Grundlagen zu den Schriftarten beim Programmieren mit Java

Ac 2016-2024

1) Fonts(Schriftarten)

In JAVA wird eine Schriftart (**Font**) durch 3 Eigenschaften definiert:

- Schriftfamilie bzw. **name** (`"MONOSPACED"`, `"SERIF"`, `"SANS_SERIF"`)
- Schriftstil bzw. **style** (`PLAIN`, `BOLD`, `ITALIC`, `BOLD ITALIC`)
- Schrifthöhe bzw. **size** (In Punkt angegeben; Datentyp `int`)

Ein Fontobjekt wird über den Konstruktor Font erzeugt mit

public Font (String name, int style, int size)

Ein Font ist in JAVA immer mit einem Grafikkontext eines Containers zu verknüpfen. Zum Beispiel können das Grafikkontexte von Panels (über paintComponent()), Textfeldern, Comboboxen, Buttons, etc. sein.
Mit **setFont()** kann man den Font in den Grafikkontext eintragen und mit **getFont()** kann der aktuelle Font abgefragt werden .
Außerdem bestehen Zugriffsmöglichkeiten mittels **getFamily()**, **getStyle()** und **getSize()** .

Beispiel: Font ("SERIF", 2, 12) bedeutet: Proportionalschrift; Italic; 12 Punkt hoch

Der Stil (style) ist also auch vom Typ int, und zwar bedeuten:
0 Font.PLAIN (Einfach; das ist der Standard)
1 Font.BOLD (Fett)
2 Font.ITALIC (Schrägschrift)
3 Font.ITALIC+Font.BOLD

In Java wird ein Font z.B. so deklariert:

```
String schriftFamilie = MONOSPACED;
int schriftStil = Font.BOLD;
int schriftHoehe = 13;
Font neueSchriftart = new Font (schriftFamilie, schriftStil, schriftHoehe );
```

Containerbeispiel: `JTextField tfTest = new JTextField();`
Eintragen des Fonts: `tfTest.setFont(neueSchriftart);`

Natürlich kann man auf einem Windows-System unter Java noch andere Schriftfamilien verwenden als die oben angegebenen drei, aber sobald man ein Programm für alle Systeme (auch Mac und Linux) schreiben will, sollte man in Java bei diesen drei Schriftfamilien bleiben, damit der Text auch lesbar bleibt !

Auf einem Windows-Rechner gilt für die Schriftfamilien folgendes:
Monospaced entspricht **Courier New**,
Serif entspricht der Proportionalschrift **Times New Roman,**
SansSerif entspricht der schnörkellosen Schrift **Arial** .

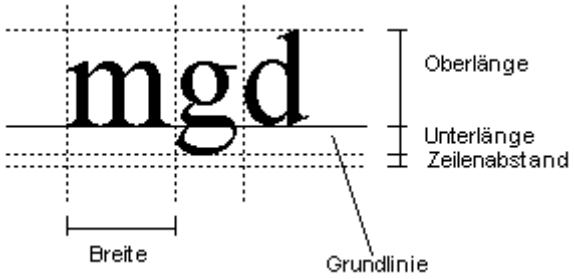
Will man andere Zeichen (z.B. mathematische oder griechische) erzeugen, so bewerkstelligt man das mit Hilfe der sog. **Unicode-Zeichensätze**. Genaueres dazu im Artikel „**unicode-codepoints-surrogates**" .
Statt des üblichen Strings für das betreffende Zeichen gibt man beim Unicode den String mit einem Backslash, gefolgt vom Buchstaben u und anschließendem vierstelligen Hexadezimalcode des Zeichens an.
Beispiel: Mit **"\u2211"** erhält man das große Sigma (∑) , mit **"\u0041"** erhält man „A" .

```
tfTest.setText("\u2211");
tfTest.setText("A");      gleichbedeutend mit    tfTest.setText("\u0041");
```

2) Metriken

Außerdem bietet Java eine Klasse **FontMetrics** an (zur Bestimmung der Größeneigenschaften, z.B. Zeichenbreite, Oberlänge etc.). Dies ist sehr vorteilhaft beim Layout-Design . Als abstrakte Klasse kann FontMetrics nicht instanziiert werden, sondern muss durch die Methode **getFontMetrics** innerhalb des betreffenden Grafikkontextes aufgerufen werden.

Die folgende Grafik zeigt die bei der Verwendung von FontMetrics wichtigen Parameter:



Alle diese Parameter können mittels entsprechender Methoden ermittelt werden , wobei alle Rückgabewerte in Bildschirmpixeln angegeben werden:

int charWidth(char zchn) liefert die Breite eines einzelnen Zeichens
int stringWidth(String s) liefert die Breite eines ganzen Strings
Es gilt dabei: `stringWidth = Textbreite ("advance") + Zwischenraum("padding")`

int getAscent() liefert die Oberlänge des Fonts
int getDescent() liefert die Unterlänge des Fonts
int getHeight() liefert die Höhe
int getLeading() liefert den Zeilenabstand

Es gilt: **Oberlänge + Unterlänge + Zeilenabstand = Höhe**

Definiert und angewandt wird FontMetrics z.B. so:

```
Font fontSys = new Font("SERIF", Font.BOLD, 15);
FontMetrics fMSys = getFontMetrics(fontSys);

System.out.println("Oberlänge = " + fMSys.getAscent());
System.out.println("Unterlänge = " + fMSys.getDescent());
System.out.println("Zeilenabstand = " + fMSys.getLeading());
System.out.println("Höhe = " + fMSys.getHeight());
System.out.println("Oberlänge + Unterlänge + Zeilenabstand = " +
                   fMSys.getAscent() + fMSys.getDescent() + fMSys.getLeading());
```

Vorsicht:
Es kann einzelne Zeichen geben (Sonderfälle), die eine größere Ober- oder Unterlänge haben als es die oben genannten Methoden ermitteln ! Dafür gibt es in JAVA-FontMetrics zusätzliche Methoden !