```java
import components.random.Random;

/**
 * Monte Carlo Estimate: compute percentage of pseudo-random points in [0.0,1.0)
 * interval that fall in the left half subinterval [0.0,0.5).
 */
public final class MonteCarlo {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private MonteCarlo() {
    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments; unused here
     */
    public static void main(String[] args) {
        /*
         * Open input and output streams
         */
        SimpleReader input = new SimpleReader1L();
        SimpleWriter output = new SimpleWriter1L();
        /*
         * Ask user for number of points to generate
         */
        output.print("Number of points: ");
        int n = input.nextInteger();
        /*
         * Declare counters and initialize them
         */
        int ptsInInterval = 0, ptsInSubinterval = 0;
        /*
         * Generate points and count how many fall in circle's interval
         */
        ptsInSubinterval = numberOfPointsInCircle(n);
        ptsInInterval = n;
        /*
         * Estimate area of circle by multiplying area of square by number of
         * points in the circle then dividing by total number of points
         */
        double area = Math.pow(2, 2) * ptsInSubinterval / ptsInInterval;
        output.println("Area of the circle is " + area);
        /*
         * Close input and output streams
         */
        input.close();
        output.close();
    }

    /**
     * Checks whether the given point (xCoord, yCoord) is inside the circle of
     * radius 1.0 centered at the point (1.0, 1.0).
     *
     * @param xCoord
     *            the x coordinate of the point
     * @param yCoord
     *            the y coordinate of the point
     * @return true if the point is inside the circle, false otherwise
     */
    private static boolean pointIsInCircle(double xCoord, double yCoord) {
        boolean ans = false;

        double distance = Math
                .sqrt(Math.pow(1 - xCoord, 2) + Math.pow(1 - yCoord, 2));
        if (distance <= 1.0) {
            ans = true;
        }

        return ans;
    }

    /**
     * Generates n pseudo-random points in the [0.0,2.0) x [0.0,2.0) square and
     * returns the number that fall in the circle of radius 1.0 centered at the
     * point (1.0, 1.0).
     *
     * @param n
     *            the number of points to generate
     * @return the number of points that fall in the circle
     */
    private static int numberOfPointsInCircle(int n) {
        Random rndX = new Random1L();
        Random rndY = new Random1L();
        int count = 0;
        int i = 0;

        while (i < n) {
            double x = 2 * rndX.nextDouble();
            double y = 2 * rndY.nextDouble();

            if (pointIsInCircle(x, y)) {
                count++;
            }

            i++;
        }

        return count;
    }

}
```