```java
package rpn.graphics;

import java.awt.*;
import rpn.doubles.Formats;

/** PlotPaths are the basic class drawn into RealCanvases.
    Drawing follows the PostScript model.
    Each path is an array of segments.
    Each segment is an array of coordinates.
    Stroking a path means drawing lines along the pieces.
    Filling it means turning each piece into a polygon.  */

class NoCurrentPointException extends Exception {
}

public class PlotPath {
    final static int INC = 128;

    RealGraphics gfx;

    Polygon segment[];
    Polygon currentSegment = null;
    int maxSegmentNo = INC;
    int segmentNo;

    // mark last moveto:
    int last[];
    int cp[];
    int N = 0;

    public PlotPath(RealGraphics gfx) {
        this.gfx = gfx;
        segment = new Polygon[INC];
        segmentNo = 0;
    }

    int[] currentPoint() {
        return(cp);
    }

    // - moves ---------------------------------------------

    void movetoPoint(int q[]) {
        if (segmentNo >= maxSegmentNo) {
            // System.out.println("overflow!");
            maxSegmentNo += INC;
            Polygon newseg[] = new Polygon[maxSegmentNo];
            for (int i=0;i<segmentNo;i++) {
                newseg[i] = segment[i];
            }
            segment = newseg;
        }
        segment[segmentNo] = new Polygon();
        currentSegment = segment[segmentNo++];
        currentSegment.addPoint(q[0], q[1]);
        last = q;
        cp = q;
```

```
    }

    public void moveto(double x, double y) {
        int q[] = gfx.toPoint(x, y);
        movetoPoint(q);
    }

    public void moveto(double p[]) {
        int q[] = gfx.toPoint(p);
        movetoPoint(q);
    }

    public void rmoveto(double x[]) throws NoCurrentPointException {
        if (cp != null) {
            int P[] = gfx.toRelativePoint(x);
            P[0] += cp[0];
            P[1] += cp[1];
            movetoPoint(P);
        } else {
            throw new NoCurrentPointException();
        }
    }

    public void rmoveto(double x, double y) throws NoCurrentPointException {
        if (cp != null) {
            int P[] = gfx.toRelativePoint(x, y);
            P[0] += cp[0];
            P[1] += cp[1];
            movetoPoint(P);
        } else {
            throw new NoCurrentPointException();
        }
    }

    // - lines ---------------------------------------------

    public void lineto(double p[]) throws NoCurrentPointException {
        if (cp != null) {
            N++;
            int q[] = gfx.toPoint(p);
            currentSegment.addPoint(q[0], q[1]);
            cp = q;
        } else {
            throw new NoCurrentPointException();
        }
    }

    public void lineto(double x, double y) throws NoCurrentPointException {
        if (cp != null) {
            N++;
            int q[] = gfx.toPoint(x, y);
            currentSegment.addPoint(q[0], q[1]);
            cp = q;
        } else {
            throw new NoCurrentPointException();
        }
    }
```

```
public void rlineto(double x, double y) throws NoCurrentPointException {
    if (cp != null) {
        N++;
        int P[] = gfx.toRelativePoint(x, y);
        cp[0] += P[0];
        cp[1] += P[1];
        currentSegment.addPoint(cp[0], cp[1]);
    } else {
        throw new NoCurrentPointException();
    }
}

// - curves --------------------------------------------

public void curveto(double P[]) throws NoCurrentPointException {
    if (cp != null) {
        double P1[] = gfx.toFinePoint(P[0], P[1]);
        double P2[] = gfx.toFinePoint(P[2], P[3]);
        double P3[] = gfx.toFinePoint(P[4], P[5]);
        Curve.addCurve(currentSegment, P1, P2, P3);
        cp = new int[2];
        cp[0] = (int) P3[0];
        cp[1] = (int) P3[1];
    } else {
        throw new NoCurrentPointException();
    }
}

public void curveto(double P[][]) throws NoCurrentPointException {
    if (cp != null) {
        System.out.println("---------------------");
        System.out.println("P0 = " + RealPoint.toString(P[0]));
        System.out.println("P1 = " + RealPoint.toString(P[1]));
        System.out.println("P2 = " + RealPoint.toString(P[2]));
        P[0] = gfx.toFinePoint(P[0]);
        P[1] = gfx.toFinePoint(P[1]);
        P[2] = gfx.toFinePoint(P[2]);
        Curve.addCurve(currentSegment, P[0], P[1], P[2]);
        cp = new int[2];
        cp[0] = (int) P[2][0];
        cp[1] = (int) P[2][1];
    } else {
        throw new NoCurrentPointException();
    }
}

// -----------------------------------------------------

void drawPath(Polygon P) {
    int n = P.npoints;
    Graphics g = gfx.getGraphics();
    if (n > 0) {
        int x = P.xpoints[0];
        int y = P.ypoints[0];
        for (int i=1;i<P.npoints;i++) {
            int x1 = P.xpoints[i];
```

```
            int y1 = P.ypoints[i];
            g.drawLine(x, y, x1, y1);
            x = x1; y = y1;
        }
    }
}

public void close() {
    if (last != null) {
        currentSegment.addPoint(last[0], last[1]);
        movetoPoint(last);
    }
}

public void stroke() {
    for (int i=0;i<segmentNo;i++) {
        drawPath(segment[i]);
    }
}

public void stroke(Color c) {
    Color tmp = gfx.getColor();
    gfx.setColor(c);
    for (int i=0;i<segmentNo;i++) {
        drawPath(segment[i]);
    }
    gfx.setColor(tmp);
}

public void fill() {
    for (int i=0;i<segmentNo;i++) {
        gfx.fillPolygon(segment[i]);
    }
}

public void fill(Color c) {
    Color tmp = gfx.getColor();
    gfx.setColor(c);
    for (int i=0;i<segmentNo;i++) {
        gfx.fillPolygon(segment[i]);
    }
    gfx.setColor(tmp);
}

public String toString() {
    StringBuffer sb = new StringBuffer();
    for (int i=0;i<segmentNo;i++) {
        Polygon p = segment[i];
        for (int j=0;j<p.npoints;j++) {
            sb.append('(');
            sb.append(Formats.toString(p.xpoints[j]));
            sb.append(',');
            sb.append(Formats.toString(p.ypoints[j]));
            sb.append(')');
        }
    }
    return(new String(sb));
```

```
    }
}
```

```
package rpn.graphics;

import java.awt.*;
import java.util.Stack;
import rpn.doubles.Formats;

interface PathSegment {
    void addTo(PlotPath p);
}

class MovetoSegment implements PathSegment {
    double x[];
    public MovetoSegment(double x[]) {
        this.x = x;
    }
    public void addTo(PlotPath p) {
        p.moveto(x);
    }
}

class LinetoSegment implements PathSegment {
    double x[];
    public LinetoSegment(double x[]) {
        this.x = x;
    }
    public void addTo(PlotPath p) {
        try {
            p.lineto(x);
        } catch(Exception e) { ; }
    }
}

class CurvetoSegment implements PathSegment {
    double x[][];
    public CurvetoSegment(double x[][]) {
        this.x = x;
    }

    public void addTo(PlotPath p) {
        try {
            p.curveto(x);
        } catch(Exception e) { ; }
    }
}

class CloseSegment implements PathSegment {
    public CloseSegment() {
    }

    public void addTo(PlotPath p) {
        p.close();
    }
}

// -- PathConstructor ------------------------------------------

class PathConstructor {
```

```
final static int Quiet = 0,
    Move = 1, Line = 2, Curve = 3;
int state;
double last[];
double currentPoint[];
boolean rendered = false;

Stack segmentStack;

public PathConstructor(RealGraphics g) {
    segmentStack = new Stack();
    state = Quiet;
}

public void setState(int s) {
    state = s;
}

int n = 0;
final static int M = 3;
public void addSegment(double p[]) {
    int n = segmentStack.size();
    switch(state) {
        case Move:  {
            last = p;
            currentPoint = p;
            moveto(p);
            break;
        }
        case Line:  {
            lineto(p);
            currentPoint = p;
            break;
        }
        case Curve:  {
            double q[][] = new double[3][];
            q[0] = new double[2];
            q[0][0] = (2*currentPoint[0] + p[0])/3;
            q[0][1] = (2*currentPoint[1] + p[1])/3;
            q[1] = new double[2];
            q[1][0] = (currentPoint[0] + 2*p[0])/3;
            q[1][1] = (currentPoint[1] + 2*p[1])/3;
            q[2] = p;
            curveto(q);
            System.out.println("current = " + RealPoint.toString(currentPoint));
            System.out.println("q[0] = " + RealPoint.toString(q[0]));
            System.out.println("q[1] = " + RealPoint.toString(q[1]));
            System.out.println("q[2] = " + RealPoint.toString(q[2]));
            currentPoint = p;
            break;
        }
    }
}

// -------------------------------------------

void moveto(double x[]) {
```

```
        PathSegment ds = new MovetoSegment(x);
        segmentStack.push(ds);
    }
    void lineto(double x[]) {
        PathSegment ds = new LinetoSegment(x);
        segmentStack.push(ds);
    }
    void curveto(double x[][]) {
        PathSegment ds = new CurvetoSegment(x);
        segmentStack.push(ds);
    }
    void close() {
        PathSegment ds = new CloseSegment();
        segmentStack.push(ds);
    }

    // ---------------------------------------------

    public void draw(RealGraphics gfx) {
        PlotPath p = new PlotPath(gfx);
        for (int i=0;i<segmentStack.size();i++) {
            PathSegment s = (PathSegment) segmentStack.elementAt(i);
            s.addTo(p);
        }
        p.stroke(Color.lightGray);
    }
}

// -- DrawingPath ----------------------------------------------

class DrawingPath {
    final static int None = 0, Stroke = 1,
        Fill = 2;
    PathSegment segment[];
    Color c;
    int type;

    DrawingPath(PathConstructor p,
            int type, Color c) {
        this.type = type;
        this.c = c;
        int n = p.segmentStack.size();
        segment = new PathSegment[n];
        for (int i=0;i<n;i++) {
            segment[i] = (PathSegment) p.segmentStack.elementAt(i);
        }
        p.rendered = true;
    }
    // --------------------------------------------------

    public void draw(RealGraphics gfx) {
        PlotPath p = new PlotPath(gfx);
        System.out.println("path size = " + segment.length);
        for (int i=0;i<segment.length;i++) {
            segment[i].addTo(p);
        }
        switch(type) {
```

```
            case Stroke:  {
                p.stroke(c);
                break;
            }
            case Fill:  {
                p.fill(c);
                break;
            }
        }
    }

}

// -- DrawCanvas ----------------------------------------------

public class DrawCanvas extends Canvas {
    int W, H, w, h;
    Stack pathStack;
    RealGraphics gfx;
    PathConstructor cp = null;
    Color colour = Color.black;

    public DrawCanvas(int w, int h) {
        this.w = w; this.h = h;
        W = w; H = h;
        resize(w, h);
        setBackground(Color.white);
        pathStack = new Stack();
    }

    /** Reverses the normal ordering of paint() and
        update() for flicker-free performance.  */
    public void paint(Graphics g) {
        update(g);
    }

    Image img;
    /** Usually overridden.  */
    public void update(Graphics g) {
        if (gfx == null) {
            img = createImage(W, H);
            Graphics G = img.getGraphics();
            gfx = new RealGraphics(G, w, h);
        }
        System.out.println("path count = " + pathStack.size());
        for (int i=0;i<pathStack.size();i++) {
            DrawingPath path = (DrawingPath) pathStack.elementAt(i);
            path.draw(gfx);
        }
        if (cp != null && !cp.rendered) {
            cp.draw(gfx);
        }
        g.drawImage(img, 0, 0, null);
        System.out.println("================================");
    }

    // - button events from the frame -------------------------
```

```
    public void setColor(Color c) {
        colour = c;
    }

    public void clear() {
        pathStack = new Stack();
        cp = null;
        gfx.clear();
        repaint();
    }

    public void newPath() {
        cp = new PathConstructor(gfx);
        cp.setState(PathConstructor.Move);
    }

    public void line() {
        if (cp != null) {
            cp.setState(PathConstructor.Line);
        }
    }

    public void curve() {
        if (cp != null) {
            cp.setState(PathConstructor.Curve);
        }
    }

    public void close() {
        if (cp != null) {
            cp.close();
        }
    }

    public void stroke() {
        if (cp != null) {
            DrawingPath dp = new DrawingPath(cp, DrawingPath.Stroke, colour);
            pathStack.push(dp);
        }
        repaint();
    }

    public void fill() {
        if (cp != null) {
            DrawingPath dp = new DrawingPath(cp, DrawingPath.Fill, colour);
            pathStack.push(dp);
        }
        repaint();
    }

    // - mouse events ------------------------------------

    public boolean mouseDown(Event e, int x, int y) {
        double P[] = gfx.toRealPoint(x, y);
        System.out.println("mouse at " + RealPoint.toString(P));
        cp.addSegment(P);
```

```
        repaint();
        return(true);
    }

    public boolean mouseDrag(Event e, int x, int y) {
        return(true);
    }

    public boolean mouseUp(Event e, int x, int y) {
        return(true);
    }
}
```