链滴

JAVA 随机数生成 Int、Long、Float、Double

作者：tomyli

原文链接：https://ld246.com/article/1497079228098

来源网站：链滴

<h2 id="toc_0">随机数Int的生成
#60;/h2>

<h3 id="toc_1">生成无边界的Int
#60;/h3>

<pre><code>
Test
public void testRandom_generatingIntegerUnbounded() throws Exception {

int intUnbounded = new Random().nextInt();

System.out.println(intUnbounded);

}

</code></pre>

<h3 id="toc_2">生成有边界的Int
#60;/h3>

<pre><code>
Test
public void testRandom_generatingIntegerBounded_withRange() throws Exception {

int min = 1;

int max = 10;

int intBounded = min + ((int) (new Random().nextFloat() * (max - min)));

System.out.println(intBounded);

}

</code></pre>

<blockquote>

<p>包含1而不包含10<
p>

</blockquote>

<h3 id="toc_3">使用Apache Common Math来生成有边界的
nt</h3>

<pre><code>
Test
public void testRandom_generatingIntegerBounded_withApacheMath() throws Exception {

int min = 1;

int max = 10;

int intBounded = new RandomDataGenerator().nextInt(min, max);

System.out.println(intBounded);

}

</code></pre>

<blockquote>

<p>包含1且包含10</p
#62;

</blockquote>

<h3 id="toc_4">使用Apache Common Lang的工具类来生
有边界的Int</h3>

<pre><code>
Test

public void testRandom_generatingIntegerBounded_withApacheLangInclusive() throws Except
on {

int min = 1;

int max = 10;

int intBounded = RandomUtils.nextInt(min, max);

System.out.println(intBounded);

}

</code></pre>

<blockquote>

<p>包含1而不包含10<
p>

</blockquote>

<h3 id="toc_5">使用TreadLocalRandom来生成有边界的Int
#60;/h3>

<pre><code>
Test

public void testRandom_generatingIntegerBounded_withThreadLocalRandom() throws Except
on {

int min = 1;

int max = 10;

int threadIntBound = ThreadLocalRandom.current().nextInt(min, max);

System.out.println(threadIntBound);

}

</code></pre>

<blockquote>

<p>包含1而不包含10<

p>

</blockquote>

<hr/>

<h2 id="toc_6">随机数Long的生成
#60;/h2>

<h3 id="toc_7">生成无边界的Long
#60;/h3>

<pre><code>
Test

public void testRandom_generatingLongUnbounded() throws Exception {

long unboundedLong = new Random().nextLong();

System.out.println(unboundedLong);

}

</code></pre>

<blockquote>

<p>因为Random类使用的种子是48bits，所以nextLong不能返回所有可能
long值，long是64bits。</p>

</blockquote>

<h3 id="toc_8">生成有边界的Long
#60;/h3>

<pre><code>
Test

public void testRandom_generatingLongBounded_withRange() throws Exception {

long min = 1;

long max = 10;

long rangeLong = min + (((long) (new Random().nextDouble() * (max - min))));

System.out.println(rangeLong);

}

</code></pre>

<blockquote>

<p>以上只会生成1到10的long类型的随机数<
p>

</blockquote>

<h3 id="toc_9">使用Apache Commons Math来生成有边界
Long</h3>

<pre><code>
Test
public void testRandom_generatingLongBounded_withApacheMath() throws Exception {

long min = 1;

long max = 10;

long rangeLong = new RandomDataGenerator().nextLong(min, max);

System.out.println(rangeLong);

}

</code></pre>

<blockquote>

<p>此方式主要使用的<
 href="http://commons.apache.org/proper/commons-math/javadocs/api-3.6/org/apache/co
mons/math3/random/RandomDataGenerator.html">RandomDataGenerator&#6
;/a>类提供的生成随机数的方法</p
#62;

</blockquote>

<h3 id="toc_10">使用Apache Commons Lang的工具类来
成有边界的Long</h3>

<pre><code>
Test
public void testRandom_generatingLongBounded_withApacheLangInclusive() throws Excepti
n {

long min = 1;

long max = 10;

long longBounded = RandomUtils.nextLong(min, max);

System.out.println(longBounded);

}

</code></pre>

<blockquote>

<p><a href=" https://commons.
pache.org/proper/commons-lang/javadocs/api-3.3/org/apache/commons/lang3/RandomUtils
html">RandomUtils
供了对java.util.Random的补充</p
#62;

</blockquote>

<h3 id="toc_11">使用ThreadLocalRandom生成有边界的L
ng</h3>

<pre><code>

Test
public void testRandom_generatingLongBounded_withThreadLocalRandom() throws Exceptio
{
long min = 1;
long max = 10;
long threadLongBound = ThreadLocalRandom.current().nextLong(min, max);
System.out.println(threadLongBound);
}

</code></pre>

<hr/>

<h2 id="toc_12">随机数Float的生成
#60;/h2>

<h3 id="toc_13">生成0.0-1.0之间的Float随机数
#60;/h3>

<pre><code>
Test
public void testRandom_generatingFloat0To1() throws Exception {
float floatUnbounded = new Random().nextFloat();
System.out.println(floatUnbounded);
}

</code></pre>

<blockquote>
<p>以上只会生成包含0.0而不包括1.0的float类型随机数<
p>
</blockquote>

<h3 id="toc_14">生成有边界的Float随机数
#60;/h3>

<pre><code>
Test
public void testRandom_generatingFloatBounded_withRange() throws Exception {
float min = 1f;
float max = 10f;
float floatBounded = min + new Random().nextFloat() * (max - min);
System.out.println(floatBounded);
}

</code></pre>

<h3 id="toc_15">使用Apache Common Math来生成有边
的Float随机数</h3>

<pre><code>
Test

```
public void testRandom_generatingFloatBounded_withApacheMath() throws Exception {

float min = 1f;

float max = 10f;

float randomFloat = new RandomDataGenerator().getRandomGenerator().nextFloat();

float generatedFloat = min + randomFloat * (max - min);

System.out.println(generatedFloat);

}
```

</code></pre>

<h3 id="toc_16">使用Apache Common Lang来生成有边
的Float随机数</h3>

<pre><code>
Test

```
public void testRandom_generatingFloatBounded_withApacheLang() throws Exception {

float min = 1f;

float max = 10f;

float generatedFloat = RandomUtils.nextFloat(min, max);

System.out.println(generatedFloat);

}
```

</code></pre>

<h3 id="toc_17">使用ThreadLocalRandom生成有边界的Fl
at随机数</h3>

<blockquote>

<p>ThreadLocalRandom类没有提供<
p>

</blockquote>

<hr/>

<h2 id="toc_18">随机数Double的生成
#60;/h2>

<h3 id="toc_19">生成0.0d-1.0d之间的Double随机数
#60;/h3>

<pre><code>
Test
public void testRandom_generatingDouble0To1() throws Exception {

double generatorDouble = new Random().nextDouble();

System.out.println(generatorDouble);

}

</code></pre>

<blockquote>

<p>与Float相同，以上方法只会生成包含0.0d而不包含1.0d的随机数
#60;/p>

</blockquote>

<h3 id="toc_20">生成带有边界的Double随机数
#60;/h3>

<pre><code>
Test
public void testRandom_generatingDoubleBounded_withRange() throws Exception {

double min = 1.0;

double max = 10.0;

double boundedDouble = min + new Random().nextDouble() * (max - min);

System.out.println(boundedDouble);

assertThat(boundedDouble, greaterThan(min));

assertThat(boundedDouble, lessThan(max));

}

</code></pre>

<h3 id="toc_21">使用Apache Common Math来生成有边
的Double随机数</h3>

<pre><code>
Test
public void testRandom_generatingDoubleBounded_withApacheMath() throws Exception {

double min = 1.0;

double max = 10.0;

double boundedDouble = new RandomDataGenerator().getRandomGenerator().nextDouble();

double generatorDouble = min + boundedDouble * (max - min);

System.out.println(generatorDouble);

assertThat(generatorDouble, greaterThan(min));

assertThat(generatorDouble, lessThan(max));

}

</code></pre>

<h3 id="toc_22">使用Apache Common Lang生成有边界
Double随机数</h3>

<pre><code>
Test
public void testRandom_generatingDoubleBounded_withApacheLang() throws Exception {

double min = 1.0;

double max = 10.0;

double generatedDouble = RandomUtils.nextDouble(min, max);

System.out.println(generatedDouble);

}

</code></pre>

<h3 id="toc_23">使用ThreadLocalRandom生成有边界的D
uble随机数</h3>

<pre><code>
Test
public void testRandom_generatingDoubleBounded_withThreadLocalRandom() throws Except
on {

double min = 1.0;

double max = 10.0;

double generatedDouble = ThreadLocalRandom.current().nextDouble(min, max);

System.out.println(generatedDouble);

}

</code></pre>

<h2 id="toc_24">JAVA中有多少可以实现随机数的类或方法?
#60;/h2>

java.util.Random 这个类提供了生成Bytes、Int、Long、Float、Doubl
、Boolean的随机数的方法</li&#6
;

java.util.Math.random 方法提供了生成Double随机数的方法，这个方
的内部实现也是调用了java.util.Random的nextDouble方法，只不过它对多线程进行了更好的支持
在多个线程并发时会减少每个随机数生成器的竞争<
li>

第三方工具类，如Apache Common Lang库与Apache Common Mat
库中提供的随机数生成类，真正使用一行代码来实现复杂的随机数生成<
li>

java.util.concurrent.ThreadLocalRandom 专为多线程并发使用的随机生成器，使用的方法为ThreadLocalRandom.current.nextInt()，此类是在JDK1.7中提供的，并且特适合ForkJoinTask框架，而且在这个类中直接提供了生成有边界的随机数的操作，如<ode>public int nextInt(int origin, int bound)<code>，这样也可以一行代码来实现复杂的随机数生成了。

<blockquote>

<p>最后的总结为单线程中使用java.util.Random类，在多线程中使用java.utl.concurrent.ThreadLocalRandom类。</p#62;

</blockquote>

<h2 id="toc_25">总结<h2>

<p>JAVA在不JDK升级中不断在完善API，现在可以使用JDK原生的API写出雅的代码了。所有的这些测试完整的代码在这里#60;/p>