

public class **File** Summary: [Fields](#) | [Ctors](#) | [Methods](#) | [Inherited Methods](#)
extends [Object](#) | [\[Expand All\]](#)
implements **Added in [API level 1](#)**
[Serializable](#) [Comparable<T>](#)

[java.lang.Object](#)
↳ [java.io.File](#)

Class Overview

An "abstract" representation of a file system entity identified by a pathname. The pathname may be absolute (relative to the root directory of the file system) or relative to the current directory in which the program is running.

The actual file referenced by a `File` may or may not exist. It may also, despite the name `File`, be a directory or other non-regular file.

This class provides limited functionality for getting/setting file permissions, file type, and last modified time.

On Android strings are converted to UTF-8 byte sequences when sending filenames to the operating system, and byte sequences returned by the operating system (from the various `list` methods) are converted to strings by decoding them as UTF-8 byte sequences.

See Also

[Serializable](#)
[Comparable](#)

Summary

Fields

| | |
|--|--|
| public static final String pathSeparator | The system-dependent string used to separate components in search paths (""). |
| public static final char pathSeparatorChar | The system-dependent character used to separate components in search paths (''). |

| | |
|---|--|
| <code>public static final String separator</code> | The system-dependent string used to separate components in filenames ('/'). |
| <code>public static final char separatorChar</code> | The system-dependent character used to separate components in filenames ('/'). |

Public Constructors

`File(File dir, String name)`

Constructs a new file using the specified directory and name.

`File(String path)`

Constructs a new file using the specified path.

`File(String dirPath, String name)`

Constructs a new File using the specified directory path and file name, placing a path separator between the two.

`File(Uri uri)`

Constructs a new File using the path of the specified URI.

Public Methods

`canExecute()`

`boolean` Tests whether or not this process is allowed to execute this file.

`canRead()`

`boolean` Indicates whether the current context is allowed to read from this file.

`canWrite()`

`boolean` Indicates whether the current context is allowed to write to this file.

`compareTo(File another)`

`int` Returns the relative sort ordering of the paths for this file and the file `another`.

`createNewFile()`

`boolean` Creates a new, empty file on the file system according to the path information stored in this file.

`createTempFile(String prefix, String suffix, File directory)`

`static File` Creates an empty temporary file in the given directory using the given prefix and suffix as part of the file name.

`createTempFile(String prefix, String suffix)`

`static File` Creates an empty temporary file using the given prefix and suffix as part of the file name.

`delete()`
boolean Deletes this file.

`deleteOnExit()`
void Schedules this file to be automatically deleted when the VM terminates normally.

`equals(Object obj)`
boolean Compares `obj` to this file and returns `true` if they represent the *same* object using a path specific comparison.

`exists()`
boolean Returns a boolean indicating whether this file can be found on the underlying file system.

`getAbsolutePath()`
File Returns a new file constructed using the absolute path of this file.

`getAbsolutePath()`
String Returns the absolute path of this file.

`getCanonicalFile()`
File Returns a new file created using the canonical path of this file.

`getCanonicalPath()`
String Returns the canonical path of this file.

`getFreeSpace()`
long Returns the number of free bytes on the partition containing this path.

`getName()`
String Returns the name of the file or directory represented by this file.

`getParent()`
String Returns the pathname of the parent of this file.

`getParentFile()`
File Returns a new file made from the pathname of the parent of this file.

`getPath()`
String Returns the path of this file.

`getTotalSpace()`
long Returns the total size in bytes of the partition containing this path.

`getUsableSpace()`
long Returns the number of usable free bytes on the partition containing this path.

`hashCode()`
`int` Returns an integer hash code for the receiver.

`isAbsolute()`
`boolean` Indicates if this file's pathname is absolute.

`isDirectory()`
`boolean` Indicates if this file represents a *directory* on the underlying file system.

`isFile()`
`boolean` Indicates if this file represents a *file* on the underlying file system.

`isHidden()`
`boolean` Returns whether or not this file is a hidden file as defined by the operating system.

`lastModified()`
`long` Returns the time when this file was last modified, measured in milliseconds since January 1st, 1970, midnight.

`length()`
`long` Returns the length of this file in bytes.

`list()`
`String[]` Returns an array of strings with the file names in the directory represented by this file.

`list(FileNameFilter filter)`
`String[]` Gets a list of the files in the directory represented by this file.

`listFiles()`
`File[]` Returns an array of files contained in the directory represented by this file.

`listFiles(FileNameFilter filter)`
`File[]` Gets a list of the files in the directory represented by this file.

`listFiles(FileFilter filter)`
`File[]` Gets a list of the files in the directory represented by this file.

`listRoots()`
`static File[]` Returns the file system roots.

`mkdir()`
`boolean` Creates the directory named by this file, assuming its parents exist.

`makedirs()`
`boolean` Creates the directory named by this file, creating missing parent directories if necessary.

`boolean renameTo (File newPath)`
 Renames this file to newPath.

`boolean setExecutable (boolean executable)`
 Equivalent to `setExecutable(executable, true)`.

`boolean setExecutable (boolean executable, boolean ownerOnly)`
 Manipulates the execute permissions for the abstract path designated by this file.

`boolean setLastModified (long time)`
 Sets the time this file was last modified, measured in milliseconds since January 1st, 1970, midnight.

`boolean setReadOnly ()`
 Equivalent to `setWritable(false, false)`.

`boolean setReadable (boolean readable)`
 Equivalent to `setReadable(readable, true)`.

`boolean setReadable (boolean readable, boolean ownerOnly)`
 Manipulates the read permissions for the abstract path designated by this file.

`boolean setWritable (boolean writable, boolean ownerOnly)`
 Manipulates the write permissions for the abstract path designated by this file.

`boolean setWritable (boolean writable)`
 Equivalent to `setWritable(writable, true)`.

`String toString ()`
 Returns a string containing a concise, human-readable description of this file.

`URI toURI ()`
 Returns a Uniform Resource Identifier for this file.

`URL toURL ()`
This method was deprecated in API level 9. Use `toURI ()` and `toURL ()` to correctly escape illegal characters.

Inherited Methods [\[Expand\]](#)

- From class `java.lang.Object`
- From interface `java.lang.Comparable`

Fields

`public static final String pathSeparator`

Added in [API level 1](#)

The system-dependent string used to separate components in search paths (":"). See [pathSeparatorChar](#) (</reference/java/io>)

[/File.html#pathSeparatorChar](#)).

public static final char **pathSeparatorChar**

Added in [API level 1](#)

The system-dependent character used to separate components in search paths ('`.`'). This is used to split such things as the PATH environment variable and classpath system properties into lists of directories to be searched.

This field is initialized from the system property "`path.separator`". Later changes to that property will have no effect on this field or this class.

public static final String **separator**

Added in [API level 1](#)

The system-dependent string used to separate components in filenames ('`/`'). See [separatorChar](#) ([/reference/java/io/File.html#separatorChar](#)).

public static final char **separatorChar**

Added in [API level 1](#)

The system-dependent character used to separate components in filenames ('`/`'). Use of this (rather than hard-coding '`/`') helps portability to other operating systems.

This field is initialized from the system property "`file.separator`". Later changes to that property will have no effect on this field or this class.

Public Constructors

public **File** (File dir, String name)

Added in [API level 1](#)

Constructs a new file using the specified directory and name.

Parameters

- dir* the directory where the file is stored.
name the file's name.

Throws

[NullPointerException](#) if name is null.

public **File** (String path)

Added in [API level 1](#)

Constructs a new file using the specified path.

Parameters

path the path to be used for the file.

public File (String dirPath, String name)

Added in API level 1

Constructs a new File using the specified directory path and file name, placing a path separator between the two.

Parameters

dirPath the path to the directory where the file is stored.

name the file's name.

Throws

NullPointerException if name == null.

public File (URI uri)

Added in API level 1

Constructs a new File using the path of the specified URI. *uri* needs to be an absolute and hierarchical Unified Resource Identifier with file scheme and non-empty path component, but with undefined authority, query or fragment components.

Parameters

uri the Unified Resource Identifier that is used to construct this file.

Throws

IllegalArgumentException if *uri* does not comply with the conditions above.

See Also

toURI()

URI

Public Methods

public boolean canExecute ()

Added in API level 9

Tests whether or not this process is allowed to execute this file. Note that this is a best-effort result; the only way to be certain is to actually attempt the operation.

Returns

true if this file can be executed, false otherwise.

public boolean canRead ()Added in [API level 1](#)

Indicates whether the current context is allowed to read from this file.

Returns

true if this file can be read, false otherwise.

public boolean canWrite ()Added in [API level 1](#)

Indicates whether the current context is allowed to write to this file.

Returns

true if this file can be written, false otherwise.

public int compareTo (File another)Added in [API level 1](#)

Returns the relative sort ordering of the paths for this file and the file *another*. The ordering is platform dependent.

Parameters

another a file to compare this file to

Returns

an int determined by comparing the two paths. Possible values are described in the Comparable interface.

See Also

[Comparable](#)

public boolean createNewFile ()Added in [API level 1](#)

Creates a new, empty file on the file system according to the path information stored in this file. This method returns true if it creates a file, false if the file already existed. Note that it returns false even if the file is not a file (because it's a directory, say).

This method is not generally useful. For creating temporary files, use [createTempFile\(String, String\)](#) ([/reference/java/io/File.html#createTempFile\(java.lang.String, java.lang.String\)](#)) instead. For reading/writing files, use [FileInputStream](#) ([/reference/java/io/FileInputStream.html](#)), [FileOutputStream](#) ([/reference/java/io/FileOutputStream.html](#)), or [RandomAccessFile](#) ([/reference/java/io/RandomAccessFile.html](#)), all of which can create files.

Note that this method does *not* throw `IOException` if the file

already exists, even if it's not a regular file. Callers should always check the return value, and may additionally want to call [`isFile\(\)`](/reference/java/io/File.html#isFile()) ([`isFile\(\)`](/reference/java/io/File.html#isFile())).

Returns

true if the file has been created, false if it already exists.

Throws

[`IOException`](#) if it's not possible to create the file.

public static [`File`](#) **createTempFile** ([`String`](#) prefix, [`String`](#) suffix, [`File`](#) directory)

Added in [`API level 1`](#)

Creates an empty temporary file in the given directory using the given prefix and suffix as part of the file name. If `suffix` is null, `.tmp` is used.

Note that this method does *not* call [`deleteOnExit\(\)`](#) ([`deleteOnExit\(\)`](/reference/java/io/File.html#deleteOnExit())), but see the documentation for that method before you call it manually.

Parameters

| | |
|------------------|---|
| <i>prefix</i> | the prefix to the temp file name. |
| <i>suffix</i> | the suffix to the temp file name. |
| <i>directory</i> | the location to which the temp file is to be written, or null for the default location for temporary files, which is taken from the "java.io.tmpdir" system property. It may be necessary to set this property to an existing, writable directory for this method to work properly. |

Returns

the temporary file.

Throws

| | |
|---|--|
| <code>IllegalArgumentException</code> | if the length of <code>prefix</code> is less than 3. |
| <code>IOException</code> | if an error occurs when writing the file. |

public static [`File`](#) **createTempFile** ([`String`](#) prefix, [`String`](#) suffix)

Added in [`API level 1`](#)

Creates an empty temporary file using the given prefix and suffix as part of the file name. If `suffix` is null, `.tmp` is used. This method is a convenience method that calls [`createTempFile\(String,`](#)

String, File) ([/reference/java/io/File.html#createTempFile\(java.lang.String, java.lang.String, java.io.File\)](/reference/java/io/File.html#createTempFile(java.lang.String, java.lang.String, java.io.File))) with the third argument being null.

Parameters

prefix the prefix to the temp file name.

suffix the suffix to the temp file name.

Returns

the temporary file.

Throws

IOException if an error occurs when writing the file.

public boolean **delete** ()

Added in [API level 1](#)

Deletes this file. Directories must be empty before they will be deleted.

Note that this method does *not* throw *IOException* on failure. Callers must check the return value.

Returns

true if this file was deleted, false otherwise.

public void **deleteOnExit** ()

Added in [API level 1](#)

Schedules this file to be automatically deleted when the VM terminates normally.

Note that on Android, the application lifecycle does not include VM termination, so calling this method will not ensure that files are deleted. Instead, you should use the most appropriate out of:

- Use a finally clause to manually invoke *delete()*.
- Maintain your own set of files to delete, and process it at an appropriate point in your application's lifecycle.
- Use the Unix trick of deleting the file as soon as all readers and writers have opened it. No new readers/writers will be able to access the file, but all existing ones will still have access until the last one closes the file.

public boolean **equals** (Object obj)

Added in [API level 1](#)

Compares obj to this file and returns true if they represent the same object using a path specific comparison.

Parameters

obj the object to compare this file with.

Returns

true if *obj* is the same as this object, false otherwise.

public boolean **exists** ()

Added in [API level 1](#)

Returns a boolean indicating whether this file can be found on the underlying file system.

Returns

true if this file exists, false otherwise.

public [File](#) **getAbsolutePath** ()

Added in [API level 1](#)

Returns a new file constructed using the absolute path of this file. Equivalent to `new File(this.getAbsolutePath())`.

public [String](#) **getAbsolutePath** ()

Added in [API level 1](#)

Returns the absolute path of this file. An absolute path is a path that starts at a root of the file system. On Android, there is only one root: `/`.

A common use for absolute paths is when passing paths to a `Process` as command-line arguments, to remove the requirement implied by relative paths, that the child must have the same working directory as its parent.

public [File](#) **getCanonicalFile** ()

Added in [API level 1](#)

Returns a new file created using the canonical path of this file. Equivalent to `new File(this.getCanonicalPath())`.

Returns

the new file constructed from this file's canonical path.

Throws

[*IOException*](#) if an I/O error occurs.

public [String](#) **getCanonicalPath** ()

Added in [API level 1](#)

Returns the canonical path of this file. An *absolute* path is one that begins at the root of the file system. A *canonical* path is an absolute path with symbolic links and references to `"."` or `".."` resolved. If a path element does not exist (or is not searchable), there is a conflict

between interpreting canonicalization as a textual operation (where "a/../b" is "b" even if "a" does not exist) .

Most callers should use `getAbsolutePath()` ([/reference/java/io/File.html#getAbsolutePath\(\)](/reference/java/io/File.html#getAbsolutePath())) instead. A canonical path is significantly more expensive to compute, and not generally useful. The primary use for canonical paths is determining whether two paths point to the same file by comparing the canonicalized paths.

It can be actively harmful to use a canonical path, specifically because canonicalization removes symbolic links. It's wise to assume that a symbolic link is present for a reason, and that that reason is because the link may need to change. Canonicalization removes this layer of indirection. Good code should generally avoid caching canonical paths.

Returns

the canonical path of this file.

Throws

IOException if an I/O error occurs.

public long `getFreeSpace` ()

Added in [API level 9](#)

Returns the number of free bytes on the partition containing this path. Returns 0 if this path does not exist.

Note that this is likely to be an optimistic over-estimate and should not be taken as a guarantee your application can actually write this many bytes.

public String `getName` ()

Added in [API level 1](#)

Returns the name of the file or directory represented by this file.

Returns

this file's name or an empty string if there is no name part in the file's path.

public String `getParent` ()

Added in [API level 1](#)

Returns the pathname of the parent of this file. This is the path up to but not including the last name. `null` is returned if there is no parent.

Returns

this file's parent pathname or `null`.

public File getParentFile ()Added in [API level 1](#)

Returns a new file made from the pathname of the parent of this file. This is the path up to but not including the last name. `null` is returned when there is no parent.

Returns

a new file representing this file's parent or `null`.

public String getPath ()Added in [API level 1](#)

Returns the path of this file.

public long getTotalSpace ()Added in [API level 9](#)

Returns the total size in bytes of the partition containing this path. Returns 0 if this path does not exist.

public long getUsableSpace ()Added in [API level 9](#)

Returns the number of usable free bytes on the partition containing this path. Returns 0 if this path does not exist.

Note that this is likely to be an optimistic over-estimate and should not be taken as a guarantee your application can actually write this many bytes. On Android (and other Unix-based systems), this method returns the number of free bytes available to non-root users, regardless of whether you're actually running as root, and regardless of any quota or other restrictions that might apply to the user. (The `getFreeSpace` method returns the number of bytes potentially available to root.)

public int hashCode ()Added in [API level 1](#)

Returns an integer hash code for the receiver. Any two objects for which `equals` returns `true` must return the same hash code.

Returns

this file's hash value.

See Also

[`equals\(Object\)`](#)

public boolean isAbsolute ()Added in [API level 1](#)

Indicates if this file's pathname is absolute. Whether a pathname is absolute is platform specific. On Android, absolute paths start with the character `'/'`.

Returns

true if this file's pathname is absolute, false otherwise.

See Also

[getPath\(\)](#)

public boolean **isDirectory** ()

Added in [API level 1](#)

Indicates if this file represents a *directory* on the underlying file system.

Returns

true if this file is a directory, false otherwise.

public boolean **isFile** ()

Added in [API level 1](#)

Indicates if this file represents a *file* on the underlying file system.

Returns

true if this file is a file, false otherwise.

public boolean **isHidden** ()

Added in [API level 1](#)

Returns whether or not this file is a hidden file as defined by the operating system. The notion of "hidden" is system-dependent. For Unix systems a file is considered hidden if its name starts with a ".". For Windows systems there is an explicit flag in the file system for this purpose.

Returns

true if the file is hidden, false otherwise.

public long **lastModified** ()

Added in [API level 1](#)

Returns the time when this file was last modified, measured in milliseconds since January 1st, 1970, midnight. Returns 0 if the file does not exist.

Returns

the time when this file was last modified.

public long **length** ()

Added in [API level 1](#)

Returns the length of this file in bytes. Returns 0 if the file does not exist. The result for a directory is not defined.

Returns

the number of bytes in this file.

public String[] list ()

Added in API level 1

Returns an array of strings with the file names in the directory represented by this file. The result is `null` if this file is not a directory.

The entries `.` and `..` representing the current and parent directory are not returned as part of the list.

Returns

an array of strings with file names or `null`.

public String[] list (FilenameFilter filter)

Added in API level 1

Gets a list of the files in the directory represented by this file. This list is then filtered through a `FilenameFilter` and the names of files with matching names are returned as an array of strings. Returns `null` if this file is not a directory. If `filter` is `null` then all filenames match.

The entries `.` and `..` representing the current and parent directories are not returned as part of the list.

Parameters

filter the filter to match names against, may be `null`.

Returns

an array of files or `null`.

public File[] listFiles ()

Added in API level 1

Returns an array of files contained in the directory represented by this file. The result is `null` if this file is not a directory. The paths of the files in the array are absolute if the path of this file is absolute, they are relative otherwise.

Returns

an array of files or `null`.

public File[] listFiles (FilenameFilter filter)

Added in API level 1

Gets a list of the files in the directory represented by this file. This list is then filtered through a `FilenameFilter` and files with matching names are returned as an array of files. Returns `null` if this file is not a directory. If `filter` is `null` then all filenames match.

The entries `.` and `..` representing the current and parent directories are not returned as part of the list.

Parameters

filter the filter to match names against, may be `null`.

Returns

an array of files or `null`.

public [File\[\]](#) `listFiles` ([FileFilter](#) filter)

Added in [API level 1](#)

Gets a list of the files in the directory represented by this file. This list is then filtered through a `FileFilter` and matching files are returned as an array of files. Returns `null` if this file is not a directory. If *filter* is `null` then all files match.

The entries `.` and `..` representing the current and parent directories are not returned as part of the list.

Parameters

filter the filter to match names against, may be `null`.

Returns

an array of files or `null`.

public static [File\[\]](#) `listRoots` ()

Added in [API level 1](#)

Returns the file system roots. On Android and other Unix systems, there is a single root, `/`.

public boolean `mkdir` ()

Added in [API level 1](#)

Creates the directory named by this file, assuming its parents exist. Use `makedirs()` ([/reference/java/io/File.html#makedirs\(\)](/reference/java/io/File.html#makedirs())) if you also want to create missing parents.

Note that this method does *not* throw `IOException` on failure. Callers must check the return value. Note also that this method returns `false` if the directory already existed. If you want to know whether the directory exists on return, either use `(f.mkdir() || f.isDirectory())` or simply ignore the return value from this method and simply call `isDirectory()` ([/reference/java/io/File.html#isDirectory\(\)](/reference/java/io/File.html#isDirectory())).

Returns

`true` if the directory was created, `false` on failure or if the directory already existed.

public boolean mkdirs ()Added in [API level 1](#)

Creates the directory named by this file, creating missing parent directories if necessary. Use `mkdir()` ([/reference/java/io/File.html#mkdir\(\)](#)) if you don't want to create missing parents.

Note that this method does *not* throw `IOException` on failure. Callers must check the return value. Note also that this method returns `false` if the directory already existed. If you want to know whether the directory exists on return, either use `(f.mkdirs() || f.isDirectory())` or simply ignore the return value from this method and simply call `isDirectory()` ([/reference/java/io/File.html#isDirectory\(\)](#)).

Returns

`true` if the directory was created, `false` on failure or if the directory already existed.

public boolean renameTo (File newPath)Added in [API level 1](#)

Renames this file to `newPath`. This operation is supported for both files and directories.

Many failures are possible. Some of the more likely failures include:

- Write permission is required on the directories containing both the source and destination paths.
- Search permission is required for all parents of both paths.
- Both paths be on the same mount point. On Android, applications are most likely to hit this restriction when attempting to copy between internal storage and an SD card.

Note that this method does *not* throw `IOException` on failure. Callers must check the return value.

Parameters

newPath the new path.

Returns

`true` on success.

public boolean setExecutable (boolean executable) Added in [API level 9](#)

Equivalent to `setExecutable(executable, true)`.

See Also

[setExecutable\(boolean, boolean\)](#)

public boolean **setExecutable** (boolean executable,
boolean ownerOnly) Added in [API level 9](#)

Manipulates the execute permissions for the abstract path designated by this file.

Note that this method does *not* throw `IOException` on failure. Callers must check the return value.

Parameters

- | | |
|-------------------|--|
| <i>executable</i> | To allow execute permission if true, otherwise disallow |
| <i>ownerOnly</i> | To manipulate execute permission only for owner if true, otherwise for everyone. The manipulation will apply to everyone regardless of this value if the underlying system does not distinguish owner and other users. |

Returns

true if and only if the operation succeeded. If the user does not have permission to change the access permissions of this abstract pathname the operation will fail. If the underlying file system does not support execute permission and the value of executable is false, this operation will fail.

public boolean **setLastModified** (long time) Added in [API level 1](#)

Sets the time this file was last modified, measured in milliseconds since January 1st, 1970, midnight.

Note that this method does *not* throw `IOException` on failure. Callers must check the return value.

Parameters

- | | |
|-------------|---|
| <i>time</i> | the last modification time for this file. |
|-------------|---|

Returns

true if the operation is successful, false otherwise.

Throws

- [`IllegalArgumentException`](#) if time < 0.

public boolean **setReadOnly** () Added in [API level 1](#)

Equivalent to `setWritable(false, false)`.

See Also

[`setWritable\(boolean, boolean\)`](#)

public boolean **setReadable** (boolean readable) Added in [API level 9](#)

Equivalent to `setReadable(readable, true)`.

See Also

[setReadable\(boolean, boolean\)](#)

public boolean **setReadable** (boolean readable, boolean ownerOnly) Added in [API level 9](#)

Manipulates the read permissions for the abstract path designated by this file.

Parameters

- | | |
|------------------|---|
| <i>readable</i> | To allow read permission if true, otherwise disallow |
| <i>ownerOnly</i> | To manipulate read permission only for owner if true, otherwise for everyone. The manipulation will apply to everyone regardless of this value if the underlying system does not distinguish owner and other users. |

Returns

true if and only if the operation succeeded. If the user does not have permission to change the access permissions of this abstract pathname the operation will fail. If the underlying file system does not support read permission and the value of readable is false, this operation will fail.

public boolean **setWritable** (boolean writable, boolean ownerOnly) Added in [API level 9](#)

Manipulates the write permissions for the abstract path designated by this file.

Parameters

- | | |
|------------------|--|
| <i>writable</i> | To allow write permission if true, otherwise disallow |
| <i>ownerOnly</i> | To manipulate write permission only for owner if true, otherwise for everyone. The manipulation will apply to everyone regardless of this value if the underlying system does not distinguish owner and other users. |

Returns

true if and only if the operation succeeded. If the user does not have permission to change the access permissions of this

abstract pathname the operation will fail.

public boolean **setWritable** (boolean writable)

Added in [API level 9](#)

Equivalent to `setWritable(writable, true)`.

See Also

[setWritable\(boolean, boolean\)](#)

public [String](#) **toString** ()

Added in [API level 1](#)

Returns a string containing a concise, human-readable description of this file.

Returns

a printable representation of this file.

public [URI](#) **toURI** ()

Added in [API level 1](#)

Returns a Uniform Resource Identifier for this file. The URI is system dependent and may not be transferable between different operating / file systems.

Returns

an URI for this file.

public [URL](#) **toURL** ()

Added in [API level 1](#)

This method was deprecated in API level 9.

Use [toURI\(\)](#) ([/reference/java/io/File.html#toURI\(\)](#)) and [toURL\(\)](#) ([/reference/java/net/URI.html#toURL\(\)](#)) to correctly escape illegal characters.

Returns a Uniform Resource Locator for this file. The URL is system dependent and may not be transferable between different operating / file systems.

Returns

a URL for this file.

Throws

[MalformedURLException](#) if the path cannot be transformed into a URL.