Module java.base
Package java.util

Class UUID

java.lang.Object
    java.util.UUID

All Implemented Interfaces:

`Serializable`, `Comparable<UUID>`

```
public final class UUID
extends Object
implements Serializable, Comparable<UUID>
```

A class that represents an immutable universally unique identifier (UUID). A UUID represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of UUID (described below).

The layout of a variant 2 (Leach-Salz) UUID is as follows: The most significant long consists of the following unsigned fields:

```
 0xFFFFFFFF00000000 time_low
 0x00000000FFFF0000 time_mid
 0x000000000000F000 version
 0x0000000000000FFF time_hi
```

The least significant long consists of the following unsigned fields:

```
 0xC000000000000000 variant
 0x3FFF000000000000 clock_seq
 0x0000FFFFFFFFFFFF node
```

The variant field contains a value which identifies the layout of the `UUID`. The bit layout described above is valid only for a `UUID` with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this `UUID`. There are four different basic types of UUIDs: time-based, DCE security, name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create `UUID`s, see *RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace* , section 4.2 "Algorithms for Creating a Time-Based UUID".

Since:

1.5

External Specifications

RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace

See Also:

Serialized Form

Constructor Summary

Constructors

Constructor	Description
`UUID(long mostSigBits, long leastSigBits)`	Constructs a new `UUID` using the specified data.

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method	Description
`int`	`clockSequence()`	The clock sequence value associated with this UUID.
`int`	`compareTo(UUID val)`	Compares this UUID with the specified UUID.
`boolean`	`equals(Object obj)`	Compares this object to the specified object.
`static UUID`	`fromString(String name)`	Creates a `UUID` from the string standard representation as described in the `toString()` method.
`long`	`getLeastSignificantBits()`	Returns the least significant 64 bits of this UUID's 128 bit value.
`long`	`getMostSignificantBits()`	Returns the most significant 64 bits of this UUID's 128 bit value.
`int`	`hashCode()`	Returns a hash code for this `UUID`.
`static UUID`	`nameUUIDFromBytes(byte[] name)`	Static factory to retrieve a type 3 (name based) `UUID` based on the specified byte array.
`long`	`node()`	The node value associated with this UUID.
`static UUID`	`randomUUID()`	Static factory to retrieve a type 4 (pseudo randomly generated)

		UUID.
`long`	**`timestamp`**`()`	The timestamp value associated with this UUID.
`String`	**`toString`**`()`	Returns a `String` object representing this `UUID`.
`int`	**`variant`**`()`	The variant number associated with this `UUID`.
`int`	**`version`**`()`	The version number associated with this `UUID`.

Methods declared in class java.lang.Object

`clone, finalize, getClass, notify, notifyAll, wait, wait, wait`

Constructor Details

UUID

```
public UUID(long mostSigBits,
            long leastSigBits)
```

Constructs a new `UUID` using the specified data. `mostSigBits` is used for the most significant 64 bits of the `UUID` and `leastSigBits` becomes the least significant 64 bits of the `UUID`.

Parameters:

`mostSigBits` - The most significant bits of the `UUID`

`leastSigBits` - The least significant bits of the `UUID`

Method Details

randomUUID

```
public static UUID randomUUID()
```

Static factory to retrieve a type 4 (pseudo randomly generated) UUID. The `UUID` is generated using a cryptographically strong pseudo random number generator.

Returns:

A randomly generated `UUID`

nameUUIDFromBytes

`public static UUID nameUUIDFromBytes(byte[] name)`

Static factory to retrieve a type 3 (name based) `UUID` based on the specified byte array.

Parameters:

`name` - A byte array to be used to construct a `UUID`

Returns:

A `UUID` generated from the specified array

fromString

`public static UUID fromString(String name)`

Creates a `UUID` from the string standard representation as described in the `toString()` method.

Parameters:

`name` - A string that specifies a `UUID`

Returns:

A `UUID` with the specified value

Throws:

`IllegalArgumentException` - If name does not conform to the string representation as described in `toString()`

getLeastSignificantBits

`public long getLeastSignificantBits()`

Returns the least significant 64 bits of this UUID's 128 bit value.

Returns:

The least significant 64 bits of this UUID's 128 bit value

getMostSignificantBits

`public long getMostSignificantBits()`

Returns the most significant 64 bits of this UUID's 128 bit value.

Returns:

The most significant 64 bits of this UUID's 128 bit value

version

`public int version()`

The version number associated with this `UUID`. The version number describes how this `UUID` was generated. The version number has the following meaning:

- 1 Time-based UUID
- 2 DCE security UUID
- 3 Name-based UUID
- 4 Randomly generated UUID

Returns:

The version number of this `UUID`

variant

```
public int variant()
```

The variant number associated with this `UUID`. The variant number describes the layout of the `UUID`. The variant number has the following meaning:

- 0 Reserved for NCS backward compatibility
- 2 IETF RFC 4122 (Leach-Salz), used by this class
- 6 Reserved, Microsoft Corporation backward compatibility
- 7 Reserved for future definition

Returns:

The variant number of this `UUID`

External Specifications

RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace

timestamp

```
public long timestamp()
```

The timestamp value associated with this UUID.

The 60 bit timestamp value is constructed from the time_low, time_mid, and time_hi fields of this `UUID`. The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based UUID, which has version type 1. If this `UUID` is not a time-based UUID then this method throws UnsupportedOperationException.

Returns:

The timestamp of this `UUID`.

Throws:

`UnsupportedOperationException` - If this UUID is not a version 1 UUID

clockSequence

```
public int clockSequence()
```

The clock sequence value associated with this UUID.

The 14 bit clock sequence value is constructed from the clock sequence field of this UUID. The clock sequence field is used to guarantee temporal uniqueness in a time-based UUID.

The `clockSequence` value is only meaningful in a time-based UUID, which has version type 1. If this UUID is not a time-based UUID then this method throws UnsupportedOperationException.

Returns:

The clock sequence of this `UUID`

Throws:

`UnsupportedOperationException` - If this UUID is not a version 1 UUID

node

```
public long node()
```

The node value associated with this UUID.

The 48 bit node value is constructed from the node field of this UUID. This field is intended to hold the IEEE 802 address of the machine that generated this UUID to guarantee spatial uniqueness.

The node value is only meaningful in a time-based UUID, which has version type 1. If this UUID is not a time-based UUID then this method throws UnsupportedOperationException.

Returns:

The node value of this `UUID`

Throws:

`UnsupportedOperationException` - If this UUID is not a version 1 UUID

toString

```
public String toString()
```

Returns a `String` object representing this `UUID`.

The UUID string representation is as described by this BNF:

```
     UUID                   = <time_low> "-" <time_mid> "-"
                              <time_high_and_version> "-"
                              <variant_and_sequence> "-"
                              <node>
     time_low               = 4*<hexOctet>
     time_mid               = 2*<hexOctet>
     time_high_and_version  = 2*<hexOctet>
     variant_and_sequence   = 2*<hexOctet>
     node                   = 6*<hexOctet>
     hexOctet               = <hexDigit><hexDigit>
```

```
    hexDigit               =
          "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
          | "a" | "b" | "c" | "d" | "e" | "f"
          | "A" | "B" | "C" | "D" | "E" | "F"
```

Overrides:

`toString` in class `Object`

Returns:

A string representation of this `UUID`

hashCode

`public int hashCode()`

Returns a hash code for this `UUID`.

Overrides:

`hashCode` in class `Object`

Returns:

A hash code value for this `UUID`

See Also:

`Object.equals(java.lang.Object)`, `System.identityHashCode(java.lang.Object)`

equals

`public boolean equals(Object obj)`

Compares this object to the specified object. The result is `true` if and only if the argument is not `null`, is a `UUID` object, has the same variant, and contains the same value, bit for bit, as this `UUID`.

Overrides:

`equals` in class `Object`

Parameters:

`obj` - The object to be compared

Returns:

`true` if the objects are the same; `false` otherwise

See Also:

`Object.hashCode()`, `HashMap`

compareTo

`public int compareTo(UUID val)`

Compares this UUID with the specified UUID.

The first of two UUIDs is greater than the second if the most significant field in which the UUIDs differ is greater for the first UUID.

Specified by:

`compareTo` in interface `Comparable<UUID>`

Parameters:

`val` - `UUID` to which this `UUID` is to be compared

Returns:

-1, 0 or 1 as this `UUID` is less than, equal to, or greater than `val`