

public final class

Bitmap

extends [Object](#)

implements [Parcelable](#)

Summary: [Nested Classes](#) | [Constants](#) | [Inherited Constants](#) | [Fields](#) | [Methods](#) |

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

Added in [API level 1](#)

[java.lang.Object](#)

↳ [android.graphics.Bitmap](#)

Summary

Nested Classes

enum [Bitmap.CompressFormat](#) Specifies the known formats a bitmap can be compressed into

enum [Bitmap.Config](#) Possible bitmap configurations.

Constants

int [DENSITY_NONE](#) Indicates that the bitmap was created for an unknown pixel density.

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

► From interface [android.os.Parcelable](#)

Fields

public static final [Creator<Bitmap>](#) CREATOR

Public Methods

boolean [compress](#)([Bitmap.CompressFormat](#) format, int quality, [OutputStream](#) stream)
Write a compressed version of the bitmap to the specified outputstream.

boolean [copy](#)([Bitmap.Config](#) config, boolean isMutable)

Bitmap [copy](#)(int width, int height, [Bitmap.Config](#) config)
Tries to make a new bitmap based on the dimensions of this bitmap, setting the new bitmap's config to the one specified, and then copying this bitmap's pixels into the new bitmap.

void [copyPixelsFromBuffer](#)([Buffer](#) src)
Copy the pixels from the buffer, beginning at the current position, overwriting the bitmap's pixels.

void [copyPixelsToBuffer](#)([Buffer](#) dst)
Copy the bitmap's pixels into the specified buffer (allocated by the caller).

static Bitmap [createBitmap](#)([DisplayMetrics](#) display, int[] colors, int width, int height, [Bitmap.Config](#) config)
Returns an immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array.

static Bitmap [createBitmap](#)([DisplayMetrics](#) display, int[] colors, int offset, int stride, int width, int height, [Bitmap.Config](#) config)
Returns an immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array.

static Bitmap [createBitmap](#)(Bitmap source, int x, int y, int width, int height)
Returns an immutable bitmap from the specified subset of the source bitmap.

static Bitmap [createBitmap](#)(Bitmap src)
Returns an immutable bitmap from the source bitmap.

static Bitmap [createBitmap](#)([DisplayMetrics](#) display, int width, int height, [Bitmap.Config](#) config)
Returns a mutable bitmap with the specified width and height.

static Bitmap [createBitmap](#)(Bitmap source, int x, int y, int width, int height, [Matrix](#) m, boolean filter)
Returns an immutable bitmap from subset of the source bitmap, transformed by the optional matrix.

static Bitmap [createBitmap](#)(int width, int height, [Bitmap.Config](#) config)
Returns a mutable bitmap with the specified width and height.

static Bitmap [createBitmap](#)(int[] colors, int offset, int stride, int width, int height, [Bitmap.Config](#) config)
Returns an immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array.

static Bitmap [createBitmap](#)(int[] colors, int width, int height, [Bitmap.Config](#) config)
Returns an immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array.

static Bitmap [createScaledBitmap](#)(Bitmap src, int dstWidth, int dstHeight, boolean filter)
Creates a new bitmap, scaled from an existing bitmap, when possible.

int [describeContents](#)()
No special parcel contents.

```

    void eraseColor (int c)
        Fills the bitmap's pixels with the specified CoLoR.

    Bitmap extractAlpha ()
        Returns a new bitmap that captures the alpha values of the original.

    Bitmap extractAlpha (Paint paint, int[] offsetXY)
        Returns a new bitmap that captures the alpha values of the original.

    final int getAllocationByteCount ()
        Returns the size of the allocated memory used to store this bitmap's pixels.

    final int getByteCount ()
        Returns the minimum number of bytes that can be used to store this bitmap's pixels.

    final Bitmap.Config getConfig ()
        If the bitmap's internal config is in one of the public formats, return that config, otherwise return null.

    int getDensity ()
        Returns the density for this bitmap.

    int getGenerationId ()
        Returns the generation ID of this bitmap.

    final int getHeight ()
        Returns the bitmap's height

    byte[] getNinePatchChunk ()
        Returns an optional array of private data, used by the UI system for some bitmaps.

    int getPixel (int x, int y)
        Returns the CoLoR at the specified location.

    void getPixels (int[] pixels, int offset, int stride, int x, int y, int width, int height)
        Returns in pixels[] a copy of the data in the bitmap.

    final int getRowBytes ()
        Return the number of bytes between rows in the bitmap's pixels.

    int getScaledHeight (DisplayMetrics metrics)
        Convenience for calling getScaledHeight (int) with the target density of the given DisplayMetrics.

    int getScaledHeight (int targetDensity)
        Convenience method that returns the height of this bitmap divided by the density scale factor.

    int getScaledHeight (Canvas canvas)
        Convenience for calling getScaledHeight (int) with the target density of the given Canvas.

    int getScaledWidth (int targetDensity)
        Convenience method that returns the width of this bitmap divided by the density scale factor.

    int getScaledWidth (DisplayMetrics metrics)
        Convenience for calling getScaledWidth (int) with the target density of the given DisplayMetrics.

    int getScaledWidth (Canvas canvas)
        Convenience for calling getScaledWidth (int) with the target density of the given Canvas.

    final int getWidth ()
        Returns the bitmap's width

    boolean hasAlpha ()
        Returns true if the bitmap's config supports per-pixel alpha, and if the pixels may contain non-opaque alpha values.

    boolean hasMipMap ()
        Indicates whether the renderer responsible for drawing this bitmap should attempt to use mipmaps when this bitmap is drawn scaled down.

    final boolean isMutable ()
        Returns true if the bitmap is marked as mutable (i.e. can be drawn into)

    boolean isPremultiplied ()
        Indicates whether pixels stored in this bitmaps are stored pre-multiplied.

    final boolean isRecycled ()
        Returns true if this bitmap has been recycled.

    void prepareToDraw ()
        Rebuilds any caches associated with the bitmap that are used for drawing it.

```

```

    reconfigure(int width, int height, Bitmap.Config config)

void    Modifies the bitmap to have a specified width, height, and Bitmap.Config (/reference/android/graphics
        /Bitmap.Config.html), without affecting the underlying allocation backing the bitmap.

void    recycle()
        Free the native object associated with this bitmap, and clear the reference to the pixel data.
boolean sameAs(Bitmap other)
        Given another bitmap, return true if it has the same dimensions, config, and pixel data as this bitmap.
        setConfig(Bitmap.Config config)

void    Convenience method for calling reconfigure(int, int, Config) (/reference/android/graphics
        /Bitmap.html#reconfigure(int, int, android.graphics.Bitmap.Config)) with the current height and width.

        setDensity(int density)
void    Specifies the density for this bitmap.

        setHasAlpha(boolean hasAlpha)
void    Tell the bitmap if all of the pixels are known to be opaque (false) or if some of the pixels may contain
        non-opaque alpha values (true).
        setHasMipMap(boolean hasMipMap)
final void Set a hint for the renderer responsible for drawing this bitmap indicating that it should attempt to use
        mipmaps when this bitmap is drawn scaled down.
        setHeight(int height)

void    Convenience method for calling reconfigure(int, int, Config) (/reference/android/graphics
        /Bitmap.html#reconfigure(int, int, android.graphics.Bitmap.Config)) with the current width and config.

        setPixel(int x, int y, int color)

void    Write the specified Color (/reference/android/graphics/Color.html) into the bitmap (assuming it is mutable)
        at the x,y coordinate.

        setPixels(int[] pixels, int offset, int stride, int x, int y, int width, int height)
void    Replace pixels in the bitmap with the colors in the array.

        setPremultiplied(boolean premultiplied)
final void Sets whether the bitmap should treat its data as pre-multiplied.
        setWidth(int width)

void    Convenience method for calling reconfigure(int, int, Config) (/reference/android/graphics
        /Bitmap.html#reconfigure(int, int, android.graphics.Bitmap.Config)) with the current height and config.

        writeToParcel(Parcel p, int flags)
void    Write the bitmap and its pixels to the parcel.

```

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

- From class `java.lang.Object`
- From interface `android.os.Parcelable`

Constants

`public static final int DENSITY_NONE`

Added in [API level 4](#)

Indicates that the bitmap was created for an unknown pixel density.

See Also

[`getDensity\(\)`](#)
[`setDensity\(int\)`](#)

Constant Value: 0 (0x00000000)

public static final [Creator](#)<[Bitmap](#)> CREATOR

Added in [API level 1](#)

Public Methods

public boolean **compress** ([Bitmap.CompressFormat](#) format, int quality, [OutputStream](#) stream)

Added in [API level 1](#)

Write a compressed version of the bitmap to the specified outputstream. If this returns true, the bitmap can be reconstructed by passing a corresponding inputstream to [BitmapFactory.decodeStream\(\)](#). Note: not all Formats support all bitmap configs directly, so it is possible that the returned bitmap from [BitmapFactory](#) could be in a different bitdepth, and/or may have lost per-pixel alpha (e.g. JPEG only supports opaque pixels).

Parameters

format The format of the compressed image

quality Hint to the compressor, 0-100. 0 meaning compress for small size, 100 meaning compress for max quality. Some formats, like PNG which is lossless, will ignore the quality setting

stream The outputstream to write the compressed data.

Returns

true if successfully compressed to the specified stream.

public [Bitmap](#) **copy** ([Bitmap.Config](#) config, boolean isMutable)

Added in [API level 1](#)

Tries to make a new bitmap based on the dimensions of this bitmap, setting the new bitmap's config to the one specified, and then copying this bitmap's pixels into the new bitmap. If the conversion is not supported, or the allocator fails, then this returns NULL. The returned bitmap initially has the same density as the original.

Parameters

config The desired config for the resulting bitmap

isMutable True if the resulting bitmap should be mutable (i.e. its pixels can be modified)

Returns

the new bitmap, or null if the copy could not be made.

public void **copyPixelsFromBuffer** ([Buffer](#) src)

Added in [API level 3](#)

Copy the pixels from the buffer, beginning at the current position, overwriting the bitmap's pixels. The data in the buffer is not changed in any way (unlike [setPixels\(\)](#), which converts from unpremultiplied 32bit to whatever the bitmap's native format is.

After this method returns, the current position of the buffer is updated: the position is incremented by the number of elements read from the buffer. If you need to read the bitmap from the buffer again you must first rewind the buffer.

public void **copyPixelsToBuffer** ([Buffer](#) dst)

Added in [API level 1](#)

Copy the bitmap's pixels into the specified buffer (allocated by the caller). An exception is thrown if the buffer is not large enough to hold all of the pixels (taking into account the number of bytes per pixel) or if the [Buffer](#) subclass is not one of the support types ([ByteBuffer](#), [ShortBuffer](#), [IntBuffer](#)).

The content of the bitmap is copied into the buffer as-is. This means that if this bitmap stores its pixels pre-multiplied (see [isPremultiplied\(\)](#) ([/reference/android/graphics/Bitmap.html#isPremultiplied\(\)](#)), the values in the buffer will also be pre-multiplied.

After this method returns, the current position of the buffer is updated: the position is incremented by the number of elements written in the buffer.

public static [Bitmap](#) **createBitmap** ([DisplayMetrics](#) display, int[] colors, int width, int height, [Bitmap.Config](#) config)

Added in [API level 17](#)

Returns an immutable bitmap with the specified width and height, with each pixel value set to the

corresponding value in the colors array. Its initial density is determined from the given [DisplayMetrics](/reference/android/util/DisplayMetrics.html) (</reference/android/util/DisplayMetrics.html>).

Parameters

- display* Display metrics for the display this bitmap will be drawn on.
- colors* Array of [Color](#) used to initialize the pixels. This array must be at least as large as width * height.
- width* The width of the bitmap
- height* The height of the bitmap
- config* The bitmap config to create. If the config does not support per-pixel alpha (e.g. RGB_565), then the alpha bytes in the colors[] will be ignored (assumed to be FF)

Throws

- [IllegalArgumentExcep](#)tion if the width or height are <= 0, or if the color array's length is less than the number of pixels.

public static [Bitmap](#) **createBitmap** ([DisplayMetrics](#) display, int[] colors, int offset, int stride, int width, int height, [Bitmap.Config](#) config) Added in [API level 17](#)

Returns an immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array. Its initial density is determined from the given [DisplayMetrics](/reference/android/util/DisplayMetrics.html) (</reference/android/util/DisplayMetrics.html>).

Parameters

- display* Display metrics for the display this bitmap will be drawn on.
- colors* Array of [Color](#) used to initialize the pixels.
- offset* Number of values to skip before the first color in the array of colors.
- stride* Number of colors in the array between rows (must be >= width or <= -width).
- width* The width of the bitmap
- height* The height of the bitmap
- config* The bitmap config to create. If the config does not support per-pixel alpha (e.g. RGB_565), then the alpha bytes in the colors[] will be ignored (assumed to be FF)

Throws

- [IllegalArgumentExcep](#)tion if the width or height are <= 0, or if the color array's length is less than the number of pixels.

public static [Bitmap](#) **createBitmap** ([Bitmap](#) source, int x, int y, int width, int height) Added in [API level 1](#)

Returns an immutable bitmap from the specified subset of the source bitmap. The new bitmap may be the same object as source, or a copy may have been made. It is initialized with the same density as the original bitmap.

Parameters

- source* The bitmap we are subsetting
- x* The x coordinate of the first pixel in source
- y* The y coordinate of the first pixel in source
- width* The number of pixels in each row
- height* The number of rows

Returns

A copy of a subset of the source bitmap or the source bitmap itself.

Throws

- [IllegalArgumentExcep](#)tion if the x, y, width, height values are outside of the dimensions of the source bitmap, or width is <= 0, or height is <= 0

public static [Bitmap](#) **createBitmap** ([Bitmap](#) src) Added in [API level 1](#)

Returns an immutable bitmap from the source bitmap. The new bitmap may be the same object as source, or a copy may have been made. It is initialized with the same density as the original bitmap.

public static [Bitmap](#) **createBitmap** ([DisplayMetrics](#) display, int width, int height,

Bitmap.Config config)

Added in [API level 17](#)

Returns a mutable bitmap with the specified width and height. Its initial density is determined from the given [DisplayMetrics](#) ([/reference/android/util/DisplayMetrics.html](#)).

Parameters

- display* Display metrics for the display this bitmap will be drawn on.
- width* The width of the bitmap
- height* The height of the bitmap
- config* The bitmap config to create.

Throws

- [IllegalArgumentException](#) if the width or height are ≤ 0

public static **Bitmap** **createBitmap** ([Bitmap](#) source, int x, int y, int width, int height, [Matrix](#) m, boolean filter)

Added in [API level 1](#)

Returns an immutable bitmap from subset of the source bitmap, transformed by the optional matrix. The new bitmap may be the same object as source, or a copy may have been made. It is initialized with the same density as the original bitmap. If the source bitmap is immutable and the requested subset is the same as the source bitmap itself, then the source bitmap is returned and no new bitmap is created.

Parameters

- source* The bitmap we are subsetting
- x* The x coordinate of the first pixel in source
- y* The y coordinate of the first pixel in source
- width* The number of pixels in each row
- height* The number of rows
- m* Optional matrix to be applied to the pixels
- filter* true if the source should be filtered. Only applies if the matrix contains more than just translation.

Returns

A bitmap that represents the specified subset of source

Throws

- [IllegalArgumentException](#) if the x, y, width, height values are outside of the dimensions of the source bitmap, or width is ≤ 0 , or height is ≤ 0

public static **Bitmap** **createBitmap** (int width, int height, [Bitmap.Config](#) config)

Added in [API level 1](#)

Returns a mutable bitmap with the specified width and height. Its initial density is as per [getDensity\(\)](#) ([/reference/android/graphics/Bitmap.html#getDensity\(\)](#)).

Parameters

- width* The width of the bitmap
- height* The height of the bitmap
- config* The bitmap config to create.

Throws

- [IllegalArgumentException](#) if the width or height are ≤ 0

public static **Bitmap** **createBitmap** (int[] colors, int offset, int stride, int width, int height, [Bitmap.Config](#) config)

Added in [API level 1](#)

Returns a immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array. Its initial density is as per [getDensity\(\)](#) ([/reference/android/graphics/Bitmap.html#getDensity\(\)](#)).

Parameters

- colors* Array of [Color](#) used to initialize the pixels.
- offset* Number of values to skip before the first color in the array of colors.

stride Number of colors in the array between rows (must be \geq width or \leq -width).
width The width of the bitmap
height The height of the bitmap
config The bitmap config to create. If the config does not support per-pixel alpha (e.g. RGB_565), then the alpha bytes in the colors[] will be ignored (assumed to be FF)

Throws

IllegalArgumentException if the width or height are \leq 0, or if the color array's length is less than the number of pixels.

public static Bitmap **createBitmap** (int[] colors, int width, int height, Bitmap.Config config) Added in API level 1

Returns a immutable bitmap with the specified width and height, with each pixel value set to the corresponding value in the colors array. Its initial density is as per getDensity() ([/reference/android/graphics/Bitmap.html#getDensity\(\)](/reference/android/graphics/Bitmap.html#getDensity())).

Parameters

colors Array of Color used to initialize the pixels. This array must be at least as large as width * height.
width The width of the bitmap
height The height of the bitmap
config The bitmap config to create. If the config does not support per-pixel alpha (e.g. RGB_565), then the alpha bytes in the colors[] will be ignored (assumed to be FF)

Throws

IllegalArgumentException if the width or height are \leq 0, or if the color array's length is less than the number of pixels.

public static Bitmap **createScaledBitmap** (Bitmap src, int dstWidth, int dstHeight, boolean filter) Added in API level 1

Creates a new bitmap, scaled from an existing bitmap, when possible. If the specified width and height are the same as the current width and height of the source bitmap, the source bitmap is returned and no new bitmap is created.

Parameters

src The source bitmap.
dstWidth The new bitmap's desired width.
dstHeight The new bitmap's desired height.
filter true if the source should be filtered.

Returns

The new scaled bitmap or the source bitmap if no scaling is required.

Throws

IllegalArgumentException if width is \leq 0, or height is \leq 0

public int **describeContents** () Added in API level 1

No special parcel contents.

Returns

a bitmask indicating the set of special object types marshalled by the Parcelable.

public void **eraseColor** (int c) Added in API level 1

Fills the bitmap's pixels with the specified Color (</reference/android/graphics/Color.html>).

Throws

IllegalStateException if the bitmap is not mutable.

public Bitmap **extractAlpha** () Added in API level 1

Returns a new bitmap that captures the alpha values of the original. This may be drawn with `Canvas.drawBitmap()`, where the color(s) will be taken from the paint that is passed to the draw call.

Returns

new bitmap containing the alpha channel of the original bitmap.

public Bitmap extractAlpha (Paint paint, int[] offsetXY)

Added in [API level 1](#)

Returns a new bitmap that captures the alpha values of the original. These values may be affected by the optional `Paint` parameter, which can contain its own alpha, and may also contain a `MaskFilter` which could change the actual dimensions of the resulting bitmap (e.g. a blur maskfilter might enlarge the resulting bitmap). If `offsetXY` is not null, it returns the amount to offset the returned bitmap so that it will logically align with the original. For example, if the paint contains a blur of radius 2, then `offsetXY[]` would contain -2, -2, so that drawing the alpha bitmap offset by (-2, -2) and then drawing the original would result in the blur visually aligning with the original.

The initial density of the returned bitmap is the same as the original's.

Parameters

- paint* Optional paint used to modify the alpha values in the resulting bitmap. Pass null for default behavior.
- offsetXY* Optional array that returns the X (index 0) and Y (index 1) offset needed to position the returned bitmap so that it visually lines up with the original.

Returns

new bitmap containing the (optionally modified by paint) alpha channel of the original bitmap. This may be drawn with `Canvas.drawBitmap()`, where the color(s) will be taken from the paint that is passed to the draw call.

public final int getAllocationByteCount ()

Added in [API level 19](#)

Returns the size of the allocated memory used to store this bitmap's pixels.

This can be larger than the result of `getByteCount()` ([/reference/android/graphics/Bitmap.html#getByteCount\(\)](#)) if a bitmap is reused to decode other bitmaps of smaller size, or by manual reconfiguration. See `reconfigure(int, int, Config)` ([/reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)](#)), `setWidth(int)` ([/reference/android/graphics/Bitmap.html#setWidth\(int\)](#)), `setHeight(int)` ([/reference/android/graphics/Bitmap.html#setHeight\(int\)](#)), `setConfig(Bitmap.Config)` ([/reference/android/graphics/Bitmap.html#setConfig\(android.graphics.Bitmap.Config\)](#)), and `BitmapFactory.Options.inBitmap` ([/reference/android/graphics/BitmapFactory.Options.html#inBitmap](#)). If a bitmap is not modified in this way, this value will be the same as that returned by `getByteCount()` ([/reference/android/graphics/Bitmap.html#getByteCount\(\)](#)).

This value will not change over the lifetime of a `Bitmap`.

See Also

[reconfigure\(int, int, Config\)](#)

public final int getByteCount ()

Added in [API level 12](#)

Returns the minimum number of bytes that can be used to store this bitmap's pixels.

As of `KITKAT` ([/reference/android/os/Build.VERSION_CODES.html#KITKAT](#)), the result of this method can no longer be used to determine memory usage of a bitmap. See `getAllocationByteCount()` ([/reference/android/graphics/Bitmap.html#getAllocationByteCount\(\)](#)).

public final Bitmap.Config getConfig ()

Added in [API level 1](#)

If the bitmap's internal config is in one of the public formats, return that config, otherwise return null.

public int getDensity ()

Added in [API level 4](#)

Returns the density for this bitmap.

The default density is the same density as the current display, unless the current application does not support different screen densities in which case it is `DENSITY_DEFAULT` ([/reference/android](#)

[/util/DisplayMetrics.html#DENSITY_DEFAULT](#)). Note that compatibility mode is determined by the application that was initially loaded into a process – applications that share the same process should all have the same compatibility, or ensure they explicitly set the density of their bitmaps appropriately.

Returns

A scaling factor of the default density or [DENSITY_NONE](#) if the scaling factor is unknown.

See Also

[setDensity\(int\)](#)
[DENSITY_DEFAULT](#)
[densityDpi](#)
[DENSITY_NONE](#)

public int **getGenerationId** ()

Added in [API level 12](#)

Returns the generation ID of this bitmap. The generation ID changes whenever the bitmap is modified. This can be used as an efficient way to check if a bitmap has changed.

Returns

The current generation ID for this bitmap.

public final int **getHeight** ()

Added in [API level 1](#)

Returns the bitmap's height

public byte[] **getNinePatchChunk** ()

Added in [API level 1](#)

Returns an optional array of private data, used by the UI system for some bitmaps. Not intended to be called by applications.

public int **getPixel** (int x, int y)

Added in [API level 1](#)

Returns the [Color](#) ([/reference/android/graphics/Color.html](#)) at the specified location. Throws an exception if x or y are out of bounds (negative or >= to the width or height respectively). The returned color is a non-premultiplied ARGB value.

Parameters

- x The x coordinate (0...width-1) of the pixel to return
- y The y coordinate (0...height-1) of the pixel to return

Returns

The argb [Color](#) at the specified coordinate

Throws

[IllegalArgumentException](#) if x, y exceed the bitmap's bounds

public void **getPixels** (int[] pixels, int offset, int stride, int x, int y, int width, int height)

Added in [API level 1](#)

Returns in pixels[] a copy of the data in the bitmap. Each value is a packed int representing a [Color](#) ([/reference/android/graphics/Color.html](#)). The stride parameter allows the caller to allow for gaps in the returned pixels array between rows. For normal packed results, just pass width for the stride value. The returned colors are non-premultiplied ARGB values.

Parameters

- pixels The array to receive the bitmap's colors
- offset The first index to write into pixels[]
- stride The number of entries in pixels[] to skip between rows (must be >= bitmap's width). Can be negative.
- x The x coordinate of the first pixel to read from the bitmap
- y The y coordinate of the first pixel to read from the bitmap
- width The number of pixels to read from each row
- height The number of rows to read

Throws

[IllegalArgumentException](#)

if x, y, width, height exceed the bounds of the bitmap, or if
abs(stride) < width.

[ArrayIndexOutOfBoundsException](#)

if the pixels array is too small to receive the specified
number of pixels.

public final int **getRowBytes** ()

Added in [API level 1](#)

Return the number of bytes between rows in the bitmap's pixels. Note that this refers to the pixels as stored natively by the bitmap. If you call `getPixels()` or `setPixels()`, then the pixels are uniformly treated as 32bit values, packed according to the Color class.

As of [KITKAT](#) ([//reference/android/os/Build.VERSION_CODES.html#KITKAT](#)), this method should not be used to calculate the memory usage of the bitmap. Instead, see [getAllocationByteCount\(\)](#) ([//reference/android/graphics/Bitmap.html#getAllocationByteCount\(\)](#)).

Returns

number of bytes between rows of the native bitmap pixels.

public int **getScaledHeight** ([DisplayMetrics](#) metrics)

Added in [API level 4](#)

Convenience for calling [getScaledHeight\(int\)](#) ([//reference/android/graphics/Bitmap.html#getScaledHeight\(int\)](#)) with the target density of the given [DisplayMetrics](#) ([//reference/android/util/DisplayMetrics.html](#)).

public int **getScaledHeight** (int targetDensity)

Added in [API level 4](#)

Convenience method that returns the height of this bitmap divided by the density scale factor.

Parameters

targetDensity The density of the target canvas of the bitmap.

Returns

The scaled height of this bitmap, according to the density scale factor.

public int **getScaledHeight** ([Canvas](#) canvas)

Added in [API level 4](#)

Convenience for calling [getScaledHeight\(int\)](#) ([//reference/android/graphics/Bitmap.html#getScaledHeight\(int\)](#)) with the target density of the given [Canvas](#) ([//reference/android/graphics/Canvas.html](#)).

public int **getScaledWidth** (int targetDensity)

Added in [API level 4](#)

Convenience method that returns the width of this bitmap divided by the density scale factor.

Parameters

targetDensity The density of the target canvas of the bitmap.

Returns

The scaled width of this bitmap, according to the density scale factor.

public int **getScaledWidth** ([DisplayMetrics](#) metrics)

Added in [API level 4](#)

Convenience for calling [getScaledWidth\(int\)](#) ([//reference/android/graphics/Bitmap.html#getScaledWidth\(int\)](#)) with the target density of the given [DisplayMetrics](#) ([//reference/android/util/DisplayMetrics.html](#)).

public int **getScaledWidth** ([Canvas](#) canvas)

Added in [API level 4](#)

Convenience for calling [getScaledWidth\(int\)](#) ([//reference/android/graphics/Bitmap.html#getScaledWidth\(int\)](#)) with the target density of the given [Canvas](#) ([//reference/android/graphics/Canvas.html](#)).

public final int **getWidth** ()

Added in [API level 1](#)

Returns the bitmap's width

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

Returns true if the bitmap's config supports per-pixel alpha, and if the pixels may contain non-opaque alpha values. For some configs, this is always false (e.g. RGB_565), since they do not support per-pixel alpha. However, for configs that do, the bitmap may be flagged to be known that all of its pixels are opaque. In this case hasAlpha() will also return false. If a config such as ARGB_8888 is not so flagged, it will return true by default.

public final boolean hasMipMap ()Added in [API level 17](#)

Indicates whether the renderer responsible for drawing this bitmap should attempt to use mipmaps when this bitmap is drawn scaled down. If you know that you are going to draw this bitmap at less than 50% of its original size, you may be able to obtain a higher quality. This property is only a suggestion that can be ignored by the renderer. It is not guaranteed to have any effect.

Returns

true if the renderer should attempt to use mipmaps, false otherwise

See Also

[setHasMipMap\(boolean\)](#)

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

Returns true if the bitmap is marked as mutable (i.e. can be drawn into)

public final boolean isPremultiplied ()Added in [API level 17](#)

Indicates whether pixels stored in this bitmaps are stored pre-multiplied. When a pixel is pre-multiplied, the RGB components have been multiplied by the alpha component. For instance, if the original color is a 50% translucent red (128, 255, 0, 0), the pre-multiplied form is (128, 128, 0, 0).

This method always returns false if [getConfig\(\)](#) ([reference/android/graphics/Bitmap.html#getConfig\(\)](#)) is [RGB_565](#) ([reference/android/graphics/Bitmap.Config.html#RGB_565](#)).

This method only returns true if [hasAlpha\(\)](#) ([reference/android/graphics/Bitmap.html#hasAlpha\(\)](#)) returns true. A bitmap with no alpha channel can be used both as a pre-multiplied and as a non pre-multiplied bitmap.

Only pre-multiplied bitmaps may be drawn by the view system or [Canvas](#) ([reference/android/graphics/Canvas.html](#)). If a non-pre-multiplied bitmap with an alpha channel is drawn to a Canvas, a RuntimeException will be thrown.

Returns

true if the underlying pixels have been pre-multiplied, false otherwise

See Also

[setPremultiplied\(boolean\)](#)

[inPremultiplied](#)

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

Returns true if this bitmap has been recycled. If so, then it is an error to try to access its pixels, and the bitmap will not draw.

Returns

true if the bitmap has been recycled

public void prepareToDraw ()Added in [API level 4](#)

Rebuilds any caches associated with the bitmap that are used for drawing it. In the case of purgeable bitmaps, this call will attempt to ensure that the pixels have been decoded. If this is called on more than one bitmap in sequence, the priority is given in LRU order (i.e. the last bitmap called will be given highest priority). For bitmaps with no associated caches, this call is effectively a no-op, and therefore is harmless.

public void reconfigure (int width, int height, [Bitmap.Config](#) config)Added in [API level 19](#)

Modifies the bitmap to have a specified width, height, and [Bitmap.Config](#) ([reference/android](#)

[/graphics/Bitmap.Config.html](#)), without affecting the underlying allocation backing the bitmap. Bitmap pixel data is not re-initialized for the new configuration.

This method can be used to avoid allocating a new bitmap, instead reusing an existing bitmap's allocation for a new configuration of equal or lesser size. If the Bitmap's allocation isn't large enough to support the new configuration, an `IllegalArgumentException` will be thrown and the bitmap will not be modified.

The result of `getByteCount()` ([/reference/android/graphics/Bitmap.html#getByteCount\(\)](#)) will reflect the new configuration, while `getAllocationByteCount()` ([/reference/android/graphics/Bitmap.html#getAllocationByteCount\(\)](#)) will reflect that of the initial configuration.

WARNING: This method should NOT be called on a bitmap currently used by the view system. It does not make guarantees about how the underlying pixel buffer is remapped to the new config, just that the allocation is reused. Additionally, the view system does not account for bitmap properties being modifying during use, e.g. while attached to drawables.

See Also

[setWidth\(int\)](#)
[setHeight\(int\)](#)
[setConfig\(Config\)](#)

public void **recycle** ()

Added in [API level 1](#)

Free the native object associated with this bitmap, and clear the reference to the pixel data. This will not free the pixel data synchronously; it simply allows it to be garbage collected if there are no other references. The bitmap is marked as "dead", meaning it will throw an exception if `getPixels()` or `setPixels()` is called, and will draw nothing. This operation cannot be reversed, so it should only be called if you are sure there are no further uses for the bitmap. This is an advanced call, and normally need not be called, since the normal GC process will free up this memory when there are no more references to this bitmap.

public boolean **sameAs** ([Bitmap](#) other)

Added in [API level 12](#)

Given another bitmap, return true if it has the same dimensions, config, and pixel data as this bitmap. If any of those differ, return false. If other is null, return false.

public void **setConfig** ([Bitmap.Config](#) config)

Added in [API level 19](#)

Convenience method for calling `reconfigure(int, int, Config)` ([/reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)](#)) with the current height and width.

WARNING: this method should not be used on bitmaps currently used by the view system, see `reconfigure(int, int, Config)` ([/reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)](#)) for more details.

See Also

[reconfigure\(int, int, Config\)](#)
[setWidth\(int\)](#)
[setHeight\(int\)](#)

public void **setDensity** (int density)

Added in [API level 4](#)

Specifies the density for this bitmap. When the bitmap is drawn to a Canvas that also has a density, it will be scaled appropriately.

Parameters

density The density scaling factor to use with this bitmap or `DENSITY_NONE` if the density is unknown.

See Also

[getDensity\(\)](#)
[DENSITY_DEFAULT](#)
[densityDpi](#)
[DENSITY_NONE](#)

public void **setHasAlpha** (boolean hasAlpha)

Tell the bitmap if all of the pixels are known to be opaque (false) or if some of the pixels may contain non-opaque alpha values (true). Note, for some configs (e.g. RGB_565) this call is ignored, since it does not support per-pixel alpha values. This is meant as a drawing hint, as in some cases a bitmap that is known to be opaque can take a faster drawing case than one that may have non-opaque per-pixel alpha values.

public final void **setHasMipMap** (boolean hasMipMap)

Added in [API level 17](#)

Set a hint for the renderer responsible for drawing this bitmap indicating that it should attempt to use mipmaps when this bitmap is drawn scaled down. If you know that you are going to draw this bitmap at less than 50% of its original size, you may be able to obtain a higher quality by turning this property on. Note that if the renderer respects this hint it might have to allocate extra memory to hold the mipmap levels for this bitmap. This property is only a suggestion that can be ignored by the renderer. It is not guaranteed to have any effect.

Parameters

hasMipMap indicates whether the renderer should attempt to use mipmaps

See Also

[hasMipMap\(\)](#)

public void **setHeight** (int height)

Added in [API level 19](#)

Convenience method for calling [reconfigure\(int, int, Config\)](#) ([/reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)](#)) with the current width and config.

WARNING: this method should not be used on bitmaps currently used by the view system, see [reconfigure\(int, int, Config\)](#) ([/reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)](#)) for more details.

See Also

[reconfigure\(int, int, Config\)](#)

[setWidth\(int\)](#)

[setConfig\(Config\)](#)

public void **setPixel** (int x, int y, int color)

Added in [API level 1](#)

Write the specified [Color](#) ([/reference/android/graphics/Color.html](#)) into the bitmap (assuming it is mutable) at the x,y coordinate. The color must be a non-premultiplied ARGB value.

Parameters

x The x coordinate of the pixel to replace (0...width-1)

y The y coordinate of the pixel to replace (0...height-1)

color The ARGB color to write into the bitmap

Throws

[IllegalStateException](#) if the bitmap is not mutable

[IllegalArgumentException](#) if x, y are outside of the bitmap's bounds.

public void **setPixels** (int[] pixels, int offset, int stride, int x, int y, int width, int height)

Added in [API level 1](#)

Replace pixels in the bitmap with the colors in the array. Each element in the array is a packed int representing a non-premultiplied ARGB [Color](#) ([/reference/android/graphics/Color.html](#)).

Parameters

pixels The colors to write to the bitmap

offset The index of the first color to read from pixels[]

stride The number of colors in pixels[] to skip between rows. Normally this value will be the same as the width of the bitmap, but it can be larger (or negative).

x The x coordinate of the first pixel to write to in the bitmap.

y The y coordinate of the first pixel to write to in the bitmap.

width The number of colors to copy from pixels[] per row

height The number of rows to write to the bitmap

Throws

<u><code>IllegalStateException</code></u>	if the bitmap is not mutable
<u><code>IllegalArgumentException</code></u>	if x, y, width, height are outside of the bitmap's bounds.
<u><code>ArrayIndexOutOfBoundsException</code></u>	if the pixels array is too small to receive the specified number of pixels.

public final void **setPremultiplied** (boolean premultiplied)

Added in [API level 19](#)

Sets whether the bitmap should treat its data as pre-multiplied.

Bitmaps are always treated as pre-multiplied by the view system and [`Canvas`](#) ([`//reference/android/graphics/Canvas.html`](#)) for performance reasons. Storing un-pre-multiplied data in a Bitmap (through [`setPixel\(int, int, int\)`](#) ([`//reference/android/graphics/Bitmap.html#setPixel\(int, int, int\)`](#)), [`setPixels\(int\[\], int, int, int, int, int, int, int\)`](#) ([`//reference/android/graphics/Bitmap.html#setPixels\(int\[\], int, int, int, int, int, int, int\)`](#)), or [`BitmapFactory.Options.inPremultiplied`](#) ([`//reference/android/graphics/BitmapFactory.Options.html#inPremultiplied`](#))) can lead to incorrect blending if drawn by the framework.

This method will not affect the behavior of a bitmap without an alpha channel, or if [`hasAlpha\(\)`](#) ([`//reference/android/graphics/Bitmap.html#hasAlpha\(\)`](#)) returns false.

See Also

[`isPremultiplied\(\)`](#)
[`inPremultiplied`](#)

public void **setWidth** (int width)

Added in [API level 19](#)

Convenience method for calling [`reconfigure\(int, int, Config\)`](#) ([`//reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)`](#)) with the current height and config.

WARNING: this method should not be used on bitmaps currently used by the view system, see [`reconfigure\(int, int, Config\)`](#) ([`//reference/android/graphics/Bitmap.html#reconfigure\(int, int, android.graphics.Bitmap.Config\)`](#)) for more details.

See Also

[`reconfigure\(int, int, Config\)`](#)
[`setHeight\(int\)`](#)
[`setConfig\(Config\)`](#)

public void **writeToParcel** ([`Parcel`](#) p, int flags)

Added in [API level 1](#)

Write the bitmap and its pixels to the parcel. The bitmap can be rebuilt from the parcel by calling `CREATOR.createFromParcel()`.

Parameters

- p* Parcel object to write the bitmap data into
- flags* Additional flags about how the object should be written. May be 0 or [`PARCELABLE_WRITE_RETURN_VALUE`](#).