Convert a stream of byte sample data into a `BufferedImage`.

To clarify a few things a 4-byte/pixel image is a 32-bit/pixel image (as 4 * 8 is 32), however the actual color components **may** only span 24-bits (3 bytes) as one 8-bit component is left for Alpha (transparency). Given this fact, is it perfectly normal to have a 24-bit image with 8-bits per color channel and no alpha channel (where the image would then be a 3-byte/pixel image).

The reason you are getting the incompatibility exception is that you are using the wrong method to create your WritableRaster.

Given that your input data is in the form of a byte array and that you are trying to create a image where each byte stores not the whole pixel but a **sample of a pixel** the method `createPackedRaster()` is immediately deemed unsuitable as the method treats every data element (i.e. every byte) as its own pixel which definitely isn't what you're trying to accomplish here.

To determine which of the "create" methods you need to use, you need to determine the format of the incoming data.

The three main types of sample encoding is displayed below:

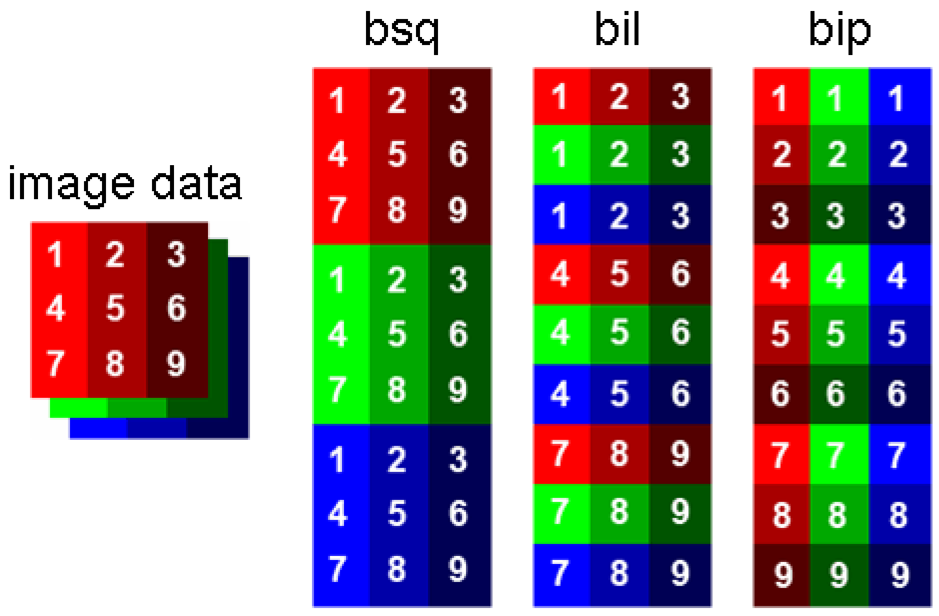

Image Source *[amor.cms.hu-berlin.de]*

They are the Band Sequential Format (bsq), Band Interleaved by Pixel Format (bip) and Band Interleaved by Line Format (bil). The image demonstrates how a 3 by 3 sample image may be encoded. For the sake of simplicity we will only consider the latter two (BIP and BIL) as BSQ is rarely used (*as far as I know*).

If the image data coming through is in the format of BIP, also simply know as pixel interleaved, you will need to use the `createInterleavedRaster()` method to read the input data. As the documentation states:

Creates a Raster based on a PixelInterleavedSampleModel with the specified data type.

Another factor we need to determine is that if the byte data coming through contains alpha, in essence whether the image data is like this (assuming BIP):

R G B R G B ....

or

R G B A R G B A ....

Since there are many overloaded methods taking different parameters we shall use the one that takes a dataType, the width, the height, the bands and a location.

For the dataType we shall use the constant `DataBuffer.TYPE_BYTE` since we are inputting a byte array.

The width and height are supplied by our method parameters, nothing funky here.

The bands corresponds to how many distinct bands of data are there. If your input stream doesn't contain alpha you will have three distinct bands (red, green and blue). If your input data contains alpha your image will have four bands (red, green, blue and alpha).

The statement will look like the following:

```
//Replace '4' with '3' if your image doesn't have alpha
WritableRaster wr = Raster.createInterleavedRaster(DataBuffer.TYPE_BYTE, w, h,
4, null);
```

Next part to worry about is the ColorModel. Since we are no longer using `createPackedRaster()` method we need to change the type of our ColorModel as well to `ComponentColorModel`. As this is getting a bit long, I will just show you what I did (please do read the documentation yourself though).

As follows:

```
ColorSpace sRGB = ColorSpace.getInstance(ColorSpace.CS_sRGB);

//Change the first 'true' to 'false' if you don't have alpha.
ComponentColorModel ccm = new ComponentColorModel(sRGB, true, false,
Transparency.TRANSLUCENT, DataBuffer.TYPE_BYTE);
```

Now, assembling it all together:

```
void fun(byte[] imgBuff,int w,int h) throws IOException{
    WritableRaster wr = Raster.createInterleavedRaster(DataBuffer.TYPE_BYTE, w,
h, 4, null);

    wr.setDataElements(0, 0, w, h, imgBuff);

    ColorSpace sRGB = ColorSpace.getInstance(ColorSpace.CS_sRGB);

    ComponentColorModel ccm = new ComponentColorModel(sRGB, true, false,
Transparency.TRANSLUCENT, DataBuffer.TYPE_BYTE);

    BufferedImage img = new BufferedImage(ccm, wr, false, null);
}
```

You will note I added a line `wr.setDataElements(0, 0, w, h, imgBuff)`. This line effectively does the actual data population into the WritableRaster using the byte array directly (no need to create a DataBufferByte object).

I mentioned earlier I would talk about both BIP and BIL. Changing the method above from BIP to BIL should as simple as swapping the `createInterleavedRaster()` method to `createBandedRaster()` and it *should* (I may be wrong here) work just the same.