| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MultiPixelPackedSampleModel.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/MemoryImageSource.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/PackedColorModel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/MultiPixelPackedSampleModel.html)    [**NO FRAMES**](http://docs.google.com/MultiPixelPackedSampleModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#26in1rg) |

## **java.awt.image**

Class MultiPixelPackedSampleModel

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

public class **MultiPixelPackedSampleModel**extends [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html)

The MultiPixelPackedSampleModel class represents one-banded images and can pack multiple one-sample pixels into one data element. Pixels are not allowed to span data elements. The data type can be DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT. Each pixel must be a power of 2 number of bits and a power of 2 number of pixels must fit exactly in one data element. Pixel bit stride is equal to the number of bits per pixel. Scanline stride is in data elements and the last several data elements might be padded with unused pixels. Data bit offset is the offset in bits from the beginning of the [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) to the first pixel and must be a multiple of pixel bit stride.

The following code illustrates extracting the bits for pixel x, y from DataBuffer data and storing the pixel data in data elements of type dataType:

int dataElementSize = DataBuffer.getDataTypeSize(dataType);  
 int bitnum = dataBitOffset + x\*pixelBitStride;  
 int element = data.getElem(y\*scanlineStride + bitnum/dataElementSize);  
 int shift = dataElementSize - (bitnum & (dataElementSize-1))  
 - pixelBitStride;  
 int pixel = (element >> shift) & ((1 << pixelBitStride) - 1);

| **Field Summary** | |
| --- | --- |

| **Fields inherited from class java.awt.image.**[**SampleModel**](http://docs.google.com/java/awt/image/SampleModel.html) |
| --- |
| [dataType](http://docs.google.com/java/awt/image/SampleModel.html#dataType), [height](http://docs.google.com/java/awt/image/SampleModel.html#height), [numBands](http://docs.google.com/java/awt/image/SampleModel.html#numBands), [width](http://docs.google.com/java/awt/image/SampleModel.html#width) |

| **Constructor Summary** | |
| --- | --- |
| [**MultiPixelPackedSampleModel**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#MultiPixelPackedSampleModel(int,%20int,%20int,%20int))(int dataType, int w, int h, int numberOfBits)            Constructs a MultiPixelPackedSampleModel with the specified data type, width, height and number of bits per pixel. |
| [**MultiPixelPackedSampleModel**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#MultiPixelPackedSampleModel(int,%20int,%20int,%20int,%20int,%20int))(int dataType, int w, int h, int numberOfBits, int scanlineStride, int dataBitOffset)            Constructs a MultiPixelPackedSampleModel with specified data type, width, height, number of bits per pixel, scanline stride and data bit offset. |

| **Method Summary** | |
| --- | --- |
| [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) | [**createCompatibleSampleModel**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#createCompatibleSampleModel(int,%20int))(int w, int h)            Creates a new MultiPixelPackedSampleModel with the specified width and height. |
| [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) | [**createDataBuffer**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#createDataBuffer())()            Creates a DataBuffer that corresponds to this MultiPixelPackedSampleModel. |
| [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) | [**createSubsetSampleModel**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#createSubsetSampleModel(int%5B%5D))(int[] bands)            Creates a new MultiPixelPackedSampleModel with a subset of the bands of this MultiPixelPackedSampleModel. |
| boolean | [**equals**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Indicates whether some other object is "equal to" this one. |
| int | [**getBitOffset**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getBitOffset(int))(int x)            Returns the offset, in bits, into the data element in which it is stored for the xth pixel of a scanline. |
| int | [**getDataBitOffset**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getDataBitOffset())()            Returns the data bit offset in bits. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getDataElements**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer))(int x, int y, [Object](http://docs.google.com/java/lang/Object.html) obj, [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)            Returns data for a single pixel in a primitive array of type TransferType. |
| int | [**getNumDataElements**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getNumDataElements())()            Returns the number of data elements needed to transfer one pixel via the [getDataElements(int, int, java.lang.Object, java.awt.image.DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)) and [setDataElements(int, int, java.lang.Object, java.awt.image.DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)) methods. |
| int | [**getOffset**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getOffset(int,%20int))(int x, int y)            Returns the offset of pixel (x, y) in data array elements. |
| int[] | [**getPixel**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getPixel(int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer))(int x, int y, int[] iArray, [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)            Returns the specified single band pixel in the first element of an int array. |
| int | [**getPixelBitStride**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getPixelBitStride())()            Returns the pixel bit stride in bits. |
| int | [**getSample**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getSample(int,%20int,%20int,%20java.awt.image.DataBuffer))(int x, int y, int b, [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)            Returns as int the sample in a specified band for the pixel located at (x, y). |
| int[] | [**getSampleSize**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getSampleSize())()            Returns the number of bits per sample for all bands. |
| int | [**getSampleSize**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getSampleSize(int))(int band)            Returns the number of bits per sample for the specified band. |
| int | [**getScanlineStride**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getScanlineStride())()            Returns the scanline stride. |
| int | [**getTransferType**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getTransferType())()            Returns the TransferType used to transfer pixels by way of the getDataElements and setDataElements methods. |
| int | [**hashCode**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#hashCode())()            Returns a hash code value for the object. |
| void | [**setDataElements**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer))(int x, int y, [Object](http://docs.google.com/java/lang/Object.html) obj, [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)            Sets the data for a single pixel in the specified DataBuffer from a primitive array of type TransferType. |
| void | [**setPixel**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setPixel(int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer))(int x, int y, int[] iArray, [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)            Sets a pixel in the DataBuffer using an int array for input. |
| void | [**setSample**](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setSample(int,%20int,%20int,%20int,%20java.awt.image.DataBuffer))(int x, int y, int b, int s, [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)            Sets a sample in the specified band for the pixel located at (x, y) in the DataBuffer using an int for input. |

| **Methods inherited from class java.awt.image.**[**SampleModel**](http://docs.google.com/java/awt/image/SampleModel.html) |
| --- |
| [getDataElements](http://docs.google.com/java/awt/image/SampleModel.html#getDataElements(int,%20int,%20int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [getDataType](http://docs.google.com/java/awt/image/SampleModel.html#getDataType()), [getHeight](http://docs.google.com/java/awt/image/SampleModel.html#getHeight()), [getNumBands](http://docs.google.com/java/awt/image/SampleModel.html#getNumBands()), [getPixel](http://docs.google.com/java/awt/image/SampleModel.html#getPixel(int,%20int,%20double%5B%5D,%20java.awt.image.DataBuffer)), [getPixel](http://docs.google.com/java/awt/image/SampleModel.html#getPixel(int,%20int,%20float%5B%5D,%20java.awt.image.DataBuffer)), [getPixels](http://docs.google.com/java/awt/image/SampleModel.html#getPixels(int,%20int,%20int,%20int,%20double%5B%5D,%20java.awt.image.DataBuffer)), [getPixels](http://docs.google.com/java/awt/image/SampleModel.html#getPixels(int,%20int,%20int,%20int,%20float%5B%5D,%20java.awt.image.DataBuffer)), [getPixels](http://docs.google.com/java/awt/image/SampleModel.html#getPixels(int,%20int,%20int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer)), [getSampleDouble](http://docs.google.com/java/awt/image/SampleModel.html#getSampleDouble(int,%20int,%20int,%20java.awt.image.DataBuffer)), [getSampleFloat](http://docs.google.com/java/awt/image/SampleModel.html#getSampleFloat(int,%20int,%20int,%20java.awt.image.DataBuffer)), [getSamples](http://docs.google.com/java/awt/image/SampleModel.html#getSamples(int,%20int,%20int,%20int,%20int,%20double%5B%5D,%20java.awt.image.DataBuffer)), [getSamples](http://docs.google.com/java/awt/image/SampleModel.html#getSamples(int,%20int,%20int,%20int,%20int,%20float%5B%5D,%20java.awt.image.DataBuffer)), [getSamples](http://docs.google.com/java/awt/image/SampleModel.html#getSamples(int,%20int,%20int,%20int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer)), [getWidth](http://docs.google.com/java/awt/image/SampleModel.html#getWidth()), [setDataElements](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [setPixel](http://docs.google.com/java/awt/image/SampleModel.html#setPixel(int,%20int,%20double%5B%5D,%20java.awt.image.DataBuffer)), [setPixel](http://docs.google.com/java/awt/image/SampleModel.html#setPixel(int,%20int,%20float%5B%5D,%20java.awt.image.DataBuffer)), [setPixels](http://docs.google.com/java/awt/image/SampleModel.html#setPixels(int,%20int,%20int,%20int,%20double%5B%5D,%20java.awt.image.DataBuffer)), [setPixels](http://docs.google.com/java/awt/image/SampleModel.html#setPixels(int,%20int,%20int,%20int,%20float%5B%5D,%20java.awt.image.DataBuffer)), [setPixels](http://docs.google.com/java/awt/image/SampleModel.html#setPixels(int,%20int,%20int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer)), [setSample](http://docs.google.com/java/awt/image/SampleModel.html#setSample(int,%20int,%20int,%20double,%20java.awt.image.DataBuffer)), [setSample](http://docs.google.com/java/awt/image/SampleModel.html#setSample(int,%20int,%20int,%20float,%20java.awt.image.DataBuffer)), [setSamples](http://docs.google.com/java/awt/image/SampleModel.html#setSamples(int,%20int,%20int,%20int,%20int,%20double%5B%5D,%20java.awt.image.DataBuffer)), [setSamples](http://docs.google.com/java/awt/image/SampleModel.html#setSamples(int,%20int,%20int,%20int,%20int,%20float%5B%5D,%20java.awt.image.DataBuffer)), [setSamples](http://docs.google.com/java/awt/image/SampleModel.html#setSamples(int,%20int,%20int,%20int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer)) |

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

### MultiPixelPackedSampleModel

public **MultiPixelPackedSampleModel**(int dataType,  
 int w,  
 int h,  
 int numberOfBits)

Constructs a MultiPixelPackedSampleModel with the specified data type, width, height and number of bits per pixel.

**Parameters:**dataType - the data type for storing samplesw - the width, in pixels, of the region of image data describedh - the height, in pixels, of the region of image data describednumberOfBits - the number of bits per pixel **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if dataType is not either DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT

### MultiPixelPackedSampleModel

public **MultiPixelPackedSampleModel**(int dataType,  
 int w,  
 int h,  
 int numberOfBits,  
 int scanlineStride,  
 int dataBitOffset)

Constructs a MultiPixelPackedSampleModel with specified data type, width, height, number of bits per pixel, scanline stride and data bit offset.

**Parameters:**dataType - the data type for storing samplesw - the width, in pixels, of the region of image data describedh - the height, in pixels, of the region of image data describednumberOfBits - the number of bits per pixelscanlineStride - the line stride of the image datadataBitOffset - the data bit offset for the region of image data described **Throws:** [RasterFormatException](http://docs.google.com/java/awt/image/RasterFormatException.html) - if the number of bits per pixel is not a power of 2 or if a power of 2 number of pixels do not fit in one data element. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if w or h is not greater than 0 [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if dataType is not either DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT

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

### createCompatibleSampleModel

public [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **createCompatibleSampleModel**(int w,  
 int h)

Creates a new MultiPixelPackedSampleModel with the specified width and height. The new MultiPixelPackedSampleModel has the same storage data type and number of bits per pixel as this MultiPixelPackedSampleModel.

**Specified by:**[createCompatibleSampleModel](http://docs.google.com/java/awt/image/SampleModel.html#createCompatibleSampleModel(int,%20int)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**w - the specified widthh - the specified height **Returns:**a [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) with the specified width and height and with the same storage data type and number of bits per pixel as this MultiPixelPackedSampleModel. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if w or h is not greater than 0

### createDataBuffer

public [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) **createDataBuffer**()

Creates a DataBuffer that corresponds to this MultiPixelPackedSampleModel. The DataBuffer object's data type and size is consistent with this MultiPixelPackedSampleModel. The DataBuffer has a single bank.

**Specified by:**[createDataBuffer](http://docs.google.com/java/awt/image/SampleModel.html#createDataBuffer()) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Returns:**a DataBuffer with the same data type and size as this MultiPixelPackedSampleModel.

### getNumDataElements

public int **getNumDataElements**()

Returns the number of data elements needed to transfer one pixel via the [getDataElements(int, int, java.lang.Object, java.awt.image.DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)) and [setDataElements(int, int, java.lang.Object, java.awt.image.DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)) methods. For a MultiPixelPackedSampleModel, this is one.

**Specified by:**[getNumDataElements](http://docs.google.com/java/awt/image/SampleModel.html#getNumDataElements()) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Returns:**the number of data elements.**See Also:**[SampleModel.getDataElements(int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.getDataElements(int, int, int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#getDataElements(int,%20int,%20int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.setDataElements(int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.setDataElements(int, int, int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.getTransferType()](http://docs.google.com/java/awt/image/SampleModel.html#getTransferType())

### getSampleSize

public int[] **getSampleSize**()

Returns the number of bits per sample for all bands.

**Specified by:**[getSampleSize](http://docs.google.com/java/awt/image/SampleModel.html#getSampleSize()) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Returns:**the number of bits per sample.

### getSampleSize

public int **getSampleSize**(int band)

Returns the number of bits per sample for the specified band.

**Specified by:**[getSampleSize](http://docs.google.com/java/awt/image/SampleModel.html#getSampleSize(int)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**band - the specified band **Returns:**the number of bits per sample for the specified band.

### getOffset

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

Returns the offset of pixel (x, y) in data array elements.

**Parameters:**x - the X coordinate of the specified pixely - the Y coordinate of the specified pixel **Returns:**the offset of the specified pixel.

### getBitOffset

public int **getBitOffset**(int x)

Returns the offset, in bits, into the data element in which it is stored for the xth pixel of a scanline. This offset is the same for all scanlines.

**Parameters:**x - the specified pixel **Returns:**the bit offset of the specified pixel.

### getScanlineStride

public int **getScanlineStride**()

Returns the scanline stride.

**Returns:**the scanline stride of this MultiPixelPackedSampleModel.

### getPixelBitStride

public int **getPixelBitStride**()

Returns the pixel bit stride in bits. This value is the same as the number of bits per pixel.

**Returns:**the pixelBitStride of this MultiPixelPackedSampleModel.

### getDataBitOffset

public int **getDataBitOffset**()

Returns the data bit offset in bits.

**Returns:**the dataBitOffset of this MultiPixelPackedSampleModel.

### getTransferType

public int **getTransferType**()

Returns the TransferType used to transfer pixels by way of the getDataElements and setDataElements methods. The TransferType might or might not be the same as the storage DataType. The TransferType is one of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT.

**Overrides:**[getTransferType](http://docs.google.com/java/awt/image/SampleModel.html#getTransferType()) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Returns:**the transfertype.**See Also:**[SampleModel.getDataElements(int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.getDataElements(int, int, int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#getDataElements(int,%20int,%20int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.setDataElements(int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.setDataElements(int, int, int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)), [SampleModel.getNumDataElements()](http://docs.google.com/java/awt/image/SampleModel.html#getNumDataElements()), [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html)

### createSubsetSampleModel

public [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **createSubsetSampleModel**(int[] bands)

Creates a new MultiPixelPackedSampleModel with a subset of the bands of this MultiPixelPackedSampleModel. Since a MultiPixelPackedSampleModel only has one band, the bands argument must have a length of one and indicate the zeroth band.

**Specified by:**[createSubsetSampleModel](http://docs.google.com/java/awt/image/SampleModel.html#createSubsetSampleModel(int%5B%5D)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**bands - the specified bands **Returns:**a new SampleModel with a subset of bands of this MultiPixelPackedSampleModel. **Throws:** [RasterFormatException](http://docs.google.com/java/awt/image/RasterFormatException.html) - if the number of bands requested is not one. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if w or h is not greater than 0

### getSample

public int **getSample**(int x,  
 int y,  
 int b,  
 [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)

Returns as int the sample in a specified band for the pixel located at (x, y). An ArrayIndexOutOfBoundsException is thrown if the coordinates are not in bounds.

**Specified by:**[getSample](http://docs.google.com/java/awt/image/SampleModel.html#getSample(int,%20int,%20int,%20java.awt.image.DataBuffer)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**x - the X coordinate of the specified pixely - the Y coordinate of the specified pixelb - the band to return, which is assumed to be 0data - the DataBuffer containing the image data **Returns:**the specified band containing the sample of the specified pixel. **Throws:** ArrayIndexOutOfBoundException - if the specified coordinates are not in bounds.**See Also:**[setSample(int, int, int, int, DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setSample(int,%20int,%20int,%20int,%20java.awt.image.DataBuffer))

### setSample

public void **setSample**(int x,  
 int y,  
 int b,  
 int s,  
 [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)

Sets a sample in the specified band for the pixel located at (x, y) in the DataBuffer using an int for input. An ArrayIndexOutOfBoundsException is thrown if the coordinates are not in bounds.

**Specified by:**[setSample](http://docs.google.com/java/awt/image/SampleModel.html#setSample(int,%20int,%20int,%20int,%20java.awt.image.DataBuffer)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**x - the X coordinate of the specified pixely - the Y coordinate of the specified pixelb - the band to return, which is assumed to be 0s - the input sample as an intdata - the DataBuffer where image data is stored **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if the coordinates are not in bounds.**See Also:**[getSample(int, int, int, DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getSample(int,%20int,%20int,%20java.awt.image.DataBuffer))

### getDataElements

public [Object](http://docs.google.com/java/lang/Object.html) **getDataElements**(int x,  
 int y,  
 [Object](http://docs.google.com/java/lang/Object.html) obj,  
 [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)

Returns data for a single pixel in a primitive array of type TransferType. For a MultiPixelPackedSampleModel, the array has one element, and the type is the smallest of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT that can hold a single pixel. Generally, obj should be passed in as null, so that the Object is created automatically and is the correct primitive data type.

The following code illustrates transferring data for one pixel from DataBuffer db1, whose storage layout is described by MultiPixelPackedSampleModel mppsm1, to DataBuffer db2, whose storage layout is described by MultiPixelPackedSampleModel mppsm2. The transfer is generally more efficient than using getPixel or setPixel.

MultiPixelPackedSampleModel mppsm1, mppsm2;  
 DataBufferInt db1, db2;  
 mppsm2.setDataElements(x, y, mppsm1.getDataElements(x, y, null,  
 db1), db2);

Using getDataElements or setDataElements to transfer between two DataBuffer/SampleModel pairs is legitimate if the SampleModels have the same number of bands, corresponding bands have the same number of bits per sample, and the TransferTypes are the same.

If obj is not null, it should be a primitive array of type TransferType. Otherwise, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if the coordinates are not in bounds, or if obj is not null and is not large enough to hold the pixel data.

**Specified by:**[getDataElements](http://docs.google.com/java/awt/image/SampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**x - the X coordinate of the specified pixely - the Y coordinate of the specified pixelobj - a primitive array in which to return the pixel data or null.data - the DataBuffer containing the image data. **Returns:**an Object containing data for the specified pixel. **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if obj is not a primitive array of type TransferType or is not null [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if the coordinates are not in bounds, or if obj is not null or not large enough to hold the pixel data**See Also:**[setDataElements(int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer))

### getPixel

public int[] **getPixel**(int x,  
 int y,  
 int[] iArray,  
 [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)

Returns the specified single band pixel in the first element of an int array. ArrayIndexOutOfBoundsException is thrown if the coordinates are not in bounds.

**Overrides:**[getPixel](http://docs.google.com/java/awt/image/SampleModel.html#getPixel(int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**x - the X coordinate of the specified pixely - the Y coordinate of the specified pixeliArray - the array containing the pixel to be returned or nulldata - the DataBuffer where image data is stored **Returns:**an array containing the specified pixel. **Throws:** [ArrayIndexOutOfBoundsException](http://docs.google.com/java/lang/ArrayIndexOutOfBoundsException.html) - if the coordinates are not in bounds**See Also:**[setPixel(int, int, int[], DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#setPixel(int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer))

### setDataElements

public void **setDataElements**(int x,  
 int y,  
 [Object](http://docs.google.com/java/lang/Object.html) obj,  
 [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)

Sets the data for a single pixel in the specified DataBuffer from a primitive array of type TransferType. For a MultiPixelPackedSampleModel, only the first element of the array holds valid data, and the type must be the smallest of DataBuffer.TYPE\_BYTE, DataBuffer.TYPE\_USHORT, or DataBuffer.TYPE\_INT that can hold a single pixel.

The following code illustrates transferring data for one pixel from DataBuffer db1, whose storage layout is described by MultiPixelPackedSampleModel mppsm1, to DataBuffer db2, whose storage layout is described by MultiPixelPackedSampleModel mppsm2. The transfer is generally more efficient than using getPixel or setPixel.

MultiPixelPackedSampleModel mppsm1, mppsm2;  
 DataBufferInt db1, db2;  
 mppsm2.setDataElements(x, y, mppsm1.getDataElements(x, y, null,  
 db1), db2);

Using getDataElements or setDataElements to transfer between two DataBuffer/SampleModel pairs is legitimate if the SampleModel objects have the same number of bands, corresponding bands have the same number of bits per sample, and the TransferTypes are the same.

obj must be a primitive array of type TransferType. Otherwise, a ClassCastException is thrown. An ArrayIndexOutOfBoundsException is thrown if the coordinates are not in bounds, or if obj is not large enough to hold the pixel data.

**Specified by:**[setDataElements](http://docs.google.com/java/awt/image/SampleModel.html#setDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**x - the X coordinate of the pixel locationy - the Y coordinate of the pixel locationobj - a primitive array containing pixel datadata - the DataBuffer containing the image data**See Also:**[getDataElements(int, int, Object, DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getDataElements(int,%20int,%20java.lang.Object,%20java.awt.image.DataBuffer))

### setPixel

public void **setPixel**(int x,  
 int y,  
 int[] iArray,  
 [DataBuffer](http://docs.google.com/java/awt/image/DataBuffer.html) data)

Sets a pixel in the DataBuffer using an int array for input. ArrayIndexOutOfBoundsException is thrown if the coordinates are not in bounds.

**Overrides:**[setPixel](http://docs.google.com/java/awt/image/SampleModel.html#setPixel(int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer)) in class [SampleModel](http://docs.google.com/java/awt/image/SampleModel.html) **Parameters:**x - the X coordinate of the pixel locationy - the Y coordinate of the pixel locationiArray - the input pixel in an int arraydata - the DataBuffer containing the image data**See Also:**[getPixel(int, int, int[], DataBuffer)](http://docs.google.com/java/awt/image/MultiPixelPackedSampleModel.html#getPixel(int,%20int,%20int%5B%5D,%20java.awt.image.DataBuffer))

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) o)

**Description copied from class:** [**Object**](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) Indicates whether some other object is "equal to" this one.

The equals method implements an equivalence relation on non-null object references:

* It is *reflexive*: for any non-null reference value x, x.equals(x) should return true.
* It is *symmetric*: for any non-null reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.
* It is *transitive*: for any non-null reference values x, y, and z, if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.
* It is *consistent*: for any non-null reference values x and y, multiple invocations of x.equals(y) consistently return true or consistently return false, provided no information used in equals comparisons on the objects is modified.
* For any non-null reference value x, x.equals(null) should return false.

The equals method for class Object implements the most discriminating possible equivalence relation on objects; that is, for any non-null reference values x and y, this method returns true if and only if x and y refer to the same object (x == y has the value true).

Note that it is generally necessary to override the hashCode method whenever this method is overridden, so as to maintain the general contract for the hashCode method, which states that equal objects must have equal hash codes.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**o - the reference object with which to compare. **Returns:**true if this object is the same as the obj argument; false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

**Description copied from class:** [**Object**](http://docs.google.com/java/lang/Object.html#hashCode()) Returns a hash code value for the object. This method is supported for the benefit of hashtables such as those provided by java.util.Hashtable.

The general contract of hashCode is:

* Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified. This integer need not remain consistent from one execution of an application to another execution of the same application.
* If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.
* It is *not* required that if two objects are unequal according to the [Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hashtables.

As much as is reasonably practical, the hashCode method defined by class Object does return distinct integers for distinct objects. (This is typically implemented by converting the internal address of the object into an integer, but this implementation technique is not required by the JavaTM programming language.)

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a hash code value for this object.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MultiPixelPackedSampleModel.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/MemoryImageSource.html)   [**NEXT CLASS**](http://docs.google.com/java/awt/image/PackedColorModel.html) | [**FRAMES**](http://docs.google.com/index.html?java/awt/image/MultiPixelPackedSampleModel.html)    [**NO FRAMES**](http://docs.google.com/MultiPixelPackedSampleModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#2s8eyo1) | [METHOD](#26in1rg) |

[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).