```java
//******************************************************************************
//
// File:    Simulation.java
// Package: ---
// Unit:    Class Simulation
//
//******************************************************************************

import edu.rit.pj2.Loop;

/**
 * Class Simulation takes the necessary input to run a specified number of
 * simulations generating random graphs and averaging the distance over all
 * the graphs.
 *
 * @author  Jimi Ford
 * @version 2-15-2015
 */
public class Simulation {

    // private data members
    private int v, n;
    private double p;
    private Task ref;
    private long seed;
    private DoubleVbl.Mean average;

    /**
     * Construct a simulation object
     *
     * @param   ref reference to the Task program in order to utilize its
     *          parallelFor loop
     * @param seed the seed value for the PRNG
     * @param v number of vertices in the graph
     * @param p edge probability of any two vertices being connected
     * @param n number of simulations to run (or graphs to generate)
     */
    public Simulation(Task ref, long seed, int v, double p, int n) {
        this.v = v;
        this.p = p;
        this.n = n;
        this.seed = seed;
        this.ref = ref;
        this.average = new DoubleVbl.Mean();
    }


    /**
     * Loop through the <I>n</I> simulations and accumulate the distances
     * between each pair of vertices. The looping in this method is where
     * most of the computation takes place, so to combat this, a parallel
     * loop is used.
     *
     * @return the results of the <I>n</I> simulations
     */
    public SimulationResult simulate() {
        // run "n" simulations
        this.ref.parallelFor(0, n - 1).exec(new Loop() {
```

```
62              Random prng;
63              DoubleVbl.Mean thrAverage;
64
65              @Override
66              public void start() {
67                  prng = new Random(seed + rank());
68                  thrAverage = threadLocal(average);
69              }
70
71              @Override
72              public void run(int i) {
73                  UndirectedGraph.randomGraph(prng, v, p).
74                      accumulateDistances(thrAverage);
75              }
76
77          });
78
79          return new SimulationResult(v, p, average.doubleValue());
80      }
81  }
82
```