FontMetrics Differences between different Java Versions

Overview

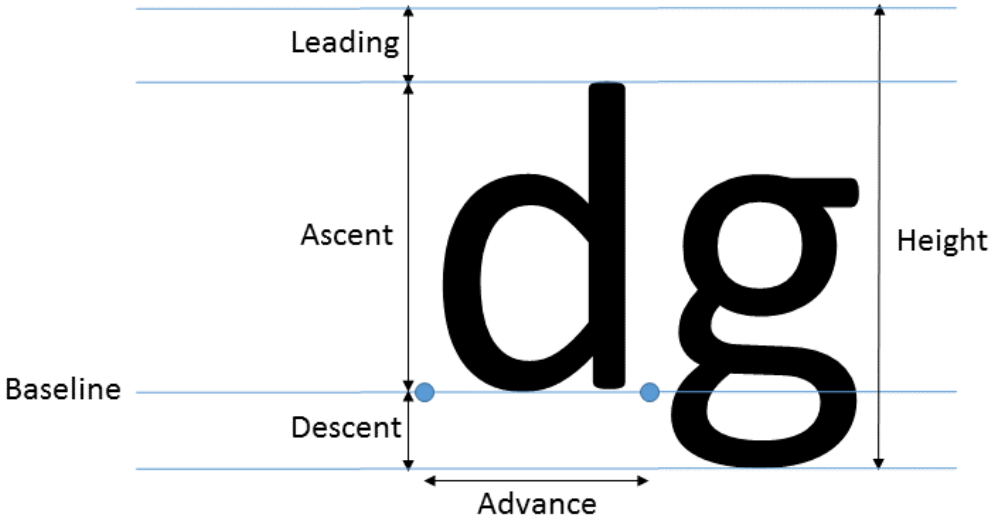
FontMetrics values differ for the same font on the same system, between different Java versions, causing a number of problems in the application for code that depends on these metrics.

This page describes those differences in more detail than was done in the initial investigation.

Terminology Overview

If you are not familiar with font metrics terminology (used throughout this investigation), check the **java.awt.FontMetrics** JavaDoc and this tutorial.

As a quick reference, the diagram below shows some of the measurements corresponding to FontMetrics methods:



A Font also has a size, which is explained in the **java.awt.Font#getSize** JavaDoc:

> `public int getSize()`
>
> Returns the point size of this `Font`, rounded to an integer. Most users are familiar with the idea of using *point size* to specify the size of glyphs in a font. This point size defines a measurement between the baseline of one line to the baseline of the following line in a single spaced text document. The point size is based on *typographic points*, approximately 1/72 of an inch.
>
> The Java(tm)2D API adopts the convention that one point is equivalent to one unit in user coordinates. When using a normalized transform for converting user space coordinates to device space coordinates 72 user space units equal 1 inch in device space. In this case one point is 1/72 of an inch.
>
> **Returns:**
>
> the point size of this Font in 1/72 of an inch units.

Conclusions

1. The position of the baseline appears to have shifted significantly upwards with FreeType for a number of fonts.
2. Rounding errors (either in T2K or FreeType) may be present in some several Font metrics

Investigation Notes

Data generation

A slightly adapted version of the code written in the initial investigation is shown below:

```
package demo;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.util.Arrays;
import java.util.List;

public class FontMetricsDataGenerator {

    private static List<Integer> STYLES = Arrays.asList(Font.PLAIN,
Font.BOLD, Font.ITALIC, Font.BOLD | Font.ITALIC);
    private static final String EXAMPLE_STRING = "The quick brown fox
jumps over the lazy dog";

    public static void main(String[] args) {
        titles();
        List<Font> fontList = getFonts();
        fontList
                .stream()
                .sorted(FontMetricsDataGenerator::comparator)
                .forEach(FontMetricsDataGenerator::reportFontMetrics);
    }

    private static List<Font> getFonts() {
        GraphicsEnvironment ge = GraphicsEnvironment.
getLocalGraphicsEnvironment();
        Font[] fonts = ge.getAllFonts();
        return Arrays.asList(fonts);
    }

    private static void titles() {
        System.err.println("** java.vendor  = " + System.getProperty
("java.vendor"));
        System.err.println("** java.version = " + System.getProperty
("java.version"));
        System.err.println("** java.vm.name = " + System.getProperty
("java.vm.name"));
        System.err.println();
        System.err.println();

        System.err.print("Name");
        System.err.print("\tFamily");
        System.err.print("\tStyle");
        System.err.print("\tSize");
        System.err.print("\tAscent");
        System.err.print("\tDescent");
        System.err.print("\tHeight");
        System.err.print("\tLeading");
        System.err.print("\tMaxAdvance");
        System.err.print("\tMaxAscent");
        System.err.print("\tMaxDescent");
        System.err.print("\tstringWidth");
        System.err.println();
    }

    private static int comparator(Font f1, Font f2) {
```

```
        int rval = f1.getFontName().compareTo(f2.getFontName());
        if (rval != 0) {
            return rval;
        }
        rval = f1.getFamily().compareTo(f2.getFamily());
        if (rval != 0) {
            return rval;
        }
        rval = f1.getName().compareTo(f2.getName());
        if (rval != 0) {
            return rval;
        }
        Integer i1 = f1.getStyle();
        Integer i2 = f1.getStyle();
        return i1.compareTo(i2);
    }

    private static void reportFontMetrics(Font baseFont) {
        BufferedImage img = new BufferedImage(100, 100, BufferedImage.
TYPE_INT_ARGB);
        Graphics2D g2d = img.createGraphics();
        for (int style : STYLES) {
            for (int size = 1; size <= 48; ++size) {
                reportFontMetrics(g2d, baseFont, style, size);
            }
        }
    }

    private static void reportFontMetrics(Graphics2D g2d, Font
baseFont, int style, int size) {
        Font font = new Font(baseFont.getName(), style, size);
        FontMetrics metrics = g2d.getFontMetrics(font);
        System.err.print(font.getName());
        System.err.print("\t" + font.getFamily());
        System.err.print("\t" + font.getStyle());
        System.err.print("\t" + size);
        System.err.print("\t" + metrics.getAscent());
        System.err.print("\t" + metrics.getDescent());
        System.err.print("\t" + metrics.getHeight());
        System.err.print("\t" + metrics.getLeading());
        System.err.print("\t" + metrics.getMaxAdvance());
        System.err.print("\t" + metrics.getMaxAscent());
        System.err.print("\t" + metrics.getMaxDescent());
        System.err.print("\t" + metrics.stringWidth(EXAMPLE_STRING));
        System.err.println();
    }
}
```

This code is independent of Capital and generates CSV output for the various FontMetrics methods with the following fields in each line:

`Font,Style,Size,Height,Ascent,Descent,MaxAdvance,Leading,MaxAscent,MaxDescent`

Date is generated for:

- All fonts on the system, apart from Lucida fonts (which can be added to OpenJDK 11, but caused some difficulties in the comparison here so have been left out for now): There are 460 such fonts on my Windows 7 laptop
- Sizes 1-48
- All four combinations of style: plain, bold, italic, bold_italic

Observations

The data from the program above was compared using a spreadsheet. The **OracleJDK 8** values were subtracted from the **OpenJDK 11** values and an analysis of these differences was done.

> The spreadsheet for this is very large and has not been attached to this page.

> Data below is for all font sizes. Variation is smaller when we restrict the data to size 12 fonts (say).
>
> This suggests that a number of the differences are proportional to the font sizes (although we haven't actually yet checked for proportionality).

Most font combinations have different FontMetrics

The majority of combinations tested have some difference in at least one of the FontMetrics values in the data:

Row Labels	Count of Has Diff?
0	25250
1	63262
Grand Total	**88512**

Generally the diffs are in the stringWidth method with affects the "horizontal" size of the rendered text, but there are also a number of diffs in height related metrics which are the ones currently causing the most difficulty with our applications.

MaxAcscent/MaxDescent and Ascent/Descent are interchangeable in practice

Although the FontMetrics JavaDoc explains the difference between Ascent and MaxAscent there are no examples in the 88512 combinations with OpenJDK 11 where MaxAscent differs from Ascent. The same is true for Descent/MaxDescent.

There are probably fonts, locales or systems where these values differ but for the purposes of the remaining discussion here, these pairs of methods are interchangeable.

Some font baselines appear to have moved up

Ascent diffs are nearly always equal to or **less** than 0: Differences range from 0 to 12 (with only 2 of the combinations having a diff of 1)

Row Labels	Count of Ascent
-17	4
-16	12
-15	16
-14	12
-13	12
-12	12
-10	12
-9	12
-8	96
-7	168
-6	276
-5	928
-4	1718
-3	2860
-2	3604
-1	6013

0	72755
1	2
Grand Total	**88512**

Descent diffs are always equal to or **greater** than 0:

Row Labels	Count of Descent
0	73222
1	5855
2	3507
3	2694
4	1788
5	846
6	224
7	204
8	80
9	12
10	12
11	12
12	16
13	12
14	12
15	12
16	4
Grand Total	**88512**

Height is always **similar** (the same or at most plus or minus 1). Suggesting that the differences in Ascent and Descent nearly cancel out, i. e. that the **baseline of the font has moved "up" a little**:

Row Labels	Count of Height
-1	3949
0	81605
1	2958
Grand Total	**88512**

Leading is always **similar**:

Row Labels	Count of Leading
-1	3061
0	82139
1	3312
Grand Total	**88512**

Ascent + **Descent** is also virtually always **similar**:

Row Labels	Count of Ascent+Descent
-2	8

-1	3286
0	83158
1	2060
Grand Total	**88512**

Lack of variation in **Ascent+Descent** but much larger variation in both **Ascent** and **Descent** adds weight to the theory that the font **baseline** has unexpectedly moved for some fonts.

Example Data

Here are the diffs in font metrics for the Calibri plain font at size 12:

Name	Font Name	Family	Style	Size	Ascent	Descent	Height	Leading
Calibri	Calibri	Calibri	0	12	-2	1	-1	0

Here are OracleJDK 8 and OpenJDK 11 font metrics for this font that result in the diffs above:

JDK	Name	Font Name	Family	Style	Size	Ascent	Descent	Height	Leading
OracleJDK 8	Calibri	Calibri	Calibri	0	12	11	2	16	3
OpenJDK 11	Calibri	Calibri	Calibri	0	12	9	3	15	3

As in the general case, this shows a small change in the height and a larger change in the baseline position.

> This could be a significant finding, pointing out where a FreeType (or T2K) bug is or how we might work around the changes...

MaxAdvance is always only slightly smaller or is equal or greater

This means that spacing between characters will be the same, imperceptively smaller or varying degrees larger.

Here is the variation of differences across all combinations checked:

Row Labels	Count of MaxAdvance
-1	27195
0	36994
1	878
2	1306
3	1037
4	1089
5	1231
6	1174
7	957
8	1311
9	1140
10	994
11	1290
12	1128
13	994
14	1279
15	1100
16	1028
17	1250

18	1025
19	1005
20	885
21	596
22	418
23	358
24	242
25	112
26	113
27	106
28	76
29	71
30	66
31	22
32	19
33	19
34	4
Grand Total	**88512**

Any diffs are likely to be greater for very large font sizes. The size of the diffs here is not necessarily a cause for concern.

We have seen differences in font spacing in our tests but all have so far been acceptable.