tion, so that is the one considered first. The following properties hold if the sequence 0 … 999 is a true uniform distribution:

- The first number is equally likely to be 0, 1, 2, …, 999.
- The *i*th number is equally likely to be 0, 1, 2, …, 999.
- The average of all the generated numbers is 499.5.

```
 1 // Random class
 2 //
 3 // CONSTRUCTION: with (a) no initializer or (b) an integer
 4 //     that specifies the initial state of the generator
 5 //
 6 // ******************PUBLIC OPERATIONS*********************
 7 //     Return a random number according to some distribution:
 8 // int randomInt( )                   --> Uniform, 1 to 2^31-1
 9 // double randomReal( )               --> Uniform, 0..1
10 // int randomInt( int low, int high ) --> Uniform low..high
11 // int poisson( double expectedVal )  --> Poisson
12 // double negExp( double expectedVal )--> Negative exponential
13 //     A related static method:
14 // void permute( Object [ ] a )       --> Randomly permute
15
16 /**
17  * Random number class, using a 31-bit
18  * linear congruential generator.
19  * Note that java.util contains a class Random,
20  * so watch out for name conflicts.
21  */
22 public class Random
23 {
24     public Random( )
25       { /* Figure 9.2 */ }
26     public Random( int initialState )
27       { /* Figure 9.2 */ }
28     public int randomInt( )
29       { /* Figure 9.3 */ }
30     public double randomReal( )
31       { return randomInt( ) / (double) M; }
32     public int randomInt( int low, int high )
33       { /* Exercise 9.8 */ }
34     public int poisson( double expectedValue )
35       { /* Figure 9.5 */ }
36     public double negExp( double expectedValue )
37       { /* Figure 9.6 */ }
38     public static final void permute( Object [ ] a )
39       { /* Figure 9.7 */}
40
41     private int state;
42 }
```

Figure 9.1 Skeleton for random-number generator class