| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/CompoundName.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/CompositeName.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/ConfigurationException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/CompoundName.html)    [**NO FRAMES**](http://docs.google.com/CompoundName.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

## **javax.naming**

Class CompoundName

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

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

This class represents a compound name -- a name from a hierarchical name space. Each component in a compound name is an atomic name.

The components of a compound name are numbered. The indexes of a compound 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 compound name has no components.

#### Compound Name Syntax

The syntax of a compound name is specified using a set of properties: jndi.syntax.direction Direction for parsing ("right\_to\_left", "left\_to\_right", "flat"). If unspecified, defaults to "flat", which means the namespace is flat with no hierarchical structure. jndi.syntax.separator Separator between atomic name components. Required unless direction is "flat". jndi.syntax.ignorecase If present, "true" means ignore the case when comparing name components. If its value is not "true", or if the property is not present, case is considered when comparing name components. jndi.syntax.escape If present, specifies the escape string for overriding separator, escapes and quotes. jndi.syntax.beginquote If present, specifies the string delimiting start of a quoted string. jndi.syntax.endquote String delimiting end of quoted string. If present, specifies the string delimiting the end of a quoted string. If not present, use syntax.beginquote as end quote. jndi.syntax.beginquote2 Alternative set of begin/end quotes. jndi.syntax.endquote2 Alternative set of begin/end quotes. jndi.syntax.trimblanks If present, "true" means trim any leading and trailing whitespaces in a name component for comparison purposes. If its value is not "true", or if the property is not present, blanks are significant. jndi.syntax.separator.ava If present, specifies the string that separates attribute-value-assertions when specifying multiple attribute/value pairs. (e.g. "," in age=65,gender=male). jndi.syntax.separator.typeval If present, specifies the string that separators attribute from value (e.g. "=" in "age=65") These properties are interpreted according to the following rules:

1. In a string without quotes or escapes, any instance of the separator delimits two atomic names. Each atomic name is referred to as a *component*.
2. A separator, quote or escape is escaped if preceded immediately (on the left) by the escape.
3. If there are two sets of quotes, a specific begin-quote must be matched by its corresponding end-quote.
4. A non-escaped begin-quote which precedes a component must be matched by a non-escaped end-quote at the end of the component. A component thus quoted is referred to as a *quoted component*. It is parsed by removing the being- and end- quotes, and by treating the intervening characters as ordinary characters unless one of the rules involving quoted components listed below applies.
5. Quotes embedded in non-quoted components are treated as ordinary strings and need not be matched.
6. A separator that is escaped or appears between non-escaped quotes is treated as an ordinary string and not a separator.
7. An escape string within a quoted component acts as an escape only when followed by the corresponding end-quote string. This can be used to embed an escaped quote within a quoted component.
8. An escaped escape string is not treated as an escape string.
9. An escape string that does not precede a meta string (quotes or separator) and is not at the end of a component is treated as an ordinary string.
10. A leading separator (the compound name string begins with a separator) denotes a leading empty atomic component (consisting of an empty string). A trailing separator (the compound name string ends with a separator) denotes a trailing empty atomic component. Adjacent separators denote an empty atomic component.

The string form of the compound name follows the syntax described above. When the components of the compound name are turned into their string representation, the reserved syntax rules described above are applied (e.g. embedded separators are escaped or quoted) so that when the same string is parsed, it will yield the same components of the original compound name.

#### Multithreaded Access

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

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

| **Field Summary** | |
| --- | --- |
| protected  javax.naming.NameImpl | [**impl**](http://docs.google.com/javax/naming/CompoundName.html#impl)            Implementation of this compound name. |
| protected  [Properties](http://docs.google.com/java/util/Properties.html) | [**mySyntax**](http://docs.google.com/javax/naming/CompoundName.html#mySyntax)            Syntax properties for this compound name. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**CompoundName**](http://docs.google.com/javax/naming/CompoundName.html#CompoundName(java.util.Enumeration,%20java.util.Properties))([Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> comps, [Properties](http://docs.google.com/java/util/Properties.html) syntax)            Constructs a new compound name instance using the components specified in comps and syntax. |
|  | [**CompoundName**](http://docs.google.com/javax/naming/CompoundName.html#CompoundName(java.lang.String,%20java.util.Properties))([String](http://docs.google.com/java/lang/String.html) n, [Properties](http://docs.google.com/java/util/Properties.html) syntax)            Constructs a new compound name instance by parsing the string n using the syntax specified by the syntax properties supplied. |

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

| **Field Detail** |
| --- |

### impl

protected transient javax.naming.NameImpl **impl**

Implementation of this compound name. This field is initialized by the constructors and cannot be null. It should be treated as a read-only variable by subclasses.

### mySyntax

protected transient [Properties](http://docs.google.com/java/util/Properties.html) **mySyntax**

Syntax properties for this compound name. This field is initialized by the constructors and cannot be null. It should be treated as a read-only variable by subclasses. Any necessary changes to mySyntax should be made within constructors and not after the compound name has been instantiated.

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

### CompoundName

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

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

**Parameters:**comps - A non-null enumeration of the components to add. Each element of the enumeration is of class String. The enumeration will be consumed to extract its elements.syntax - A non-null properties that specify the syntax of this compound name. See class description for contents of properties.

### CompoundName

public **CompoundName**([String](http://docs.google.com/java/lang/String.html) n,  
 [Properties](http://docs.google.com/java/util/Properties.html) syntax)  
 throws [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html)

Constructs a new compound name instance by parsing the string n using the syntax specified by the syntax properties supplied.

**Parameters:**n - The non-null string to parse.syntax - A non-null list of properties that specify the syntax of this compound name. See class description for contents of properties. **Throws:** [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If 'n' violates the syntax specified by syntax.

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

### toString

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

Generates the string representation of this compound name, using the syntax rules of the compound name. The syntax rules are described in the class description. An empty component is represented by an empty string. The string representation thus generated can be passed to the CompoundName constructor with the same syntax properties to create a new equivalent compound 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 compound name.

### equals

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

Determines whether obj is syntactically equal to this compound name. If obj is null or not a CompoundName, false is returned. Two compound names are equal if each component in one is "equal" to the corresponding component in the other.

Equality is also defined in terms of the syntax of this compound name. The default implementation of CompoundName uses the syntax properties jndi.syntax.ignorecase and jndi.syntax.trimblanks when comparing two components for equality. If case is ignored, two strings with the same sequence of characters but with different cases are considered equal. If blanks are being trimmed, leading and trailing blanks are ignored for the purpose of the comparison.

Both compound names must have the same number of components.

Implementation note: Currently the syntax properties of the two compound names are not compared for equality. They might be in the future.

**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 compound name, false otherwise.**See Also:**[compareTo(java.lang.Object obj)](http://docs.google.com/javax/naming/CompoundName.html#compareTo(java.lang.Object))

### hashCode

public int **hashCode**()

Computes the hash code of this compound name. The hash code is the sum of the hash codes of the "canonicalized" forms of individual components of this compound name. Each component is "canonicalized" according to the compound name's syntax before its hash code is computed. For a case-insensitive name, for example, the uppercased form of a name has the same hash code as its lowercased equivalent.

**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:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### clone

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

Creates a copy of this compound name. Changes to the components of this compound name won't affect the new copy and vice versa. The clone and this compound name share the same syntax.

**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 compound name.**See Also:**[Cloneable](http://docs.google.com/java/lang/Cloneable.html)

### compareTo

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

Compares this CompoundName 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 CompoundName, ClassCastException is thrown.

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

Ordering of compound names depend on the syntax of the compound name. By default, they follow lexicographical rules for string comparison with the extension that this applies to all the components in the compound name and that comparison of individual components is affected by the jndi.syntax.ignorecase and jndi.syntax.trimblanks properties, identical to how they affect equals(). If this compound name is "lexicographically" lesser than obj, a negative number is returned. If this compound name is "lexicographically" greater than obj, a positive number is returned.

Implementation note: Currently the syntax properties of the two compound names are not compared when checking order. They might be in the future.

**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 CompoundName.**See Also:**[equals(java.lang.Object)](http://docs.google.com/javax/naming/CompoundName.html#equals(java.lang.Object))

### size

public int **size**()

Retrieves the number of components in this compound 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 compound name.

### isEmpty

public boolean **isEmpty**()

Determines whether this compound name is empty. A compound 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 compound 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 compound name as an enumeration of strings. The effects of updates to this compound 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 compound 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 compound 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 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 compound name whose components consist of a prefix of the components in this compound name. The result and this compound name share the same syntax. Subsequent changes to this compound name does not affect the name that is returned and vice versa.

**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 compound 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 compound name whose components consist of a suffix of the components in this compound name. The result and this compound name share the same syntax. Subsequent changes to this compound 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 compound name consisting of the components at indexes in the range [posn,size()). If posn is equal to size(), an empty compound 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 compound name is a prefix of this compound name. A compound name 'n' is a prefix if it is equal to getPrefix(n.size())--in other words, this compound name starts with 'n'. If n is null or not a compound name, false is returned.

Implementation note: Currently the syntax properties of n are not used when doing the comparison. They might be in the future.

**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 compound name to check. **Returns:**true if n is a CompoundName and is a prefix of this compound name, false otherwise.

### endsWith

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

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

Implementation note: Currently the syntax properties of n are not used when doing the comparison. They might be in the future.

**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 compound name to check. **Returns:**true if n is a CompoundName and is a suffix of this compound 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 compound name -- in order -- to the end of this compound name.

Implementation note: Currently the syntax properties of suffix is not used or checked. They might be in the future.

**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 CompoundName, not a new one. Cannot be null. **Throws:** [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If suffix is not a compound name, or if the addition of the components violates the syntax of this compound name (e.g. exceeding number of components).

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

Implementation note: Currently the syntax properties of suffix is not used or checked. They might be in the future.

**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 CompoundName, 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 n is not a compound name, or if the addition of the components violates the syntax of this compound name (e.g. exceeding number of components).

### 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 compound 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 CompoundName, 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 compound 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 compound name. Components of this compound 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 CompoundName, 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 compound 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 compound name. The component of this compound 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 compound name is empty). [InvalidNameException](http://docs.google.com/javax/naming/InvalidNameException.html) - If deleting the component would violate the compound 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/CompoundName.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/CompositeName.html)   [**NEXT CLASS**](http://docs.google.com/javax/naming/ConfigurationException.html) | [**FRAMES**](http://docs.google.com/index.html?javax/naming/CompoundName.html)    [**NO FRAMES**](http://docs.google.com/CompoundName.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#lnxbz9) |

[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).