```
public class FeynmanPlateView {
   Element plate = new ElementCylinder();
   Element omega = new ElementArrow();
   Element angularMomentum = new ElementArrow();
   Element bodyX = new ElementArrow(); // x-axis in body frame
   Element bodyY = new ElementArrow(); // y-axis in body frame
   ElementTrail trailX = new ElementTrail();
   ElementTrail trailY = new ElementTrail();
   // note how OSP does Greek symbols
   ElementText labelOmega = new ElementText("$\\omega$");
   Group bodyGroup = new Group();
   Display3DFrame frame = new Display3DFrame("Space View");
   FeynmanPlate rigidBody;

   public FeynmanPlateView(FeynmanPlate rigidBody) {
      this.rigidBody = rigidBody; // save a reference to the model
      plate.setSizeXYZ(2, 2, 0.1);
      bodyX.setSizeXYZ(1.4, 0, 0);
      bodyY.setSizeXYZ(0, 1.4, 0);
      frame.setPreferredMinMax(-2, 2, -2, 2, -2, 2);
      frame.setSize(600, 600);
      plate.getStyle().setFillColor(java.awt.Color.LIGHT_GRAY);
      frame.setDecorationType(VisualizationHints.DECORATION_NONE);
      omega.getStyle().setFillColor(java.awt.Color.YELLOW);
      angularMomentum.getStyle().setFillColor(java.awt.Color.BLUE);
      bodyX.getStyle().setResolution(new Resolution(10));
      bodyY.getStyle().setResolution(new Resolution(10));
      bodyX.getStyle().setFillColor(java.awt.Color.RED);
      bodyY.getStyle().setFillColor(java.awt.Color.GREEN);
      trailX.getStyle().setLineColor(java.awt.Color.RED);
      trailY.getStyle().setLineColor(java.awt.Color.GREEN);
      bodyGroup.addElement(plate);
      bodyGroup.addElement(omega);
      bodyGroup.addElement(labelOmega);
      bodyGroup.addElement(bodyX);
      bodyGroup.addElement(bodyY);
      // coordinates in space
      frame.addElement(trailX);
      frame.addElement(trailY);
      frame.addElement(angularMomentum);
      frame.addElement(bodyGroup);
      // scene is simple, so draw it properly when rotating
      frame.setAllowQuickRedraw(false);
      bodyGroup.setTransformation(rigidBody.getTransformation());
   }

   void initialize() {
      trailX.clear();
      trailY.clear();
      update();
   }

   void update() {
      bodyGroup.setTransformation(rigidBody.getTransformation());
      double[] vec =
           new double[] {rigidBody.wx, rigidBody.wy, rigidBody.wz};
      double norm =
           Math.sqrt(vec[0]*vec[0]+vec[1]*vec[1]+vec[2]*vec[2]);
      norm = Math.max(norm, 1.0e-6);
      double s = VectorMath.magnitude(rigidBody.spaceL)/norm;
      // scale omega to same length as angular momentum
      omega.setSizeXYZ(s*vec[0], s*vec[1], s*vec[2]);
      labelOmega.setXYZ(s*vec[0], s*vec[1], s*vec[2]);
      angularMomentum.setSizeXYZ(rigidBody.spaceL);
      double[] tipX = new double[] {bodyX.getSizeX(), 0, 0};
      bodyGroup.toSpaceFrame(tipX);
      trailX.addPoint(tipX[0], tipX[1], tipX[2]);
      double[] tipY = new double[] {0, bodyY.getSizeY(), 0};
      bodyGroup.toSpaceFrame(tipY);
      trailY.addPoint(tipY[0], tipY[1], tipY[2]);
   }
}
```


`FeynmanPlateView` shows a spinning plate as seen from the space frame. The elements needed to visualize the plate as seen in the laboratory (space) frame are instantiated in the constructor. The plate is represented by an `ElementCylinder` object and the x- and y-body axes are `ElementArrow` objects. Additional arrows are used to show the angular momentum and angular velocity. As the body frame axes rotate, we add points to trail objects to display the trajectories of the body axis arrows. These trails show the wobble. The plate, angular velocity arrow, and axes arrows are placed in a group so that their orientation can be set with a single transformation. The angular momentum arrow is added to the 3D frame because it is constant in the space frame. Because the trails are also added to the 3D frame, we will need to transform the tips of the body axes vectors into the space frame in order to add points to the trails.

The `update` method in the `FeynmanPlateView` is invoked after every time step. The method begins by setting the body group's transformation using the transformation computed in `FeynmanPlate`. The body frame's angular velocity components are then scaled so that the angular velocity arrow has the same length as the angular momentum arrow. (Relative arrow length has little meaning because angular velocity has dimension $1/L$ and angular momentum has dimension ML^2/T.) Note that $\boldsymbol{\omega}$ is not transformed separately because this arrow is in the body group and this group has been transformed. However, the coordinates of the tips of the body axes arrows are transformed into the space frame because the trails are not part of the body group.

Problem 17.13 Simulation of a wobbling plate

(a) Run the target class `FeynmanPlateApp`. Does your simulation confirm Feynman's observation of the wobbling plate in the Cornell cafeteria? Are the discrepancies between Feynman's description and the simulation due to the fact that the wobble is not small or due to numerical inaccuracies? Justify your answer.

(b) How well does the algorithm conserve energy and angular momentum?

(c) Will a rectangular food tray wobble the same way as a plate?

(d) Does the plate wobble if it is spun about an axis close to a diameter rather than close to a line perpendicular to the flat side of the disk? ■