```java
import components.random.Random;
import components.random.Random1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * 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() {
    }

    /**
     * * 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) {
        Random rnd = new Random1L();
        double x = 2 * rnd.nextDouble();
        double y = 2 * rnd.nextDouble();
        boolean fax = false;
        if (x == 1.0 && y == 1.0) {
            fax = true;
        } else {
            fax = false;
        }

        return fax;

    }

    /**
     * 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) {
        int pts2 = 0, pts = 0;
        /*
         * Create pseudo-random number generator
         */
        Random rnd = new Random1L();
        double y = 0.0;
        double x = 0.0;
        /*
```

```
58          * Generate points and count how many fall in [0.0,0.5) interval
59          */
60         while (pts2 < n) {
61             /*
62              * Generate pseudo-random number in [0.0,1.0) interval
63              */
64             x = 2 * rnd.nextDouble();
65             y = 2 * rnd.nextDouble();
66             /*
67              * Increment total number of generated points
68              */
69             pts2++;
70             /*
71              * Check if point is in [0.0,0.5) interval and increment counter if
72              * it is
73              */
74             if ((x - 1) * (x - 1) + (y - 1) * (y - 1) <= 1) {
75                 pts++;
76             }
77         }
78
79         return pts;
80     }
81
82     /**
83      * Main method.
84      *
85      * @param args
86      *            the command line arguments; unused here
87      */
88     public static void main String[] args) {
89         /*
90          * Open input and output streams
91          */
92         SimpleReader input = new SimpleReader1L();
93         SimpleWriter output = new SimpleWriter1L();
94         /*
95          * Ask user for number of points to generate
96          */
97         output.print("Number of darts: ");
98         int n = input.nextInteger();
99         /*
100         * Declare counters and initialize them
101         */
102        int ptsInInterval = 0, ptsInSubinterval = 0;
103        /*
104         * Create pseudo-random number generator
105         */
106        Random rnd = new Random1L();
107        double y = 0.0, x = 0.0;
108        int i = 0;
109        /*
110         * Generate points and count how many fall in [0.0,0.5) interval
111         */
112        while (ptsInInterval < n) {
113            /*
114             * Generate pseudo-random number in [0.0,1.0) interval
```

```
115             */
116            x = 2 * rnd.nextDouble();
117            y = 2 * rnd.nextDouble();
118            /*
119             * Increment total number of generated points
120             */
121            ptsInInterval++;
122            /*
123             * Check if point is in [0.0,0.5) interval and increment counter if
124             * it is
125             */
126            if (pointIsInCircle(x, y) == true) {
127                ;
128            }
129            {
130                i++;
131            }
132            if ((x - 1) * (x - 1) + (y - 1) * (y - 1) <= 1) {
133                ptsInSubinterval++;
134            }
135        }
136
137        output.println(
138                "Number of darts in the circle: " + numberOfPointsInCircle(n));
139
140        /*
141         * Estimate percentage of points generated in [0.0,1.0) interval that
142         * fall in the [0.0,0.5) subinterval
143         */
144        double estimate = (numberOfPointsInCircle(n) / (1.0 * i));
145        output.println(
146                "Estimate of darts that hit 1x1 (absolute center) of the ones that were in
    radius 1 of the center: "
147                        + estimate + "%");
148        /*
149         * Close input and output streams
150         */
151        input.close();
152        output.close();
153    }
154
155}
```