**public class Activity**

**public boolean onKeyDown (int keyCode,** [**KeyEvent**](http://developer.android.com/reference/android/view/KeyEvent.html) **event)**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Called when a key was pressed down and not handled by any of the views inside of the activity. So, for example, key presses while the cursor is inside a TextView will not trigger the event (unless it is a navigation to another object) because TextView handles its own key presses.

If the focused view didn't want this event, this method is called.

The default implementation takes care of [KEYCODE\_BACK](http://developer.android.com/reference/android/view/KeyEvent.html#KEYCODE_BACK) by calling [onBackPressed()](http://developer.android.com/reference/android/app/Activity.html#onBackPressed%28%29), though the behavior varies based on the application compatibility mode: for [ECLAIR](http://developer.android.com/reference/android/os/Build.VERSION_CODES.html#ECLAIR) or later applications, it will set up the dispatch to call [onKeyUp(int, KeyEvent)](http://developer.android.com/reference/android/app/Activity.html#onKeyUp%28int,%20android.view.KeyEvent%29) where the action will be performed; for earlier applications, it will perform the action immediately in on-down, as those versions of the platform behaved.

Other additional default key handling may be performed if configured with [setDefaultKeyMode(int)](http://developer.android.com/reference/android/app/Activity.html#setDefaultKeyMode%28int%29).

**Parameters**

|  |  |
| --- | --- |
| **keyCode** | The value in event.getKeyCode(). |
| **event** | Description of the key event. |

**Returns**

* Return true to prevent this event from being propagated further, or false to indicate that you have not handled this event and it should continue to be propagated.

**See Also**

* [onKeyUp(int, KeyEvent)](http://developer.android.com/reference/android/app/Activity.html#onKeyUp%28int,%20android.view.KeyEvent%29)
* [KeyEvent](http://developer.android.com/reference/android/view/KeyEvent.html)

**public class MediaPlayer**

extends Object

**Class Overview**

MediaPlayer class can be used to control playback of audio/video files and streams. An example on how to use the methods in this class can be found in [VideoView](http://developer.android.com/reference/android/widget/VideoView.html). Please see [Audio and Video](http://developer.android.com/guide/topics/media/index.html) for additional help using MediaPlayer.

**public static** [**MediaPlayer**](http://developer.android.com/reference/android/media/MediaPlayer.html) **create (**[**Context**](http://developer.android.com/reference/android/content/Context.html) **context, int resid)**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Convenience method to create a MediaPlayer for a given resource id. On success, [prepare()](http://developer.android.com/reference/android/media/MediaPlayer.html#prepare%28%29) will already have been called and must not be called again.

When done with the MediaPlayer, you should call [release()](http://developer.android.com/reference/android/media/MediaPlayer.html#release%28%29), to free the resources. If not released, too many MediaPlayer instances will result in an exception.

**Parameters**

|  |  |
| --- | --- |
| **context** | the Context to use |
| **resid** | the raw resource id (*R.raw.<something>*) for the resource to use as the datasource |

**Returns**

* a MediaPlayer object, or null if creation failed

**public void release ()**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Releases resources associated with this MediaPlayer object. It is considered good practice to call this method when you're done using the MediaPlayer. For instance, whenever the Activity of an application is paused, this method should be invoked to release the MediaPlayer object. In addition to unnecessary resources (such as memory and instances of codecs) being hold, failure to call this method immediately if a MediaPlayer object is no longer needed may also lead to continuous battery consumption for mobile devices, and playback failure if no multiple instances of the same codec is supported on a device.

**public void start ()**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Starts or resumes playback. If playback had previously been paused, playback will continue from where it was paused. If playback had been stopped, or never started before, playback will start at the beginning.

**Throws**

|  |  |
| --- | --- |
| [**IllegalStateException**](http://developer.android.com/reference/java/lang/IllegalStateException.html) | if it is called in an invalid state |

**public void stop ()**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Stops playback after playback has been stopped or paused.

**Throws**

|  |  |
| --- | --- |
| [**IllegalStateException**](http://developer.android.com/reference/java/lang/IllegalStateException.html) | if the internal player engine has not been initialized. |
|  |  |
|  |  |

**public class**

**VideoView**

**extends SurfaceView**

**implements MediaController.MediaPlayerControl**

**java.lang.Object**

**↳ android.view.View**

**↳ android.view.SurfaceView**

**↳ android.widget.VideoView**

**Class Overview**

**Displays a video file. The VideoView class can load images from various sources (such as resources or content providers), takes care of computing its measurement from the video so that it can be used in any layout manager, and provides various display options such as scaling and tinting.**

#### public void setVideoPath ([String](http://developer.android.com/reference/java/lang/String.html) path)

**public void start ()**

**public class PreferenceManager**

**public static** [**SharedPreferences**](http://developer.android.com/reference/android/content/SharedPreferences.html) **getDefaultSharedPreferences (**[**Context**](http://developer.android.com/reference/android/content/Context.html) **context)**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Gets a SharedPreferences instance that points to the default file that is used by the preference framework in the given context.

**Parameters**

|  |  |
| --- | --- |
| **context** | The context of the preferences whose values are wanted. |

**Returns**

* A SharedPreferences instance that can be used to retrieve and listen to values of the preferences.

**public interface SharedPreferencesInterface**

**public abstract boolean getBoolean (**[**String**](http://developer.android.com/reference/java/lang/String.html) **key, boolean defValue)**

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Retrieve a boolean value from the preferences.

**Parameters**

|  |  |
| --- | --- |
| **key** | The name of the preference to retrieve. |
| **defValue** | Value to return if this preference does not exist. |

**Returns**

* Returns the preference value if it exists, or defValue. Throws ClassCastException if there is a preference with this name that is not a boolean.

**Throws**

|  |  |
| --- | --- |
| [**ClassCastException**](http://developer.android.com/reference/java/lang/ClassCastException.html) |  |

#### public [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) getPreferences (int mode)

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Retrieve a [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) object for accessing preferences that are private to this activity. This simply calls the underlying [getSharedPreferences(String, int)](http://developer.android.com/reference/android/content/ContextWrapper.html#getSharedPreferences%28java.lang.String,%20int%29) method by passing in this activity's class name as the preferences name.

##### Parameters

|  |  |
| --- | --- |
| **mode** | Operating mode. Use [MODE\_PRIVATE](http://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE) for the default operation, [MODE\_WORLD\_READABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_READABLE) and [MODE\_WORLD\_WRITEABLE](http://developer.android.com/reference/android/content/Context.html#MODE_WORLD_WRITEABLE) to control permissions. |

##### Returns

* Returns the single SharedPreferences instance that can be used to retrieve and modify the preference values.

public abstract [SharedPreferences.Editor](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html) edit ()

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Create a new Editor for these preferences, through which you can make modifications to the data in the preferences and atomically commit those changes back to the SharedPreferences object.

Note that you must call [commit()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#commit%28%29) to have any changes you perform in the Editor actually show up in the SharedPreferences.

##### Returns

* Returns a new instance of the [SharedPreferences.Editor](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html) interface, allowing you to modify the values in this SharedPreferences object.

#### public abstract [SharedPreferences.Editor](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html) putString ([String](http://developer.android.com/reference/java/lang/String.html) key, [String](http://developer.android.com/reference/java/lang/String.html) value)

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Set a String value in the preferences editor, to be written back once [commit()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#commit%28%29) or [apply()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#apply%28%29) are called.

##### Parameters

|  |  |
| --- | --- |
| **key** | The name of the preference to modify. |
| **value** | The new value for the preference. |

##### Returns

* Returns a reference to the same Editor object, so you can chain put calls together.

#### public abstract boolean commit ()

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Commit your preferences changes back from this Editor to the [SharedPreferences](http://developer.android.com/reference/android/content/SharedPreferences.html) object it is editing. This atomically performs the requested modifications, replacing whatever is currently in the SharedPreferences.

Note that when two editors are modifying preferences at the same time, the last one to call commit wins.

If you don't care about the return value and you're using this from your application's main thread, consider using [apply()](http://developer.android.com/reference/android/content/SharedPreferences.Editor.html#apply%28%29) instead.

##### Returns

* Returns true if the new values were successfully written to persistent storage.

#### public abstract [String](http://developer.android.com/reference/java/lang/String.html) getString ([String](http://developer.android.com/reference/java/lang/String.html) key, [String](http://developer.android.com/reference/java/lang/String.html) defValue)

Since: [API Level 1](http://developer.android.com/guide/appendix/api-levels.html#level1)

Retrieve a String value from the preferences.

##### Parameters

|  |  |
| --- | --- |
| **key** | The name of the preference to retrieve. |
| **defValue** | Value to return if this preference does not exist. |

##### Returns

* Returns the preference value if it exists, or defValue. Throws ClassCastException if there is a preference with this name that is not a String.

##### Throws

|  |  |
| --- | --- |
| [**ClassCastException**](http://developer.android.com/reference/java/lang/ClassCastException.html) |  |