```
// CubePers.java: A cube in perspective.
// Uses: Point2D (Section 1.5), Point3D (Section 3.9).

// Copied from Section 5.4 of
//    Ammeraal, L. (1998) Computer Graphics for Java Programmers,
//       Chichester: John Wiley.
//
// Modified by Juan Gomez - 10/24/99

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

/**
 * this class draws a perspective representation of a cube, with all edges
 * visible - A wire frame model
 */
public class CubePers extends Frame {
   private static final long serialVersionUID = 1L;

   public static void main(String[] args) {
      new CubePers();
   }

   CubePers() {
      super("A cube in perspective");
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            System.exit(0);
         }
      });
      setLayout(new BorderLayout());
      add("Center", new CvCubePers());
      Dimension dim = getToolkit().getScreenSize();
      setSize(dim.width / 2, dim.height / 2);
      setLocation(dim.width / 4, dim.height / 4);
      setVisible(true);
   }
}

/**
 * draw the perspective cube on the canvas container
 */

class CvCubePers extends Canvas {
   private static final long serialVersionUID = 1L;
   int centerX, centerY;
   Obj obj = new Obj();

   int iX(float x) {
      return Math.round(centerX + x);
   }

   int iY(float y) {
      return Math.round(centerY - y);
   }

   void line(Graphics g, int i, int j) {
      Point2D P = obj.vScr[i], Q = obj.vScr[j];
      if (obj.w[i].x + obj.w[j].x + obj.w[i].y + obj.w[j].y + obj.w[i].z + obj.w[j].z == 0) {
         g.setColor(Color.BLACK);
         g.drawLine(iX(P.x), iY(P.y), iX(Q.x), iY(Q.y));
      } else if (obj.w[i].x + obj.w[j].x + obj.w[i].y + obj.w[j].y + obj.w[i].z + obj.w[j].z == 4)
      {
         g.setColor(Color.BLACK);
         g.drawLine(iX(P.x), iY(P.y), iX(Q.x), iY(Q.y));
      } else {
         g.setColor(Color.BLUE);
         g.drawLine(iX(P.x), iY(P.y), iX(Q.x), iY(Q.y));
      }
   }
```

```
   void fill(Graphics g, int i, int j, int k, int l) {
      Point2D P = obj.vScr[i], Q = obj.vScr[j], R = obj.vScr[k], S = obj.vScr[l];
      float pq = obj.w[i].x + obj.w[j].x + obj.w[i].y + obj.w[j].y + obj.w[i].z + obj.w[j].z;
      float qr = obj.w[j].x + obj.w[k].x + obj.w[j].y + obj.w[k].y + obj.w[j].z + obj.w[k].z;
      float rs = obj.w[k].x + obj.w[l].x + obj.w[k].y + obj.w[l].y + obj.w[k].z + obj.w[l].z;
      float sp = obj.w[l].x + obj.w[i].x + obj.w[l].y + obj.w[i].y + obj.w[l].z + obj.w[i].z;

      int[] xPoints = { iX(P.x), iX(Q.x), iX(R.x), iX(S.x) };
      int[] yPoints = { iY(P.y), iY(Q.y), iY(R.y), iY(S.y) };
      if (pq + qr + rs + sp == 8) {
         g.setColor(Color.RED);
         g.fillPolygon(xPoints, yPoints, 4);
      }
   }

   public void paint(Graphics g) {
      Dimension dim = getSize();
      int maxX = dim.width - 1, maxY = dim.height - 1, minMaxXY = Math.min(maxX, maxY);
      centerX = maxX / 2;
      centerY = maxY / 2;
      obj.d = obj.rho * minMaxXY / obj.objSize; //
      obj.eyeAndScreen();
      // Faces
      fill(g, 0, 1, 2, 3);
      fill(g, 0, 4, 5, 1);
      fill(g, 2, 3, 6, 7);
      fill(g, 4, 5, 6, 7);
      fill(g, 0, 3, 4, 7);
      fill(g, 1, 5, 6, 2);
      // Horizontal edges at the bottom:
      // Black
      line(g, 0, 1);
      line(g, 1, 2);
      // Blue
      line(g, 2, 3);
      line(g, 3, 0);
      // Horizontal edges at the top:
      // Black
      line(g, 4, 5);
      line(g, 5, 6);
      line(g, 6, 7);
      line(g, 7, 4);
      // Vertical edges:
      // Black
      line(g, 0, 4);
      line(g, 1, 5);
      line(g, 2, 6);
      // Blue
      line(g, 3, 7);
   }
}

/**
 * store the world, eye, and screen coordinates for each of the 8 vertices of
 * the cube in this class
 */
class Obj // Contains 3D object data
{
   float rho, theta = 0.3F, phi = 1.3F, d, objSize, v11, v12, v13, v21, v22, v23, v32, v33, v43;
   // Elements of viewing matrix V
   Point3D[] w; // World coordinates
   Point2D[] vScr; // Screen coordinates

   Obj() {
      w = new Point3D[8];
      vScr = new Point2D[8];
      // Bottom surface:
      w[0] = new Point3D(1, -1, -1);
      w[1] = new Point3D(1, 1, -1);
```

```
      w[2] = new Point3D(-1, 1, -1);
      w[3] = new Point3D(-1, -1, -1);
      // Top surface:
      w[4] = new Point3D(1, -1, 1);
      w[5] = new Point3D(1, 1, 1);
      w[6] = new Point3D(-1, 1, 1);
      w[7] = new Point3D(-1, -1, 1);
      objSize = (float) Math.sqrt(12F);
      // = sqrt(2 * 2 + 2 * 2 + 2 * 2)
      // = distance between two opposite vertices.
      rho = 5 * objSize; // For reasonable perspective effect
   }

   void initPersp() {
      float costh = (float) Math.cos(theta), sinth = (float) Math.sin(theta), cosph = (float)
      Math.cos(phi),
            sinph = (float) Math.sin(phi);
      v11 = -sinth;
      v12 = -cosph * costh;
      v13 = sinph * costh;
      v21 = costh;
      v22 = -cosph * sinth;
      v23 = sinph * sinth;
      v32 = sinph;
      v33 = cosph;
      v43 = -rho;
   }

   void eyeAndScreen() {
      initPersp();
      for (int i = 0; i < 8; i++) {
         Point3D P = w[i];
         float x = v11 * P.x + v21 * P.y, y = v12 * P.x + v22 * P.y + v32 * P.z,
               z = v13 * P.x + v23 * P.y + v33 * P.z + v43;
         vScr[i] = new Point2D(-d * x / z, -d * y / z);
      }
   }
}
```