Informatik Klasse 13, Foliensatz 4
Eingabe und Fensteranordnung

Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

31. August 2009

Widget-Klassen

- Frame: Kontruktionsrahmen zur Gruppierung von Widgets
- Button: Tastenelement zum Start eines Kommandos
- Entry: Texteingabe
- Label: Anzeige einesn einzeiligen Textes oder eines Bildes
- Message: Textanzeige mit Zeilenumbruch

- Canvas: Zeichenfeld, Grundbaustein Graphikeditor
- Checkbutton: binäres Eingabeelement
- Listbox: Anzeige einer Liste von Alternativen
- Menue, Menuebutton: Pull-Up- und Pull-Down-Menues
- Radiobutton, Scale: numerische Eingabe
- Scrollbar: Scrollelemente für Festerausschnitt
- Text: Formatierte Textanzeige, Grundbaustein Texteditor
- Toplevel: Konstruktionrahmen für ein Hauptfenster

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()
  self.Label = Label(frame, text="Eingabe:")
  self.Label.pack()
 def Ausgabe(self):
   x = self.Eingabe.get()
   self.Label.config(text="Eingabe: " + x)
root = Tk(); app = App(root); root.mainloop()
```

mixins (gemeinsame Elternklassen)

Die Widget-Klassen sind Geschwister, die ihre Basismethoden von denselben Elternklassen erben: Anordnung, Konfiguration, Bindung von Funktionen an Ereignisse, ...

- Geometie-Mixins:
  - Pack: Anordnung nach dem Stapelprinzip
  - Grid: Anordnung als 2D-Matrix
  - Place: Anordnung mit expliziter Positionsangabe
- Konfiguration (Festlegung/Veränderung von Attributen)
- Ereignisbindung, ...

Experiment mit »Grid«

```
from Tkinter import *
class App:
 def __init__(self, master):
  frame = Frame(master)
  frame.pack()
  Label(frame, text="First")(1).grid(row=0)(2)
  Label(frame, text="Second").grid(row=1)
  e1 = Entry(frame)
  e2 = Entry(frame)
  e1.grid(row=0, column=1)
  e2.grid(row=1, column=1)

root = Tk()
app = App(root)
root.mainloop()
```

(1) keine Zuweisung erforderlich, da Referenz im Elternfenster

(2) auch ein Aufruf der Form »*Widget.Methode*([*Argumente*])«

Experiment mit »Pack«

- die Methode »pack« kann Widgets nur unter- oder nebeneinander anordnen; 2D-Anordnung erfordert Hilfsrahmen

```
from Tkinter import *
class App:
 def __init__(self, master):
  m_frame = Frame(master)
  m_frame.pack()
  l_frame = Frame(m_frame); l_frame.pack(side="left")
  r_frame = Frame(m_frame); r_frame.pack(side="left")
  Label(l_frame, text="First").pack()
  Label(l_frame, text="Second").pack()
  e1 = Entry(r_frame); e1.pack()
  e2 = Entry(r_frame); e2.pack()

root = Tk()
app = App(root)
root.mainloop()
```

Aufgabe 4.1: Text-Klasse

Schreiben Sie in Anlehnung an Inf12/Foliensatz21 eine Klasse »text« mit den Methoden:

- __str__(self): Rückgabe des Inhalts als Strings)
- append(self, txt): Anhänges des Textes in txt
- undo(self): löschen des zuletzt angehängten Textes
- search(self, txt): Suche, ob »txt« enthalten ist; wenn ja, Rückgabe der Position als String, sonst Rückgabe des Strings "nicht enthalten"

Testen Sie die Klasse und ihrer Methoden.

Hinweis:
Statt eins Konstruktors genügt es im Beispiel, ein Attribut mit einer leeren Zeichenkette zu vereinbaren.

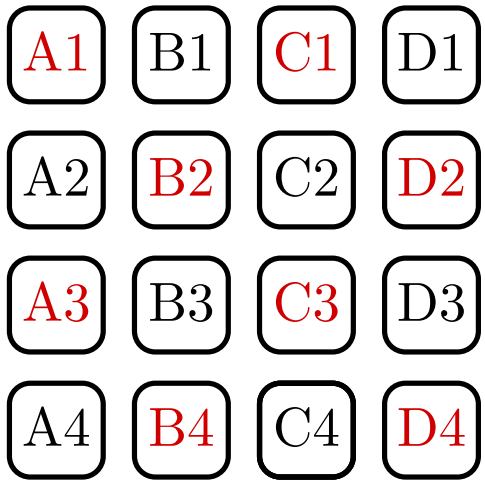
Aufgabe 4.2: GUI-Erweiterung der Text-Klasse mit

Schreiben Sie eine Klasse »App« für eine graphische Anwendung mit »text« als Elternklasse, bei der

- die Methoden »append«, »undo« und »search« mit gleichnamigen Tasten (Button) gestartet werden,
- die Eingabevariable »txt« mit einem Texteingabeelement (Entry) eingegeben wird
- bei jedem Methodenaufruf von »append« und »undo« auch der gesamte Text in einem Message-Widget aktualisiert wird und
- bei »search« das Ergebnis der Suchfunktion in einem weiteren Message-Widget angezeigt wird.

Aufgabe 4.3: Label-Matrix

Schreiben Sie eine Applikationsklasse, die aus Labeln folgende Oberfläche erzeugt, und ein Applikationsprogramm zum Test.



Hinweise:

- Das Label in Zeile 0, Spalte 0 wird mit »Label(frame, text="A1", fg="red")«, das Label in Zeile 0, Spalte 1 wird mit »Label(frame, text="B1")« etc. erzeugen.
- Erzeugung der Labels mit einer Schleife. Anordnung mit der »grid()«-Methode.

Aufgabe 4.4: Label-Matrix mit Farbumschaltung

Erweiterung Sie die Klasse aus der Aufgabe zuvor um eine Methode, die für alle Label die Textfarbe zwischen schwarz und rot umschaltet, und einen Button, der diese Methode aufruft.

Hinweise:

- Der Konstruktor muss zusätzliche eine Liste mit 4×4 Elementen erzeugen und den Listenelementen z.B. mit

  ```
  L[i][j]=Label(...)
  ```

  die Referenzen auf die Labels zuweisen.
- Änderung der Textfarbe mit

  ```
  L[i][j].config(fg="red")
  ```

  bzw.

  ```
  L[i][j].config(fg="black")
  ```