| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BufferStrategy.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/image/BufferedImageOp.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/ByteLookupTable.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/BufferStrategy.html)    [**NO FRAMES**](http://docs.google.com/BufferStrategy.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

## **java.awt.image**

Class BufferStrategy

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.awt.image.BufferStrategy**

**Direct Known Subclasses:** [Component.BltBufferStrategy](http://docs.google.com/java/awt/Component.BltBufferStrategy.html), [Component.FlipBufferStrategy](http://docs.google.com/java/awt/Component.FlipBufferStrategy.html)

public abstract class **BufferStrategy**extends [Object](http://docs.google.com/java/lang/Object.html)

The BufferStrategy class represents the mechanism with which to organize complex memory on a particular Canvas or Window. Hardware and software limitations determine whether and how a particular buffer strategy can be implemented. These limitations are detectible through the capabilities of the GraphicsConfiguration used when creating the Canvas or Window.

It is worth noting that the terms *buffer* and *surface* are meant to be synonymous: an area of contiguous memory, either in video device memory or in system memory.

There are several types of complex buffer strategies, including sequential ring buffering and blit buffering. Sequential ring buffering (i.e., double or triple buffering) is the most common; an application draws to a single *back buffer* and then moves the contents to the front (display) in a single step, either by copying the data or moving the video pointer. Moving the video pointer exchanges the buffers so that the first buffer drawn becomes the *front buffer*, or what is currently displayed on the device; this is called *page flipping*.

Alternatively, the contents of the back buffer can be copied, or *blitted* forward in a chain instead of moving the video pointer.

Double buffering:  
  
 \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*  
 \* \* ------> \* \*  
 [To display] <---- \* Front B \* Show \* Back B. \* <---- Rendering  
 \* \* <------ \* \*  
 \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*  
  
 Triple buffering:  
  
 [To \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*  
 display] \* \* --------+---------+------> \* \*  
 <---- \* Front B \* Show \* Mid. B. \* \* Back B. \* <---- Rendering  
 \* \* <------ \* \* <----- \* \*  
 \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*

Here is an example of how buffer strategies can be created and used:

// Check the capabilities of the GraphicsConfiguration  
 ...  
  
 // Create our component  
 Window w = new Window(gc);  
  
 // Show our window  
 w.setVisible(true);  
  
 // Create a general double-buffering strategy  
 w.createBufferStrategy(2);  
 BufferStrategy strategy = w.getBufferStrategy();  
  
 // Main loop  
 while (!done) {  
 // Prepare for rendering the next frame  
 // ...  
  
 // Render single frame  
 do {  
 // The following loop ensures that the contents of the drawing buffer  
 // are consistent in case the underlying surface was recreated  
 do {  
 // Get a new graphics context every time through the loop  
 // to make sure the strategy is validated  
 Graphics graphics = strategy.getDrawGraphics();  
   
 // Render to graphics  
 // ...  
  
 // Dispose the graphics  
 graphics.dispose();  
  
 // Repeat the rendering if the drawing buffer contents   
 // were restored  
 } while (strategy.contentsRestored());  
  
 // Display the buffer  
 strategy.show();  
  
 // Repeat the rendering if the drawing buffer was lost  
 } while (strategy.contentsLost());  
 }  
  
 // Dispose the window  
 w.setVisible(false);  
 w.dispose();

**Since:** 1.4 **See Also:**[Component](http://docs.google.com/java/awt/Component.html), [GraphicsConfiguration](http://docs.google.com/java/awt/GraphicsConfiguration.html), [VolatileImage](http://docs.google.com/java/awt/image/VolatileImage.html)

| **Constructor Summary** | |
| --- | --- |
| [**BufferStrategy**](http://docs.google.com/java/awt/image/BufferStrategy.html#BufferStrategy())() |

| **Method Summary** | |
| --- | --- |
| abstract  boolean | [**contentsLost**](http://docs.google.com/java/awt/image/BufferStrategy.html#contentsLost())()            Returns whether the drawing buffer was lost since the last call to getDrawGraphics. |
| abstract  boolean | [**contentsRestored**](http://docs.google.com/java/awt/image/BufferStrategy.html#contentsRestored())()            Returns whether the drawing buffer was recently restored from a lost state and reinitialized to the default background color (white). |
| void | [**dispose**](http://docs.google.com/java/awt/image/BufferStrategy.html#dispose())()            Releases system resources currently consumed by this BufferStrategy and removes it from the associated Component. |
| abstract  [BufferCapabilities](http://docs.google.com/java/awt/BufferCapabilities.html) | [**getCapabilities**](http://docs.google.com/java/awt/image/BufferStrategy.html#getCapabilities())()            Returns the BufferCapabilities for this BufferStrategy. |
| abstract  [Graphics](http://docs.google.com/java/awt/Graphics.html) | [**getDrawGraphics**](http://docs.google.com/java/awt/image/BufferStrategy.html#getDrawGraphics())()            Creates a graphics context for the drawing buffer. |
| abstract  void | [**show**](http://docs.google.com/java/awt/image/BufferStrategy.html#show())()            Makes the next available buffer visible by either copying the memory (blitting) or changing the display pointer (flipping). |

| **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()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [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)) |

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

### BufferStrategy

public **BufferStrategy**()

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

### getCapabilities

public abstract [BufferCapabilities](http://docs.google.com/java/awt/BufferCapabilities.html) **getCapabilities**()

Returns the BufferCapabilities for this BufferStrategy.

**Returns:**the buffering capabilities of this strategy

### getDrawGraphics

public abstract [Graphics](http://docs.google.com/java/awt/Graphics.html) **getDrawGraphics**()

Creates a graphics context for the drawing buffer. This method may not be synchronized for performance reasons; use of this method by multiple threads should be handled at the application level. Disposal of the graphics object obtained must be handled by the application.

**Returns:**a graphics context for the drawing buffer

### contentsLost

public abstract boolean **contentsLost**()

Returns whether the drawing buffer was lost since the last call to getDrawGraphics. Since the buffers in a buffer strategy are usually type VolatileImage, they may become lost. For a discussion on lost buffers, see VolatileImage.

**Returns:**Whether or not the drawing buffer was lost since the last call to getDrawGraphics.**See Also:**[VolatileImage](http://docs.google.com/java/awt/image/VolatileImage.html)

### contentsRestored

public abstract boolean **contentsRestored**()

Returns whether the drawing buffer was recently restored from a lost state and reinitialized to the default background color (white). Since the buffers in a buffer strategy are usually type VolatileImage, they may become lost. If a surface has been recently restored from a lost state since the last call to getDrawGraphics, it may require repainting. For a discussion on lost buffers, see VolatileImage.

**Returns:**Whether or not the drawing buffer was restored since the last call to getDrawGraphics.**See Also:**[VolatileImage](http://docs.google.com/java/awt/image/VolatileImage.html)

### show

public abstract void **show**()

Makes the next available buffer visible by either copying the memory (blitting) or changing the display pointer (flipping).

### dispose

public void **dispose**()

Releases system resources currently consumed by this BufferStrategy and removes it from the associated Component. After invoking this method, getBufferStrategy will return null. Trying to use a BufferStrategy after it has been disposed will result in undefined behavior.

**Since:** 1.6 **See Also:**[Component.createBufferStrategy(int)](http://docs.google.com/java/awt/Component.html#createBufferStrategy(int)), [Component.getBufferStrategy()](http://docs.google.com/java/awt/Component.html#getBufferStrategy())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BufferStrategy.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/awt/image/BufferedImageOp.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/ByteLookupTable.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/BufferStrategy.html)    [**NO FRAMES**](http://docs.google.com/BufferStrategy.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#4d34og8) |

[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).