z) <= halfLength);
       p2Inside = (center.distanceZ(p2) <= radius &&
         Math.abs(center.z - p2.z) <= halfLength);
       if (p1Inside && p2Inside)
       {  return p1.distance(p2);
       }
       else
       {  final double a,b,c,determinant;
         a = Math.pow(dX,2) + Math.pow(dY,2);
         b = 2 * (dX * (p1.x - center.x) + dY * (p1.y - center.y));
         c = Math.pow(center.x,2) + Math.pow(center.y,2) +
           Math.pow(p1.x,2) + Math.pow(p1.y,2) -
           2 * (center.x * p1.x + center.y * p1.y) -
           Math.pow(radius,2);
         determinant = Math.pow(b,2) - 4 * a * c;
         Coordinate end1 = null;
         Coordinate end2 = null;
         if (p1Inside)
         {  end1 = p1;
         }
         else if (determinant <= 0)
         {  if (dX == 0 && dY == 0)
            {  end1 = new Coordinate();
              end1.x = p1.x;
              end1.y = p1.y;
              if (p1.z > p2.z)
              {  end1.z = max.z;
              }
              else
              {  end1.z = min.z;
              }
            }
            else
            {  //System.err.println("Intersection Distance error");
              return 0.0;
            }
         }
         else
         {  final double u1 = (-b - Math.sqrt(determinant)) /
             (2 * a);
            end1 = new Coordinate();
            end1.x = p1.x + u1 * dX;
            end1.y = p1.y + u1 * dY;
//               double t = (end1.x - p1.x) / dX;
            end1.z = p1.z + u1 * dZ;
            if (end1.z > max.z)
            {  double u = (max.z - p1.z) / dZ;
              end1.x = p1.x + u * dX;
              end1.y = p1.y + u * dY;
              end1.z = max.z;
```

```
            }
            else if (end1.z < min.z)
            {  double u = (min.z - p1.z) / dZ;
              end1.x = p1.x + u * dX;
              end1.y = p1.y + u * dY;
              end1.z = min.z;
            }
         }
         if (p2Inside)
         {  end2 = p2;
         }
         else if (determinant <= 0)
         {  if (dX == 0 && dY == 0)
            {  end2 = new Coordinate();
              end2.x = p2.x;
              end2.y = p2.y;
              if (p2.z > p1.z)
              {  end2.z = max.z;
              }
              else
              {  end2.z = min.z;
              }
            }
            else
            {  //System.err.println("Intersection Distance error");
              return 0.0;
            }
         }
         else
         {  final double u2 = (-b + Math.sqrt(determinant)) /
             (2 * a);
            end2 = new Coordinate();
            end2.x = p1.x + u2 * dX;
            end2.y = p1.y + u2 * dY;
//               double t = (end2.x - p1.x) / dX;
            end2.z = p1.z + u2 * dZ;
            if (end2.z > max.z)
            {  double u = (max.z - p1.z) / dZ;
              end2.x = p1.x + u * dX;
              end2.y = p1.y + u * dY;
              end2.z = max.z;
            }
            else if (end2.z < min.z)
            {  double u = (min.z - p1.z) / dZ;
              end2.x = p1.x + u * dX;
              end2.y = p1.y + u * dY;
              end2.z = min.z;
            }
         }
//             System.out.println("*" + end1 + " * " + end2);
         return end1.distance(end2);
       }
     }
   }
   public void setCenter(Coordinate center)
   {  this.center = center;
     setMaxMin();
   }
/*
   public static void main(String[] args)
   {  Pencil p = new Pencil(new Coordinate(1,1,1),new RadBundle());
     System.out.println(p);
     System.out.println(p.intersectDistance(new Coordinate(0,0,1),
      new Coordinate(2,2,1)));
```

```
     System.out.println(p.intersectDistance(new Coordinate(1.1,1,1),
      new Coordinate(-0.9,1,1)));
     System.out.println(p.intersectDistance(new Coordinate(0,-1,0),
      new Coordinate(0,1,0)));
     System.out.println(p.intersectDistance(new Coordinate(0,-1,3),
      new Coordinate(0,1,3)));
     System.out.println(p.intersectDistance(new Coordinate(1,0,10),
      new Coordinate(1,0,10)));
     System.out.println(p.intersectDistance(new Coordinate(0.15,0,10),
      new Coordinate(0.15,0,0)));
     System.out.println(p.intersectDistance(new
Coordinate(p.min.x,0,p.max.z),
      new Coordinate(p.max.x,0,p.min.z)));
     System.out.println(p.intersectDistance(new Coordinate(-0.2,0,p.max.z),
      new Coordinate(0.2,0,p.min.z)));
     System.out.println(p.intersectDistance(new Coordinate(1,0,10),
      new Coordinate(1,0,10)));

   }
*/

   private void setMaxMin()
   {  min.x = center.x - radius;
     max.x = center.x + radius;
     min.y = center.y - radius;
     max.y = center.y + radius;
     min.z = center.z - halfLength;
     max.z = center.z + halfLength;
   }
   public void setSource(RadBundle source)
   {  this.source = source;
   }
   public String toString()
   {  return "" + Units.outLength(center) + " " + source;
   }
   public void translate(double dX,double dY,double dZ)
   {  center.translate(dX,dY,dZ);
     setMaxMin();
   }
}
```

Target.java

```
/* written by Brian Chow
 * created March 19, 1998
 * last modified November 13, 1998
 */

/**
 * Representation of a target.
 */
abstract class Target implements Cloneable
{  /**
   * Array of points to represent target.
   */
   protected Coordinate[] points;
   /**
   * Array containing the dose at the corresponding target point.
   */
   protected double[] dose;
   /**
   * Calculation time in milliseconds.
   */
   protected int calculationTime;
/**
* Calculates doses at target points and keeps track of the
* calculation time.
*/
public void calculate(Fixture currentFixture) {
   long timeBegin = System.currentTimeMillis();
   dose = Dose.pointSource(currentFixture, points);
   doPostCalculations();
   long timeEnd = System.currentTimeMillis();
   calculationTime = (int) (timeEnd - timeBegin);
}
/**
* Clone this target.
*/
public Object clone() {
   try {
      return super.clone();
   }
   catch (CloneNotSupportedException e) { // this shouldn't happen, since
we are Cloneable
      return null;
   }
}
/**
 * Allows subclasses to do additional analysis after
 * dose calculations have been made.  Currently this
 * method returns immediately.
 */
protected void doPostCalculations() {
}
}
```

TargetPoints.java

```
/* written by Brian Chow
 * created March 20, 1998
 * last modified November 13, 1998
 */

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, 75);
   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";
}
}
```

TargetObject.java

```
/* written by Brian Chow
 * created March 23, 1998
 * last modified November 13, 1998
 */

import java.util.*;

/**
 *
 */
abstract class TargetObject extends TargetPoints {
   protected Coordinate center;
   protected int centerIndex;
   protected double height;
   protected double[] doseDeviation;
/**
*
*/
public Object clone() {
   return super.clone();
}
abstract protected void constructObject();
/**
 * Calculates percent deviation from center.
 */
protected void doPostCalculations() {
   final int numPoints = points.length;
   final double centerDose = dose[centerIndex];
   doseDeviation = new double[numPoints];
   for (int i = 0; i < numPoints; i++) {
      doseDeviation[i] = (dose[i] - centerDose) / centerDose;
   }
}
/**
 * Return a vector of string headings to use to identify
 * calculation results.
 * @return java.util.Vector
 */
protected Vector getCalcColumnHeadings() {
   Vector returnVal = super.getCalcColumnHeadings();
```

```
   returnVal.addElement("Deviation from Center");
   return returnVal;
}
/**
 * Return a vector of strings for one row of
 * calculation results.
 * @return java.util.Vector
 */
protected Vector getCalcResultsRow(int row) {
   Vector returnVal = super.getCalcResultsRow(row);
   java.text.DecimalFormat df = new java.text.DecimalFormat("#.00 %");
   returnVal.addElement(df.format(doseDeviation[row]));
   return returnVal;
}
public Coordinate getCenter() {
   return center;
}
public double getHeight() {
   return height;
}
public void setCenter(Coordinate c) {
   center = c;
   constructObject();
}
public void setHeight(double h) {
   height = h;
   constructObject();
}
protected final void translate(Coordinate c) {
   for (int i = 0; i < points.length; i++) {
      points[i].translate(c.x, c.y, c.z);
   }
}
}
```

TargetCyl.java

```
/* written by Brian Chow
 * created April 7, 1997
 * last modified November 13, 1998
 */

/**
 *
 */
class TargetCyl extends TargetObject {
   private double radius;
/**
*
*/
public TargetCyl() {
   this(new Coordinate(), 1, 1);
}
public TargetCyl(Coordinate c, double h, double r) {
   center = c;
   centerIndex = 7;
   height = h;
   radius = r;
   constructObject();
}
public Object clone() {
   return super.clone();
}
protected void constructObject() {
   points = new Coordinate[15];
   for (int i = -1; i <= 1; i++) {
      final double z = height / 2 * i;
      points[5 * (i + 1)] = new Coordinate(0, -radius, z);
      points[5 * (i + 1) + 1] = new Coordinate(-radius, 0, z);
      points[5 * (i + 1) + 2] = new Coordinate(0, 0, z);
      points[5 * (i + 1) + 3] = new Coordinate(radius, 0, z);
      points[5 * (i + 1) + 4] = new Coordinate(0, radius, z);
   }
   translate(center);
}
public double getRadius() {
   return radius;
}
public void setCyl(Coordinate c, double h, double r) {
   center = c;
   height = h;
   radius = r;
   constructObject();
}
public String toString() {
   return "Cylinder: Center " + Units.outLength(center) + " Height " +
Units.outLength(height) + " Radius " + Units.outLength(radius);
}
}
```

TargetRect.java

```
/* written by Brian Chow
 * created April 7, 1997
 * last modified November 13, 1998
 */

/**
 *
 */
class TargetRect extends TargetObject {
   private double width, length;
/**
*
*/
public TargetRect() {
   this(new Coordinate(), 1, 1, 1);
}
public TargetRect(Coordinate c, double h, double w, double l) {
   center = c;
   centerIndex = 13;
   height = h;
   width = w;
   length = l;
   constructObject();
}
public Object clone() {
   return super.clone();
}
protected void constructObject() {
   points = new Coordinate[27];
   int i = 0;
   for (int j = -1; j <= 1; j++) {
      for (int k = -1; k <= 1; k++) {
         for (int m = -1; m <= 1; m++) {
            points[i] = new Coordinate(width / 2 * m, length / 2 * k,
height / 2 * j);
            i++;
```

```
         }
      }
   }
   translate(center);
}
public double getLength() {
   return length;
}
public double getWidth() {
   return width;
}
public void setRect(Coordinate c, double h, double w, double l) {
   center = c;
   height = h;
   width = w;
   length = l;
   constructObject();
}
public String toString() {
   return "Rectangular Parallelepiped: Center " + Units.outLength(center) +
" Height " + Units.outLength(height) + " Width " + Units.outLength(width) +
" Length " + Units.outLength(length);
}
}
```

TargetGradient.java

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

/**
 *
 */
public class TargetGradient extends Target {
   private GradientPlotCalculator gpc = new GradientPlotCalculator();
   private int height,width;
   private int numIndices = 16;
/**
 * This method was created in VisualAge.
 */
public TargetGradient() {
   width = 51;
   height = 51;
   calculatePoints();
}
/**
* 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.
   gpc.setNumIndices(numIndices);
   gpc.setMaxMin(new double[] {0, 0});
   gpc.setWidth(width);
   gpc.setHeight(height);
   gpc.setData(dose);
   GradientPlot gp = new GradientPlot(gpc);
   gp.setVisible(true);
}
```

```
/**
 * This method was created in VisualAge.
 */
private void calculatePoints() {
   points = new Coordinate[width * height];
   for (int i = 0; i < width; i++) {
      for (int j = 0; j < height; j++) {
         points[i * width + j] = new Coordinate((i - (height - 1) / 2) /
10.0, (j - (width - 1) / 2) / 10.0, 0);
      }
   }
}
/**
 * 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 December 11, 1998
 */

import java.awt.*;
import java.awt.event.*;
/**
 *
 */
public class GradientPlot extends Frame implements WindowListener {
   private Panel ivjContentsPane = null;
   private GradientPlotLegendCanvas ivjLegendCanvas = null;
   private Label ivjLegendLabel = null;
   private GradientPlotFieldCanvas ivjPlotCanvas = null;
   private GradientPlotCalculator ivjGradientPlotCalculator = null;
   private Label ivjPlotLabel = 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);
   }
}
/**
 * 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();
   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);
         // 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 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 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();
}
/**
 * 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);
   connPtoP1SetTarget();
   connPtoP2SetTarget();
}
/**
 * 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, 293);
   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 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
}
}
```

GradientPlotCanvas.java

```
/* written by Brian Chow
 * created November 15, 1998
 * last modified December 11, 1998
 */

import java.awt.*;
/**
 * This type was created in VisualAge.
 */
public abstract class GradientPlotCanvas extends Canvas {
   protected Dimension minSize = new Dimension();
   protected Color[] colors;
   protected int numIndices;
/**
 * This method was created in VisualAge.
 */
public GradientPlotCanvas() {
}
/**
 * This method was created in VisualAge.
 * @return java.awt.Color[]
 */
public Color [] getColors() {
   return colors;
}
/**
 * This method was created in VisualAge.
 * @return java.awt.Dimension
 */
public Dimension getMinimumSize() {
   return minSize;
}
/**
 * This method was created in VisualAge.
 * @return java.awt.Dimension
 */
public Dimension getPreferredSize() {
   return getMinimumSize();
}
/**
 * Sets the colors property (java.awt.Color[]) value.
 * @param colors The new value for the property.
 * @see #getColors
 */
public void setColors(Color[] colors) {
   this.colors = colors;
   numIndices = colors.length;
}
/**
 * This method was created in VisualAge.
 */
```

```
protected void setMinimumSize(int w,int h) {
   minSize.setSize(w,h);
}
}
```

GradientPlotFieldCanvas.java

```
/* written by Brian Chow
 * created November 15, 1998
 * last modified December 11, 1998
 */

import java.awt.*;
/**
 * This type was created in VisualAge.
 */
public class GradientPlotFieldCanvas extends GradientPlotCanvas {
   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;
/**
 * This method was created in VisualAge.
 */
public GradientPlotFieldCanvas() {
   super();
   setFont(new Font("SansSerif", Font.PLAIN, 12));
   fm = getToolkit().getFontMetrics(getFont());
   final int height = fm.getHeight();
   textOffset = height;
}
/**
 * This method was created in VisualAge.
 */
protected void checkDimensions() {
   if (data != null && data.length == width * height) {
      setMinimumSize();
   }
   else {
      setMinimumSize(0, 0);
   }
}
/**
 * This method was created in VisualAge.
 * @return int[]
 */
public int[] getData() {
   return data;
}
/**
 * This method was created in VisualAge.
 * @return int
 */
public int getHeight() {
   return height;
}
/**
 * This method was created in VisualAge.
 * @return int
 */
```

```
public int getWidth() {
   return width;
}
/**
 * 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();
   int pixelWidth = size.width / width;
   int 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();
}
/**
 * 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();
}
/**
 * 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);
}
/**
 * This method was created in VisualAge.
 * @param width int
 */
public void setWidth(int width) {
   this.width = width;
   checkDimensions();
}
}
```

GradientPlotLegendCanvas.java

```
/* written by Brian Chow
 * created November 15, 1998
 * last modified December 11, 1998
 */

import java.awt.*;
/**
 * This type was created in VisualAge.
 */
public class GradientPlotLegendCanvas extends GradientPlotCanvas {
   protected static final int BOXHEIGHT = 20;
   protected static final int BOXWIDTH = 40;
   protected static final int HORIZGAP = 10;
   protected String[] labels;
   protected FontMetrics fm;
   private int textOffset;
/**
 * This method was created in VisualAge.
 */
public GradientPlotLegendCanvas() {
   super();
   setFont(new Font("SanSerif", Font.PLAIN, 12));
   fm = getToolkit().getFontMetrics(getFont());
   final int height = fm.getHeight();
   textOffset = ((height - BOXHEIGHT) / 2) + height;
}
/**
 * This method was created in VisualAge.
 * @param g java.awt.Graphics
 */
public void paint(Graphics g) {
   if (colors == null || labels == null)
   {  return;
   }
   for (int i = 0;i < numIndices;i++)
   {  // Draw colored box
      g.setColor(colors[i]);
      g.fillRect(0,i * BOXHEIGHT,BOXWIDTH,BOXHEIGHT);

      // Label box with number
      g.setColor(Color.black);
      g.drawString(labels[i],BOXWIDTH + HORIZGAP, i * BOXHEIGHT +
textOffset);
   }
   g.dispose();
}
/**
 * This method was created in VisualAge.
 * @param labels java.lang.String[]
 */
public void setLabels(String[] labels) {
   this.labels = labels;
   setMinimumSize();
}
/**
 * This method was created in VisualAge.
 */
protected void setMinimumSize() {
   int maxLength = 0;
   int curLength = 0;
   for (int i = 0; i < numIndices; i++) {
```

```
      curLength = fm.stringWidth(labels[i]);
      if (curLength > maxLength) {
         maxLength = curLength;
      }
   }
   setMinimumSize(BOXWIDTH + HORIZGAP + maxLength + 10, numIndices *
BOXHEIGHT);
}
}
```

GradientPlotCalculator.java

```
/* written by Brian Chow
 * created November 15, 1998
 * last modified December 14, 1998
 */

import java.awt.*;
/**
 *
 */
public class GradientPlotCalculator {
   protected int numIndices;
   protected Color[] colors;
   protected double[] ranges;
   protected String[] labels;
   protected double[] data;
   protected int[] colorData;
   protected int width,height;
   protected boolean autoRange = false;
   protected double maxValue;
   protected double minValue;
   protected GradientPlotLegendCanvas legendCanvas;
   protected GradientPlotFieldCanvas fieldCanvas;
/**
 * This method was created in VisualAge.
 */
public GradientPlotCalculator() {
   this(10);
}
/**
 * This method was created in VisualAge.
 * @param numIndices int
 */
public GradientPlotCalculator(int numIndices) {
   this(numIndices, 0, 0);
}
/**
 * This method was created in VisualAge.
 * @param numIndices int
 * @param max double
 * @param min double
 */
public GradientPlotCalculator(int numIndices, double max, double min) {
   setNumIndices(numIndices);
   setMaxMin(new double[] {max, min});
}
/**
 * This method was created in VisualAge.
 * @param numIndices int
 * @param max double
 * @param min double
 * @param width int
 * @param height int
```

```
 * @param data double[]
 */
public GradientPlotCalculator(int numIndices,double max,double min,int
width,int height,double[] data) {
   setNumIndices(numIndices);
   setMaxMin(new double[] {max, min});
}
/**
 * This method was created in VisualAge.
 */
protected void calculateAutoRange() {
   double max = 1;
   double min = 0;
   if (data != null) {
      max = data[0];
      min = data[0];
      for (int i = 1; i < data.length; i++) {
         if (data[i] < min) {
            min = data[i];
         }
         else
            if (data[i] > max) {
               max = data[i];
            }
      }
   }
   maxValue = max;
   minValue = min;
}
/**
 * This method was created in VisualAge.
 */
protected void calculateColors() {
   for (int i = 0; i < numIndices; i++) {
      colors[i] = new Color(Color.HSBtoRGB((float) ((1 - i / (numIndices -
1f)) / 1.25), 0.85f, 0.85f));
   }
}
/**
 * This method was created in VisualAge.
 */
protected void calculateData() {
   final double maxMinDiff = maxValue - minValue;
   for (int i = 0; i < data.length; i++) {
      colorData[i] = (int)((data[i] - minValue) / maxMinDiff * numIndices);
      colorData[i] = colorData[i] >= numIndices ? numIndices - 1 :
colorData[i] < 0 ? 0 : colorData[i];
   }
}
/**
 * This method was created in VisualAge.
 */
protected void calculateLabels() {
   for (int i = 0; i < numIndices; i++) {
      labels[i] = new String("> " + Units.outDose(ranges[i]));
   }
}
/**
 * This method was created in VisualAge.
 */
protected void calculateRanges() {
   if (autoRange) {
      calculateAutoRange();
   }
```

```
   for (int i = 0; i < numIndices; i++) {
      ranges[i] = (maxValue - minValue) * i / numIndices + minValue;
   }
}
/**
 * This method was created in VisualAge.
 * @return double[]
 */
public double[] getData() {
   return data;
}
/**
 * This method was created in VisualAge.
 * @return java.awt.Canvas
 */
public GradientPlotFieldCanvas getFieldCanvas() {
   return fieldCanvas;
}
/**
 * This method was created in VisualAge.
 * @return int
 */
public int getHeight() {
   return height;
}
/**
 * This method was created in VisualAge.
 * @return java.awt.Canvas
 */
public GradientPlotLegendCanvas getLegendCanvas() {
   return legendCanvas;
}
/**
 * This method was created in VisualAge.
 * @return double[]
 */
public double[] getMaxMin() {
   return new double[] {maxValue, minValue};
}
/**
 * This method was created in VisualAge.
 * @return int
 */
public int getNumIndices() {
   return numIndices;
}
/**
 * This method was created in VisualAge.
 * @return int
 */
public int getWidth() {
   return width;
}
/**
 * This method was created in VisualAge.
 * @param data double[]
 */
public void setData(double[] data) {
   this.data = data;
   if (data == null) {
      return;
   }
   else
      if (colorData == null || colorData.length != data.length) {
         colorData = new int[data.length];
      }

   // See if auto scale is on
   if (autoRange) {
      calculateRanges();
      calculateLabels();
   }
   calculateData();
   if (fieldCanvas != null) {
      fieldCanvas.setData(colorData);
   }
}
/**
 * This method was created in VisualAge.
 * @param legendCanvas GradientPlotLegendCanvas
 */
public void setFieldCanvas(GradientPlotFieldCanvas fieldCanvas) {
   this.fieldCanvas = fieldCanvas;
   if (fieldCanvas == null) {
      return;
   }
   fieldCanvas.setColors(colors);
   fieldCanvas.setHeight(height);
   fieldCanvas.setWidth(width);
   fieldCanvas.setData(colorData);
}
/**
 * This method was created in VisualAge.
 * @param height int
 */
public void setHeight(int height) {
   this.height = height;
   if (fieldCanvas != null) {
      fieldCanvas.setHeight(height);
   }
}
/**
 * This method was created in VisualAge.
 * @param legendCanvas GradientPlotLegendCanvas
 */
public void setLegendCanvas(GradientPlotLegendCanvas legendCanvas) {
   this.legendCanvas = legendCanvas;
   if (legendCanvas == null) {
      return;
   }
   legendCanvas.setColors(colors);
   legendCanvas.setLabels(labels);
}
/**
 * This method was created in VisualAge.
 * @param max double
 * @param min double
 */
public void setMaxMin(double[] maxmin) {
   if (maxmin[0] == maxmin[1]) {
      autoRange = true;
   }
   if (maxmin[0] > maxmin[1]) {
      this.maxValue = maxmin[0];
      this.minValue = maxmin[1];
   }
   else {
      this.maxValue = maxmin[1];
```

```
      this.minValue = maxmin[0];
   }
   calculateRanges();
   calculateLabels();
   if (legendCanvas != null) {
      legendCanvas.setLabels(labels);
   }
}
/**
 * This method was created in VisualAge.
 * @param numIndices int
 */
public void setNumIndices(int numIndices) {
   if (this.numIndices == numIndices) {
      return;
   }
   this.numIndices = numIndices;
   colors = new Color[numIndices];
   ranges = new double[numIndices];
   labels = new String[numIndices];
   calculateColors();
   if (legendCanvas != null) {
      legendCanvas.setColors(colors);
   }
   if (fieldCanvas != null) {
      fieldCanvas.setColors(colors);
   }
}
/**
 * This method was created in VisualAge.
 * @param width int
 */
public void setWidth(int width) {
   this.width = width;
   if (fieldCanvas != null) {
      fieldCanvas.setWidth(width);
   }
}
}
```

ClickChecker.java

```
/* written by Brian Chow
 * created October 11, 1998
 * last modified October 11, 1998
 */

/**
 *
 */
public class ClickChecker {
   /**
   * Variable to keep track of the last click time.  Workaround for
   * extraneous mouse clicks when user double clicks.
   */
   private static long lastClick = 0;
   private static int clickThreshold = 750;
/**
*
*/
public static boolean isDouble() {
   final long currentTimeStamp = System.currentTimeMillis();
   if (currentTimeStamp - clickThreshold < lastClick) {
      return true;
```

```
   }
   lastClick = currentTimeStamp;
   return false;
}
/**
*
*/
public static void setClickThreshold(int threshold) {
   if (threshold > 0) {
      clickThreshold = threshold;
   }
   else {
      throw new IllegalArgumentException();
   }
}
}
```

GradientDialog.java

```
/* written by Brian Chow
 * created November 21, 1998
 * last modified November 21, 1998
 */
/**
 *
 */
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.Label ivjNumPointsLabel = null;
   private java.awt.TextField ivjNumPointsTextField = null;
   private java.awt.FlowLayout ivjPlanePanelFlowLayout = null;
   private java.awt.Label ivjSizeLabel = null;
   private java.awt.TextField ivjSizeTextField = null;
   private java.awt.Button ivjButton2 = null;
   private java.awt.Panel ivjButtonPanel = null;
   private java.awt.Button ivjPlotButton = 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() == getButton2()) ) {
      connEtoM1(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);
   }
}
/**
 * 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 Button2 property value.
 * @return java.awt.Button
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Button getButton2() {
   if (ivjButton2 == null) {
      try {
         ivjButton2 = new java.awt.Button();
         ivjButton2.setName("Button2");
         ivjButton2.setLabel("Cancel");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjButton2;
}
/**
 * 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(getButton2(), getButton2().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 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);
         // 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 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 constraintsNumPointsLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsNumPointsTextField = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsSizeLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsCenterLabel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsCenterPanel = new
java.awt.GridBagConstraints();
   java.awt.GridBagConstraints constraintsSizeTextField = 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.WEST;
         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.WEST;
         constraintsPlanePanel.weightx = 100.0;
         constraintsPlanePanel.weighty = 0.0;
         getContentsPane().add(getPlanePanel(), constraintsPlanePanel);

         constraintsNumPointsLabel.gridx = 0;
constraintsNumPointsLabel.gridy = 3;
         constraintsNumPointsLabel.gridwidth = 1;
constraintsNumPointsLabel.gridheight = 1;
         constraintsNumPointsLabel.anchor =
java.awt.GridBagConstraints.WEST;
         constraintsNumPointsLabel.weightx = 0.0;
         constraintsNumPointsLabel.weighty = 0.0;
         getContentsPane().add(getNumPointsLabel(),
constraintsNumPointsLabel);

         constraintsNumPointsTextField.gridx = 1;
constraintsNumPointsTextField.gridy = 3;
         constraintsNumPointsTextField.gridwidth = 1;
constraintsNumPointsTextField.gridheight = 1;
         constraintsNumPointsTextField.anchor =
java.awt.GridBagConstraints.WEST;
         constraintsNumPointsTextField.weightx = 0.0;
         constraintsNumPointsTextField.weighty = 0.0;
         getContentsPane().add(getNumPointsTextField(),
constraintsNumPointsTextField);

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

         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.anchor = java.awt.GridBagConstraints.WEST;
         constraintsCenterPanel.weightx = 0.0;
         constraintsCenterPanel.weighty = 0.0;
         getContentsPane().add(getCenterPanel(), constraintsCenterPanel);

         constraintsSizeTextField.gridx = 1; constraintsSizeTextField.gridy
= 1;
         constraintsSizeTextField.gridwidth = 1;
constraintsSizeTextField.gridheight = 1;
```

```
         constraintsSizeTextField.anchor =
java.awt.GridBagConstraints.WEST;
         constraintsSizeTextField.weightx = 0.0;
         constraintsSizeTextField.weighty = 0.0;
         getContentsPane().add(getSizeTextField(),
constraintsSizeTextField);
         // 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 NumPointsLabel property value.
 * @return java.awt.Label
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.Label getNumPointsLabel() {
   if (ivjNumPointsLabel == null) {
      try {
         ivjNumPointsLabel = new java.awt.Label();
         ivjNumPointsLabel.setName("NumPointsLabel");
         ivjNumPointsLabel.setText("Number of Points");
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjNumPointsLabel;
}
/**
 * Return the NumPointsTextField property value.
 * @return java.awt.TextField
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private java.awt.TextField getNumPointsTextField() {
   if (ivjNumPointsTextField == null) {
      try {
         ivjNumPointsTextField = new java.awt.TextField();
         ivjNumPointsTextField.setName("NumPointsTextField");
         ivjNumPointsTextField.setText("25");
         ivjNumPointsTextField.setColumns(10);
         // user code begin {1}
         // user code end
      } catch (java.lang.Throwable ivjExc) {
         // user code begin {2}
         // user code end
         handleException(ivjExc);
      }
   };
   return ivjNumPointsTextField;
}
/**
 * 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 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 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("10");
```

```
         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 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);
   getButton2().addActionListener(this);
   connPtoP1SetTarget();
   connPtoP2SetTarget();
   connPtoP3SetTarget();
   connPtoP4SetTarget();
}
/**
 * Initialize the class.
 */
/* WARNING: THIS METHOD WILL BE REGENERATED. */
private void initialize() {
   // user code begin {1}
   ivjCenterPanel = new CoordinateTextFieldPanel();
   // user code end
   setName("GradientDialog");
   setLayout(new java.awt.BorderLayout());
   setSize(426, 240);
   setModal(true);
   add(getContentsPane(), "Center");
   add(getButtonPanel(), "South");
   initConnections();
   // 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 {
      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.setVisible(true);
   } catch (Throwable exception) {
      System.err.println("Exception occurred in main() of
java.awt.Dialog");
      exception.printStackTrace(System.out);
   }
}
/**
 * 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
}
}
```