Bölüm 8

Konuşlandırıcılar

Görsel arayüz araçlarının (widgets) taşıyıcılar üzerine yerleştirilmesine *konuşlandırma* (layout) diyeceğiz. Konuşlandırma eylemini yapan özel sınıflar vardır. Onlar da birer widget'tir. Ama işlevlerine uysun diye, onları konuşlandırıcılar (Layout Managers) diye adlandırıyoruz. Konuşlandırıcılar bir kaç tanedir. Widgetlerin taşıyıcılara nasıl yerleştirileceğine bağlı olarak uygun bir konuşlandırıcı seçilmelidir. Çünkü farklı konuşlandırıcılar farklı tür yerleşim yaparlar.

window, pane, tab, dialog,... gibi bazı taşıyıcıların kendi konuşlandırıcıları vardır. Onlar için ayrı konuşlandırıcı kullanmaya gerek yoktur.

Başlıca konuşlandırıcılar şunlardır.

- Pack
- Grid
- Place
-

Bir widget'i konuşlandırıcı ile yerleştirirken daima şu iki işi yapmalıyız:

1. widget'i yarat
2. Widget'i konuşlandırıcıya ekle

8.1 Pack Konuşlandırıcısı

Pack, tkinter'in üç konuşlandırıcısı arasında en basiti sayılır. Pack konuşlandırıcısında, widgetlerin taşıyıcıda nereye konulacağını belirtmek yerine, widgetlerin birbirlerine göre konumları belirtilir. Sonra pack() fonksiyonu gerekeni yapar. Basit ve kolay kullanılır olmasına karşın, konuşlandırma eyleminde Grid ve Place konuşlandırıcılarının yeteneklerine sahip değildir. Ama basit arayüzlerde çekinmeden kullanabileceğimiz bir konuşlandırıcıdır.

8.1.1 Pack Örnekleri

Module 8.1.

```
from tkinter import *

win = Tk()

Label(win, text="Al yazmalım", bg="red", fg="white").pack()
Label(win, text="Yeşil elma", bg="green", fg="black").pack()
Label(win, text="Mavi gök", bg="blue", fg="white").pack()

mainloop()
```

Açıklamalar:

5,6,7.satırlar, win taşıyıcı üzerine konulacak birer label tanımlıyor. `Label().pack()` ifadesi, label'deki *text*'i *win* üzerine bir satır olarak yerleştiriyor. 5.satır *"Al yazmalım"* etiketidir *win* üzerine ortalayıp koyuyor. Onun altına 6.ve 7.satırlar aynı biçimde yerleşiyor.

text içindeki fg (foreground) değeri yazının rengini belirliyor. Zemin koyu ise, *fg="white"* değeri seçilebilir. Bu durmda yazılar zemin üzerine beyaz yazılır.

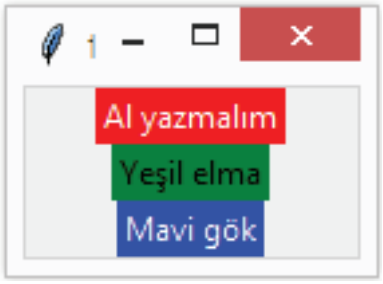
Açılan pencereyi genişletirseniz, etiketlerin win üzerindeki konumlarının, genişliğe göre, hep ortalı kaldığını görebilirsiniz. Benzer olarak, pencerenin yüksekliğini artırırsanız, etiketlerin daima N (North, kuzey) kenarına yanaşık kaldığını görebilirsiniz.

Bu örnekte widgetlerin (etiketler) genişliği, win'in başlığı için gerekli genişlikten küçük kaldığı için, win taşıyıcının genişliğine etki etmezler. Ama her satır için win taşıyıcının yüksekliği artar.

Ekrana gelen arayüzü incelersek, şunları gözlemleriz:

- widgetler yazılış sırasına göre alt alta win üzerine konuşlanırlar.
- Widgetler win taşıyıcının genişliğine göre ortalanırlar.

- Widgetler win taşıyıcının üst kenarına (N, North, kuzey) yanaşık düzende konuşlanırlar.



Şekil 8.1: pack() Konuşlandırıcısı

fill seçeneği

Ekrana gelen arayüze bakarsak, label'lerin uzunlukları farklı olunca, backgroud (bg, zemin rengi) 'un etiket genişliğinde olduğunu, bu genişliğin win taşıyıcının genişliğine eşit olmadığını, kenarlarda gri renkli bölgeler kaldığını görebiliriz. Görsel açıdan böyle olması hoşumuza gitmiyorsa, label'in zemin rengini win taşıyıcının sol (W,West, batı) kenarından sağ (E,East,doğu) kenarına kadar yatay doğrultuda uzatabiliriz. Bunun için fill=X seçeneğini kullanıyoruz.

Module 8.2.

```
1 from tkinter import *

  win = Tk()

  e = Label(root, text="Al yazmalım", bg="red", fg="white")
6 e.pack(fill=X)
  e = Label(root, text="Yeşil elma", bg="green", fg="black")
  e.pack(fill=X)
  e = Label(root, text="Mavi gök", bg="blue", fg="white")
  e.pack(fill=X)
11

  mainloop()
```

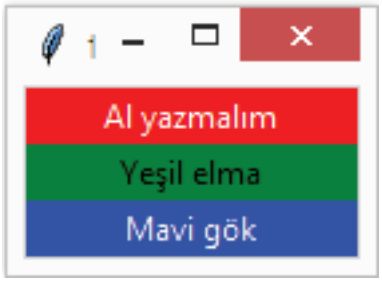
Açıklamalar:

5.satır, Module 8.1'nin 5.satırı ile aynıdır. Tek fark, burada yaratılan label'a e adı verilmiş olmasıdır.

6.satırdaki fill=X değeri, label'ın zemin rengini taşıyıcının E-W kenarlarına kadar uzatıyor. win'in genişliğini artırırsanız, zemin genginin de

beraber arttığını görebilirsiniz. X sabiti yerine Y sabitş kullanılırsa, fill=Y değeri, zemin rengini,satırın düşey doğrutusunda (N-S) uzatacaktır

10.satıra kadar olanlar benzer işi yapıyorlar.



Şekil 8.2: fill=X parametresi

padx, pady

Yatay doğrultuda Yastıklama

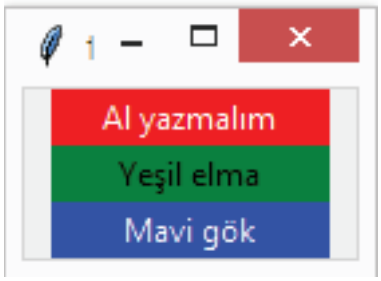
Bazen, widget'in zemin rengini taşıyıcının genişliği kadar yatay doğreultuda uzatmak yerine, alt alta gelen widgetlerin zemin renklerinin genişliklerinin birbirlerine eşit ve düşey doğrultuda hizalı olmasını isteyebiliriz. Böyle yapmak için yastıklama (padding) seçeneği kullanılır.

Module 8.3.

```
from tkinter import *

win = Tk()

e = Label(win, text="Al yazmalım", bg="red", fg="white")
e.pack(fill=X,padx=10)
e = Label(win, text="Yeşil elma", bg="green", fg="black")
e.pack(fill=X,padx=10)
e = Label(win
, text="Mavi gök", bg="blue", fg="white")
e.pack(fill=X,padx=10)

mainloop()
```

Açıklamalar:

Öncekine göre farklı olan 6.satırı açıklamak yetecektir. `pack()` fonksiyonuna `padx =10` parametresi eklenmiştir. Bu parametre, widget'in E-W kenarları ile taşıyıcının E-W kenarları arasına 10 pixel genişliğinde yastıklar koyar.



Şekil 8.3: padx parametresi

Düşey Doğrultuda Yastıklama

padx parametresi yerine pady parametresi kullanılırsa, widget kenarları düşey doğrultu boyunca yastıklanır.

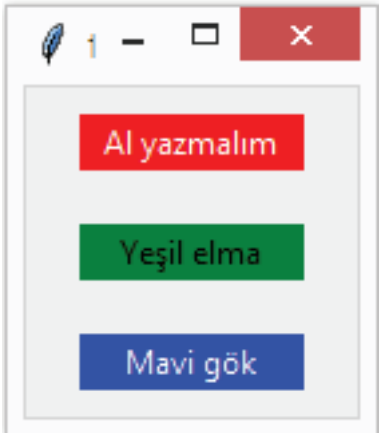
Module 8.4.

```
from tkinter import *

win = Tk()

e = Label(win, text="Al yazmalım", bg="red", fg="white")
e.pack(fill=X,pady=10)
e = Label(win, text="Yeşil elma", bg="green", fg="black")
e.pack(fill=X,pady=10)
e = Label(win
, text="Mavi gök", bg="blue", fg="white")
e.pack(fill=X,pady=10)

mainloop()
```

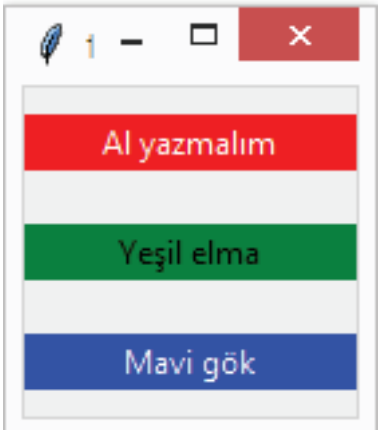
İstenirse, parametreler birlikte kullanılabilir. Module 8.5'de fill, padx ve pady parametreleri birlikte kullanılıyor.

Module 8.5.

```
from tkinter import *

win = Tk()

e = Label(win, text="Al yazmalım", bg="red", fg="white")
e.pack(fill=X, pady=10, padx=20)
e = Label(win, text="Yeşil elma", bg="green", fg="black")
e.pack(fill=X,pady=10, padx=20)
e = Label(win
, text="Mavi gök", bg="blue", fg="white")
e.pack(fill=X,pady=10, padx=20)

mainloop()
```



Şekil 8.4: pady parametresi



Şekil 8.5: Çoklu parametre

Doğal olarak, fill ve padx, pady parametrelerini her widget için aynı olmak yerine farklı değwerlerle de yazabiliriz. Bazı widgetlerden bu parametreleri ekleyebilir ya da kaldırabiliriz.

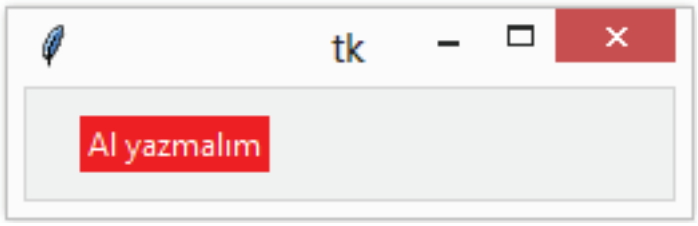
8.2 Widgetleri yan yana koymak

Aksi istenmedikçe, pack() konuşlandırıcısı her kullanılışında widgeti yeni bir satıra koyar. Böylece widgetler alt alta gelir. Bazen widgetlerin yan yana konuşlanmasını isteyebiliriz. Onu yapmak için `side` parametresine LEFT ya da RIGHT sabit değerlerinden birisini atamak yetecektir.

Taşıyıcıya tek widget konuluyorsa, side = LEFT değeri, widgeti taşıyıcının W (West, batı, sol) kenarına yanaştırır. Benzer olarak, side = RIGHT değeri, widgeti taşıyıcının E (East, doğu, sağ) kenarına yanaştırır. Bkz. Module 8.6

Module 8.6.

```
from tkinter import *

win = Tk()

e = Label(win, text="Al yazmalım", bg="red", fg="white")
e.pack(fill=X, pady=10, padx=20, side=LEFT)

mainloop()
```

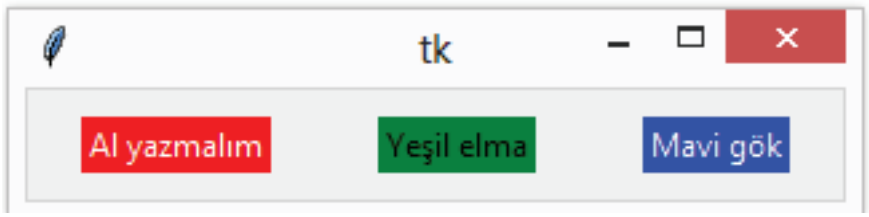


Şekil 8.6: side=LEFT

Taşıyıcıya birden çok widget konuluyorsa, side = LEFT değeri ilk widgeti taşıyıcıya göre, sonraki widgetleri birbirlerine göre yan yana konulandırı. Başka bir değişle, snra gelen widget öncekini kendi soluna alır. Bkz. Module 8.7 ve Şekil 8.7.

Module 8.7.

```
from tkinter import *

win = Tk()

e = Label(win, text="Al yazmalım", bg="red", fg="white")
e.pack(fill=X, pady=10, padx=20, side=LEFT)
e = Label(win, text="Yeşil elma", bg="green", fg="black")
e.pack(fill=X,pady=10, padx=20, side=LEFT)
e = Label(win, text="Mavi gök", bg="blue", fg="white")
e.pack(fill=X,pady=10, padx=20, side=LEFT)

mainloop()
```



Şekil 8.7: Widgetlerin Yan Yana dizilişi

Taşıyıcıya göre, widgetlerin bazısını sola bazısını sağa yanaşık yapabiliriz. Sola gidecekler için side=LEFT, sağa yanaşacaklar için side=RIGHT değerleri verilir.

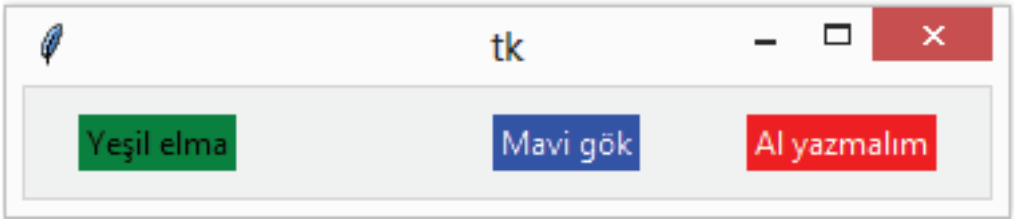
Module 8.8.

```
from tkinter import *

win = Tk()

e = Label(win, text="Al yazmalım", bg="red", fg="white")
e.pack(fill=X, pady=10, padx=20, side=RIGHT)
e = Label(win, text="Yeşil elma", bg="green", fg="black")
e.pack(fill=X,pady=10, padx=20, side=LEFT)
e = Label(win, text="Mavi gök", bg="blue", fg="white")
e.pack(fill=X,pady=10, padx=20, side=RIGHT)

mainloop()
```

8.3 Place Konuşlandırıcısı

Place konuşlaandırıcısı, widget'in konuşlanacağı yeri mutlak ya da göreli olarak belirler. Sözdizimi şöyledir:

Module 8.9.



Şekil 8.8: Widgetlerin Yan Yana dizilişi

```
widget.place( konuşlandırma_seçenekleri )
```

Konuşlandırma_seçeneklerinin başlıcaları şunlardır:

anchor Widgetin, taşıyıcının hangi kenarına yanaşık olacağını belirler. Taşıyıcının kenarları yatay ve düşey doğrultudadır ve kendisi dikdörtgen biçimindedir. Bunun kenarlarını üst, sağ, alt ve sol diye nitelersek, tkinter bu kenarları N (North, kuzey), E (East, doğu), S (South, güney), W (West, batı) sabitleriyle belirler. Yanaşıklık (anchor) kenarlar yerine köşelere doğru da olabilir. Köşeleri belirtmek için NE (North-East, kuzey-doğu), NW (North-West, kuzey-batı), SE (South-East, güney-doğu) ve SW (South-West, Güney-batı) yönlri kullanılır.

bordermode INSIDE ve OUTSIDE sabit değerlerini alabilir. INSIDE öndeğerdir (default). Bu değer, taşıyıcının kenarlarının dikkate alınmamasını sağlar.

height, width Pixel cinsindn widgetin yüksekliğini ve genişliğini belirler.

relheight, relwidth 0.0 dan 1.0 a kadar değişmek üzere, widget'in boyutlarını, taşıyıcının boyutlarına oranlar.

relx,rely Yatay ve düşey offset değerleridir.

Bunları örneklerle inceleyeceğiz.

8.4 Place Örnekleri

Module 8.10.

```
from Tkinter import *
import tkMessageBox
```

```
import Tkinter

top = Tkinter.Tk()

def helloCallBack():
   tkMessageBox.showinfo( "Hello Python", "Hello World")

B = Tkinter.Button(top, text ="Hello", command = helloCallBack)

B.pack()
B.place(bordermode=OUTSIDE, height=100, width=100)
top.mainloop()
```

Window ve Dialog için place() konuşlandırıcısının kullanılması greksiz bir iş olur. Onlar için pack() ya da grid() konuşlandırıcısı daha uygundur. Karmaşık arayüzlerde, widgetlerin nereye konuşlanacağını ve boyutlarını mutlak ya da göreli olarak belirtmek gerektiğinde place() konuşlandırıcısı kullanılmalıdır.

place() konuşlandırıcısını bir w taşıyıcında yaratmak için

Module 8.11.

```
  w.place(relx=0.5, rely=0.5, anchor=CENTER)
  w.place(x=5, y=5, relwidth=1, relheight=1, width=-10, height=-10)
```

deyimlerine benzer bir deyim kullanılır. İlk satırdaki *rlx* ve *rely* parametreleri w taşıyıcısına göredir; yani konuşlanacağı yer ekrana göre değil, w taşıyıcısına göre belirleniyor. İkinci satırda x ve y parametreleri ekrana göre mutlak koordinatlardır.

Module 8.12.

```
import Tkinter as tk
import random

root = tk.Tk()
# width x height + x_offset + y_offset:
root.geometry("170x200+30+30")

languages = ['Python', 'Perl', 'C++', 'Java', 'Tcl/Tk']
labels = range(5)
for i in range(5):
    ct = [random.randrange(256) for x in range(3)]
    brightness = int(round(0.299*ct[0] + 0.587*ct[1] + 0.114*ct[2]))
    ct_hex = "%02x%02x%02x" % tuple(ct)
    bg_colour = '#' + "".join(ct_hex)
    l = tk.Label(root,
                 text=languages[i],
                 fg='White' if brightness < 120 else 'Black',
                 bg=bg_colour)
```

```
    l.place(x = 20, y = 30 + i*30, width=120, height=25)

root.mainloop()
```

8.5 Grid Konuşlandırıcısı

`Grid` (ızgara) konuşlandırıcısı, taşıyıcı üzerine sanki bir ızgara gerilmiş gibi, widgetleri ızgaranın gözelerine yerleştirir. Yerleştirme eyleminde, widget, grid'in istenen gözesine yerleştirilebilir. Grid'in gözelerini bir spreadsheet (Excel, Open Office Calc) hücreleri gibi düşününüz.

Grid'in gözeleri satır ve kolonların kesişiminden oluşur. Satır ve kolonlar 0'dan başlayarak birer artan tamsayılarla numaralanır. İlk satır row= 0, ilk kolon column= 0 dır. (column=0, row=0) gözesi, grid'in sol üstteki ilk gözesidir.

`tkinter` modülünde bir `Grid` nesnesi yaratmak için `grid()` kurucusu kullanılır.

```
ata.grid()
```

deyimi *ata* üzerinde bir *grid* konuşlandırıcısı yaratır ve onu *ata* taşıyıcısının kullanmasını sağlar. Örneğin, Liste 2.1 modülüne *ata* yerine *self* koyarak bu deyimi eklersek, anataşıyıcı üzerinde bir *grid* konuşlandırıcısı yaratmış oluruz.

Örnek olarak, anataşıyıcıya, yukarıda yarattığımız *grid* konuşlandırıcısı ile bir `Entry` widgeti yerleştirmek için

```
        self.entry = tkinter.Entry(self)
        self.entry.grid(column=0,row=0,sticky='EW')
```

deyimlerini kullanabiliriz. Burada ilk satır `Entry` widgeti (nesnesi) yaratıyor, ikinci satır nesneyi grid'in ilk gözesine yerleştiriyor. Bu değerler, Grid'in sol üst köşesindeki ilk gözenin değerleridir. Grid'in her gözesi, satır ve kolon numarası ile belirlenebilir.

Her iki satırdaki self ifadesi taşıyıcıyı (parent) belirtir. Örneğimizde, self, anataşıyıcı olan penceredir. Bu durumda, `Entry` widgeti anataşıyıcı üzerine konulacaktır.

Gözelerin kenarları E (East, doğu), W(West, batı), N (North, kuzey), S (South, güney) sabitleri ile belirlenir.

`grid()` fonksiyonunun *sticky* parametresi, gözenek, kendi içine konulan widget'ten daha büyükse, widget'in gözede bir kenara yanaşık olmasını sağlar. Örnekteki *sticky='EW'* değeri, gözeye konulan text'in doğu ve batı

kenarlarına yanaşık olmasını istiyor. Pencere büyütüldüğünde, gözedeki text doğu-batı yönünde sünecektir. Bu özel bir görünüş oluşturur.

Bu söylediklerimizi Module 2.1'ye uygularsak Module 8.13 biçimini alır.

Module 8.13.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import tkinter

class Basit(tkinter.Tk):
    def __init__(self,parent):
        tkinter.Tk.__init__(self,parent)
        self.parent = parent
        self.initialize()



    def initialize(self):
        self.grid()
        self.entry = tkinter.Entry(self)
        self.entry.grid(column=0,row=0,sticky='EW')

if __name__ == "__main__":
    uygulama = Basit(None)
    uygulama.title('İlk Uygulamam')


    uygulama.mainloop()
```

Açıklamalar:

Module 8.14.

```
from Tkinter import *

colours = ['red','green','orange','white','yellow','blue']

r = 0
for c in colours:
    Label(text=c, relief=RIDGE,width=15).grid(row=r,column=0)
    Entry(bg=c, relief=SUNKEN,width=10).grid(row=r,column=1)
    r = r + 1

mainloop()
```

Açıklamalar: