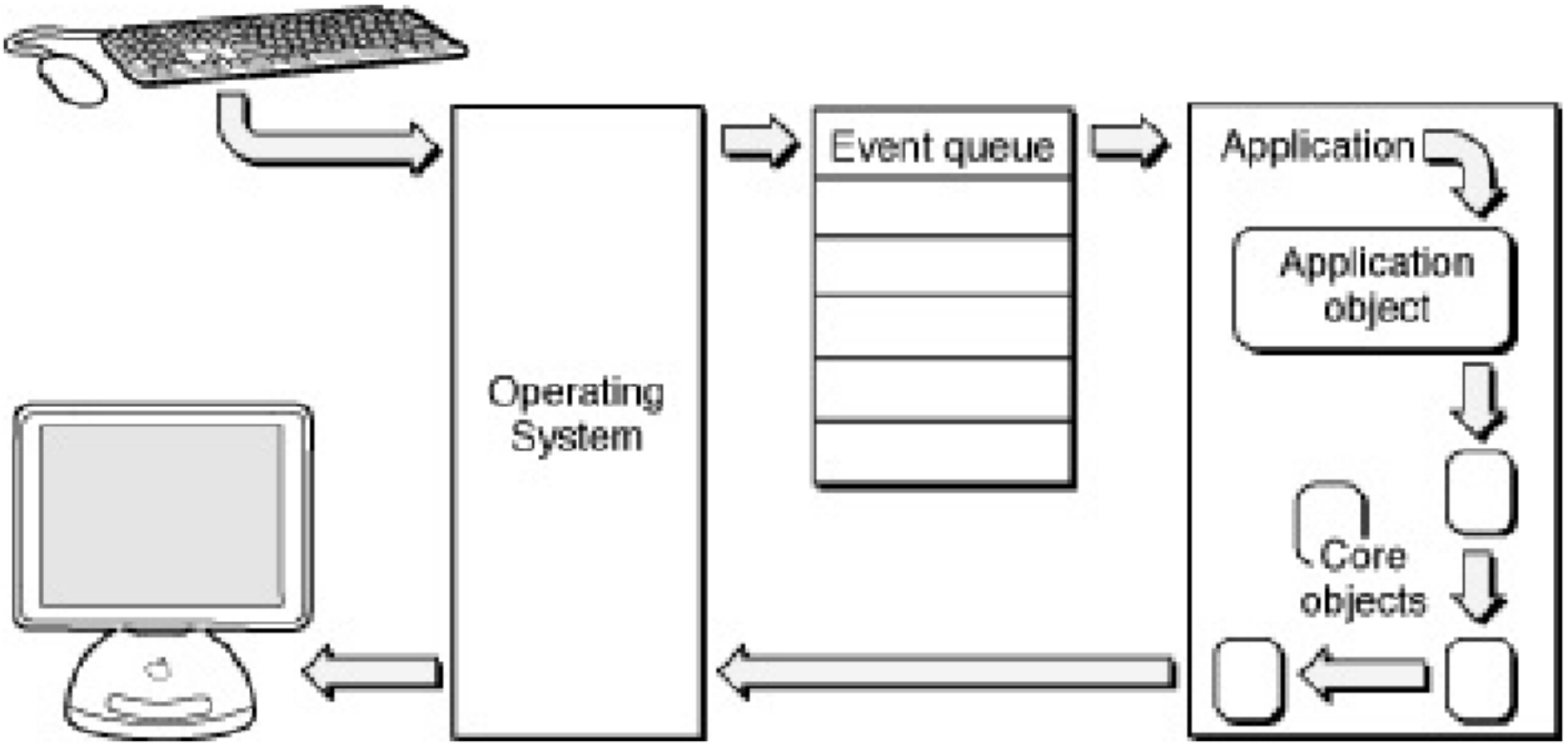
Paradigma Orientata Eveniment

Cursul nr. 5

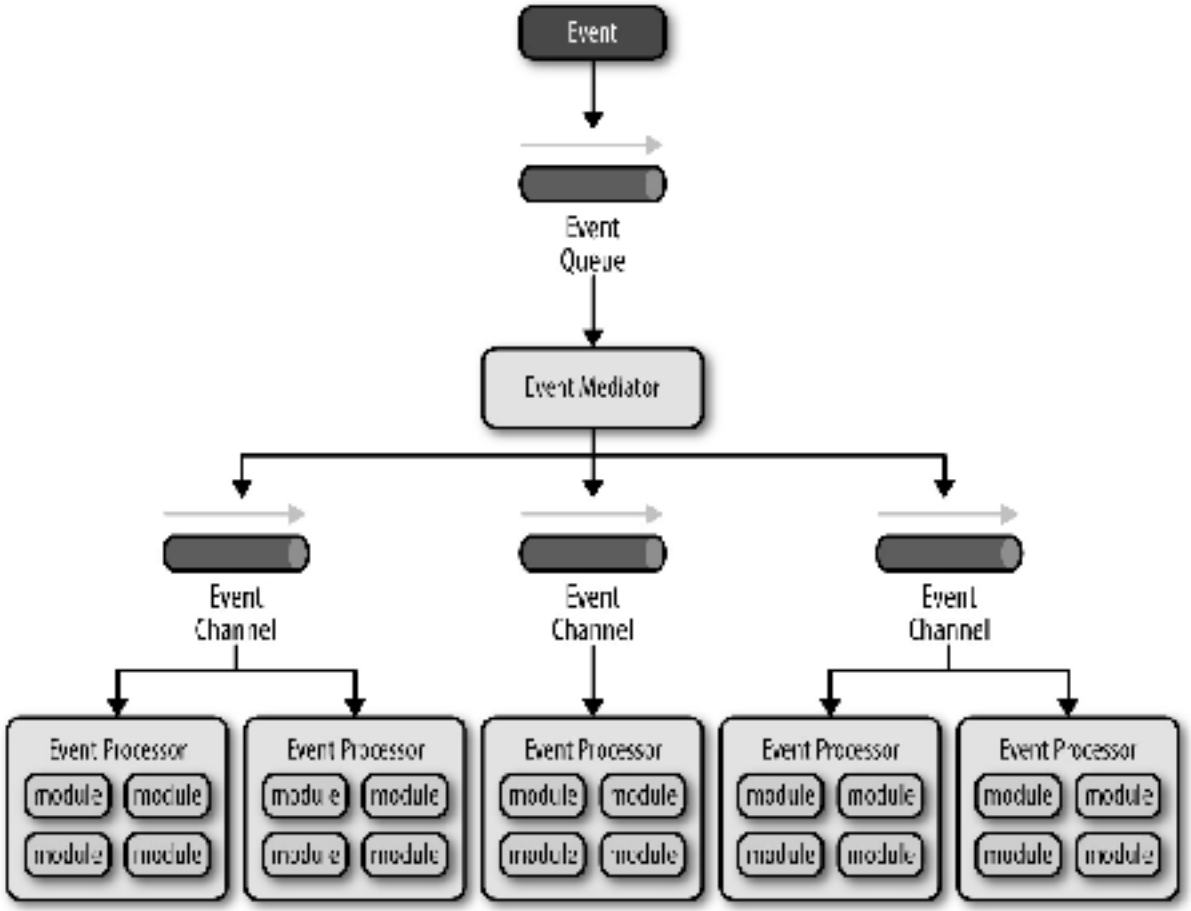
Mihai Zaharia

Ce este un eveniment?

O manieră de gestionare/tratare



Abordarea modernă de proiectare arhitecturală



Tratare evenimente asincrone

Polling

- **Interrupt-driven**

Event-driven

Paradigma orientată eveniment

Metoda 1 - Polling

Interacțiunea este guvernată de o buclă infinită:

```
Loop forever:
{ i=1..n
   read input i
   answer to input i
  inc i }
```

Metoda 2 - Interrupt-driven

1. Activează dispozitivul, apoi
2. Începe procesarea de bază (instalează sistemul de gestiune a evenimentelor/intreruperi)
3. Așteaptă apariția unei întreruperi
4. La apariția unei întreruperi
   1. Salvează starea curentă (schimbare context)
   2. Încarcă și execută metoda de tratare a întreruperii
   3. Restaurează contextul anterior
   4. Go to #1

Metoda #3: Event-driven

Interacțiunea este din nou guvernată de o buclă:

```
main()
{
 ...inițializează structurile de date ale aplicației ...
 ...inițializează și lansează în execuție  GUI....
 // intră în bucla de eveniment
 while(true)
        {
        Event e = get_event();//primește evenimentul
        process_event(e); // tratează evenimentul
        }
}
```

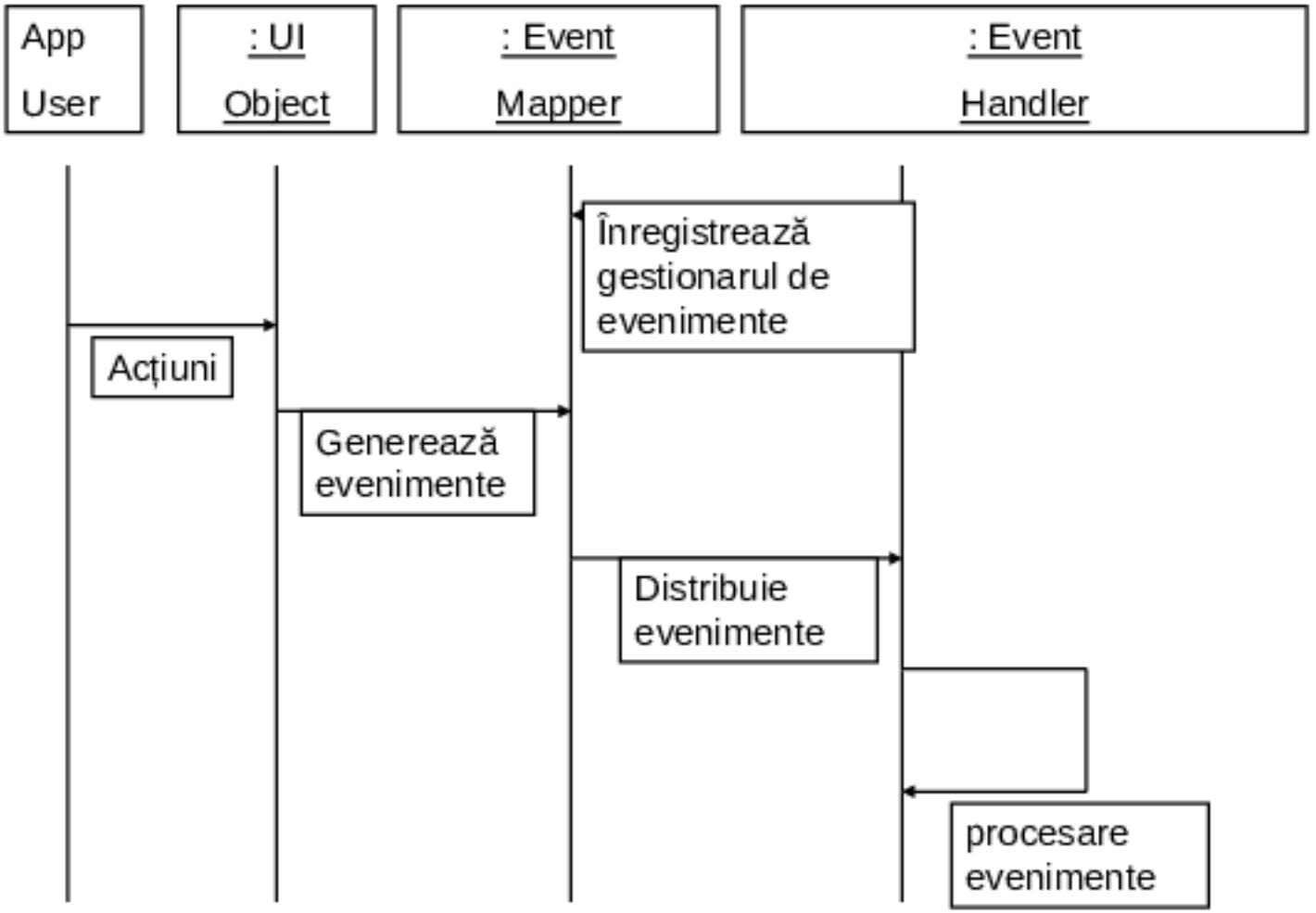
Avantajele procesării orientate eveniment

- Sunt mai portabile
- Permit tratarea mai rapidă
- Se pot folosi în time-slicing)
- Încurajează reutilizarea codului
- se potrivesc cu oop

Componentele unui program simplu bazat pe EDP

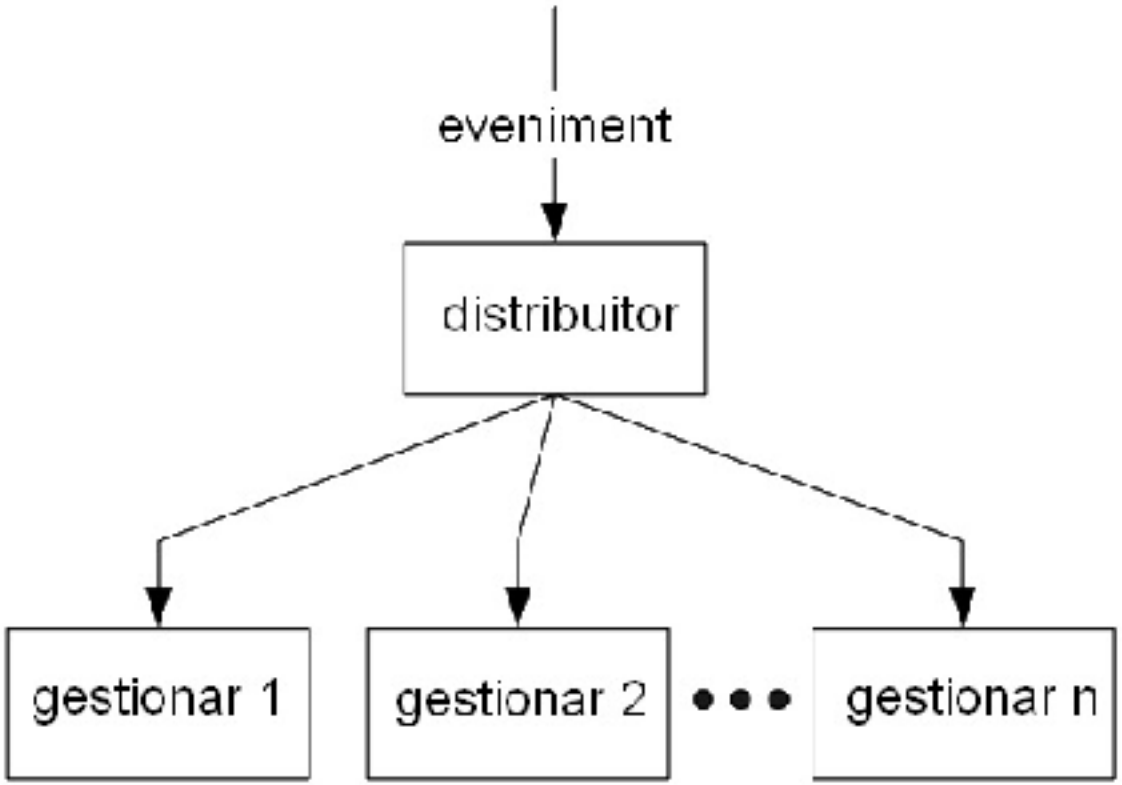
1. Generatoare de evenimente
2. Sursa evenimentelor
3. Bucla de evenimente
4. Gestionari de evenimente
5. Event mapper
6. Înregistrarea evenimentelor

Diagrama de secvență pentru EDP



Tratarea evenimentelor

Dispatcher/Distribuitor/Mapper:



Programarea orientată pe eveniment

EDP – aplicații consolă

```
#include <iostream>
using namespace std;
int value; // globală
int main() { // Inițializare
        char s = '+';
        value = 0;
        while(1) { // buclă tratare eveniment
                cout << "Selectați operația (+,-,q): \n";
                cin >> s; //aștept evenimentul de la utilizator
                switch(s)
                 { //event mapper
                 case '+': //event registration
                         add(); break; //event handler
                 case '-': //înregistrare eveniment
                         sub(); break; //gestionare eveniment
                 case 'q': //înregistrare eveniment
                         exit(1);  //gestionare eveniment
                  }
                }
        return(1);
        }
```

EDP – aplicații consolă

```
// gestionare pentru evenimente
void add ()
{ // gestionar eveniment "+"
int in;
cout << "Introduceți un întreg: \n";
cin >> in;
value += in;
cout << "Valoarea curentă este: " << value << "\n";
}
void sub ()
{ // "-" event handler
int in;
cout << "Introduceți un întreg: \n";
cin >> in;
value -= in;
cout << "Valoarea curentă este: " << value << "\n";
}
```

EDP – aplicații GUI

	Secvențial	EDP
Text	puțin	modest
GUI	inutil	majoritar

Proiectarea unei aplicații EDP - GUI

- Stabilirea interacțiunii vizuale și a evenimentelor: **Look & Feel**

GUI proiectată corect

- O interfață este bună dacă are următoarele caracteristici:
  - **Eleganța**
  - **Îl ghidează** pe looser
  - Oferă **informații ajutătoare**
  - Folosește o **ierarhie** de interfețe
  - Permite ca utilizatorul să facă **greșeli**

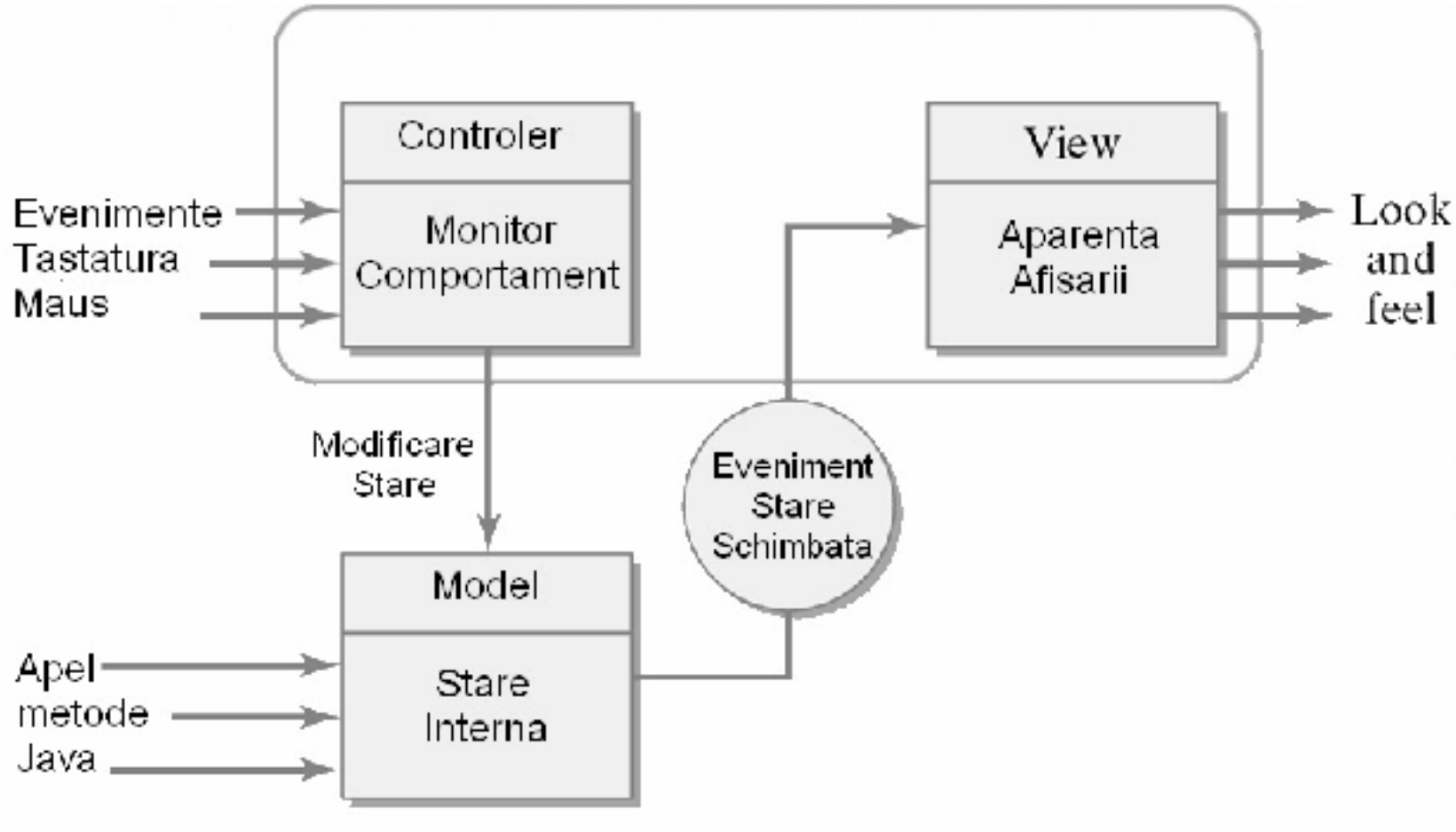
Pick Correlation

- Procesul de selecție a unei ferestre sau aplicații care trebuie sa trateze un eveniment oarecare (deoarece le aparține) se numește corelația de selecție (pick correlation)

Ce sunt widgets?

- Sunt obiectele din cadrul unui GUI orientat obiect.

Model-View-Controller (MVC)



Ce este SDL?

- SDL = Simple DirectMedia Layer
- unde=*www.libsdl.org*
- *alte surse*
  - http://tlahoda.github.io/sdlpp/index.html
  - https://wiki.libsdl.org/
  - https://docs.sdl.com/LiveContent/content/en-US/SDL%20Web-v5

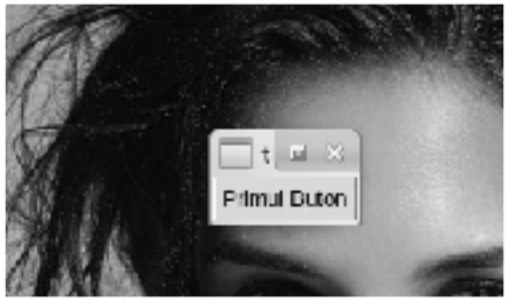
Grafică în Python - TKInter

Primul Buton

```
from tkinter import *
from tkinter import ttk
baza = Tk()
ttk.Button(baza, text="Primul Buton").grid()
baza.mainloop()
```

Pt disperati

instalați pycharm exact ca și intelij
apoi din terminal
sudo apt-get install python3-tk
repornește pycharm
execuția și restul ca la intelij
rezultatul --> pe fruntea fetei

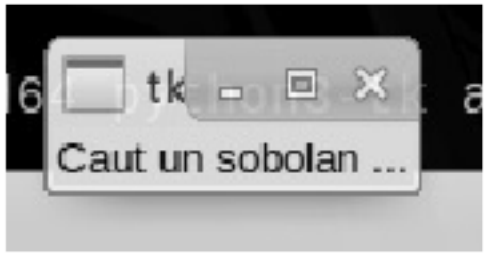


Primul Widget

```
from tkinter import *
from tkinter import ttk

root = Tk()
content = ttk.Frame(root)
button = ttk.Button(content)
root.mainloop()
```

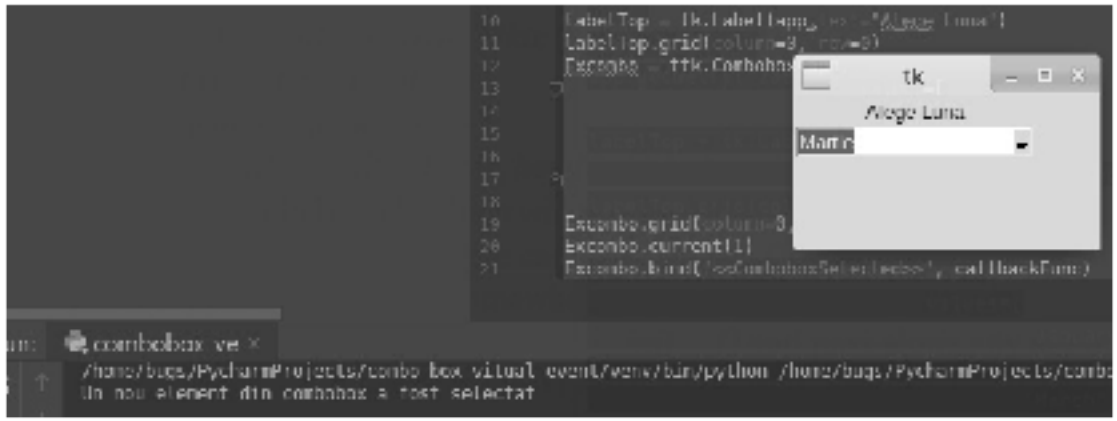
Asocierea unor evenimente

```
from tkinter import *
from tkinter import ttk

root = Tk()

l =ttk.Label(root, text="Caut un sobolan ...")
l.grid()
l.bind('<Enter>', lambda e: l.configure(text='Sobolanul este in interior'))
l.bind('<Leave>', lambda e: l.configure(text='Sobolanul a fugit din zona'))
l.bind('<1>', lambda e: l.configure(text='Sobolanul a miscat din urechea stanga'))
l.bind('<Double-1>', lambda e: l.configure(text='L-am tras de doua ori la rand de
urechi pe sobolan'))
l.bind('<B3-Motion>', lambda e: l.configure(text='Urechea dreapta a fost folosita
pentru o incercare de mutare la %d,%d' %(e.x, e.y)))
root.mainloop()
```



Evenimente virtuale

```
import tkinter as tk
from tkinter import ttk
def callbackFunc(event):
    print("Un nou element din combobox a fost selectat")
app = tk.Tk()
app.geometry('200x100')
labelTop = tk.Label(app,text="Alege Luna")
labelTop.grid(column=0, row=0)
Excombo = ttk.Combobox(app,
                       values=[
                           "Ianuarie",
                           "Februarie",
                           "Martie",
                           "Aprilie"])
Excombo.grid(column=0, row=1)
Excombo.current(1)
Excombo.bind("<<ComboboxSelected>>", callbackFunc)
app.mainloop()
```

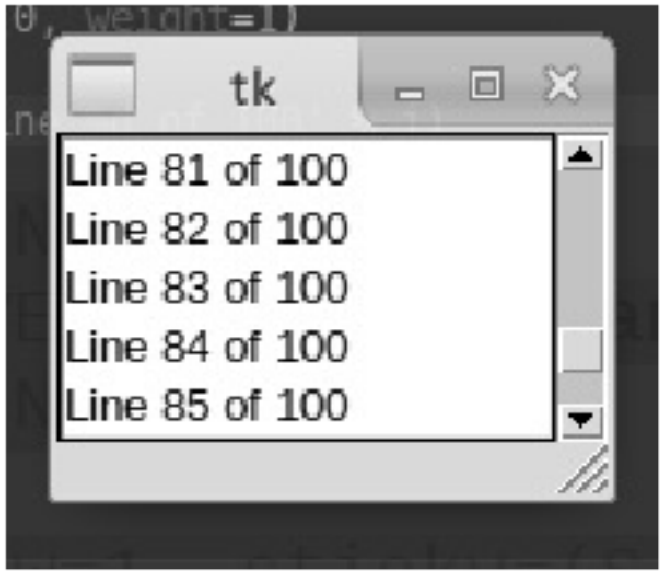


Organizarea matricială a widgets

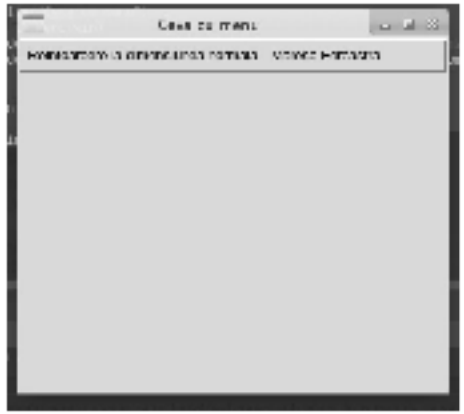
```
from tkinter import *
from tkinter import ttk
root = Tk()
content = ttk.Frame(root)
frame = ttk.Frame(content, borderwidth=5, relief="sunken", width=200,height=100)
namelbl = ttk.Label(content, text="Name")
name = ttk.Entry(content)
onevar = BooleanVar()
twovar = BooleanVar()
threevar = BooleanVar()
onevar.set(True)
twovar.set(False)
threevar.set(True)
one = ttk.Checkbutton(content, text="One", variable=onevar, onvalue=True)
two = ttk.Checkbutton(content, text="Two", variable=twovar, onvalue=True)
three = ttk.Checkbutton(content, text="Three", variable=threevar, onvalue=True)
ok = ttk.Button(content, text="Okay")
cancel = ttk.Button(content, text="Cancel")
content.grid(column=0, row=0)
frame.grid(column=0, row=0, columnspan=3, rowspan=2)
namelbl.grid(column=3, row=0, columnspan=2)
name.grid(column=3, row=1, columnspan=2)
one.grid(column=0, row=3)
two.grid(column=1, row=3)
three.grid(column=2, row=3)
ok.grid(column=3, row=3)
cancel.grid(column=4, row=3)
root.mainloop()
```

O bară pentru derulare

```
from tkinter import *
from tkinter import ttk
root = Tk()
l = Listbox(root, height=5)
l.grid(column=0, row=0, sticky=(N,W,E,S))
s = ttk.Scrollbar(root, orient=VERTICAL, command=l.yview)
s.grid(column=1, row=0, sticky=(N,S))
l['yscrollcommand'] = s.set
ttk.Sizegrip().grid(column=1, row=1, sticky=(S,E))
root.grid_columnconfigure(0, weight=1)
root.grid_rowconfigure(0, weight=1)
for i in range(1,101):
    l.insert('end', 'Line %d of 100' % i)
root.mainloop()
```



Creare Menu-uri simple cu comenzi directe



```
import tkinter as tk

win = tk.Tk()

win.geometry('400x300')

win.title("Ceva cu menu")

menu = tk.Menu(win)

menu.add_command(label='Reintoarcere la dimensiunea normala',
command=lambda: win.geometry('400x300') + win.title("Dimensiune
400x300"))

menu.add_command(label='Maresc Fereastra', command=lambda:
win.geometry('600x600') + win.title("Dimensiune 600x600"))

win.configure(menu=menu)

win.mainloop()
```

Creare Menu-uri ierarhizate

```
from tkinter import *
def donothing():
   filewin = Toplevel(root)
   button = Button(filewin, text="Do nothing button")
   button.pack()
root = Tk()
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)#
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu) ##
editmenu = Menu(menubar, tearoff=0)#
editmenu.add_command(label="Undo", command=donothing)
editmenu.add_separator()
editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
menubar.add_cascade(label="Edit", menu=editmenu) ##
helpmenu = Menu(menubar, tearoff=0)#
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)##
root.config(menu=menubar)
root.mainloop()
```

Creare Ferestre cu Ferestre

```
from tkinter import *
m1 = PanedWindow()
m1.pack(fill = BOTH, expand = 1)

left = Entry(m1, bd = 5)
m1.add(left)

m2 = PanedWindow(m1, orient = VERTICAL)
m1.add(m2)

top = Scale( m2, orient = HORIZONTAL)
m2.add(top)

bottom = Button(m2, text = "OK")
m2.add(bottom)

mainloop()
```

Eveniment la Frame

```
from tkinter import Tk, Label, Button, StringVar
from tkinter import ttk
import tkinter as tk
class UnGUI:
    LABEL_TEXT = [
        "Primul GUI!", "Dzeu cu mila.",
        "chiar merge?...", "...eticheta asta interactiva.",
        "apasa poate merge.", ]
    def __init__(self, master):
        self.master = master
        master.title("Ceva simplu")
        self.label_index = 0
        self.label_text = StringVar()
self.label_text.set(self.LABEL_TEXT[self.label_index])
        self.label = Label(master, textvariable=self.label_text)
        self.label.bind("<Button-1>", self.cycle_label_text)
        self.label.pack()

        self.greet_button = Button(master, text="Fa ceva",
command=self.validateButton)

        self.greet_button.pack()
  self.close_button = Button(master, text="Close",
                                        command=master.quit)
        self.close_button.pack()
    def cycle_label_text(self, event):
        self.label_index += 1
        self.label_index %= len(self.LABEL_TEXT)
self.label_text.set(self.LABEL_TEXT[self.label_index])
    def validateButton(self):
        win = tk.Toplevel()
        win.wm_title("Window")
        l = tk.Label(win, text="Apasa-l")
        l.grid(row=0, column=0)
        b = ttk.Button(win, text="Am inteles!",
                                        command=win.destroy)
        b.grid(row=1, column=0)

root = tk.Tk()
my_gui = UnGUI(root)
root.mainloop()
```

Un exemplu pentru disperați

```
from tkinter import *

from tkinter import messagebox

top = Tk()
top.geometry("300x100")
def hello():
   messagebox.showinfo("Zii", "Zii Ceva")

B1 = Button(top, text = "Zii Inca Ceva", command = hello)
B1.place(x = 35,y = 50)

top.mainloop()
```

O altă versiune cu butoane

```
import tkinter as tk
app = tk.Tk()
labelExample = tk.Button(app, text="0")

def change_label_number():
    counter = int(str(labelExample['text']))
    counter += 1
    labelExample.config(text=str(counter))
buttonExample = tk.Button(app, text="Increase", width=30,
command=change_label_number)
buttonExample.pack()
labelExample.pack()
app.mainloop()
```

UN desen simplu

```
from tkinter import *

top = Tk()

C = Canvas(top, bg = "blue", height = 250, width = 300)

coord = 10, 50, 240, 210
arc = C.create_arc(coord, start = 0, extent = 150, fill = "red")
line = C.create_line(10,10,200,200,fill = 'white')
C.pack()
top.mainloop()
```