Random Number Generation in Java

Need some randomness in your numbers? Check out a few different techniques you can avail yourself of to create random (or mostly random) numbers in Java.

While developing applications, we often need to generate random numbers. Java provides support for generating random numbers primarily through the java.lang.Math and java.util.Random classes.

In this post, I will discuss different ways to generate random numbers based on different types of requirements.

Random Numbers Using the Math Class

Java provides the Math class in the java.util package to generate random numbers.

The Math class contains the static Math.random() method to generate random numbers of the double type.

The random() method returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. When you call Math.random(), under the hood, a java.util.Random pseudorandom-number generator object is created and used.

You can use the Math.random() method with or without passing parameters. If you provide parameters, the method produces random numbers within the given parameters.

The code to use the Math.random() method:

```java
public static double getRandomNumber(){
    double x = Math.random();
    return x;
}
```

The getRandomNumber() method uses the Math.random() method to return a positive double value that is greater than or equal to 0.0 and less than 1.0.

The output of running the code is:

```
Double between 0.0 and 1.0: SimpleRandomNumber = 0.21753313144345698
```

Random Numbers Within a Given Range

For generating random numbers between a given a range, you need to specify the range. A standard expression for accomplishing this is:

```
(Math.random() * ((max - min) + 1)) + min
```

Let us break this expression into steps:

1. First, multiply the magnitude of the range of values you want to cover by the result that Math.random() produces.

   `Math.random() * ( max - min )` returns a value in the range [0, max – min] where max is excluded. For example, if you want [5,10], you need to cover 5 integer values so you can use Math.random()*5. This would return a value in the range [0,5], where 5 is not included.
2. Next, shift this range up to the range that you are targeting. You do this by adding the min value.

`(Math.random() * ( max - min )) + min`

But this still does not include the maximum value.

- To get the max value included, you need to add 1 to your range parameter (max - min). This will return a random double within the specified range.

`double x = (Math.random()*((max-min)+1))+min;`

There are different ways of implementing the above expression. Let us look at a couple of them.

Random Double Within a Given Range

By default, the Math.random() method returns a random number of the type double whenever it is called. The code to generate a random double value between a specified range is:

```
public static double getRandomDoubleBetweenRange(double min, double max){
    double x = (Math.random()*((max-min)+1))+min;
    return x;
}
```

You can call the preceding method from the main method by passing the arguments like this.

```
System.out.println("Double between 5.0 and 10.00: RandomDoubleNumber = "+getR
andomDoubleBetweenRange(5.0, 10.00));
```

The output is this.

```
System.out.println("Double between 5.0 and 10.00: RandomDoubleNumber = "+getR
andomDoubleBetweenRange(5.0, 10.00));
```

Random Integer Within a Given Range

The code to generate a random integer value between a specified range is this.

```
public static double getRandomIntegerBetweenRange(double min, double max){
    double x = (int)(Math.random()*((max-min)+1))+min;
    return x;
}
```

The preceding getRandomIntegerBetweenRange() method produces a random integer between the given range. As Math.random() method generates random numbers of double type, you need to truncate the decimal part and cast it to int in order to get the integer random number. You can call this method from the main method by passing the arguments as follows:

```
System.out.println("Integer between 2 and 6: RandomIntegerNumber = "+getRando
mIntegerBetweenRange(2,6));
```

The output is this.

```
Integer between 2 and 6: RandomIntegerNumber = 5
```

Note: You can pass a range of negative values to generate a random negative number within the range.

Random Number Generation Using the Random Class

You can use the java.util.Random class to generate random numbers of different types, such as int, float, double, long, and boolean.

To generate random numbers, first, create an instance of the Random class and then call one of the random value generator methods, such as nextInt(), nextDouble(), or nextLong().

The nextInt() method of Random accepts a bound integer and returns a random integer from 0 (inclusive) to the specified bound (exclusive).

The code to use the nextInt() method is this.

```
public static int generateRandomInt(int upperRange){
    Random random = new Random();
    return random.nextInt(upperRange);
}
```

The code to use the nextInt() method to generate an integer within a range is:

```
public static int generateRandomIntIntRange(int min, int max) {
    Random r = new Random();
    return r.nextInt((max - min) + 1) + min;
}
```

The nextFloat() and nextDouble() methods allow generating float and double values between 0.0 and 1.0.

The code to use both the methods is:

```
public static double generateRandomDouble(){
    Random random = new Random();
    return random.nextDouble();
}
public static float generateRandomFloat(){
    Random random = new Random();
    return random.nextFloat();
}
```

Random Number Generation Features in Java 8

Java 8 introduced a new method, ints(), in the java.util.Random class. The ints() method returns an unlimited stream of pseudorandom int values. You can limit the random numbers between a specified range by providing the minimum and the maximum values.

The code to use the Random.ints() method to generate random integer values within a specified range is this.

```
public static int getRandomNumberInts(int min, int max){
    Random random = new Random();
    return random.ints(min,(max+1)).findFirst().getAsInt();
}
```

The getRandomNumberInts() method generates a stream of random integers between the min (inclusive) and max (exclusive). As ints() method produces an IntStream, the code calls the findFirst() method that returns an OptionalInt object that describes the first element of this stream. The code then calls the getAsInt()method to return the int value in OptionalInt.

The code to use the Random.ints() method to generate a stream of specified random integer values is:

```
public static void getStreamOfRandomInts(int num) {
    Random random = new Random();
    random.ints(num).sorted().forEach(System.out::println);
}
```

The code to call the preceding method is:

```
System.out.println("Random int stream: RandomIntStreamofSize = ");
RandomDemo.getStreamOfRandomInts(5);
```

The output of the preceding code is:

```
Random int stream: RandomIntStreamofSize =
-1861317227
-1205557317
453883217
762357682
1725970934
```

The code to use the Random.ints() method to generate a stream of a specified number of random integer values between a range is:

```
public static void getStreamOfRandomIntsWithRange(int num, int min, int max)
{
    Random random = new Random();
    random.ints(num,min, max).sorted().forEach(System.out::println);
}
```

The code to call the preceding method is:

```
System.out.println("Random int stream of specified size in range: RandomIntSt
reamofSizeInRange = ");
RandomDemo.getStreamOfRandomIntsWithRange(5,0,10);
```

The output of the preceding code is:

```
Random int stream of specified size in range: RandomIntStreamofSizeInRange =
2
2
3
4
6
```

In addition to ints(), some other frequently used methods that Java 8 introduced to the Random class — which can return a sequential stream of random numbers — are:

- LongStream longs()
- DoubleStream doubles()

Summary

The java.util.Random class implements what is generally called a linear congruential generator (LCG). It is designed to be fast but does not meet requirements for real-time use, such as use in unique session ID generation on a web server, scientific experiments, cryptography, or lotteries and sweepstakes where a monetary stake is involved. For such scenarios, there are other alternatives, which I will cover in a later post.

For the impatient readers, you can have a look at the SecureRandom class and Xorshift random number generators.

Also, an interesting resource is random.org, a true random number service that generates randomness via atmospheric noise.