```
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;
```

```
    /*
     * Generate points and count how many fall in [0.0,0.5) interval
     */
    while (pts2 < n) {
        /*
         * Generate pseudo-random number in [0.0,1.0) interval
         */
        x = 2 * rnd.nextDouble();
        y = 2 * rnd.nextDouble();
        /*
         * Increment total number of generated points
         */
        pts2++;
        /*
         * Check if point is in [0.0,0.5) interval and increment counter if
         * it is
         */
        if ((x - 1) * (x - 1) + (y - 1) * (y - 1) <= 1) {
            pts++;
        }
    }

    return pts;
}

/**
 * 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 darts: ");
  int n = input.nextInteger();
  /*
   * Declare counters and initialize them
   */
  int ptsInInterval = 0, ptsInSubinterval = 0;
  /*
   * Create pseudo-random number generator
   */
  Random rnd = new Random1L();
  double y = 0.0, x = 0.0;
  int i = 0;
  /*
   * Generate points and count how many fall in [0.0,0.5) interval
   */
  while (ptsInInterval < n) {
    /*
     * Generate pseudo-random number in [0.0,1.0) interval
```

```
     */
    x = 2 * rnd.nextDouble();
    y = 2 * rnd.nextDouble();
    /*
     * Increment total number of generated points
     */
    ptsInInterval++;
    /*
     * Check if point is in [0.0,0.5) interval and increment counter if
     * it is
     */
    if (pointIsInCircle(x, y) == true) {
        ;
    }
    {
        i++;
    }
    if ((x - 1) * (x - 1) + (y - 1) * (y - 1) <= 1) {
        ptsInSubinterval++;
    }
}

output.println(
        "Number of darts in the circle: " + numberOfPointsInCircle(n));

/*
 * Estimate percentage of points generated in [0.0,1.0) interval that
 * fall in the [0.0,0.5) subinterval
 */
```

```
        double estimate = ((1.0 * i) / numberOfPointsInCircle(n));
        output.println(
                "Estimate of darts that hit 1x1 (absolute center) of the ones that were in radius 1 of the center:
"
                        + estimate + "%");
        /*
         * Close input and output streams
         */
        input.close();
        output.close();
    }


}
```