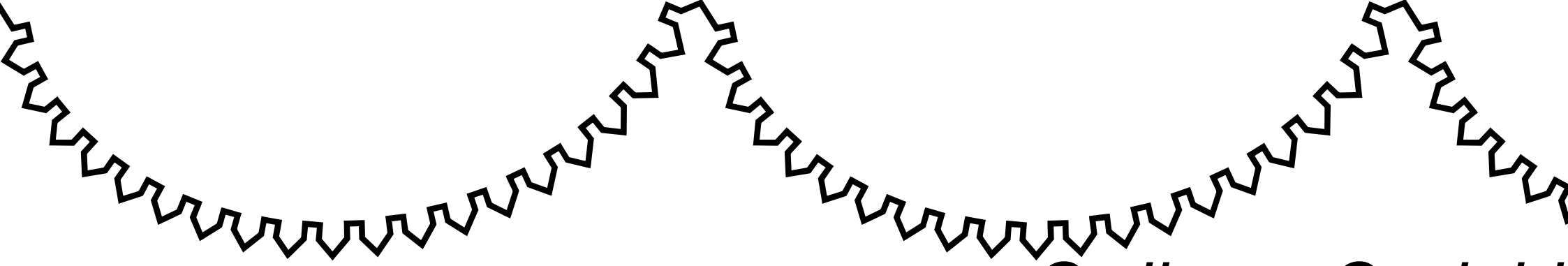

Ordinary Cycloid

Source code:

```
import java.awt.Rectangle;
import java.awt.Shape;
import java.awt.geom.AffineTransform;
import java.awt.geom.PathIterator;
import java.awt.geom.Point2D;
import java.awt.geom.Rectangle2D;

public class MyShape implements Shape {
    private int centerX;
    private int centerY;
    private int param;

    MyShape(int centerX, int centerY, int param) {
        this.centerX = centerX;
        this.centerY = centerY;
        this.param = param;
    }

    @Override
    public Rectangle getBounds() {
        return null;
    }

    @Override
    public Rectangle2D getBounds2D() {
        return null;
    }

    @Override
    public boolean contains(double x, double y) {
        return false;
    }

    @Override
    public boolean contains(Point2D p) {
        return false;
    }

    @Override
    public boolean intersects(double x, double y, double w, double h) {
        return false;
    }

    @Override
    public boolean intersects(Rectangle2D r) {
        return false;
    }

    @Override
    public boolean contains(double x, double y, double w, double h) {
        return false;
    }

    @Override
    public boolean contains(Rectangle2D r) {
        return false;
    }

    @Override
    public PathIterator getPathIterator(AffineTransform at) {
        return new ShapeIterator();
    }

    @Override
    public PathIterator getPathIterator(AffineTransform at, double flatness) {
        return new ShapeIterator();
    }

    class ShapeIterator implements PathIterator {
        boolean done = false;
        double h = Math.PI / 500;
        boolean start = true;
        double t = -2;

        @Override
        public int getWindingRule() {
            return 0;
        }

        @Override
        public boolean isDone() {
            return done;
        }

        @Override
        public void next() {
            t += h;
        }

        @Override
        public int currentSegment(float[] coordinate) {
            if (start) {
                coordinate[0] = (float) (param * t - param * Math.sin(t)); // x
                coordinate[1] = (float) (param - param * Math.cos(t)); // y
                start = false;
                return SEG_MOVETO;
            }
            if (t >= 10*Math.PI) {
                done = true;
                return SEG_CLOSE;
            }
            coordinate[0] = (float) (param * t - param * Math.sin(t)); // x
            coordinate[1] = (float) (param - param * Math.cos(t)); // y
            return SEG_LINETO;
        }

        @Override
        public int currentSegment(double[] coordinate) {
            if (start) {
                coordinate[0] = (double) (param * t - param * Math.sin(t)); // x
                coordinate[1] = (double) (param - param * Math.cos(t)); // y
                start = false;
                return SEG_MOVETO;
            }
            if (t >= 10*Math.PI) {
                done = true;
                return SEG_CLOSE;
            }
            coordinate[0] = (double) (param * t - param * Math.sin(t)); // x
            coordinate[1] = (double) (param - param * Math.cos(t)); // y
            return SEG_LINETO;
        }
    }
}
```