| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CompositeName.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/naming/CommunicationException.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/CompoundName.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/CompositeName.html)    [**NO FRAMES**](http://docs.google.com/CompositeName.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#17dp8vu) |

## **javax.naming**

Class CompositeName

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.naming.CompositeName**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Object](http://docs.google.com/java/lang/Object.html)>, [Name](http://docs.google.com/javax/naming/Name.html)

public class **CompositeName**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Name](http://docs.google.com/javax/naming/Name.html)

This class represents a composite name -- a sequence of component names spanning multiple namespaces. Each component is a string name from the namespace of a naming system. If the component comes from a hierarchical namespace, that component can be further parsed into its atomic parts by using the CompoundName class.

The components of a composite name are numbered. The indexes of a composite name with N components range from 0 up to, but not including, N. This range may be written as [0,N). The most significant component is at index 0. An empty composite name has no components.

#### JNDI Composite Name Syntax

JNDI defines a standard string representation for composite names. This representation is the concatenation of the components of a composite name from left to right using the component separator (a forward slash character (/)) to separate each component. The JNDI syntax defines the following meta characters:

* escape (backward slash \),
* quote characters (single (') and double quotes (")), and
* component separator (forward slash character (/)).

Any occurrence of a leading quote, an escape preceding any meta character, an escape at the end of a component, or a component separator character in an unquoted component must be preceded by an escape character when that component is being composed into a composite name string. Alternatively, to avoid adding escape characters as described, the entire component can be quoted using matching single quotes or matching double quotes. A single quote occurring within a double-quoted component is not considered a meta character (and need not be escaped), and vice versa.

When two composite names are compared, the case of the characters is significant.

A leading component separator (the composite name string begins with a separator) denotes a leading empty component (a component consisting of an empty string). A trailing component separator (the composite name string ends with a separator) denotes a trailing empty component. Adjacent component separators denote an empty component.

#### Composite Name Examples

This table shows examples of some composite names. Each row shows the string form of a composite name and its corresponding structural form (CompositeName).

| String Name | CompositeName |
| --- | --- |
| "" | {} (the empty name == new CompositeName("") == new CompositeName()) |
| "x" | {"x"} |
| "x/y" | {"x", "y"} |
| "x/" | {"x", ""} |
| "/x" | {"", "x"} |
| "/" | {""} |
| "//" | {"", ""} |
| "/x/" | {"", "x", ""} |
| "x//y" | {"x", "", "y"} |

#### Composition Examples

Here are some composition examples. The right column shows composing string composite names while the left column shows composing the corresponding CompositeNames. Notice that composing the string forms of two composite names simply involves concatenating their string forms together.

| String Names | CompositeNames |
| --- | --- |
| "x/y" + "/" = x/y/ | {"x", "y"} + {""} = {"x", "y", ""} |
| "" + "x" = "x" | {} + {"x"} = {"x"} |
| "/" + "x" = "/x" | {""} + {"x"} = {"", "x"} |
| "x" + "" + "" = "x" | {"x"} + {} + {} = {"x"} |

#### Multithreaded Access

A CompositeName instance is not synchronized against concurrent multithreaded access. Multiple threads trying to access and modify a CompositeName should lock the object.

**Since:** 1.3 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#javax.naming.CompositeName)

| **Constructor Summary** | |
| --- | --- |
|  | [**CompositeName**](http://docs.google.com/javax/naming/CompositeName.html#CompositeName())()            Constructs a new empty composite name. |
| protected | [**CompositeName**](http://docs.google.com/javax/naming/CompositeName.html#CompositeName(java.util.Enumeration))([Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> comps)            Constructs a new composite name instance using the components specified by 'comps'. |
|  | [**CompositeName**](http://docs.google.com/javax/naming/CompositeName.html#CompositeName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) n)            Constructs a new composite name instance by parsing the string n using the composite name syntax (left-to-right, slash separated). |

| **Method Summary** | |
| --- | --- |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**add**](http://docs.google.com/javax/naming/CompositeName.html#add(int,%20java.lang.String))(int posn, [String](http://docs.google.com/java/lang/String.html) comp)            Adds a single component at a specified position within this composite name. |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**add**](http://docs.google.com/javax/naming/CompositeName.html#add(java.lang.String))([String](http://docs.google.com/java/lang/String.html) comp)            Adds a single component to the end of this composite name. |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**addAll**](http://docs.google.com/javax/naming/CompositeName.html#addAll(int,%20javax.naming.Name))(int posn, [Name](http://docs.google.com/javax/naming/Name.html) n)            Adds the components of a composite name -- in order -- at a specified position within this composite name. |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**addAll**](http://docs.google.com/javax/naming/CompositeName.html#addAll(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) suffix)            Adds the components of a composite name -- in order -- to the end of this composite name. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/javax/naming/CompositeName.html#clone())()            Generates a copy of this composite name. |
| int | [**compareTo**](http://docs.google.com/javax/naming/CompositeName.html#compareTo(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this CompositeName with the specified Object for order. |
| boolean | [**endsWith**](http://docs.google.com/javax/naming/CompositeName.html#endsWith(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) n)            Determines whether a composite name is a suffix of this composite name. |
| boolean | [**equals**](http://docs.google.com/javax/naming/CompositeName.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Determines whether two composite names are equal. |
| [String](http://docs.google.com/java/lang/String.html) | [**get**](http://docs.google.com/javax/naming/CompositeName.html#get(int))(int posn)            Retrieves a component of this composite name. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getAll**](http://docs.google.com/javax/naming/CompositeName.html#getAll())()            Retrieves the components of this composite name as an enumeration of strings. |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**getPrefix**](http://docs.google.com/javax/naming/CompositeName.html#getPrefix(int))(int posn)            Creates a composite name whose components consist of a prefix of the components in this composite name. |
| [Name](http://docs.google.com/javax/naming/Name.html) | [**getSuffix**](http://docs.google.com/javax/naming/CompositeName.html#getSuffix(int))(int posn)            Creates a composite name whose components consist of a suffix of the components in this composite name. |
| int | [**hashCode**](http://docs.google.com/javax/naming/CompositeName.html#hashCode())()            Computes the hash code of this composite name. |
| boolean | [**isEmpty**](http://docs.google.com/javax/naming/CompositeName.html#isEmpty())()            Determines whether this composite name is empty. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**remove**](http://docs.google.com/javax/naming/CompositeName.html#remove(int))(int posn)            Deletes a component from this composite name. |
| int | [**size**](http://docs.google.com/javax/naming/CompositeName.html#size())()            Retrieves the number of components in this composite name. |
| boolean | [**startsWith**](http://docs.google.com/javax/naming/CompositeName.html#startsWith(javax.naming.Name))([Name](http://docs.google.com/javax/naming/Name.html) n)            Determines whether a composite name is a prefix of this composite name. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/javax/naming/CompositeName.html#toString())()            Generates the string representation of this composite name. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### CompositeName

protected **CompositeName**([Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> comps)

Constructs a new composite name instance using the components specified by 'comps'. This protected method is intended to be to be used by subclasses of CompositeName when they override methods such as clone(), getPrefix(), getSuffix().

**Parameters:**comps - A non-null enumeration containing the components for the new composite name. Each element is of class String. The enumeration will be consumed to extract its elements.

### CompositeName

public **CompositeName**([String](http://docs.google.com/java/lang/String.html) n)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Constructs a new composite name instance by parsing the string n using the composite name syntax (left-to-right, slash separated). The composite name syntax is described in detail in the class description.

**Parameters:**n - The non-null string to parse. **Throws:** [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If n has invalid composite name syntax.

### CompositeName

public **CompositeName**()

Constructs a new empty composite name. Such a name returns true when isEmpty() is invoked on it.

| **Method Detail** |
| --- |

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Generates the string representation of this composite name. The string representation consists of enumerating in order each component of the composite name and separating each component by a forward slash character. Quoting and escape characters are applied where necessary according to the JNDI syntax, which is described in the class description. An empty component is represented by an empty string. The string representation thus generated can be passed to the CompositeName constructor to create a new equivalent composite name.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**A non-null string representation of this composite name.

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Determines whether two composite names are equal. If obj is null or not a composite name, false is returned. Two composite names are equal if each component in one is equal to the corresponding component in the other. This implies both have the same number of components, and each component's equals() test against the corresponding component in the other name returns true.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**obj - The possibly null object to compare against. **Returns:**true if obj is equal to this composite name, false otherwise.**See Also:**[hashCode()](http://docs.google.com/javax/naming/CompositeName.html#hashCode())

### hashCode

public int **hashCode**()

Computes the hash code of this composite name. The hash code is the sum of the hash codes of individual components of this composite name.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**An int representing the hash code of this name.**See Also:**[equals(java.lang.Object)](http://docs.google.com/javax/naming/CompositeName.html#equals(java.lang.Object))

### compareTo

public int **compareTo**([Object](http://docs.google.com/java/lang/Object.html) obj)

Compares this CompositeName with the specified Object for order. Returns a negative integer, zero, or a positive integer as this Name is less than, equal to, or greater than the given Object.

If obj is null or not an instance of CompositeName, ClassCastException is thrown.

See equals() for what it means for two composite names to be equal. If two composite names are equal, 0 is returned.

Ordering of composite names follows the lexicographical rules for string comparison, with the extension that this applies to all the components in the composite name. The effect is as if all the components were lined up in their specified ordered and the lexicographical rules applied over the two line-ups. If this composite name is "lexicographically" lesser than obj, a negative number is returned. If this composite name is "lexicographically" greater than obj, a positive number is returned.

**Specified by:**[compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) in interface [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Object](http://docs.google.com/java/lang/Object.html)>**Specified by:**[compareTo](http://docs.google.com/javax/naming/Name.html#compareTo(java.lang.Object)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**obj - The non-null object to compare against. **Returns:**a negative integer, zero, or a positive integer as this Name is less than, equal to, or greater than the given Object. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if obj is not a CompositeName.**See Also:**[Comparable.compareTo(Object)](http://docs.google.com/java/lang/Comparable.html#compareTo(T))

### clone

public [Object](http://docs.google.com/java/lang/Object.html) **clone**()

Generates a copy of this composite name. Changes to the components of this composite name won't affect the new copy and vice versa.

**Specified by:**[clone](http://docs.google.com/javax/naming/Name.html#clone()) in interface [Name](http://docs.google.com/javax/naming/Name.html)**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**A non-null copy of this composite name.**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### size

public int **size**()

Retrieves the number of components in this composite name.

**Specified by:**[size](http://docs.google.com/javax/naming/Name.html#size()) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Returns:**The nonnegative number of components in this composite name.

### isEmpty

public boolean **isEmpty**()

Determines whether this composite name is empty. A composite name is empty if it has zero components.

**Specified by:**[isEmpty](http://docs.google.com/javax/naming/Name.html#isEmpty()) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Returns:**true if this composite name is empty, false otherwise.

### getAll

public [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> **getAll**()

Retrieves the components of this composite name as an enumeration of strings. The effects of updates to this composite name on this enumeration is undefined.

**Specified by:**[getAll](http://docs.google.com/javax/naming/Name.html#getAll()) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Returns:**A non-null enumeration of the components of this composite name. Each element of the enumeration is of class String.

### get

public [String](http://docs.google.com/java/lang/String.html) **get**(int posn)

Retrieves a component of this composite name.

**Specified by:**[get](http://docs.google.com/javax/naming/Name.html#get(int)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**posn - The 0-based index of the component to retrieve. Must be in the range [0,size()). **Returns:**The non-null component at index posn. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if posn is outside the specified range.

### getPrefix

public [Name](http://docs.google.com/javax/naming/Name.html) **getPrefix**(int posn)

Creates a composite name whose components consist of a prefix of the components in this composite name. Subsequent changes to this composite name does not affect the name that is returned.

**Specified by:**[getPrefix](http://docs.google.com/javax/naming/Name.html#getPrefix(int)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**posn - The 0-based index of the component at which to stop. Must be in the range [0,size()]. **Returns:**A composite name consisting of the components at indexes in the range [0,posn). **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - If posn is outside the specified range.

### getSuffix

public [Name](http://docs.google.com/javax/naming/Name.html) **getSuffix**(int posn)

Creates a composite name whose components consist of a suffix of the components in this composite name. Subsequent changes to this composite name does not affect the name that is returned.

**Specified by:**[getSuffix](http://docs.google.com/javax/naming/Name.html#getSuffix(int)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**posn - The 0-based index of the component at which to start. Must be in the range [0,size()]. **Returns:**A composite name consisting of the components at indexes in the range [posn,size()). If posn is equal to size(), an empty composite name is returned. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - If posn is outside the specified range.

### startsWith

public boolean **startsWith**([Name](http://docs.google.com/javax/naming/Name.html) n)

Determines whether a composite name is a prefix of this composite name. A composite name 'n' is a prefix if it is equal to getPrefix(n.size())--in other words, this composite name starts with 'n'. If 'n' is null or not a composite name, false is returned.

**Specified by:**[startsWith](http://docs.google.com/javax/naming/Name.html#startsWith(javax.naming.Name)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**n - The possibly null name to check. **Returns:**true if n is a CompositeName and is a prefix of this composite name, false otherwise.

### endsWith

public boolean **endsWith**([Name](http://docs.google.com/javax/naming/Name.html) n)

Determines whether a composite name is a suffix of this composite name. A composite name 'n' is a suffix if it it is equal to getSuffix(size()-n.size())--in other words, this composite name ends with 'n'. If n is null or not a composite name, false is returned.

**Specified by:**[endsWith](http://docs.google.com/javax/naming/Name.html#endsWith(javax.naming.Name)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**n - The possibly null name to check. **Returns:**true if n is a CompositeName and is a suffix of this composite name, false otherwise.

### addAll

public [Name](http://docs.google.com/javax/naming/Name.html) **addAll**([Name](http://docs.google.com/javax/naming/Name.html) suffix)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Adds the components of a composite name -- in order -- to the end of this composite name.

**Specified by:**[addAll](http://docs.google.com/javax/naming/Name.html#addAll(javax.naming.Name)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**suffix - The non-null components to add. **Returns:**The updated CompositeName, not a new one. Cannot be null. **Throws:** [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If suffix is not a composite name.

### addAll

public [Name](http://docs.google.com/javax/naming/Name.html) **addAll**(int posn,  
 [Name](http://docs.google.com/javax/naming/Name.html) n)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Adds the components of a composite name -- in order -- at a specified position within this composite name. Components of this composite name at or after the index of the first new component are shifted up (away from index 0) to accommodate the new components.

**Specified by:**[addAll](http://docs.google.com/javax/naming/Name.html#addAll(int,%20javax.naming.Name)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**n - The non-null components to add.posn - The index in this name at which to add the new components. Must be in the range [0,size()]. **Returns:**The updated CompositeName, not a new one. Cannot be null. **Throws:** [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If n is not a composite name. [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - If posn is outside the specified range.

### add

public [Name](http://docs.google.com/javax/naming/Name.html) **add**([String](http://docs.google.com/java/lang/String.html) comp)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Adds a single component to the end of this composite name.

**Specified by:**[add](http://docs.google.com/javax/naming/Name.html#add(java.lang.String)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**comp - The non-null component to add. **Returns:**The updated CompositeName, not a new one. Cannot be null. **Throws:** [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If adding comp at end of the name would violate the name's syntax.

### add

public [Name](http://docs.google.com/javax/naming/Name.html) **add**(int posn,  
 [String](http://docs.google.com/java/lang/String.html) comp)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Adds a single component at a specified position within this composite name. Components of this composite name at or after the index of the new component are shifted up by one (away from index 0) to accommodate the new component.

**Specified by:**[add](http://docs.google.com/javax/naming/Name.html#add(int,%20java.lang.String)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**comp - The non-null component to add.posn - The index at which to add the new component. Must be in the range [0,size()]. **Returns:**The updated CompositeName, not a new one. Cannot be null. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - If posn is outside the specified range. [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If adding comp at the specified position would violate the name's syntax.

### remove

public [Object](http://docs.google.com/java/lang/Object.html) **remove**(int posn)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Deletes a component from this composite name. The component of this composite name at position 'posn' is removed, and components at indices greater than 'posn' are shifted down (towards index 0) by one.

**Specified by:**[remove](http://docs.google.com/javax/naming/Name.html#remove(int)) in interface [Name](http://docs.google.com/javax/naming/Name.html) **Parameters:**posn - The index of the component to delete. Must be in the range [0,size()). **Returns:**The component removed (a String). **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - If posn is outside the specified range (includes case where composite name is empty). [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If deleting the component would violate the name's syntax.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CompositeName.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/naming/CommunicationException.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/CompoundName.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/CompositeName.html)    [**NO FRAMES**](http://docs.google.com/CompositeName.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#17dp8vu) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).