

XDES02

Programação

Orientada a Objetos

Aula 14: Programação de Interfaces

Tkinter

- Framework para construção de interfaces gráficas
- Baseado no uso de containers e componentes
- Containers podem conter componentes e outros containers
- São implementados por meio de widgets
 - ▣ Button
 - ▣ Entry
 - ▣ Label
 - ▣ Frame
 - ▣ Etc.

Criando uma janela

```
import tkinter as tk

def main():
    # Cria a janela
    janela = tk.Tk()
    janela.title('Primeira janela')

    # Entra no mainloop, o que faz com que
    # a janela seja renderizada na tela
    janela.mainloop()

main()
```

Mostrando texto

Label

```
# tk02.py
# Usando classes para criar interfaces
import tkinter as tk

class GUI:
    def __init__(self):
        # Cria a janela
        self.janela = tk.Tk()
        # Note que a janela é passada como argumento
        self.label = tk.Label(self.janela, text="Hello world")
        # Chama o método pack()
        self.label.pack()
        self.janela.mainloop()

def main():
    GUI()

main()
```

Mostrando texto

Label – Ajustando o tamanho da fonte

```
self.label = tk.Label(self.janela,  
                       text="Hello world",  
                       font=("Arial Bold", 50))
```

Frames



- Tipo de container usado para organizar componentes
- Quando um componente é criado ele deve ser inserido num container
- Frames podem ser adicionados a outros frames ou à janela principal

Organizando widgets com Frames

```
# tk04.py
# Este programa cria labels em dois frames diferentes

import tkinter as tk

class MyGUI:
    def __init__(self):
        # Cria a janela
        self.janela = tk.Tk()

        # Cria dois frames, um para o topo da janela
        # e outro para a base da janela
        self.frameTopo = tk.Frame(self.janela)
        self.frameBase = tk.Frame(self.janela)

        # Cria 3 labels para o frame do topo
        self.label1 = tk.Label(self.frameTopo, text='Um')
        self.label2 = tk.Label(self.frameTopo, text='Dois')
        self.label3 = tk.Label(self.frameTopo, text='Três')
```

Organizando widgets com Frames

```
# Empacota os labels do frame do topo
# Use side='top' para empilhar os labels
self.label11.pack(side='top')
self.label12.pack(side='top')
self.label13.pack(side='top')

# Cria 3 labels para o frame da base
self.label14 = tk.Label(self.frameBase, text='Um')
self.label15 = tk.Label(self.frameBase, text='Dois')
self.label16 = tk.Label(self.frameBase, text='Três')

# Empacota os labels do frame da base
# Use side='left' para colocá-lo horizontalmente
self.label14.pack(side='left')
self.label15.pack(side='left')
self.label16.pack(side='left')
```


Organizando widgets com Frames

```
# Finalmente, empacote os frames
self.frameTopo.pack()
self.frameBase.pack()

# Entra no mainloop
self.janela.mainloop()

def main():
    MyGUI()

main()
```

Eventos

- O tratamento de eventos no Tkinter é feito utilizando funções de callback
- Uma função de callback contém código que é executado quando um evento ocorrer
- No exemplo a seguir vamos associar duas funções de callback a dois botões
 - ▣ Assim, quando o botão for clicado, sua função de callback correspondente será executada

Botões

```
# tk05.py
import tkinter as tk

class MyGUI:
    def __init__(self):
        # Cria a janela
        self.janela = tk.Tk()

        # Criando 2 botões
        self.botao1 = tk.Button(self.janela, \
                                text='Botão 1', \
                                command=self.processaB1)
        self.botao2 = tk.Button(self.janela, \
                                text='Botão 2', \
                                command=self.processaB2)

        self.botao1.pack()
        self.botao2.pack()
```

Botões

```
self.label = tk.Label(self.janela,text="Escolha...")

self.label.pack()

# Entra no mainloop
tk.mainloop()

# Define as funções de callback
def processaB1(self):
    self.label.configure(text="Botão 1 foi clicado")
def processaB2(self):
    self.label.configure(text="Botão 2 foi clicado")

def main():
    MyGUI()

main()
```

Entrada de dados

```
# tk06.py
import tkinter as tk

class GUI:
    def __init__(self):
        self.janela = tk.Tk()
        self.frame1 = tk.Frame(self.janela)
        self.frame2 = tk.Frame(self.janela)
        self.frame1.pack()
        self.frame2.pack()

        self.labelInfo = tk.Label(self.frame1, text="Digite algo:")
        self.labelResult = tk.Label(self.frame2, text="Nada")
        self.labelInfo.pack(side="left")
        self.labelResult.pack(side="left")
```

Entrada de dados

```
self.buttonSubmit = tk.Button(self.janela, text="Enter", \
                                command=self.submit)
self.buttonSubmit.pack(side="left")

self.buttonClear = tk.Button(self.janela, text="Clear", \
                                command=self.clear)
self.buttonClear.pack(side="left")

# Criando o objeto Entry
self.inputText = tk.Entry(self.frame1, width=20)
self.inputText.pack(side="left")

self.janela.mainloop()
```

Entrada de dados

```
# Criando as funções de callback
def submit(self):
    # O texto pode ser recuperado do com o comando get()
    self.labelResult["text"] = self.inputText.get()

def clear(self):
    self.inputText.delete(0, len(self.inputText.get()))
    self.labelResult["text"] = "Nada"

def main():
    GUI()

main()
```

CheckButtons

```
# tk07.py
import tkinter as tk
from tkinter import messagebox

class MyGUI:
    def __init__(self):
        self.janela = tk.Tk()

        # Cria dois frames, um para os checkbuttons
        # e outro para o botão
        self.frameTopo = tk.Frame(self.janela)
        self.frameBase = tk.Frame(self.janela)

        # Cria 3 objetos IntVar para usar com os checkbuttons

        self.cbVar1 = tk.IntVar()
        self.cbVar2 = tk.IntVar()
        self.cbVar3 = tk.IntVar()
```


CheckButtons

```
# Ajusta os objetos IntVar para 0
self.cbVar1.set(0)
self.cbVar2.set(0)
self.cbVar3.set(0)

# Cria os Checkbuttons no frameTopo
self.cb1 = tk.Checkbutton(self.frameTopo, \
                           text='Opção 1', variable=self.cbVar1)
self.cb2 = tk.Checkbutton(self.frameTopo, \
                           text='Opção 2', variable=self.cbVar2)
self.cb3 = tk.Checkbutton(self.frameTopo, \
                           text='Opção 3', variable=self.cbVar3)

# Empacota os Checkbuttons
self.cb1.pack()
self.cb2.pack()
self.cb3.pack()
```

CheckButtons

```
# Cria botões Ok e Finaliza
self.okButton = tk.Button(self.frameBase, \
                           text='OK', command=self.mostraEscolha)
self.finalizaButton = tk.Button(self.frameBase, \
                                 text='Finaliza', command=self.janela.destroy)

# Empacota os botões
self.okButton.pack(side='left')
self.finalizaButton.pack(side='left')

# Empacota os frames
self.frameTopo.pack()
self.frameBase.pack()

# Inicia o mainloop
tk.mainloop()
```

CheckButtons

```
# Função de callback para o okButton
def mostraEscolha(self):
    # cria uma mensagem
    self.message = 'Você selecionou:\n'
    # Verifica quais CheckButtons foram selecionados
    # e monta a mensagem
    if self.cbVar1.get() == 1:
        self.message = self.message + '1\n'
    if self.cbVar2.get() == 1:
        self.message = self.message + '2\n'
    if self.cbVar3.get() == 1:
        self.message = self.message + '3\n'
    # Mostra a mensagem no messagebox
    messagebox.showinfo('Seleção', self.message)
```

```
def main():
    MyGUI()
```

```
main()
```

Exercício

- Considere o programa tk06.py
 - ▣ Implemente um segundo campo Entry e um segundo Label
 - ▣ Quando clicar ok, mostrar os dois Labels contendo o que foi digitado nos dois Entry
 - ▣ Quando limpar, realizar a limpeza nos dois Entry e nos dois Labels