```
package structure;


import structure.exceptions.PointException;

import structure.exceptions.TriangleException;

import structure.exceptions.VectorException;


public class Triangle {


	public Triangle(Vector<Point> v) throws TriangleException {

		if (v == null)

			throw new TriangleException(new String("null vector in argument"));


		if (v.size() != 3)

			throw new TriangleException(new String("vertices count exception"));


		try {

			if (pool.get(0) == null || pool.get(1) == null || pool.get(2) == null)

				throw new TriangleException(new String("one of vertices is
null"));

		} catch (VectorException e) {

			e.printStackTrace();

		}

		pool = v;

	}


	public Triangle(Point a, Point b, Point c) {
        try {
            pool = new Vector<Point>(3);
        } catch (VectorException e) {
            e.printStackTrace();
        }
        pool.set(0, a);
        pool.set(1, b);
        pool.set(2, c);
    }

    public void replaceVertice(int n, Point p) throws TriangleException {
        if (!(n >= 0 && n < 3) || p == null)
            throw new TriangleException(new String("wrong index or Point"));
        pool.set(n, p);
    }

    public Point getVertice(int n) throws TriangleException {
        if (!(n >= 0 && n < 3) )
            throw new TriangleException(new String("wrong number"));
        Point res = null;
        try {
            res = pool.get(n);
        } catch (VectorException e) {
            e.printStackTrace();
        }
        return res;
    }

    public double perimeter() throws TriangleException {
        double p = 0;

        try {
            Point last = pool.get(pool.size() - 1);
            for (int i = 0; i < pool.size(); ++i) {
                Point cur = pool.get(i);
                try {
                    p += last.distance(cur);
                } catch (PointException e) {
                    e.printStackTrace();
                }
                last = cur;
            }
        } catch(VectorException e) {
            throw new TriangleException(new String());
        }
        return p;
    }
```

```
public double square() throws TriangleException {

    double p = perimeter() / 2;
    double s = p;

    try {
        Point last = pool.get(pool.size() - 1);
        for (int i = 0; i < pool.size(); ++i) {
            Point cur = pool.get(i);
            try {
                double dist = last.distance(cur);
                s *= p - dist;
            } catch (PointException e) {
                e.printStackTrace();
            }
            last = cur;
        }
    } catch (VectorException e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }

    return Math.sqrt(s);
```

```
        }

        protected Vector<Point> pool;

}
```