Clase 22: Interfaces Gráficas y Ventanas (Clase Tkinter)

Alejandro Hevia

Diciembre 2020

Clases para manejar ventanas (módulo Tkinter)

Ejemplo 1: Programa que muestra una ventana con la palabra “hola”


```
#incluye clases predefinidas
from tkinter import *

#crear ventana (con barra de título)
ventana = Tk()

#crear componente (con texto) y agregar a la ventana
saludo = Label(ventana,text="Hola")

#acomodar componente en ventana
saludo.pack()

#mostrar ventana y esperar cierre (click en botón X)
ventana.mainloop()
```

```
from tkinter import *
```
 # deja disponibles (visibles) clases predefinidas (Tk, Label)

```
ventana = Tk()
```
crea objeto de la clase predefinida Tk que
- inicializa Tkinter (interfaz con sistema operativo)
- representa una ventana en la pantalla (con una barra de título)

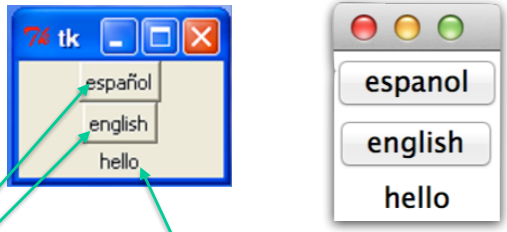

```
saludo = Label(ventana, text="Hola")
```
crea objeto de la clase predefinida **Label** que
- es una componente (*widget*) de la ventana
- contiene el texto "`Hola`"

Un componente es un objeto que puede ubicarse (visualmente) en la ventana.

```
saludo.pack()
```
invoca a método pack del objeto componente que
- acomoda la componente en la ventana para hacerla visible

```
ventana.mainloop()
```
invoca a método mainloop del objeto ventana que
- muestra la ventana en la pantalla
- ejecuta un "loop" de espera de eventos (ej: cerrar la ventana)

Ejemplo 2: Programa para Saludar en inglés y español



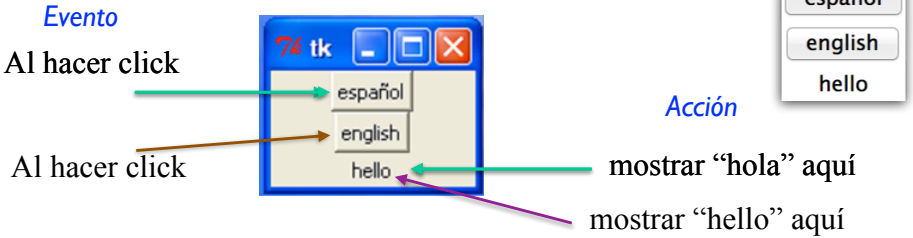
Ventana con 3 componentes:

- 2 botones (objetos de la clase **Button**)
- 1 componente de la clase **Label**

¿Qué queremos que haga?

- click en el primer botón debe mostrar “hola” en Label
- click en el 2° botón debe mostrar “hello” en Label
- Se necesitan 2 métodos, asociados a los 2 botones, que se ejecuten al hacer click con el ratón
- programa termina cuando se cierre ventana: botón externo X (en Windows) o rojo (en Mac).

Cómo se interactúa con la ventana: eventos y acciones



Cómo funciona

Click en 

debe tener asociada *una función* que hace algo

```
def saludar():
    ... # cambia label
```

por ej. cambia el label

Click en 

debe tener asociada *una función* que hace algo

```
def greet():
    ... # cambia label
```

por ej. cambia el label

`label.config(text="abc")` Pone “abc” en label `label`

```
from tkinter import *
#metodos para ejecutar en caso de click en botones
def saludar():
  saludo.config(text="hola")
def greet():
  saludo.config(text="hello")

#ventana
ventana = Tk()
#boton "español". Un click ejecuta método saludar()
boton1 = Button(ventana,text="español",command=saludar)
boton1.pack()  #ubica botón en ventana
#boton "english". Un click ejecuta greet()
boton2 = Button(ventana,text="english",command=greet)
boton2.pack()  #ubica botón debajo de anterior

#componente para mostrar saludo
saludo= Label(ventana)
saludo.pack() #ubica Label debajo
#loop para clicks en botones (hasta cerrar ventana)
ventana.mainloop()
```

Crea botón en ventana

Indica qué función llamar al hacer click

Estructura general de las interfaces en este curso

```
from tkinter import *
```

importar tkinter

```
#metodos para acciones (resptas. a eventos)
def saludar():
   ...
def greet():
   ...
```

Funciones para acciones

```
#ventana
ventana = Tk()
```

Crea ventana (vacía)

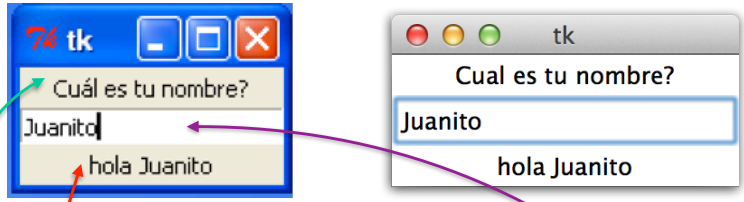
```
#boton …
boton1 = Button(ventana,text="español",command=saludar)
boton1.pack()
#boton …
boton2 = Button(ventana,text="english",command=greet)
boton2.pack()
#label …
saludo= Label(ventana)
saludo.pack()
```

Crea, configura y agrega componentes de ventana

```
#loop para clicks en botones (hasta cerrar ventana)
ventana.mainloop()
```

Espera eventos

Ejemplo 3: Saludador personalizado :-)



Ventana con 3 componentes:

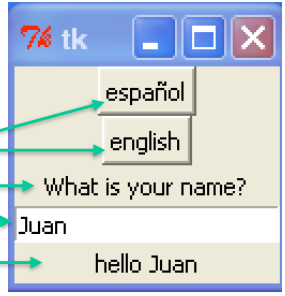
- 1 **Label** con pregunta "Cuál es tu nombre?
- 1 componente para recibir respuesta del usuario (de clase **Entry**)
- 1 **Label** para saludo personalizado

¿Qué queremos que haga?

- después de ingresar nombre se debe mostrar saludo "`hola xxx`"
  - se necesita un método, asociado a la 2a componente, que se ejecute al terminar de ingresar un nombre (caracteres hasta tecla <Return>)
- programa saluda a todas las personas que quieran ;-)
- programa termina cuando se cierre ventana (botón externo X/rojo)

```
from tkinter import *

#metodo que se ejecuta en caso de ingresar un nombre
def saludar(x):
  r=respuesta.get()  #obtener respuesta
  saludo.config(text="hola "+r)

#programa que saluda
ventana = Tk()
pregunta = Label(ventana,text="Cuál es tu nombre?")
pregunta.pack()

#componente para ingresar nombre
respuesta = Entry(ventana)
respuesta.pack()
respuesta.bind("<Return>",saludar)

#componente para mostrar saludo
saludo=Label(ventana)
saludo.pack()
ventana.mainloop()
```

Ejemplo 4: Saludador personalizado bilingüe

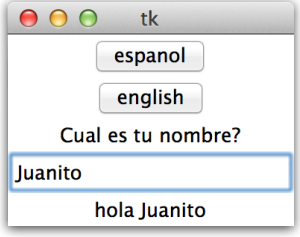


Ventana con 5 componentes:

- 2 botones (para elegir idioma)
- 1 Label para preguntar nombre
- 1 Entry para recibir nombre
- 1 Label para mostrar saludo

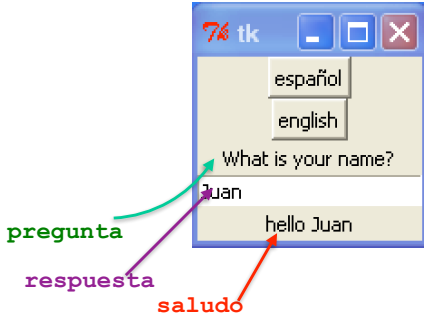
¿Operación?

- seleccionar idioma con botones
- preguntar y saludar en el idioma seleccionado
- programa termina cuando se cierra ventana (botón externo X o rojo en mac)



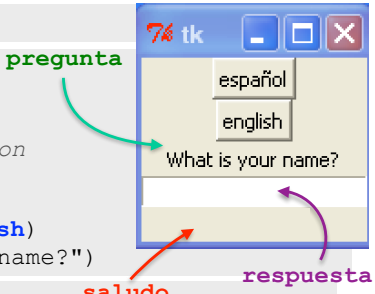
Evento	*Acción*
click en botón “español”	recordar selección, mostrar pregunta
click en botón “english”	recordar selección, mostrar pregunta
presionar return en entry	mostrar saludo

```
#saludador bilingue
ventana=Tk()

#botones
b1=Button(ventana,text="español",command=saludar)
b1.pack()
b2=Button(ventana,text="english",command=greet)
b2.pack()

#otras componentes
pregunta = Label(ventana)
pregunta.pack()
respuesta = Entry(ventana)
respuesta.pack()
saludo=Label(ventana)
saludo.pack()

#loop esperando eventos
ventana.mainloop()
```



```
from tkinter import *
```

```
#metodos para saludar en inglés

# greet: None -> None
# efecto: modifica el texto de etiqueta con
#         pregunta y el saludo
def greet():
    respuesta.bind("<Return>", greetEnglish)
    pregunta.config(text = "What is your name?")
```




```
# greetings: Event -> None
# efecto: modifica el texto de saludo y setea respuesta a ingles
def greetEnglish(event):
    saludo.config(text = "hello " + respuesta.get())
```

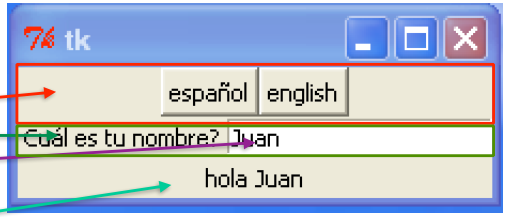
```
#metodos para saludar en español

# saludar: None -> None
# efecto: modifica texto de pregunta y setea respuesta a español
def saludar():
    respuesta.bind("<Return>", saludarEspanol)
    pregunta.config(text = "Cuál es tu nombre?")
```

```
# saludarEnEspañol: Event -> None
# efecto: modifica el texto del saludo en español
def saludarEspanol(event):
    saludo.config(text = "hola " + respuesta.get())
```

Ejemplo 5: Rediseño de ventana del Saludador personalizado bilingüe

Ventana con 3 componentes:

- un "marco" con los botones
- un "marco" con la pregunta y respuesta
- 1 Label para mostrar saludo

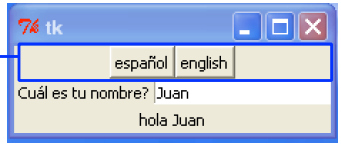


¿"marco"?

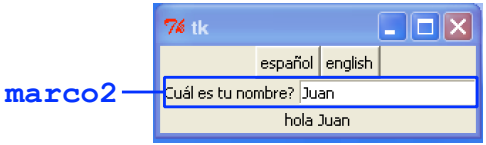
- objeto de clase **Frame**
- objeto para agrupar (poner juntos) componentes

la idea es agrupar componentes que tengan alguna relación lógica

```
from tkinter import *
…
… #funciones ya hechas para saludar en español e inglés
…

#saludador bilingüe
ventana = Tk()

#marco para agrupar botones en la ventana
marco1=Frame(ventana)
marco1.pack()

#botón 1 se ubica a la izquierda del marco1
b1=Button(marco1,text="español",command=saludar)
b1.pack(side=LEFT)

#botón 2 se ubica a continuación
b2=Button(marco1,text="english",command=greet)
b2.pack()
```



```
#marco para agrupar pregunta y respuesta
marco2=Frame(ventana)
marco2.pack()


#ubicar pregunta y respuesta en marco2
pregunta = Label(marco2)
pregunta.pack(side=LEFT)


respuesta = Entry(marco2)
respuesta.pack()

#saludo
saludo=Label(ventana)
saludo.pack()
ventana.mainloop()
```



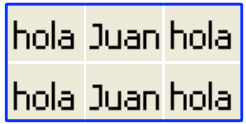
Dato útil: Alternativa a `pack()`

Si se desea organizar muchos widgets en forma de matriz, usar

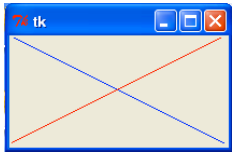
```
comp.grid(row=num, column=num)
```

donde *comp* es una componente como Label, Button, Entry, etc.

```
marco=Frame(ventana)
marco.pack()
e1 = Label(marco)
e1.config(text="hola")
e1.grid(0,0)
e2 = Label(marco)
e2.config(text="Juan")
e2.grid(0,1)
 …
```

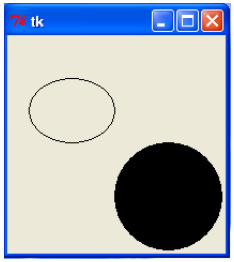


Ejemplo 6: Dibujar diagonales de la ventana (en colores azul y rojo)



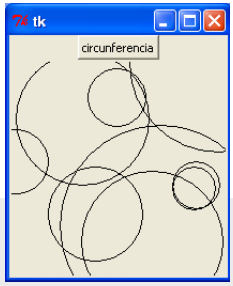

```
from tkinter import *
ventana=Tk()

#Canvas("tela") para dibujar
ancho=200; alto=100 #pixeles
cv=Canvas(ventana,width=ancho,height=alto)
cv.pack()

#dibujar diagonales (azul y roja)
#cv.create_line(h1,v1,h2,v2,fill="color")
cv.create_line(0,0,ancho,alto,fill="blue")
cv.create_line(0,alto,ancho,0,fill="red")

ventana.mainloop()
```

- coordenadas se expresan en la forma horizontal,vertical
- (0,0): coordenadas de *esquina superior izquierda*



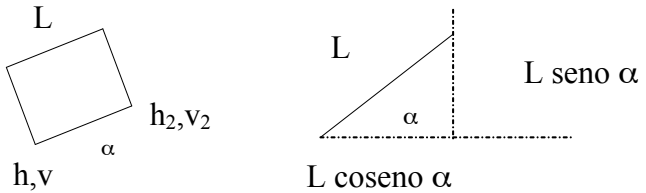

```
from tkinter import *
ventana=Tk()

cv=Canvas(ventana,width=200,height=200)
cv.pack()
cv.create_oval(20,40,100,100)
cv.create_oval(100,100,200,200,fill="black")

ventana.mainloop()
```

Nota: create_rectangle(…) dibuja rectángulo




```
#circunferencias al azar
from tkinter import *
import random
def dibujar():
    x=random.randint(0,ancho)
    y=random.randint(0,alto)
    r=random.randint(1,min(alto,ancho))/2
    cv.create_oval(x-r,y-r,x+r,y+r)

v=Tk()
b=Button(v,text="circunferencia",command=dibujar)
alto=200; ancho=200
cv=Canvas(v,width=ancho,height=alto)
b.pack(); cv.pack()
v.mainloop()
```

Problema: dibujar la siguiente figura (cuadrado girado en α)



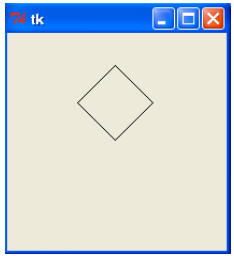
Solución 1:

```
cv=Canvas(…)
h2=h+int(L*math.cos(alfa*math.pi/180))
v2=v-int(L*math.sin(alfa*math.pi/180))
cv.create_line(h,v,h2,v2)    #primera línea
h3=...; v3=...
cv.create_line(h2,v2,h3,v3) #segunda línea
h4=...; v4=...
cv.create_line(h3,v3,h4,v4) #tercera línea
cv.create_line(h4,v4,h,v)    #cuarta línea
```

Solución 2. Usando la idea de la "tortuga" (de Logo).

Supongamos que tenemos una clase **Tortuga** (hecha por nosotros) que dibuja sobre un canvas dado

```
ventana = Tk()
ancho=200; alto=200  #en pixeles
cv=Canvas(ventana,width=200,height=200)
cv.pack()
#ubicar tortuga al centro de cv
t=Tortuga(ancho/2,alto/2,cv)
#girar la tortuga en angulo alfa
t.girar(45)

#repetir 4 veces
for i in range(4):
  #avanzar tortuga en L(dibujando linea)
  t.avanzar(50)
  #girar tortuga en 90°
  t.girar(90)
ventana.mainloop()
```



```
class Tortuga:
  #__init__: int int Canvas -> Tortuga
  #crea tortuga en canvas cv en coords(x,y) y ang. 0
  #ej: t=Tortuga(W/2,H/2,cv)
  def __init__(self,x,y,cv):
    self.h = x       #coordenada horizontal
    self.v = y       #coordenada vertical
    self.cv = cv     #canvas
    self.angulo = 0 #radianes

  #girar: int -> None
  #actualizar angulo: sumar x (en grados) al angulo
  #ej: t.girar(45)
  def girar(self,x):  … # completar

  #avanzar: int -> None
  #dibujar línea de x pixeles y reubicar tortuga
  # ej: t.avanzar(50)
  def avanzar(self,x):  ... # completar
```

Propuesto. Terminar la clase Tortuga y dibujar el cuadrado en la ventana