| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractPreferences.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   [**NEXT CLASS**](http://docs.google.com/java/util/prefs/BackingStoreException.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/prefs/AbstractPreferences.html)    [**NO FRAMES**](http://docs.google.com/AbstractPreferences.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#26in1rg) | [METHOD](#35nkun2) |

## **java.util.prefs**

Class AbstractPreferences

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.util.prefs.Preferences](http://docs.google.com/java/util/prefs/Preferences.html)  
 **java.util.prefs.AbstractPreferences**

public abstract class **AbstractPreferences**extends [Preferences](http://docs.google.com/java/util/prefs/Preferences.html)

This class provides a skeletal implementation of the [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) class, greatly easing the task of implementing it.

**This class is for Preferences implementers only. Normal users of the Preferences facility should have no need to consult this documentation. The** [**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html) **documentation should suffice.**

Implementors must override the nine abstract service-provider interface (SPI) methods: [getSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getSpi(java.lang.String)), [putSpi(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putSpi(java.lang.String,%20java.lang.String)), [removeSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeSpi(java.lang.String)), [childSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)), [removeNodeSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNodeSpi()), [keysSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#keysSpi()), [childrenNamesSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNamesSpi()), [syncSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#syncSpi()) and [flushSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#flushSpi()). All of the concrete methods specify precisely how they are implemented atop these SPI methods. The implementor may, at his discretion, override one or more of the concrete methods if the default implementation is unsatisfactory for any reason, such as performance.

The SPI methods fall into three groups concerning exception behavior. The getSpi method should never throw exceptions, but it doesn't really matter, as any exception thrown by this method will be intercepted by [get(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)), which will return the specified default value to the caller. The removeNodeSpi, keysSpi, childrenNamesSpi, syncSpi and flushSpi methods are specified to throw [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html), and the implementation is required to throw this checked exception if it is unable to perform the operation. The exception propagates outward, causing the corresponding API method to fail.

The remaining SPI methods [putSpi(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putSpi(java.lang.String,%20java.lang.String)), [removeSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeSpi(java.lang.String)) and [childSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)) have more complicated exception behavior. They are not specified to throw BackingStoreException, as they can generally obey their contracts even if the backing store is unavailable. This is true because they return no information and their effects are not required to become permanent until a subsequent call to [Preferences.flush()](http://docs.google.com/java/util/prefs/Preferences.html#flush()) or [Preferences.sync()](http://docs.google.com/java/util/prefs/Preferences.html#sync()). Generally speaking, these SPI methods should not throw exceptions. In some implementations, there may be circumstances under which these calls cannot even enqueue the requested operation for later processing. Even under these circumstances it is generally better to simply ignore the invocation and return, rather than throwing an exception. Under these circumstances, however, all subsequent invocations of flush() and sync should return false, as returning true would imply that all previous operations had successfully been made permanent.

There is one circumstance under which putSpi, removeSpi and childSpi *should* throw an exception: if the caller lacks sufficient privileges on the underlying operating system to perform the requested operation. This will, for instance, occur on most systems if a non-privileged user attempts to modify system preferences. (The required privileges will vary from implementation to implementation. On some implementations, they are the right to modify the contents of some directory in the file system; on others they are the right to modify contents of some key in a registry.) Under any of these circumstances, it would generally be undesirable to let the program continue executing as if these operations would become permanent at a later time. While implementations are not required to throw an exception under these circumstances, they are encouraged to do so. A [SecurityException](http://docs.google.com/java/lang/SecurityException.html) would be appropriate.

Most of the SPI methods require the implementation to read or write information at a preferences node. The implementor should beware of the fact that another VM may have concurrently deleted this node from the backing store. It is the implementation's responsibility to recreate the node if it has been deleted.

Implementation note: In Sun's default Preferences implementations, the user's identity is inherited from the underlying operating system and does not change for the lifetime of the virtual machine. It is recognized that server-side Preferences implementations may have the user identity change from request to request, implicitly passed to Preferences methods via the use of a static [ThreadLocal](http://docs.google.com/java/lang/ThreadLocal.html) instance. Authors of such implementations are *strongly* encouraged to determine the user at the time preferences are accessed (for example by the [get(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)) or [put(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String)) method) rather than permanently associating a user with each Preferences instance. The latter behavior conflicts with normal Preferences usage and would lead to great confusion.

**Since:** 1.4 **See Also:**[Preferences](http://docs.google.com/java/util/prefs/Preferences.html)

| **Field Summary** | |
| --- | --- |
| protected  [Object](http://docs.google.com/java/lang/Object.html) | [**lock**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#lock)            An object whose monitor is used to lock this node. |
| protected  boolean | [**newNode**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#newNode)            This field should be true if this node did not exist in the backing store prior to the creation of this object. |

| **Fields inherited from class java.util.prefs.**[**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html) |
| --- |
| [MAX\_KEY\_LENGTH](http://docs.google.com/java/util/prefs/Preferences.html#MAX_KEY_LENGTH), [MAX\_NAME\_LENGTH](http://docs.google.com/java/util/prefs/Preferences.html#MAX_NAME_LENGTH), [MAX\_VALUE\_LENGTH](http://docs.google.com/java/util/prefs/Preferences.html#MAX_VALUE_LENGTH) |

| **Constructor Summary** | |
| --- | --- |
| protected | [**AbstractPreferences**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#AbstractPreferences(java.util.prefs.AbstractPreferences,%20java.lang.String))([AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html) parent, [String](http://docs.google.com/java/lang/String.html) name)            Creates a preference node with the specified parent and the specified name relative to its parent. |

| **Method Summary** | |
| --- | --- |
| [String](http://docs.google.com/java/lang/String.html) | [**absolutePath**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#absolutePath())()            Implements the absolutePath method as per the specification in [Preferences.absolutePath()](http://docs.google.com/java/util/prefs/Preferences.html#absolutePath()). |
| void | [**addNodeChangeListener**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#addNodeChangeListener(java.util.prefs.NodeChangeListener))([NodeChangeListener](http://docs.google.com/java/util/prefs/NodeChangeListener.html) ncl)            Registers the specified listener to receive *node change events* for this node. |
| void | [**addPreferenceChangeListener**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#addPreferenceChangeListener(java.util.prefs.PreferenceChangeListener))([PreferenceChangeListener](http://docs.google.com/java/util/prefs/PreferenceChangeListener.html) pcl)            Registers the specified listener to receive *preference change events* for this preference node. |
| protected  [AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html)[] | [**cachedChildren**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#cachedChildren())()            Returns all known unremoved children of this node. |
| [String](http://docs.google.com/java/lang/String.html)[] | [**childrenNames**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNames())()            Implements the children method as per the specification in [Preferences.childrenNames()](http://docs.google.com/java/util/prefs/Preferences.html#childrenNames()). |
| protected abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**childrenNamesSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNamesSpi())()            Returns the names of the children of this preference node. |
| protected abstract  [AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html) | [**childSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Returns the named child of this preference node, creating it if it does not already exist. |
| void | [**clear**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#clear())()            Implements the clear method as per the specification in [Preferences.clear()](http://docs.google.com/java/util/prefs/Preferences.html#clear()). |
| void | [**exportNode**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#exportNode(java.io.OutputStream))([OutputStream](http://docs.google.com/java/io/OutputStream.html) os)            Implements the exportNode method as per the specification in [Preferences.exportNode(OutputStream)](http://docs.google.com/java/util/prefs/Preferences.html#exportNode(java.io.OutputStream)). |
| void | [**exportSubtree**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#exportSubtree(java.io.OutputStream))([OutputStream](http://docs.google.com/java/io/OutputStream.html) os)            Implements the exportSubtree method as per the specification in [Preferences.exportSubtree(OutputStream)](http://docs.google.com/java/util/prefs/Preferences.html#exportSubtree(java.io.OutputStream)). |
| void | [**flush**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#flush())()            Implements the flush method as per the specification in [Preferences.flush()](http://docs.google.com/java/util/prefs/Preferences.html#flush()). |
| protected abstract  void | [**flushSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#flushSpi())()            This method is invoked with this node locked. |
| [String](http://docs.google.com/java/lang/String.html) | [**get**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) key, [String](http://docs.google.com/java/lang/String.html) def)            Implements the get method as per the specification in [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String)). |
| boolean | [**getBoolean**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getBoolean(java.lang.String,%20boolean))([String](http://docs.google.com/java/lang/String.html) key, boolean def)            Implements the getBoolean method as per the specification in [Preferences.getBoolean(String,boolean)](http://docs.google.com/java/util/prefs/Preferences.html#getBoolean(java.lang.String,%20boolean)). |
| byte[] | [**getByteArray**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getByteArray(java.lang.String,%20byte%5B%5D))([String](http://docs.google.com/java/lang/String.html) key, byte[] def)            Implements the getByteArray method as per the specification in [Preferences.getByteArray(String,byte[])](http://docs.google.com/java/util/prefs/Preferences.html#getByteArray(java.lang.String,%20byte%5B%5D)). |
| protected  [AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html) | [**getChild**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getChild(java.lang.String))([String](http://docs.google.com/java/lang/String.html) nodeName)            Returns the named child if it exists, or null if it does not. |
| double | [**getDouble**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getDouble(java.lang.String,%20double))([String](http://docs.google.com/java/lang/String.html) key, double def)            Implements the getDouble method as per the specification in [Preferences.getDouble(String,double)](http://docs.google.com/java/util/prefs/Preferences.html#getDouble(java.lang.String,%20double)). |
| float | [**getFloat**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getFloat(java.lang.String,%20float))([String](http://docs.google.com/java/lang/String.html) key, float def)            Implements the getFloat method as per the specification in [Preferences.getFloat(String,float)](http://docs.google.com/java/util/prefs/Preferences.html#getFloat(java.lang.String,%20float)). |
| int | [**getInt**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getInt(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) key, int def)            Implements the getInt method as per the specification in [Preferences.getInt(String,int)](http://docs.google.com/java/util/prefs/Preferences.html#getInt(java.lang.String,%20int)). |
| long | [**getLong**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getLong(java.lang.String,%20long))([String](http://docs.google.com/java/lang/String.html) key, long def)            Implements the getLong method as per the specification in [Preferences.getLong(String,long)](http://docs.google.com/java/util/prefs/Preferences.html#getLong(java.lang.String,%20long)). |
| protected abstract  [String](http://docs.google.com/java/lang/String.html) | [**getSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getSpi(java.lang.String))([String](http://docs.google.com/java/lang/String.html) key)            Return the value associated with the specified key at this preference node, or null if there is no association for this key, or the association cannot be determined at this time. |
| protected  boolean | [**isRemoved**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#isRemoved())()            Returns true iff this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. |
| boolean | [**isUserNode**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#isUserNode())()            Implements the isUserNode method as per the specification in [Preferences.isUserNode()](http://docs.google.com/java/util/prefs/Preferences.html#isUserNode()). |
| [String](http://docs.google.com/java/lang/String.html)[] | [**keys**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#keys())()            Implements the keys method as per the specification in [Preferences.keys()](http://docs.google.com/java/util/prefs/Preferences.html#keys()). |
| protected abstract  [String](http://docs.google.com/java/lang/String.html)[] | [**keysSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#keysSpi())()            Returns all of the keys that have an associated value in this preference node. |
| [String](http://docs.google.com/java/lang/String.html) | [**name**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#name())()            Implements the name method as per the specification in [Preferences.name()](http://docs.google.com/java/util/prefs/Preferences.html#name()). |
| [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) | [**node**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#node(java.lang.String))([String](http://docs.google.com/java/lang/String.html) path)            Implements the node method as per the specification in [Preferences.node(String)](http://docs.google.com/java/util/prefs/Preferences.html#node(java.lang.String)). |
| boolean | [**nodeExists**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#nodeExists(java.lang.String))([String](http://docs.google.com/java/lang/String.html) path)            Implements the nodeExists method as per the specification in [Preferences.nodeExists(String)](http://docs.google.com/java/util/prefs/Preferences.html#nodeExists(java.lang.String)). |
| [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) | [**parent**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#parent())()            Implements the parent method as per the specification in [Preferences.parent()](http://docs.google.com/java/util/prefs/Preferences.html#parent()). |
| void | [**put**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) key, [String](http://docs.google.com/java/lang/String.html) value)            Implements the put method as per the specification in [Preferences.put(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#put(java.lang.String,%20java.lang.String)). |
| void | [**putBoolean**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putBoolean(java.lang.String,%20boolean))([String](http://docs.google.com/java/lang/String.html) key, boolean value)            Implements the putBoolean method as per the specification in [Preferences.putBoolean(String,boolean)](http://docs.google.com/java/util/prefs/Preferences.html#putBoolean(java.lang.String,%20boolean)). |
| void | [**putByteArray**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putByteArray(java.lang.String,%20byte%5B%5D))([String](http://docs.google.com/java/lang/String.html) key, byte[] value)            Implements the putByteArray method as per the specification in [Preferences.putByteArray(String,byte[])](http://docs.google.com/java/util/prefs/Preferences.html#putByteArray(java.lang.String,%20byte%5B%5D)). |
| void | [**putDouble**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putDouble(java.lang.String,%20double))([String](http://docs.google.com/java/lang/String.html) key, double value)            Implements the putDouble method as per the specification in [Preferences.putDouble(String,double)](http://docs.google.com/java/util/prefs/Preferences.html#putDouble(java.lang.String,%20double)). |
| void | [**putFloat**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putFloat(java.lang.String,%20float))([String](http://docs.google.com/java/lang/String.html) key, float value)            Implements the putFloat method as per the specification in [Preferences.putFloat(String,float)](http://docs.google.com/java/util/prefs/Preferences.html#putFloat(java.lang.String,%20float)). |
| void | [**putInt**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putInt(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) key, int value)            Implements the putInt method as per the specification in [Preferences.putInt(String,int)](http://docs.google.com/java/util/prefs/Preferences.html#putInt(java.lang.String,%20int)). |
| void | [**putLong**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putLong(java.lang.String,%20long))([String](http://docs.google.com/java/lang/String.html) key, long value)            Implements the putLong method as per the specification in [Preferences.putLong(String,long)](http://docs.google.com/java/util/prefs/Preferences.html#putLong(java.lang.String,%20long)). |
| protected abstract  void | [**putSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putSpi(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) key, [String](http://docs.google.com/java/lang/String.html) value)            Put the given key-value association into this preference node. |
| void | [**remove**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#remove(java.lang.String))([String](http://docs.google.com/java/lang/String.html) key)            Implements the remove(String) method as per the specification in [Preferences.remove(String)](http://docs.google.com/java/util/prefs/Preferences.html#remove(java.lang.String)). |
| void | [**removeNode**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode())()            Implements the removeNode() method as per the specification in [Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode()). |
| void | [**removeNodeChangeListener**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNodeChangeListener(java.util.prefs.NodeChangeListener))([NodeChangeListener](http://docs.google.com/java/util/prefs/NodeChangeListener.html) ncl)            Removes the specified NodeChangeListener, so it no longer receives change events. |
| protected abstract  void | [**removeNodeSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNodeSpi())()            Removes this preference node, invalidating it and any preferences that it contains. |
| void | [**removePreferenceChangeListener**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removePreferenceChangeListener(java.util.prefs.PreferenceChangeListener))([PreferenceChangeListener](http://docs.google.com/java/util/prefs/PreferenceChangeListener.html) pcl)            Removes the specified preference change listener, so it no longer receives preference change events. |
| protected abstract  void | [**removeSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeSpi(java.lang.String))([String](http://docs.google.com/java/lang/String.html) key)            Remove the association (if any) for the specified key at this preference node. |
| void | [**sync**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#sync())()            Implements the sync method as per the specification in [Preferences.sync()](http://docs.google.com/java/util/prefs/Preferences.html#sync()). |
| protected abstract  void | [**syncSpi**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#syncSpi())()            This method is invoked with this node locked. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/prefs/AbstractPreferences.html#toString())()            Returns the absolute path name of this preferences node. |

| **Methods inherited from class java.util.prefs.**[**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html) |
| --- |
| [importPreferences](http://docs.google.com/java/util/prefs/Preferences.html#importPreferences(java.io.InputStream)), [systemNodeForPackage](http://docs.google.com/java/util/prefs/Preferences.html#systemNodeForPackage(java.lang.Class)), [systemRoot](http://docs.google.com/java/util/prefs/Preferences.html#systemRoot()), [userNodeForPackage](http://docs.google.com/java/util/prefs/Preferences.html#userNodeForPackage(java.lang.Class)), [userRoot](http://docs.google.com/java/util/prefs/Preferences.html#userRoot()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [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** |
| --- |

### newNode

protected boolean **newNode**

This field should be true if this node did not exist in the backing store prior to the creation of this object. The field is initialized to false, but may be set to true by a subclass constructor (and should not be modified thereafter). This field indicates whether a node change event should be fired when creation is complete.

### lock

protected final [Object](http://docs.google.com/java/lang/Object.html) **lock**

An object whose monitor is used to lock this node. This object is used in preference to the node itself to reduce the likelihood of intentional or unintentional denial of service due to a locked node. To avoid deadlock, a node is *never* locked by a thread that holds a lock on a descendant of that node.

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

### AbstractPreferences

protected **AbstractPreferences**([AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html) parent,  
 [String](http://docs.google.com/java/lang/String.html) name)

Creates a preference node with the specified parent and the specified name relative to its parent.

**Parameters:**parent - the parent of this preference node, or null if this is the root.name - the name of this preference node, relative to its parent, or "" if this is the root. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if name contains a slash ('/'), or parent is null and name isn't "".

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

### put

public void **put**([String](http://docs.google.com/java/lang/String.html) key,  
 [String](http://docs.google.com/java/lang/String.html) value)

Implements the put method as per the specification in [Preferences.put(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#put(java.lang.String,%20java.lang.String)).

This implementation checks that the key and value are legal, obtains this preference node's lock, checks that the node has not been removed, invokes [putSpi(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#putSpi(java.lang.String,%20java.lang.String)), and if there are any preference change listeners, enqueues a notification event for processing by the event dispatch thread.

**Specified by:**[put](http://docs.google.com/java/util/prefs/Preferences.html#put(java.lang.String,%20java.lang.String)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the specified value is to be associated.value - value to be associated with the specified key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key or value is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH or if value.length exceeds MAX\_VALUE\_LENGTH. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.

### get

public [String](http://docs.google.com/java/lang/String.html) **get**([String](http://docs.google.com/java/lang/String.html) key,  
 [String](http://docs.google.com/java/lang/String.html) def)

Implements the get method as per the specification in [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String)).

This implementation first checks to see if key is null throwing a NullPointerException if this is the case. Then it obtains this preference node's lock, checks that the node has not been removed, invokes [getSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getSpi(java.lang.String)), and returns the result, unless the getSpi invocation returns null or throws an exception, in which case this invocation returns def.

**Specified by:**[get](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned.def - the value to be returned in the event that this preference node has no value associated with key. **Returns:**the value associated with key, or def if no value is associated with key. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. (A null default *is* permitted.)

### remove

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

Implements the remove(String) method as per the specification in [Preferences.remove(String)](http://docs.google.com/java/util/prefs/Preferences.html#remove(java.lang.String)).

This implementation obtains this preference node's lock, checks that the node has not been removed, invokes [removeSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeSpi(java.lang.String)) and if there are any preference change listeners, enqueues a notification event for processing by the event dispatch thread.

**Specified by:**[remove](http://docs.google.com/java/util/prefs/Preferences.html#remove(java.lang.String)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose mapping is to be removed from the preference node. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.

### clear

public void **clear**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the clear method as per the specification in [Preferences.clear()](http://docs.google.com/java/util/prefs/Preferences.html#clear()).

This implementation obtains this preference node's lock, invokes [keys()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#keys()) to obtain an array of keys, and iterates over the array invoking [remove(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#remove(java.lang.String)) on each key.

**Specified by:**[clear](http://docs.google.com/java/util/prefs/Preferences.html#clear()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode())

### putInt

public void **putInt**([String](http://docs.google.com/java/lang/String.html) key,  
 int value)

Implements the putInt method as per the specification in [Preferences.putInt(String,int)](http://docs.google.com/java/util/prefs/Preferences.html#putInt(java.lang.String,%20int)).

This implementation translates value to a string with [Integer.toString(int)](http://docs.google.com/java/lang/Integer.html#toString(int)) and invokes [put(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String)) on the result.

**Specified by:**[putInt](http://docs.google.com/java/util/prefs/Preferences.html#putInt(java.lang.String,%20int)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the string form of value is to be associated.value - value whose string form is to be associated with key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.getInt(String,int)](http://docs.google.com/java/util/prefs/Preferences.html#getInt(java.lang.String,%20int))

### getInt

public int **getInt**([String](http://docs.google.com/java/lang/String.html) key,  
 int def)

Implements the getInt method as per the specification in [Preferences.getInt(String,int)](http://docs.google.com/java/util/prefs/Preferences.html#getInt(java.lang.String,%20int)).

This implementation invokes [get(key, null)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)). If the return value is non-null, the implementation attempts to translate it to an int with [Integer.parseInt(String)](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String)). If the attempt succeeds, the return value is returned by this method. Otherwise, def is returned.

**Specified by:**[getInt](http://docs.google.com/java/util/prefs/Preferences.html#getInt(java.lang.String,%20int)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned as an int.def - the value to be returned in the event that this preference node has no value associated with key or the associated value cannot be interpreted as an int. **Returns:**the int value represented by the string associated with key in this preference node, or def if the associated value does not exist or cannot be interpreted as an int. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null.**See Also:**[Preferences.putInt(String,int)](http://docs.google.com/java/util/prefs/Preferences.html#putInt(java.lang.String,%20int)), [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String))

### putLong

public void **putLong**([String](http://docs.google.com/java/lang/String.html) key,  
 long value)

Implements the putLong method as per the specification in [Preferences.putLong(String,long)](http://docs.google.com/java/util/prefs/Preferences.html#putLong(java.lang.String,%20long)).

This implementation translates value to a string with [Long.toString(long)](http://docs.google.com/java/lang/Long.html#toString(long)) and invokes [put(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String)) on the result.

**Specified by:**[putLong](http://docs.google.com/java/util/prefs/Preferences.html#putLong(java.lang.String,%20long)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the string form of value is to be associated.value - value whose string form is to be associated with key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.getLong(String,long)](http://docs.google.com/java/util/prefs/Preferences.html#getLong(java.lang.String,%20long))

### getLong

public long **getLong**([String](http://docs.google.com/java/lang/String.html) key,  
 long def)

Implements the getLong method as per the specification in [Preferences.getLong(String,long)](http://docs.google.com/java/util/prefs/Preferences.html#getLong(java.lang.String,%20long)).

This implementation invokes [get(key, null)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)). If the return value is non-null, the implementation attempts to translate it to a long with [Long.parseLong(String)](http://docs.google.com/java/lang/Long.html#parseLong(java.lang.String)). If the attempt succeeds, the return value is returned by this method. Otherwise, def is returned.

**Specified by:**[getLong](http://docs.google.com/java/util/prefs/Preferences.html#getLong(java.lang.String,%20long)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned as a long.def - the value to be returned in the event that this preference node has no value associated with key or the associated value cannot be interpreted as a long. **Returns:**the long value represented by the string associated with key in this preference node, or def if the associated value does not exist or cannot be interpreted as a long. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null.**See Also:**[Preferences.putLong(String,long)](http://docs.google.com/java/util/prefs/Preferences.html#putLong(java.lang.String,%20long)), [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String))

### putBoolean

public void **putBoolean**([String](http://docs.google.com/java/lang/String.html) key,  
 boolean value)

Implements the putBoolean method as per the specification in [Preferences.putBoolean(String,boolean)](http://docs.google.com/java/util/prefs/Preferences.html#putBoolean(java.lang.String,%20boolean)).

This implementation translates value to a string with [String.valueOf(boolean)](http://docs.google.com/java/lang/String.html#valueOf(boolean)) and invokes [put(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String)) on the result.

**Specified by:**[putBoolean](http://docs.google.com/java/util/prefs/Preferences.html#putBoolean(java.lang.String,%20boolean)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the string form of value is to be associated.value - value whose string form is to be associated with key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.getBoolean(String,boolean)](http://docs.google.com/java/util/prefs/Preferences.html#getBoolean(java.lang.String,%20boolean)), [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String))

### getBoolean

public boolean **getBoolean**([String](http://docs.google.com/java/lang/String.html) key,  
 boolean def)

Implements the getBoolean method as per the specification in [Preferences.getBoolean(String,boolean)](http://docs.google.com/java/util/prefs/Preferences.html#getBoolean(java.lang.String,%20boolean)).

This implementation invokes [get(key, null)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)). If the return value is non-null, it is compared with "true" using [String.equalsIgnoreCase(String)](http://docs.google.com/java/lang/String.html#equalsIgnoreCase(java.lang.String)). If the comparison returns true, this invocation returns true. Otherwise, the original return value is compared with "false", again using [String.equalsIgnoreCase(String)](http://docs.google.com/java/lang/String.html#equalsIgnoreCase(java.lang.String)). If the comparison returns true, this invocation returns false. Otherwise, this invocation returns def.

**Specified by:**[getBoolean](http://docs.google.com/java/util/prefs/Preferences.html#getBoolean(java.lang.String,%20boolean)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned as a boolean.def - the value to be returned in the event that this preference node has no value associated with key or the associated value cannot be interpreted as a boolean. **Returns:**the boolean value represented by the string associated with key in this preference node, or def if the associated value does not exist or cannot be interpreted as a boolean. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null.**See Also:**[Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String)), [Preferences.putBoolean(String,boolean)](http://docs.google.com/java/util/prefs/Preferences.html#putBoolean(java.lang.String,%20boolean))

### putFloat

public void **putFloat**([String](http://docs.google.com/java/lang/String.html) key,  
 float value)

Implements the putFloat method as per the specification in [Preferences.putFloat(String,float)](http://docs.google.com/java/util/prefs/Preferences.html#putFloat(java.lang.String,%20float)).

This implementation translates value to a string with [Float.toString(float)](http://docs.google.com/java/lang/Float.html#toString(float)) and invokes [put(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String)) on the result.

**Specified by:**[putFloat](http://docs.google.com/java/util/prefs/Preferences.html#putFloat(java.lang.String,%20float)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the string form of value is to be associated.value - value whose string form is to be associated with key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.getFloat(String,float)](http://docs.google.com/java/util/prefs/Preferences.html#getFloat(java.lang.String,%20float))

### getFloat

public float **getFloat**([String](http://docs.google.com/java/lang/String.html) key,  
 float def)

Implements the getFloat method as per the specification in [Preferences.getFloat(String,float)](http://docs.google.com/java/util/prefs/Preferences.html#getFloat(java.lang.String,%20float)).

This implementation invokes [get(key, null)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)). If the return value is non-null, the implementation attempts to translate it to an float with [Float.parseFloat(String)](http://docs.google.com/java/lang/Float.html#parseFloat(java.lang.String)). If the attempt succeeds, the return value is returned by this method. Otherwise, def is returned.

**Specified by:**[getFloat](http://docs.google.com/java/util/prefs/Preferences.html#getFloat(java.lang.String,%20float)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned as a float.def - the value to be returned in the event that this preference node has no value associated with key or the associated value cannot be interpreted as a float. **Returns:**the float value represented by the string associated with key in this preference node, or def if the associated value does not exist or cannot be interpreted as a float. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null.**See Also:**[Preferences.putFloat(String,float)](http://docs.google.com/java/util/prefs/Preferences.html#putFloat(java.lang.String,%20float)), [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String))

### putDouble

public void **putDouble**([String](http://docs.google.com/java/lang/String.html) key,  
 double value)

Implements the putDouble method as per the specification in [Preferences.putDouble(String,double)](http://docs.google.com/java/util/prefs/Preferences.html#putDouble(java.lang.String,%20double)).

This implementation translates value to a string with [Double.toString(double)](http://docs.google.com/java/lang/Double.html#toString(double)) and invokes [put(String,String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#put(java.lang.String,%20java.lang.String)) on the result.

**Specified by:**[putDouble](http://docs.google.com/java/util/prefs/Preferences.html#putDouble(java.lang.String,%20double)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the string form of value is to be associated.value - value whose string form is to be associated with key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.getDouble(String,double)](http://docs.google.com/java/util/prefs/Preferences.html#getDouble(java.lang.String,%20double))

### getDouble

public double **getDouble**([String](http://docs.google.com/java/lang/String.html) key,  
 double def)

Implements the getDouble method as per the specification in [Preferences.getDouble(String,double)](http://docs.google.com/java/util/prefs/Preferences.html#getDouble(java.lang.String,%20double)).

This implementation invokes [get(key, null)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#get(java.lang.String,%20java.lang.String)). If the return value is non-null, the implementation attempts to translate it to an double with [Double.parseDouble(String)](http://docs.google.com/java/lang/Double.html#parseDouble(java.lang.String)). If the attempt succeeds, the return value is returned by this method. Otherwise, def is returned.

**Specified by:**[getDouble](http://docs.google.com/java/util/prefs/Preferences.html#getDouble(java.lang.String,%20double)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned as a double.def - the value to be returned in the event that this preference node has no value associated with key or the associated value cannot be interpreted as a double. **Returns:**the double value represented by the string associated with key in this preference node, or def if the associated value does not exist or cannot be interpreted as a double. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null.**See Also:**[Preferences.putDouble(String,double)](http://docs.google.com/java/util/prefs/Preferences.html#putDouble(java.lang.String,%20double)), [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String))

### putByteArray

public void **putByteArray**([String](http://docs.google.com/java/lang/String.html) key,  
 byte[] value)

Implements the putByteArray method as per the specification in [Preferences.putByteArray(String,byte[])](http://docs.google.com/java/util/prefs/Preferences.html#putByteArray(java.lang.String,%20byte%5B%5D)).

**Specified by:**[putByteArray](http://docs.google.com/java/util/prefs/Preferences.html#putByteArray(java.lang.String,%20byte%5B%5D)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key with which the string form of value is to be associated.value - value whose string form is to be associated with key. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key or value is null. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if key.length() exceeds MAX\_KEY\_LENGTH or if value.length exceeds MAX\_VALUE\_LENGTH\*3/4. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.getByteArray(String,byte[])](http://docs.google.com/java/util/prefs/Preferences.html#getByteArray(java.lang.String,%20byte%5B%5D)), [Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String))

### getByteArray

public byte[] **getByteArray**([String](http://docs.google.com/java/lang/String.html) key,  
 byte[] def)

Implements the getByteArray method as per the specification in [Preferences.getByteArray(String,byte[])](http://docs.google.com/java/util/prefs/Preferences.html#getByteArray(java.lang.String,%20byte%5B%5D)).

**Specified by:**[getByteArray](http://docs.google.com/java/util/prefs/Preferences.html#getByteArray(java.lang.String,%20byte%5B%5D)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**key - key whose associated value is to be returned as a byte array.def - the value to be returned in the event that this preference node has no value associated with key or the associated value cannot be interpreted as a byte array. **Returns:**the byte array value represented by the string associated with key in this preference node, or def if the associated value does not exist or cannot be interpreted as a byte array. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if key is null. (A null value for def *is* permitted.)**See Also:**[Preferences.get(String,String)](http://docs.google.com/java/util/prefs/Preferences.html#get(java.lang.String,%20java.lang.String)), [Preferences.putByteArray(String,byte[])](http://docs.google.com/java/util/prefs/Preferences.html#putByteArray(java.lang.String,%20byte%5B%5D))

### keys

public [String](http://docs.google.com/java/lang/String.html)[] **keys**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the keys method as per the specification in [Preferences.keys()](http://docs.google.com/java/util/prefs/Preferences.html#keys()).

This implementation obtains this preference node's lock, checks that the node has not been removed and invokes [keysSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#keysSpi()).

**Specified by:**[keys](http://docs.google.com/java/util/prefs/Preferences.html#keys()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**an array of the keys that have an associated value in this preference node. **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.

### childrenNames

public [String](http://docs.google.com/java/lang/String.html)[] **childrenNames**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the children method as per the specification in [Preferences.childrenNames()](http://docs.google.com/java/util/prefs/Preferences.html#childrenNames()).

This implementation obtains this preference node's lock, checks that the node has not been removed, constructs a TreeSet initialized to the names of children already cached (the children in this node's "child-cache"), invokes [childrenNamesSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNamesSpi()), and adds all of the returned child-names into the set. The elements of the tree set are dumped into a String array using the toArray method, and this array is returned.

**Specified by:**[childrenNames](http://docs.google.com/java/util/prefs/Preferences.html#childrenNames()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**the names of the children of this preference node. **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[cachedChildren()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#cachedChildren())

### cachedChildren

protected final [AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html)[] **cachedChildren**()

Returns all known unremoved children of this node.

**Returns:**all known unremoved children of this node.

### parent

public [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **parent**()

Implements the parent method as per the specification in [Preferences.parent()](http://docs.google.com/java/util/prefs/Preferences.html#parent()).

This implementation obtains this preference node's lock, checks that the node has not been removed and returns the parent value that was passed to this node's constructor.

**Specified by:**[parent](http://docs.google.com/java/util/prefs/Preferences.html#parent()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**the parent of this preference node. **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.

### node

public [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **node**([String](http://docs.google.com/java/lang/String.html) path)

Implements the node method as per the specification in [Preferences.node(String)](http://docs.google.com/java/util/prefs/Preferences.html#node(java.lang.String)).

This implementation obtains this preference node's lock and checks that the node has not been removed. If path is "", this node is returned; if path is "/", this node's root is returned. If the first character in path is not '/', the implementation breaks path into tokens and recursively traverses the path from this node to the named node, "consuming" a name and a slash from path at each step of the traversal. At each step, the current node is locked and the node's child-cache is checked for the named node. If it is not found, the name is checked to make sure its length does not exceed MAX\_NAME\_LENGTH. Then the [childSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)) method is invoked, and the result stored in this node's child-cache. If the newly created Preferences object's [newNode](http://docs.google.com/java/util/prefs/AbstractPreferences.html#newNode) field is true and there are any node change listeners, a notification event is enqueued for processing by the event dispatch thread.

When there are no more tokens, the last value found in the child-cache or returned by childSpi is returned by this method. If during the traversal, two "/" tokens occur consecutively, or the final token is "/" (rather than a name), an appropriate IllegalArgumentException is thrown.

If the first character of path is '/' (indicating an absolute path name) this preference node's lock is dropped prior to breaking path into tokens, and this method recursively traverses the path starting from the root (rather than starting from this node). The traversal is otherwise identical to the one described for relative path names. Dropping the lock on this node prior to commencing the traversal at the root node is essential to avoid the possibility of deadlock, as per the [locking invariant](http://docs.google.com/java/util/prefs/AbstractPreferences.html#lock).

**Specified by:**[node](http://docs.google.com/java/util/prefs/Preferences.html#node(java.lang.String)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**path - the path name of the preference node to return. **Returns:**the specified preference node. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the path name is invalid (i.e., it contains multiple consecutive slash characters, or ends with a slash character and is more than one character long). [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[Preferences.flush()](http://docs.google.com/java/util/prefs/Preferences.html#flush())

### nodeExists

public boolean **nodeExists**([String](http://docs.google.com/java/lang/String.html) path)  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the nodeExists method as per the specification in [Preferences.nodeExists(String)](http://docs.google.com/java/util/prefs/Preferences.html#nodeExists(java.lang.String)).

This implementation is very similar to [node(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#node(java.lang.String)), except that [getChild(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getChild(java.lang.String)) is used instead of [childSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)).

**Specified by:**[nodeExists](http://docs.google.com/java/util/prefs/Preferences.html#nodeExists(java.lang.String)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**path - the path name of the node whose existence is to be checked. **Returns:**true if the specified node exists. **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the path name is invalid (i.e., it contains multiple consecutive slash characters, or ends with a slash character and is more than one character long). [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method and pathname is not the empty string ("").

### removeNode

public void **removeNode**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the removeNode() method as per the specification in [Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode()).

This implementation checks to see that this node is the root; if so, it throws an appropriate exception. Then, it locks this node's parent, and calls a recursive helper method that traverses the subtree rooted at this node. The recursive method locks the node on which it was called, checks that it has not already been removed, and then ensures that all of its children are cached: The [childrenNamesSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNamesSpi()) method is invoked and each returned child name is checked for containment in the child-cache. If a child is not already cached, the [childSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)) method is invoked to create a Preferences instance for it, and this instance is put into the child-cache. Then the helper method calls itself recursively on each node contained in its child-cache. Next, it invokes [removeNodeSpi()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNodeSpi()), marks itself as removed, and removes itself from its parent's child-cache. Finally, if there are any node change listeners, it enqueues a notification event for processing by the event dispatch thread.

Note that the helper method is always invoked with all ancestors up to the "closest non-removed ancestor" locked.

**Specified by:**[removeNode](http://docs.google.com/java/util/prefs/Preferences.html#removeNode()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Throws:** [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has already been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. [UnsupportedOperationException](http://docs.google.com/java/lang/UnsupportedOperationException.html) - if this method is invoked on the root node. [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.**See Also:**[Preferences.flush()](http://docs.google.com/java/util/prefs/Preferences.html#flush())

### name

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

Implements the name method as per the specification in [Preferences.name()](http://docs.google.com/java/util/prefs/Preferences.html#name()).

This implementation merely returns the name that was passed to this node's constructor.

**Specified by:**[name](http://docs.google.com/java/util/prefs/Preferences.html#name()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**this preference node's name, relative to its parent.

### absolutePath

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

Implements the absolutePath method as per the specification in [Preferences.absolutePath()](http://docs.google.com/java/util/prefs/Preferences.html#absolutePath()).

This implementation merely returns the absolute path name that was computed at the time that this node was constructed (based on the name that was passed to this node's constructor, and the names that were passed to this node's ancestors' constructors).

**Specified by:**[absolutePath](http://docs.google.com/java/util/prefs/Preferences.html#absolutePath()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**this preference node's absolute path name.

### isUserNode

public boolean **isUserNode**()

Implements the isUserNode method as per the specification in [Preferences.isUserNode()](http://docs.google.com/java/util/prefs/Preferences.html#isUserNode()).

This implementation compares this node's root node (which is stored in a private field) with the value returned by [Preferences.userRoot()](http://docs.google.com/java/util/prefs/Preferences.html#userRoot()). If the two object references are identical, this method returns true.

**Specified by:**[isUserNode](http://docs.google.com/java/util/prefs/Preferences.html#isUserNode()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**true if this preference node is in the user preference tree, false if it's in the system preference tree.

### addPreferenceChangeListener

public void **addPreferenceChangeListener**([PreferenceChangeListener](http://docs.google.com/java/util/prefs/PreferenceChangeListener.html) pcl)

**Description copied from class:** [**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html#addPreferenceChangeListener(java.util.prefs.PreferenceChangeListener)) Registers the specified listener to receive *preference change events* for this preference node. A preference change event is generated when a preference is added to this node, removed from this node, or when the value associated with a preference is changed. (Preference change events are *not* generated by the [Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode()) method, which generates a *node change event*. Preference change events *are* generated by the clear method.)

Events are only guaranteed for changes made within the same JVM as the registered listener, though some implementations may generate events for changes made outside this JVM. Events may be generated before the changes have been made persistent. Events are not generated when preferences are modified in descendants of this node; a caller desiring such events must register with each descendant.

**Specified by:**[addPreferenceChangeListener](http://docs.google.com/java/util/prefs/Preferences.html#addPreferenceChangeListener(java.util.prefs.PreferenceChangeListener)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**pcl - The preference change listener to add.**See Also:**[Preferences.removePreferenceChangeListener(PreferenceChangeListener)](http://docs.google.com/java/util/prefs/Preferences.html#removePreferenceChangeListener(java.util.prefs.PreferenceChangeListener)), [Preferences.addNodeChangeListener(NodeChangeListener)](http://docs.google.com/java/util/prefs/Preferences.html#addNodeChangeListener(java.util.prefs.NodeChangeListener))

### removePreferenceChangeListener

public void **removePreferenceChangeListener**([PreferenceChangeListener](http://docs.google.com/java/util/prefs/PreferenceChangeListener.html) pcl)

**Description copied from class:** [**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html#removePreferenceChangeListener(java.util.prefs.PreferenceChangeListener)) Removes the specified preference change listener, so it no longer receives preference change events.

**Specified by:**[removePreferenceChangeListener](http://docs.google.com/java/util/prefs/Preferences.html#removePreferenceChangeListener(java.util.prefs.PreferenceChangeListener)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**pcl - The preference change listener to remove.**See Also:**[Preferences.addPreferenceChangeListener(PreferenceChangeListener)](http://docs.google.com/java/util/prefs/Preferences.html#addPreferenceChangeListener(java.util.prefs.PreferenceChangeListener))

### addNodeChangeListener

public void **addNodeChangeListener**([NodeChangeListener](http://docs.google.com/java/util/prefs/NodeChangeListener.html) ncl)

**Description copied from class:** [**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html#addNodeChangeListener(java.util.prefs.NodeChangeListener)) Registers the specified listener to receive *node change events* for this node. A node change event is generated when a child node is added to or removed from this node. (A single [Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode()) invocation results in multiple *node change events*, one for every node in the subtree rooted at the removed node.)

Events are only guaranteed for changes made within the same JVM as the registered listener, though some implementations may generate events for changes made outside this JVM. Events may be generated before the changes have become permanent. Events are not generated when indirect descendants of this node are added or removed; a caller desiring such events must register with each descendant.

Few guarantees can be made regarding node creation. Because nodes are created implicitly upon access, it may not be feasible for an implementation to determine whether a child node existed in the backing store prior to access (for example, because the backing store is unreachable or cached information is out of date). Under these circumstances, implementations are neither required to generate node change events nor prohibited from doing so.

**Specified by:**[addNodeChangeListener](http://docs.google.com/java/util/prefs/Preferences.html#addNodeChangeListener(java.util.prefs.NodeChangeListener)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**ncl - The NodeChangeListener to add.**See Also:**[Preferences.removeNodeChangeListener(NodeChangeListener)](http://docs.google.com/java/util/prefs/Preferences.html#removeNodeChangeListener(java.util.prefs.NodeChangeListener)), [Preferences.addPreferenceChangeListener(PreferenceChangeListener)](http://docs.google.com/java/util/prefs/Preferences.html#addPreferenceChangeListener(java.util.prefs.PreferenceChangeListener))

### removeNodeChangeListener

public void **removeNodeChangeListener**([NodeChangeListener](http://docs.google.com/java/util/prefs/NodeChangeListener.html) ncl)

**Description copied from class:** [**Preferences**](http://docs.google.com/java/util/prefs/Preferences.html#removeNodeChangeListener(java.util.prefs.NodeChangeListener)) Removes the specified NodeChangeListener, so it no longer receives change events.

**Specified by:**[removeNodeChangeListener](http://docs.google.com/java/util/prefs/Preferences.html#removeNodeChangeListener(java.util.prefs.NodeChangeListener)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**ncl - The NodeChangeListener to remove.**See Also:**[Preferences.addNodeChangeListener(NodeChangeListener)](http://docs.google.com/java/util/prefs/Preferences.html#addNodeChangeListener(java.util.prefs.NodeChangeListener))

### putSpi

protected abstract void **putSpi**([String](http://docs.google.com/java/lang/String.html) key,  
 [String](http://docs.google.com/java/lang/String.html) value)

Put the given key-value association into this preference node. It is guaranteed that key and value are non-null and of legal length. Also, it is guaranteed that this node has not been removed. (The implementor needn't check for any of these things.)

This method is invoked with the lock on this node held.

### getSpi

protected abstract [String](http://docs.google.com/java/lang/String.html) **getSpi**([String](http://docs.google.com/java/lang/String.html) key)

Return the value associated with the specified key at this preference node, or null if there is no association for this key, or the association cannot be determined at this time. It is guaranteed that key is non-null. Also, it is guaranteed that this node has not been removed. (The implementor needn't check for either of these things.)

Generally speaking, this method should not throw an exception under any circumstances. If, however, if it does throw an exception, the exception will be intercepted and treated as a null return value.

This method is invoked with the lock on this node held.

**Returns:**the value associated with the specified key at this preference node, or null if there is no association for this key, or the association cannot be determined at this time.

### removeSpi

protected abstract void **removeSpi**([String](http://docs.google.com/java/lang/String.html) key)

Remove the association (if any) for the specified key at this preference node. It is guaranteed that key is non-null. Also, it is guaranteed that this node has not been removed. (The implementor needn't check for either of these things.)

This method is invoked with the lock on this node held.

### removeNodeSpi

protected abstract void **removeNodeSpi**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Removes this preference node, invalidating it and any preferences that it contains. The named child will have no descendants at the time this invocation is made (i.e., the [Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode()) method invokes this method repeatedly in a bottom-up fashion, removing each of a node's descendants before removing the node itself).

This method is invoked with the lock held on this node and its parent (and all ancestors that are being removed as a result of a single invocation to [Preferences.removeNode()](http://docs.google.com/java/util/prefs/Preferences.html#removeNode())).

The removal of a node needn't become persistent until the flush method is invoked on this node (or an ancestor).

If this node throws a BackingStoreException, the exception will propagate out beyond the enclosing [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) invocation.

**Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.

### keysSpi

protected abstract [String](http://docs.google.com/java/lang/String.html)[] **keysSpi**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Returns all of the keys that have an associated value in this preference node. (The returned array will be of size zero if this node has no preferences.) It is guaranteed that this node has not been removed.

This method is invoked with the lock on this node held.

If this node throws a BackingStoreException, the exception will propagate out beyond the enclosing [keys()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#keys()) invocation.

**Returns:**an array of the keys that have an associated value in this preference node. **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.

### childrenNamesSpi

protected abstract [String](http://docs.google.com/java/lang/String.html)[] **childrenNamesSpi**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Returns the names of the children of this preference node. (The returned array will be of size zero if this node has no children.) This method need not return the names of any nodes already cached, but may do so without harm.

This method is invoked with the lock on this node held.

If this node throws a BackingStoreException, the exception will propagate out beyond the enclosing [childrenNames()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNames()) invocation.

**Returns:**an array containing the names of the children of this preference node. **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.

### getChild

protected [AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html) **getChild**([String](http://docs.google.com/java/lang/String.html) nodeName)  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Returns the named child if it exists, or null if it does not. It is guaranteed that nodeName is non-null, non-empty, does not contain the slash character ('/'), and is no longer than [Preferences.MAX\_NAME\_LENGTH](http://docs.google.com/java/util/prefs/Preferences.html#MAX_NAME_LENGTH) characters. Also, it is guaranteed that this node has not been removed. (The implementor needn't check for any of these things if he chooses to override this method.)

Finally, it is guaranteed that the named node has not been returned by a previous invocation of this method or [childSpi(java.lang.String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)) after the last time that it was removed. In other words, a cached value will always be used in preference to invoking this method. (The implementor needn't maintain his own cache of previously returned children if he chooses to override this method.)

This implementation obtains this preference node's lock, invokes [childrenNames()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childrenNames()) to get an array of the names of this node's children, and iterates over the array comparing the name of each child with the specified node name. If a child node has the correct name, the [childSpi(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#childSpi(java.lang.String)) method is invoked and the resulting node is returned. If the iteration completes without finding the specified name, null is returned.

**Parameters:**nodeName - name of the child to be searched for. **Returns:**the named child if it exists, or null if it does not. **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.

### childSpi

protected abstract [AbstractPreferences](http://docs.google.com/java/util/prefs/AbstractPreferences.html) **childSpi**([String](http://docs.google.com/java/lang/String.html) name)

Returns the named child of this preference node, creating it if it does not already exist. It is guaranteed that name is non-null, non-empty, does not contain the slash character ('/'), and is no longer than [Preferences.MAX\_NAME\_LENGTH](http://docs.google.com/java/util/prefs/Preferences.html#MAX_NAME_LENGTH) characters. Also, it is guaranteed that this node has not been removed. (The implementor needn't check for any of these things.)

Finally, it is guaranteed that the named node has not been returned by a previous invocation of this method or [getChild(String)](http://docs.google.com/java/util/prefs/AbstractPreferences.html#getChild(java.lang.String)) after the last time that it was removed. In other words, a cached value will always be used in preference to invoking this method. Subclasses need not maintain their own cache of previously returned children.

The implementer must ensure that the returned node has not been removed. If a like-named child of this node was previously removed, the implementer must return a newly constructed AbstractPreferences node; once removed, an AbstractPreferences node cannot be "resuscitated."

If this method causes a node to be created, this node is not guaranteed to be persistent until the flush method is invoked on this node or one of its ancestors (or descendants).

This method is invoked with the lock on this node held.

**Parameters:**name - The name of the child node to return, relative to this preference node. **Returns:**The named child node.

### toString

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

Returns the absolute path name of this preferences node.

**Specified by:**[toString](http://docs.google.com/java/util/prefs/Preferences.html#toString()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Returns:**a string representation of the object.

### sync

public void **sync**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the sync method as per the specification in [Preferences.sync()](http://docs.google.com/java/util/prefs/Preferences.html#sync()).

This implementation calls a recursive helper method that locks this node, invokes syncSpi() on it, unlocks this node, and recursively invokes this method on each "cached child." A cached child is a child of this node that has been created in this VM and not subsequently removed. In effect, this method does a depth first traversal of the "cached subtree" rooted at this node, calling syncSpi() on each node in the subTree while only that node is locked. Note that syncSpi() is invoked top-down.

**Specified by:**[sync](http://docs.google.com/java/util/prefs/Preferences.html#sync()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it. [IllegalStateException](http://docs.google.com/java/lang/IllegalStateException.html) - if this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.**See Also:**[flush()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#flush())

### syncSpi

protected abstract void **syncSpi**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

This method is invoked with this node locked. The contract of this method is to synchronize any cached preferences stored at this node with any stored in the backing store. (It is perfectly possible that this node does not exist on the backing store, either because it has been deleted by another VM, or because it has not yet been created.) Note that this method should *not* synchronize the preferences in any subnodes of this node. If the backing store naturally syncs an entire subtree at once, the implementer is encouraged to override sync(), rather than merely overriding this method.

If this node throws a BackingStoreException, the exception will propagate out beyond the enclosing [sync()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#sync()) invocation.

**Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.

### flush

public void **flush**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the flush method as per the specification in [Preferences.flush()](http://docs.google.com/java/util/prefs/Preferences.html#flush()).

This implementation calls a recursive helper method that locks this node, invokes flushSpi() on it, unlocks this node, and recursively invokes this method on each "cached child." A cached child is a child of this node that has been created in this VM and not subsequently removed. In effect, this method does a depth first traversal of the "cached subtree" rooted at this node, calling flushSpi() on each node in the subTree while only that node is locked. Note that flushSpi() is invoked top-down.

If this method is invoked on a node that has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method, flushSpi() is invoked on this node, but not on others.

**Specified by:**[flush](http://docs.google.com/java/util/prefs/Preferences.html#flush()) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.**See Also:**[flush()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#flush())

### flushSpi

protected abstract void **flushSpi**()  
 throws [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

This method is invoked with this node locked. The contract of this method is to force any cached changes in the contents of this preference node to the backing store, guaranteeing their persistence. (It is perfectly possible that this node does not exist on the backing store, either because it has been deleted by another VM, or because it has not yet been created.) Note that this method should *not* flush the preferences in any subnodes of this node. If the backing store naturally flushes an entire subtree at once, the implementer is encouraged to override flush(), rather than merely overriding this method.

If this node throws a BackingStoreException, the exception will propagate out beyond the enclosing [flush()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#flush()) invocation.

**Throws:** [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if this operation cannot be completed due to a failure in the backing store, or inability to communicate with it.

### isRemoved

protected boolean **isRemoved**()

Returns true iff this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method. This method locks this node prior to returning the contents of the private field used to track this state.

**Returns:**true iff this node (or an ancestor) has been removed with the [removeNode()](http://docs.google.com/java/util/prefs/AbstractPreferences.html#removeNode()) method.

### exportNode

public void **exportNode**([OutputStream](http://docs.google.com/java/io/OutputStream.html) os)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the exportNode method as per the specification in [Preferences.exportNode(OutputStream)](http://docs.google.com/java/util/prefs/Preferences.html#exportNode(java.io.OutputStream)).

**Specified by:**[exportNode](http://docs.google.com/java/util/prefs/Preferences.html#exportNode(java.io.OutputStream)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**os - the output stream on which to emit the XML document. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if writing to the specified output stream results in an IOException. [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if preference data cannot be read from backing store.**See Also:**[Preferences.importPreferences(InputStream)](http://docs.google.com/java/util/prefs/Preferences.html#importPreferences(java.io.InputStream))

### exportSubtree

public void **exportSubtree**([OutputStream](http://docs.google.com/java/io/OutputStream.html) os)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html)

Implements the exportSubtree method as per the specification in [Preferences.exportSubtree(OutputStream)](http://docs.google.com/java/util/prefs/Preferences.html#exportSubtree(java.io.OutputStream)).

**Specified by:**[exportSubtree](http://docs.google.com/java/util/prefs/Preferences.html#exportSubtree(java.io.OutputStream)) in class [Preferences](http://docs.google.com/java/util/prefs/Preferences.html) **Parameters:**os - the output stream on which to emit the XML document. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - if writing to the specified output stream results in an IOException. [BackingStoreException](http://docs.google.com/java/util/prefs/BackingStoreException.html) - if preference data cannot be read from backing store.**See Also:**[Preferences.importPreferences(InputStream)](http://docs.google.com/java/util/prefs/Preferences.html#importPreferences(java.io.InputStream)), [Preferences.exportNode(OutputStream)](http://docs.google.com/java/util/prefs/Preferences.html#exportNode(java.io.OutputStream))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AbstractPreferences.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   [**NEXT CLASS**](http://docs.google.com/java/util/prefs/BackingStoreException.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/prefs/AbstractPreferences.html)    [**NO FRAMES**](http://docs.google.com/AbstractPreferences.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: [FIELD](#2s8eyo1) | [CONSTR](#26in1rg) | [METHOD](#35nkun2) |

[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).