Module java.base
Package java.util.concurrent

Class ThreadLocalRandom

java.lang.Object
    java.util.Random
        java.util.concurrent.ThreadLocalRandom

All Implemented Interfaces:

Serializable, RandomGenerator

```
public final class ThreadLocalRandom
extends Random
```

A random number generator (with period 2^{64}) isolated to the current thread. Like the global `Random` generator used by the `Math` class, a `ThreadLocalRandom` is initialized with an internally generated seed that may not otherwise be modified. When applicable, use of `ThreadLocalRandom` rather than shared `Random` objects in concurrent programs will typically encounter much less overhead and contention. Use of `ThreadLocalRandom` is particularly appropriate when multiple tasks (for example, each a `ForkJoinTask`) use random numbers in parallel in thread pools.

Usages of this class should typically be of the form: `ThreadLocalRandom.current().nextX(...)` (where `X` is `Int`, `Long`, etc). When all usages are of this form, it is never possible to accidentally share a `ThreadLocalRandom` across multiple threads.

This class also provides additional commonly used bounded random generation methods.

Instances of `ThreadLocalRandom` are not cryptographically secure. Consider instead using `SecureRandom` in security-sensitive applications. Additionally, default-constructed instances do not use a cryptographically random seed unless the system property `java.util.secureRandomSeed` is set to `true`.

Since:

1.7

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces declared in interface java.util.random.RandomGenerator

RandomGenerator.ArbitrarilyJumpableGenerator, RandomGenerator.JumpableGenerator, RandomGenerator.LeapableGenerator, RandomGenerator.SplittableGenerator, RandomGenerator.StreamableGenerator

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
static ThreadLocalRandom	current()	Returns the current thread's ThreadLocalRandom object.
DoubleStream	doubles()	Returns an effectively unlimited stream of pseudorandom double values, each between zero (inclusive) and one (exclusive).
DoubleStream	doubles(double randomNumberOrigin, double randomNumberBound)	Returns an effectively unlimited stream of pseudorandom double values, each conforming to the given origin (inclusive) and bound (exclusive).
DoubleStream	doubles(long streamSize)	Returns a stream producing the given streamSize number of pseudorandom double values, each between zero (inclusive) and one (exclusive).
DoubleStream	doubles(long streamSize, double randomNumberOrigin, double randomNumberBound)	Returns a stream producing the given streamSize number of pseudorandom double values, each conforming to the given origin (inclusive) and bound (exclusive).
IntStream	ints()	Returns an effectively unlimited stream of pseudorandom int values.
IntStream	ints(int randomNumberOrigin, int randomNumberBound)	Returns an effectively unlimited stream of pseudorandom int values, each conforming

		to the given origin (inclusive) and bound (exclusive).
IntStream	ints(long streamSize)	Returns a stream producing the given streamSize number of pseudorandom int values.
IntStream	ints(long streamSize, int randomNumberOrigin, int randomNumberBound)	Returns a stream producing the given streamSize number of pseudorandom int values, each conforming to the given origin (inclusive) and bound (exclusive).
LongStream	longs()	Returns an effectively unlimited stream of pseudorandom long values.
LongStream	longs(long streamSize)	Returns a stream producing the given streamSize number of pseudorandom long values.
LongStream	longs(long randomNumberOrigin, long randomNumberBound)	Returns an effectively unlimited stream of pseudorandom long values, each conforming to the given origin (inclusive) and bound (exclusive).
LongStream	longs(long streamSize, long randomNumberOrigin, long randomNumberBound)	Returns a stream producing the given streamSize number of pseudorandom long, each conforming to the given origin (inclusive) and bound (exclusive).
protected int	next(int bits)	Generates a pseudorandom number with the indicated number of low-order bits.
double	nextDouble(double bound)	Returns a pseudorandomly chosen double value between zero (inclusive) and the specified bound (exclusive).
double	nextDouble(double origin, double bound)	Returns a pseudorandomly chosen double value between the specified origin (inclusive) and the specified bound (exclusive).
float	nextFloat(float bound)	Returns a pseudorandomly chosen float value between zero (inclusive) and the specified bound (exclusive).
float	nextFloat(float origin, float bound)	Returns a pseudorandomly chosen float value between the specified origin (inclusive) and the specified bound (exclusive).
int	nextInt(int bound)	Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.
int	nextInt(int origin, int bound)	Returns a pseudorandomly chosen int value between the specified origin (inclusive) and the specified bound (exclusive).
long	nextLong(long bound)	Returns a pseudorandomly chosen long value between zero (inclusive) and the specified bound (exclusive).
long	nextLong(long origin, long bound)	Returns a pseudorandomly chosen long value between the specified origin (inclusive) and the specified bound (exclusive).
void	setSeed(long seed)	Throws UnsupportedOperationException.

Methods declared in class java.util.Random

from, nextBoolean, nextBytes, nextDouble, nextFloat, nextGaussian, nextInt, nextLong

Methods declared in class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods declared in interface java.util.random.RandomGenerator

isDeprecated, nextExponential, nextGaussian

Method Details

current

`public static ThreadLocalRandom current()`

Returns the current thread's `ThreadLocalRandom` object. Methods of this object should be called only by the current thread, not by other threads.

Returns:

the current thread's `ThreadLocalRandom`

setSeed

`public void setSeed(long seed)`

Throws `UnsupportedOperationException`. Setting seeds in this generator is not supported.

Overrides:

`setSeed` in class `Random`

Parameters:

`seed` - the seed value

Throws:

`UnsupportedOperationException` - always

next

`protected int next(int bits)`

Generates a pseudorandom number with the indicated number of low-order bits. Because this class has no subclasses, this method cannot be invoked or overridden.

Overrides:

`next` in class `Random`

Parameters:

`bits` - random bits

Returns:

the next pseudorandom value from this random number generator's sequence

nextInt

`public int nextInt(int bound)`

Returns a pseudorandom, uniformly distributed `int` value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence. The general contract of `nextInt` is that one `int` value in the specified range is pseudorandomly generated and returned. All `bound` possible `int` values are produced with (approximately) equal probability.

Specified by:

`nextInt` in interface `RandomGenerator`

Overrides:

`nextInt` in class `Random`

Parameters:

`bound` - the upper bound (exclusive). Must be positive.

Returns:

the next pseudorandom, uniformly distributed `int` value between zero (inclusive) and `bound` (exclusive) from this random number generator's sequence

Throws:

`IllegalArgumentException` - if bound is not positive

nextInt

```
public int nextInt(int origin,
                   int bound)
```

Returns a pseudorandomly chosen `int` value between the specified origin (inclusive) and the specified bound (exclusive).

Parameters:

`origin` - the least value that can be returned

`bound` - the upper bound (exclusive) for the returned value

Returns:

a pseudorandomly chosen `int` value between the origin (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `origin` is greater than or equal to `bound`

nextLong

```
public long nextLong(long bound)
```

Returns a pseudorandomly chosen `long` value between zero (inclusive) and the specified bound (exclusive).

Parameters:

`bound` - the upper bound (exclusive) for the returned value. Must be positive.

Returns:

a pseudorandomly chosen `long` value between zero (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `bound` is not positive

nextLong

```
public long nextLong(long origin,
                     long bound)
```

Returns a pseudorandomly chosen `long` value between the specified origin (inclusive) and the specified bound (exclusive).

Parameters:

`origin` - the least value that can be returned

`bound` - the upper bound (exclusive) for the returned value

Returns:

a pseudorandomly chosen `long` value between the origin (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `origin` is greater than or equal to `bound`

nextFloat

```
public float nextFloat(float bound)
```

Returns a pseudorandomly chosen `float` value between zero (inclusive) and the specified bound (exclusive).

Implementation Note:

Parameters:

`bound` - the upper bound (exclusive) for the returned value. Must be positive and finite

Returns:

a pseudorandomly chosen `float` value between zero (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `bound` is not both positive and finite

nextFloat

```
public float nextFloat(float origin,
                       float bound)
```

Returns a pseudorandomly chosen `float` value between the specified origin (inclusive) and the specified bound (exclusive).

Implementation Note:

Parameters:

`origin` - the least value that can be returned

`bound` - the upper bound (exclusive)

Returns:

a pseudorandomly chosen `float` value between the origin (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `origin` is not finite, or `bound` is not finite, or `origin` is greater than or equal to `bound`

nextDouble

```
public double nextDouble(double bound)
```

Returns a pseudorandomly chosen `double` value between zero (inclusive) and the specified bound (exclusive).

Implementation Note:

Parameters:

`bound` - the upper bound (exclusive) for the returned value. Must be positive and finite

Returns:

a pseudorandomly chosen `double` value between zero (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `bound` is not both positive and finite

nextDouble

```
public double nextDouble(double origin,
                         double bound)
```

Returns a pseudorandomly chosen `double` value between the specified origin (inclusive) and the specified bound (exclusive).

Implementation Note:

Parameters:

`origin` - the least value that can be returned

`bound` - the upper bound (exclusive) for the returned value

Returns:

a pseudorandomly chosen `double` value between the origin (inclusive) and the bound (exclusive)

Throws:

`IllegalArgumentException` - if `origin` is not finite, or `bound` is not finite, or `origin` is greater than or equal to `bound`

ints

```
public IntStream ints(long streamSize)
```

Returns a stream producing the given `streamSize` number of pseudorandom `int` values.

A pseudorandom `int` value is generated as if it's the result of calling the method `Random.nextInt()`.

Specified by:

`ints` in interface `RandomGenerator`

Overrides:

`ints` in class `Random`

Parameters:

`streamSize` - the number of values to generate

Returns:

a stream of pseudorandom `int` values

Throws:

`IllegalArgumentException` - if `streamSize` is less than zero

Since:

1.8

ints

```
public IntStream ints()
```

Returns an effectively unlimited stream of pseudorandom `int` values.

A pseudorandom `int` value is generated as if it's the result of calling the method `Random.nextInt()`.

Specified by:

`ints` in interface `RandomGenerator`

Overrides:

`ints` in class `Random`

Implementation Note:

This method is implemented to be equivalent to `ints(Long.MAX_VALUE)`.

Returns:

a stream of pseudorandom `int` values

Since:

1.8

ints

```
public IntStream ints(long streamSize,
                      int randomNumberOrigin,
                      int randomNumberBound)
```

Returns a stream producing the given `streamSize` number of pseudorandom `int` values, each conforming to the given origin (inclusive) and bound (exclusive).

A pseudorandom `int` value is generated as if it's the result of calling the following method with the origin and bound:

```
int nextInt(int origin, int bound) {
  int n = bound - origin;
  if (n > 0) {
    return nextInt(n) + origin;
  }
  else {  // range not representable as int
    int r;
    do {
      r = nextInt();
    } while (r < origin || r >= bound);
    return r;
  }
}
```

Specified by:

`ints` in interface `RandomGenerator`

Overrides:

`ints` in class `Random`

Parameters:

`streamSize` - the number of values to generate

`randomNumberOrigin` - the origin (inclusive) of each random value

`randomNumberBound` - the bound (exclusive) of each random value

Returns:

a stream of pseudorandom `int` values, each with the given origin (inclusive) and bound (exclusive)

Throws:

`IllegalArgumentException` - if `streamSize` is less than zero, or `randomNumberOrigin` is greater than or equal to `randomNumberBound`

Since:

1.8

ints

```
public IntStream ints(int randomNumberOrigin,
                      int randomNumberBound)
```

Returns an effectively unlimited stream of pseudorandom `int` values, each conforming to the given origin (inclusive) and bound (exclusive).

A pseudorandom `int` value is generated as if it's the result of calling the following method with the origin and bound:

```
int nextInt(int origin, int bound) {
  int n = bound - origin;
  if (n > 0) {
    return nextInt(n) + origin;
  }
  else {  // range not representable as int
    int r;
    do {
      r = nextInt();
    } while (r < origin || r >= bound);
    return r;
  }
}
```

Specified by:

`ints` in interface `RandomGenerator`

Overrides:

`ints` in class `Random`

Implementation Note:

This method is implemented to be equivalent to `ints(Long.MAX_VALUE, randomNumberOrigin, randomNumberBound)`.

Parameters:

`randomNumberOrigin` - the origin (inclusive) of each random value

`randomNumberBound` - the bound (exclusive) of each random value

Returns:

a stream of pseudorandom `int` values, each with the given origin (inclusive) and bound (exclusive)

Throws:

`IllegalArgumentException` - if `randomNumberOrigin` is greater than or equal to `randomNumberBound`

Since:

1.8

longs

```
public LongStream longs(long streamSize)
```

Returns a stream producing the given `streamSize` number of pseudorandom `long` values.

A pseudorandom `long` value is generated as if it's the result of calling the method `Random.nextLong()`.

Specified by:

`longs` in interface `RandomGenerator`

Overrides:

`longs` in class `Random`

Parameters:

`streamSize` - the number of values to generate

Returns:

a stream of pseudorandom `long` values

Throws:

`IllegalArgumentException` - if `streamSize` is less than zero

Since:

1.8

longs

```
public LongStream longs()
```

Returns an effectively unlimited stream of pseudorandom `long` values.

A pseudorandom `long` value is generated as if it's the result of calling the method `Random.nextLong()`.

Specified by:

`longs` in interface `RandomGenerator`

Overrides:

`longs` in class `Random`

Implementation Note:

This method is implemented to be equivalent to `longs(Long.MAX_VALUE)`.

Returns:

a stream of pseudorandom `long` values

Since:

1.8

longs

```
public LongStream longs(long streamSize,
                        long randomNumberOrigin,
                        long randomNumberBound)
```

Returns a stream producing the given `streamSize` number of pseudorandom `long`, each conforming to the given origin (inclusive) and bound (exclusive).

A pseudorandom `long` value is generated as if it's the result of calling the following method with the origin and bound:

```
 long nextLong(long origin, long bound) {
   long r = nextLong();
   long n = bound - origin, m = n - 1;
   if ((n & m) == 0L)  // power of two
     r = (r & m) + origin;
   else if (n > 0L) {  // reject over-represented candidates
     for (long u = r >>> 1;            // ensure nonnegative
```

```
          u + m - (r = u % n) < 0L;    // rejection check
          u = nextLong() >>> 1) // retry
         ;
     r += origin;
   }
   else {              // range not representable as long
     while (r < origin || r >= bound)
       r = nextLong();
   }
   return r;
 }
```

Specified by:

`longs` in interface `RandomGenerator`

Overrides:

`longs` in class `Random`

Parameters:

`streamSize` - the number of values to generate

`randomNumberOrigin` - the origin (inclusive) of each random value

`randomNumberBound` - the bound (exclusive) of each random value

Returns:

a stream of pseudorandom `long` values, each with the given origin (inclusive) and bound (exclusive)

Throws:

`IllegalArgumentException` - if `streamSize` is less than zero, or `randomNumberOrigin` is greater than or equal to `randomNumberBound`

Since:

1.8

longs

```
public LongStream longs(long randomNumberOrigin,
                        long randomNumberBound)
```

Returns an effectively unlimited stream of pseudorandom `long` values, each conforming to the given origin (inclusive) and bound (exclusive).

A pseudorandom `long` value is generated as if it's the result of calling the following method with the origin and bound:

```
 long nextLong(long origin, long bound) {
   long r = nextLong();
   long n = bound - origin, m = n - 1;
   if ((n & m) == 0L)  // power of two
     r = (r & m) + origin;
   else if (n > 0L) {  // reject over-represented candidates
     for (long u = r >>> 1;            // ensure nonnegative
          u + m - (r = u % n) < 0L;    // rejection check
          u = nextLong() >>> 1) // retry
         ;
     r += origin;
   }
   else {              // range not representable as long
     while (r < origin || r >= bound)
       r = nextLong();
   }
   return r;
 }
```

Specified by:

`longs` in interface `RandomGenerator`

Overrides:

`longs` in class `Random`

Implementation Note:

This method is implemented to be equivalent to `longs(Long.MAX_VALUE, randomNumberOrigin, randomNumberBound)`.

Parameters:

`randomNumberOrigin` - the origin (inclusive) of each random value

`randomNumberBound` - the bound (exclusive) of each random value

Returns:

a stream of pseudorandom `long` values, each with the given origin (inclusive) and bound (exclusive)

Throws:

`IllegalArgumentException` - if `randomNumberOrigin` is greater than or equal to `randomNumberBound`

Since:

1.8

doubles

```
public DoubleStream doubles(long streamSize)
```

Returns a stream producing the given `streamSize` number of pseudorandom `double` values, each between zero (inclusive) and one (exclusive).

A pseudorandom `double` value is generated as if it's the result of calling the method `Random.nextDouble()`.

Specified by:

`doubles` in interface `RandomGenerator`

Overrides:

`doubles` in class `Random`

Parameters:

`streamSize` - the number of values to generate

Returns:

a stream of `double` values

Throws:

`IllegalArgumentException` - if `streamSize` is less than zero

Since:

1.8

doubles

```
public DoubleStream doubles()
```

Returns an effectively unlimited stream of pseudorandom `double` values, each between zero (inclusive) and one (exclusive).

A pseudorandom `double` value is generated as if it's the result of calling the method `Random.nextDouble()`.

Specified by:

`doubles` in interface `RandomGenerator`

Overrides:

`doubles` in class `Random`

Implementation Note:

This method is implemented to be equivalent to `doubles(Long.MAX_VALUE)`.

Returns:

a stream of pseudorandom `double` values

Since:

1.8

doubles

```
public DoubleStream doubles(long streamSize,
                            double randomNumberOrigin,
                            double randomNumberBound)
```

Returns a stream producing the given `streamSize` number of pseudorandom `double` values, each conforming to the given origin (inclusive) and bound (exclusive).

Specified by:

`doubles` in interface `RandomGenerator`

Overrides:

`doubles` in class `Random`

Parameters:

`streamSize` - the number of values to generate

`randomNumberOrigin` - the origin (inclusive) of each random value

`randomNumberBound` - the bound (exclusive) of each random value

Returns:

a stream of pseudorandom `double` values, each with the given origin (inclusive) and bound (exclusive)

Throws:

`IllegalArgumentException` - if `streamSize` is less than zero, or `randomNumberOrigin` is not finite, or `randomNumberBound` is not finite, or `randomNumberOrigin` is greater than or equal to `randomNumberBound`

Since:

1.8

doubles

```
public DoubleStream doubles(double randomNumberOrigin,
                            double randomNumberBound)
```

Returns an effectively unlimited stream of pseudorandom `double` values, each conforming to the given origin (inclusive) and bound (exclusive).

Specified by:

`doubles` in interface `RandomGenerator`

Overrides:

`doubles` in class `Random`

Implementation Note:

This method is implemented to be equivalent to `doubles(Long.MAX_VALUE, randomNumberOrigin, randomNumberBound)`.

Parameters:

`randomNumberOrigin` - the origin (inclusive) of each random value

`randomNumberBound` - the bound (exclusive) of each random value

Returns:

a stream of pseudorandom `double` values, each with the given origin (inclusive) and bound (exclusive)

Throws:

`IllegalArgumentException` - if `randomNumberOrigin` is not finite, or `randomNumberBound` is not finite, or `randomNumberOrigin` is greater than or equal to `randomNumberBound`

Since:

1.8