Міністерствоосвіти і науки України

НТУУ «Київський політехнічний інститут»

Фізико - технічний інститут

# Програмування4

# Лабораторна робота №9

«Побудова об’єктної ієрархії геометричних фігур»

**Виконав:**

Студент 2 курсу

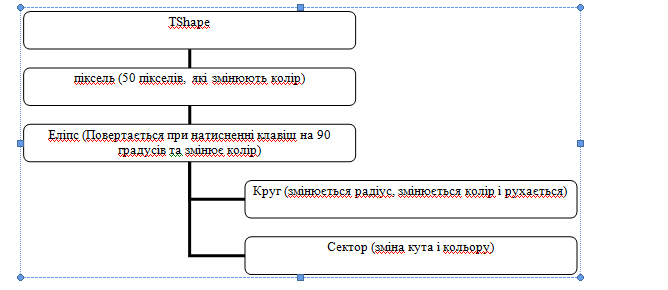
Групи ФЕ-81

Проц Роман

2020

**Мета роботи:** засвоїти базові поняття ООП на прикладі побудови ієрархії геометричних фігур засобами мови програмування Python.

**Завдання.**



Код програми:

from tkinter import \*

from tkinter import messagebox as mb

class TShape:

def \_\_init\_\_(self, canvas, id, on\_click):

self.\_canvas = canvas

self.\_id = id

self.\_canvas.tag\_bind(self.\_id, '<ButtonPress-1>', on\_click)

class Pixel(TShape):

def \_\_init\_\_(self, canvas, x, y, on\_click, color='black'):

super().\_\_init\_\_(canvas, canvas.create\_rectangle(x, y, x + 10, y + 10, fill=color, outline=""), on\_click)

def set\_color(self, color):

self.\_canvas.itemconfig(self.\_id, fill=color)

def get\_color(self):

return self.\_canvas.itemcget(self.\_id, 'fill')

class Ellipse(Pixel):

def \_\_init\_\_(self, canvas, x0, y0, x1, y1, on\_click, color='black'):

TShape.\_\_init\_\_(self, canvas, canvas.create\_oval(x0, y0, x1, y1, fill=color, outline=""), on\_click)

def rotate(self):

(x0, y0, x1, y1) = self.\_canvas.coords(self.\_id)

a = (y1 - y0) / 2

b = (x1 - x0) / 2

self.\_canvas.coords(self.\_id, x0 + b - a, y0 - b + a, x1 - b + a, y1 + b - a)

class Circle(Ellipse):

def \_\_init\_\_(self, canvas, x, y, radius, on\_click, color='black'):

r = radius

TShape.\_\_init\_\_(self, canvas,

canvas.create\_oval(x - r, y - r, x + r, y + r, fill=color, outline=""),

on\_click)

def set\_radius(self, r):

(x0, y0, x1, y1) = self.\_canvas.coords(self.\_id)

x = 0.5 \* (x1 - x0) + x0

y = 0.5 \* (y1 - y0) + y0

self.\_canvas.coords(self.\_id, x - r, y - r, x + r, y + r)

def get\_radius(self):

(x0, y0, x1, y1) = self.\_canvas.coords(self.\_id)

return (x1 - x0) / 2

def move(self, direction, distance):

(x0, y0, x1, y1) = self.\_canvas.coords(self.\_id)

if direction == 'top':

y0 -= distance

y1 -= distance

elif direction == 'right':

x0 += distance

x1 += distance

elif direction == 'left':

x0 -= distance

x1 -= distance

elif direction == 'bot':

y0 += distance

y1 += distance

self.\_canvas.coords(self.\_id, x0, y0, x1, y1)

class Sector(Ellipse):

def \_\_init\_\_(self, canvas, x, y, radius, angle, on\_click, color='black'):

r = radius

TShape.\_\_init\_\_(self, canvas,

canvas.create\_arc(x - r, y - r, x + r, y + r, start=0, extent=angle, fill=color, outline=""),

on\_click)

def set\_angle(self, angle):

self.\_canvas.itemconfig(self.\_id, extent=angle)

def get\_angle(self):

return self.\_canvas.itemcget(self.\_id, 'extent')

class App:

def \_\_init\_\_(self):

self.\_\_root = Tk()

self.\_\_shapes = []

self.\_\_canvas = Canvas(self.\_\_root, width=800, height=300)

self.\_\_root.resizable(False, False)

self.\_\_root.geometry("800x500")

self.\_\_canvas.grid(row=0)

self.\_\_menu\_frame = LabelFrame(self.\_\_root, text="Menu")

self.\_\_factory = self.creation\_menu()

self.\_\_factory.grid(row=1, column=0, sticky=N + S + W)

self.\_\_menu\_frame.grid(row=1, )

def creation\_menu(self):

menu = LabelFrame(self.\_\_root, text="Create")

x = self.\_\_add\_property\_change\_menu('x', menu,

lambda \_: 100,

lambda \_: None, None, 0, False)

y = self.\_\_add\_property\_change\_menu('y', menu,

lambda \_: 100,

lambda \_: None, None, 1, False)

color = self.\_\_add\_property\_change\_menu('color', menu,

lambda \_: 'black',

lambda \_: None, None, 2, False)

def button(name, action, row):

b = Button(menu, text=name)

def on\_click():

try:

action(int(x.get()), int(y.get()), color.get())

except ValueError:

mb.showerror("", 'Bad input')

except TclError:

mb.showerror("", 'Bad color')

b.config(command=lambda: on\_click())

b.grid(row=row, columnspan=100, sticky=N + S + E + W)

button('Pixel', lambda x, y, c: self.add\_pixel(x, y, c), 3)

button('Ellipse',

lambda x, y, c: self.add\_ellipse(x, y, x + 20, y + 40, c),

4)

button('Circle', lambda x, y, c: self.add\_circle(x, y, 40, c), 5)

button('Sector', lambda x, y, c: self.add\_sector(x, y, 40, -70, c), 6)

return menu

def wait(self):

self.\_\_root.mainloop()

def \_\_clear\_menu(self):

# for w in self.\_\_menu:

# w.pack\_forget()

for widget in self.\_\_menu\_frame.winfo\_children():

widget.destroy()

def \_\_add\_property\_change\_menu(self, name, frame, init, action, shape, line, clear\_menu):

if clear\_menu:

self.\_\_clear\_menu()

label\_text = StringVar()

label\_text.set(name)

label = Label(frame, textvariable=label\_text, height=1)

text = StringVar()

text.set(init(shape))

entry = Entry(frame, textvariable=text)

entry.bind('<Return>', lambda \_: action(entry))

label.grid(row=line, column=0, sticky=N + S + E + W)

entry.grid(row=line, column=1, sticky=N + S + E + W)

return entry

def \_\_add\_color\_change\_menu(self, shape, line, clear\_menu=False):

def change\_color(entry):

try:

shape.set\_color(entry.get())

except TclError:

mb.showerror("", 'Bad color')

self.\_\_add\_property\_change\_menu('Color: ',

self.\_\_menu\_frame,

lambda shape\_: shape\_.get\_color(),

change\_color,

shape,

line,

clear\_menu)

def \_\_add\_rotate\_menu(self, shape, line, clear\_menu=False):

if clear\_menu:

self.\_\_clear\_menu()

button = Button(self.\_\_menu\_frame, text="Rotate")

button.config(command=lambda: shape.rotate())

button.grid(row=line, columnspan=100, sticky=N + S + E + W)

def \_\_add\_radius\_change\_menu(self, shape, line, clear\_menu=False):

self.\_\_add\_property\_change\_menu('Radius: ',

self.\_\_menu\_frame,

lambda entry: entry.get\_radius(),

lambda entry: shape.set\_radius(float(entry.get())),

shape,

line,

clear\_menu)

def \_\_add\_move\_menu(self, shape, line, clear\_menu=False):

distance = 20

if clear\_menu:

self.\_\_clear\_menu()

frame = Frame(self.\_\_menu\_frame)

frame.grid(row=line, columnspan=100)

top = Button(frame, text="🡅", command=lambda: shape.move('top', distance), width=23)

top.grid(row=0, column=0, columnspan=100, sticky=N + S + E + W)

left = Button(frame, text="🡄", command=lambda: shape.move('left', distance), width=11)

left.grid(row=1, column=0, sticky=N + S + E + W)

right = Button(frame, text="🡆", command=lambda: shape.move('right', distance), width=11)

right.grid(row=1, column=1, sticky=N + S + E + W)

bottom = Button(frame, text="🡇", command=lambda: shape.move('bot', distance), width=23)

bottom.grid(row=2, column=0, columnspan=100, sticky=N + S + E + W)

def \_\_add\_angle\_change\_menu(self, shape, line, clear\_menu=False):

self.\_\_add\_property\_change\_menu('Angle: ',

self.\_\_menu\_frame,

lambda shape: shape.get\_angle(),

lambda entry: shape.set\_angle(float(entry.get())),

shape,

line,

clear\_menu)

def add\_pixel(self, x, y, color='black'):

pixel = Pixel(self.\_\_canvas, x, y, lambda \_: self.\_\_add\_color\_change\_menu(pixel, 0, True), color)

self.\_\_shapes.append(pixel)

def add\_ellipse(self, x0, y0, x1, y1, color='black'):

def menu(shape):

self.\_\_add\_rotate\_menu(shape, 0, True)

self.\_\_add\_color\_change\_menu(shape, 1)

ellipse = Ellipse(self.\_\_canvas, x0, y0, x1, y1, lambda \_: menu(ellipse), color)

self.\_\_shapes.append(ellipse)

def add\_circle(self, x, y, radius, color='black'):

def menu(shape):

self.\_\_add\_color\_change\_menu(shape, 0, True)

self.\_\_add\_radius\_change\_menu(shape, 1)

self.\_\_add\_move\_menu(shape, 2)

circle = Circle(self.\_\_canvas, x, y, radius, lambda \_: menu(circle), color)

self.\_\_shapes.append(circle)

def add\_sector(self, x, y, radius, angle, color='black'):

def menu(shape):

self.\_\_add\_color\_change\_menu(shape, 0, True)

# self.\_\_add\_radius\_change\_menu(shape, 1)

self.\_\_add\_angle\_change\_menu(shape, 2)

sector = Sector(self.\_\_canvas, x, y, radius, angle, lambda \_: menu(sector), color)

self.\_\_shapes.append(sector)

def main():

app = App()

app.wait()

if \_\_name\_\_ == '\_\_main\_\_':

main()

