# Informatik Klasse 13, Foliensatz 5
# Konfigurationsschnittstelle

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

10. September 2009

# Konfiguration

- Attribute von Widgets sind Ausgabetext, Farbe, Größe, ...; alle diese Attribute haben Standardwerte, die verändert werden können
- Veränderung mit dem Konstruktor

widgetclass(master {, attribut=Wert}) => widget

- Abfrage

  `widget.cget(attribut) => Wert`

- Verändern

  `widget.config(`*`attribut=Wert {, attribut=Wert}`*`)`

- Lesen aller Attribute:

  `widget.config()`→`Wörterbuch`

# Experiment

```
from Tkinter import *
root = Tk()
w = Label(root, text="Hello, world!")
w.pack()
wb= w.config()(1)
for key in wb.keys()(2):
 print '%20s' % key(3), '||', wb[key]
#root.mainloop()
```

(1) Erzeugt eine Wörterbuch aller Parameter

(2) Iterator über alle Schlüssel

(3) '%20s' % Zeichenkette – Darstellung als Zeichenkette mit 20 Zeichen rechts ausgerichtet

```
highlightthickness || ('highlightthickness', 'highlightThickness', 'HighlightThickn
              text || ('text', 'text', 'Text', '', ('Hello,', 'world!'))
             image || ('image', 'image', 'Image', '', '')
          compound || ('compound', 'compound', 'Compound', <index object at 0x81e55
            height || ('height', 'height', 'Height', 0, 0)
       borderwidth || ('borderwidth', 'borderWidth', 'BorderWidth', <pixel object a
              pady || ('pady', 'padY', 'Pad', <pixel object at 0x81e5348>, <pixel o
              padx || ('padx', 'padX', 'Pad', <pixel object at 0x81e5360>, <pixel o
              font || ('font', 'font', 'Font', <font object at 0x81e5450>, ('Helvet
  activeforeground || ('activeforeground', 'activeForeground', 'Background', <color
  activebackground || ('activebackground', 'activeBackground', 'Foreground', <borde
         underline || ('underline', 'underline', 'Underline', -1, -1)
             width || ('width', 'width', 'Width', 0, 0)
             state || ('state', 'state', 'State', <index object at 0x81e54c8>, 'nor
    highlightcolor || ('highlightcolor', 'highlightColor', 'HighlightColor', '#0000
      textvariable || ('textvariable', 'textVariable', 'Variable', '', '')
        takefocus || ('takefocus', 'takeFocus', 'TakeFocus', '0', '0')
               bd || ('bd', '-borderwidth')
       foreground || ('foreground', 'foreground', 'Foreground', '#000000', '#10101
               bg || ('bg', '-background')
       background || ('background', 'background', 'Background', '#d9d9d9', '#
               fg || ('fg', '-foreground')
           bitmap || ('bitmap', 'bitmap', 'Bitmap', '', '')
highlightbackground || ('highlightbackground', 'highlightBackground', 'Highlig
disabledforeground || ('disabledforeground', 'disabledForeground', 'DisabledFo
```

# Farbattribute

fg (Vordergrundfarbe), bg (Hintergrundfarbe), higlightcolorbackground, ...

- Datenbasis mit Grundfarben: Red, Green, Blue, Yellow, LightBlue, ... + viele weitere vordefinierte Farben z.B. MistyRose (siehe Systemdatei /etc/X11/rgb.txt)
- Darstellung als RGB-Wert im Format:

  `'#RRGGBB'`

  (R, G, B – Hex-Ziffern für die Farbanteile von rot, grün und blau)

# Experiment

```
from Tkinter import *
class App:
 def __init__(self, master):
    frame = Frame(master)
    frame.pack()
    self.Taste = Button(frame, text="Taste", fg="red",
     command=self.Ausgabe)
    self.Taste.pack()
    self.Eingabe = Entry(frame)
    self.Eingabe.pack()
 def Ausgabe(self):
    x = self.Eingabe.get()
    self.Eingabe.config(bg=x)
root = Tk(); app = App(root);
root.mainloop()
```

- Wie könnte man Vordergrundfarbe der Taste geändern?

# Textfont und und Textausrichtung

- Ein Font ist ein Tupel aus dem Namen der Font-Familie, der Höhe in Punkten und einer optionalen Zeichenkette für Stiel (fett, kursiv etc.). Beispielse:

```
("Times", 10, "bold")
("Helvetica", 10, "bold italic")
("Symbol", 8)
```

- Zeilenumbruch: '\n'
- Ausrichtung (justify): mögliche Werte 'center', 'left', 'right'

# Cursor

Attribut zur Festlegung, wie der Cursor über dem Widget dargestellt wird; Beispielwerte:
X_cursor, arrow, based_arrow_down, based_arrow_up, boat, bogosity, bottom_left_corner, bottom_right_corner, bottom_side, bottom_tee, box_spiral, center_ptr, circle, clock, coffee_mug cross cross_reverse, crosshair, diamond_cross, dot, dotbox, double_arrow, draft_large, draft_small, draped_box, exchange, fleur, gobbler, gumby, hand1, hand2, heart, icon, iron_cross, left_ptr, left_side, left_tee, leftbutton, ll_angle, lr_angle, man, middlebutton, mouse, pencil, pirate, plus, question_arrow, right_ptr, right_side, right_tee, rightbutton, rtl_logo, sailboat, sb_down,_arrow, sb_h_double_arrow, sb_left_arrow, sb_right_arrow, sb_up_arrow, sb_v_double_arrow, shuttle, sizing, spider, spraycan, star, target, tcross, top_left_arrow, top_left_corner, top_right_corner, top_side top_tee, trek ul_angle, umbrella, ur_angle, watch, xterm

# Aufgabe 5.1: Test unterschiedlicher Cursor

Schreiben Sie eine graphische Applikationsklasse mit einem Message-Widget, einem Button-Widget, einem Entry-Widget und einer Methode, die bei Tastendbetätigung ausgeführt wird und den Text aus dem Eingabefeld an das Cursor-Attribute des Frames zuweist.

Hinweise:

- Um ein Attribut des Frames zu ändern muss eine Referenz auf das Frame als Attribute der Klasse gespeichert werden

# Aufgabe 5.2: Aufzeichnen von Attribut-Wörterbücher

- Schreiben Sie in Anlehnung an das erste Experiment ein Programm, dass ein Widget-Objekt erzeugt, das Attribut-Wörterbuch des erzeugten Widgets in lesbarere Form in Textdateien schreibt und sich beendet.
- Erzeugen Sie jeweils eine solche Textdatei für ein Objekt der Klasse Frame, der Klasse Entry und der Klasse Label.
- Vergleichen Sie die Attribut-Wörterbücher.

# Aufgabe 5.3: Testprogramm für Attribute

- Schreiben Sie eine graphische Applikation mit folgender Oberfläche:


- Bei Betätigung der Taste soll für das Widget mit dem angegebenen Bezeichner, dem angegebenen Attribut der angegebene Wert zugeordnet werden.
- Probieren Sie die Wirkung unterschiedlicher Attribute aus.

Hinweise:
Die Konfigurationsanweisung in der mit der Taste zu startenden Methode

```
self.Widget.config(Attribute="Wert")
```

(*kursiv* – zu ersetzende Texte) ist zuerst als Zeichenkette zu erzeugen, testweise im Terminal auszugeben und dann mit dem exec-Kommando auszuführen:

```
s = 'self.' + ...
print s
exec s
```