**Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут імені Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки  
Кафедра обчислювальної техніки**

**Лабораторна робота №2**

з дисципліни  
«Об’єктно-орієнтоване програмування»

Виконав: Перевірив:

студент групи IM-43 Порєв В. М.

Олексійчук Станіслав Юрійович

номер у списку групи: 22

Київ 2025

***Варіант завдання та основні вимоги***

Варіант 22:

1. Для усіх варіантів завдань необхідно дотримуватися вимог та положень, викладених вище у порядку виконання роботи та методичних рекомендаціях.

2. У звіті повинна бути схема успадкування класів – діаграма класів

3. Для вибору типу об’єкта в графічному редакторі Lab2 повинно бути меню "Об’єкти" з чотирма підпунктами. Меню "Об’єкти" повинно бути праворуч меню "Файл" та ліворуч меню "Довідка". Підпункти меню "Об’єкти" містять назви українською мовою геометричних форм – так, як наведено вище у порядку виконання роботи та методичних рекомендаціях.

Геометричні форми згідно варіанту завдання.

4. Для вибору варіанту використовується Ж – номер студента в журналі.

5. Масив вказівників для динамічних об’єктів типу Shape

- статичний масив ShapeObjectsEditor.\_\_shapes = [None] \* 122

причому, кількість елементів масиву вказівників як для статичного, так і

динамічного має бути N = Ж+100.

Динамічний масив обирають студенти, у яких варіант (Ж mod 3 = 0).

Решта студентів – статичний масив. Позначка mod означає залишок від

ділення.

6. "Гумовий" слід при вводі об’єктів

- суцільна лінія синього кольору для варіантів (Ж mod 4 = 2)

7. Чотири геометричні форми (крапка, лінія, прямокутник, еліпс)

можуть мати наступні різновиди вводу та відображення.

7.1. Прямокутник

Увід прямокутника:

- по двом протилежним кутам для варіантів (Ж mod 2 = 0)

Відображення прямокутника:

- чорний контур з кольоровим заповненням для (Ж mod 5 = 1 або 2)

Кольори заповнення прямокутника:

- помаранчевий для (Ж mod 6 = 4)

7.2. Еліпс

Увід еліпсу:

- від центру до одного з кутів охоплюючого прямокутника для варіантів

(Ж mod 2 = 0)

Відображення еліпсу:

- чорний контур еліпсу без заповнення для (Ж mod 5 = 0 або 2)

8. Позначка поточного типу об’єкту, що вводиться

- в меню (метод OnInitMenuPopup) для варіантів (Ж mod 2 = 0)

***Текст програми***

Лабораторна робота виконана мовою програмування Python з використанням бібліотеки Tkinter. Це завдання складається з головного файлу та трьох модулів:  
1) main.py

import tkinter as tk

from tkinter import messagebox

from shape\_objects\_editor import ShapeObjectsEditor

class Main:

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

self.main\_window = None

self.canvas = None

self.editor\_manager = None

self.current\_mode = None

self.menu\_bar = None

def run(self):

self.\_create\_window()

self.\_create\_canvas()

self.\_create\_editor()

self.\_create\_menu()

self.\_bind\_events()

self.\_set\_default\_editor()

self.main\_window.mainloop()

def \_create\_window(self):

self.main\_window = tk.Tk()

self.main\_window.geometry("600x400+400+150")

self.main\_window.title("Lab2")

self.main\_window.resizable(False, False)

def \_create\_canvas(self):

self.canvas = tk.Canvas(self.main\_window, width=600, height=400, bg="white")

self.canvas.pack()

def \_create\_editor(self):

self.editor\_manager = ShapeObjectsEditor(self.canvas, self.main\_window)

def \_create\_menu(self):

self.current\_mode = tk.StringVar(value="Point")

self.menu\_bar = tk.Menu(self.main\_window)

self.\_create\_file\_menu()

self.\_create\_objects\_menu()

self.\_create\_help\_menu()

self.main\_window.config(menu=self.menu\_bar)

def \_create\_file\_menu(self):

file\_menu = tk.Menu(self.menu\_bar, tearoff=0)

file\_menu.add\_command(label="Exit", command=self.main\_window.quit)

self.menu\_bar.add\_cascade(label="File", menu=file\_menu)

def \_create\_objects\_menu(self):

objects\_menu = tk.Menu(self.menu\_bar, tearoff=0)

objects\_menu.add\_radiobutton(

label="Крапка",

command=self.\_set\_point\_editor,

variable=self.current\_mode,

value="Point"

)

objects\_menu.add\_radiobutton(

label="Лінія",

command=self.\_set\_line\_editor,

variable=self.current\_mode,

value="Line"

)

objects\_menu.add\_radiobutton(

label="Прямокутник",

command=self.\_set\_rect\_editor,

variable=self.current\_mode,

value="Rectangle"

)

objects\_menu.add\_radiobutton(

label="Еліпс",

command=self.\_set\_ellipse\_editor,

variable=self.current\_mode,

value="Ellipse"

)

self.menu\_bar.add\_cascade(label="Objects", menu=objects\_menu)

def \_create\_help\_menu(self):

help\_menu = tk.Menu(self.menu\_bar, tearoff=0)

help\_menu.add\_command(label="About", command=self.\_show\_lab\_information)

self.menu\_bar.add\_cascade(label="Help", menu=help\_menu)

def \_set\_point\_editor(self):

self.editor\_manager.start\_point\_editor()

self.current\_mode.set("Point")

self.editor\_manager.on\_menu\_popup()

self.main\_window.title("Lab2 - Point Mode")

def \_set\_line\_editor(self):

self.editor\_manager.start\_line\_editor()

self.current\_mode.set("Line")

self.editor\_manager.on\_menu\_popup()

self.main\_window.title("Lab2 - Line Mode")

def \_set\_rect\_editor(self):

self.editor\_manager.start\_rect\_editor()

self.current\_mode.set("Rectangle")

self.editor\_manager.on\_menu\_popup()

self.main\_window.title("Lab2 - Rectangle Mode")

def \_set\_ellipse\_editor(self):

self.editor\_manager.start\_ellipse\_editor()

self.current\_mode.set("Ellipse")

self.editor\_manager.on\_menu\_popup()

self.main\_window.title("Lab2 - Ellipse Mode")

def \_show\_lab\_information(self):

messagebox.showinfo("Info", "Lab 2 is working fine\n(c) Copyright 2025")

def \_bind\_events(self):

self.canvas.bind("<Button-1>", self.\_on\_click)

self.canvas.bind("<B1-Motion>", self.\_on\_drag)

self.canvas.bind("<ButtonRelease-1>", self.\_on\_drop)

def \_on\_click(self, event):

self.editor\_manager.on\_click(event.x, event.y)

def \_on\_drag(self, event):

self.editor\_manager.on\_drag(event.x, event.y)

def \_on\_drop(self, event):

self.editor\_manager.on\_drop(event.x, event.y)

def \_set\_default\_editor(self):

self.\_set\_point\_editor()

if \_\_name\_\_ == "\_\_main\_\_":

app = Main()

app.run()

2) shape.py

from abc import ABC, abstractmethod

class Shape(ABC):

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

self.\_x1 = 0

self.\_y1 = 0

self.\_x2 = 0

self.\_y2 = 0

def set(self, x1, y1, x2, y2):

self.\_x1 = x1

self.\_y1 = y1

self.\_x2 = x2

self.\_y2 = y2

@abstractmethod

def show(self, canvas):

pass

class PointShape(Shape):

def show(self, canvas):

canvas.create\_oval(self.\_x1-1, self.\_y1-1,

self.\_x1+1, self.\_y1+1,

fill="black", outline="black")

class LineShape(Shape):

def show(self, canvas):

canvas.create\_line(self.\_x1, self.\_y1,

self.\_x2, self.\_y2,

fill="black", width=1)

class RectShape(Shape):

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

super().\_\_init\_\_()

self.\_\_fill\_color = "orange"

def show(self, canvas):

canvas.create\_rectangle(self.\_x1, self.\_y1,

self.\_x2, self.\_y2,

outline="black", fill=self.\_\_fill\_color, width=1)

class EllipseShape(Shape):

def show(self, canvas):

center\_x = self.\_x1

center\_y = self.\_y1

current\_x = self.\_x2

current\_y = self.\_y2

x1 = 2 \* center\_x - current\_x

y1 = 2 \* center\_y - current\_y

x2 = current\_x

y2 = current\_y

canvas.create\_oval(x1, y1, x2, y2, outline="black", width=1)

3) editor.py

from abc import ABC, abstractmethod

from shape import PointShape, LineShape, RectShape, EllipseShape

class Editor(ABC):

@abstractmethod

def on\_click(self, x, y): pass

@abstractmethod

def on\_drag(self, x, y): pass

@abstractmethod

def on\_drop(self, x, y): pass

class ShapeEditor(Editor):

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

self.\_canvas = canvas

self.\_\_current\_shape = None

self.\_\_preview\_id = None

self.\_\_start\_x = 0

self.\_\_start\_y = 0

def on\_click(self, x, y):

self.\_\_start\_x = x

self.\_\_start\_y = y

def on\_drag(self, x, y):

if self.\_\_current\_shape and self.\_\_preview\_id:

self.\_canvas.delete(self.\_\_preview\_id)

self.\_\_current\_shape.set(self.\_\_start\_x, self.\_\_start\_y, x, y)

self.\_\_preview\_id = self.\_draw\_preview()

def on\_drop(self, x, y):

if self.\_\_current\_shape:

self.\_\_current\_shape.set(self.\_\_start\_x, self.\_\_start\_y, x, y)

if self.\_\_preview\_id:

self.\_canvas.delete(self.\_\_preview\_id)

shape = self.\_\_current\_shape

self.\_\_current\_shape = None

self.\_\_preview\_id = None

return shape

return None

def on\_menu\_popup(self):

print(f"Active editor: {self.get\_menu\_label()}")

def \_get\_start\_x(self):

return self.\_\_start\_x

def \_get\_start\_y(self):

return self.\_\_start\_y

def \_set\_current\_shape(self, shape):

self.\_\_current\_shape = shape

def \_get\_current\_shape(self):

return self.\_\_current\_shape

def \_set\_preview\_id(self, preview\_id):

self.\_\_preview\_id = preview\_id

def \_get\_preview\_id(self):

return self.\_\_preview\_id

@abstractmethod

def get\_menu\_label(self):

pass

@abstractmethod

def \_draw\_preview(self):

pass

class PointEditor(ShapeEditor):

def on\_click(self, x, y):

super().on\_click(x, y)

self.\_set\_current\_shape(PointShape())

self.\_get\_current\_shape().set(x, y, x, y)

if self.\_get\_preview\_id():

self.\_canvas.delete(self.\_get\_preview\_id())

self.\_set\_preview\_id(self.\_draw\_preview())

def get\_menu\_label(self):

return "Point"

def \_draw\_preview(self):

return self.\_canvas.create\_oval(

self.\_get\_start\_x()-2, self.\_get\_start\_y()-2,

self.\_get\_start\_x()+2, self.\_get\_start\_y()+2,

outline="blue", width=1

)

class LineEditor(ShapeEditor):

def on\_click(self, x, y):

super().on\_click(x, y)

self.\_set\_current\_shape(LineShape())

self.\_get\_current\_shape().set(x, y, x, y)

if self.\_get\_preview\_id():

self.\_canvas.delete(self.\_get\_preview\_id())

self.\_set\_preview\_id(self.\_draw\_preview())

def get\_menu\_label(self):

return "Line"

def \_draw\_preview(self):

current\_shape = self.\_get\_current\_shape()

return self.\_canvas.create\_line(

self.\_get\_start\_x(), self.\_get\_start\_y(),

current\_shape.\_x2, current\_shape.\_y2,

fill="blue", width=1

)

class RectEditor(ShapeEditor):

def on\_click(self, x, y):

super().on\_click(x, y)

self.\_set\_current\_shape(RectShape())

self.\_get\_current\_shape().set(x, y, x, y)

if self.\_get\_preview\_id():

self.\_canvas.delete(self.\_get\_preview\_id())

self.\_set\_preview\_id(self.\_draw\_preview())

def get\_menu\_label(self):

return "Rectangle"

def \_draw\_preview(self):

current\_shape = self.\_get\_current\_shape()

return self.\_canvas.create\_rectangle(

self.\_get\_start\_x(), self.\_get\_start\_y(),

current\_shape.\_x2, current\_shape.\_y2,

outline="blue", width=1

)

class EllipseEditor(ShapeEditor):

def on\_click(self, x, y):

super().on\_click(x, y)

self.\_set\_current\_shape(EllipseShape())

self.\_get\_current\_shape().set(x, y, x, y)

if self.\_get\_preview\_id():

self.\_canvas.delete(self.\_get\_preview\_id())

self.\_set\_preview\_id(self.\_draw\_preview())

def get\_menu\_label(self):

return "Ellipse"

def \_draw\_preview(self):

center\_x = self.\_get\_start\_x()

center\_y = self.\_get\_start\_y()

current\_shape = self.\_get\_current\_shape()

current\_x = current\_shape.\_x2

current\_y = current\_shape.\_y2

x1 = 2 \* center\_x - current\_x

y1 = 2 \* center\_y - current\_y

x2 = current\_x

y2 = current\_y

return self.\_canvas.create\_oval(

x1, y1, x2, y2,

outline="blue", width=1

)

4) shape\_objects\_editor.py

from editor import PointEditor, LineEditor, RectEditor, EllipseEditor

class ShapeObjectsEditor:

def \_\_init\_\_(self, canvas, main\_window=None):

self.\_canvas = canvas

self.\_main = main\_window

self.\_\_shapes = [None] \* 122

self.\_\_shapes\_count = 0

self.\_\_current\_editor = None

def start\_point\_editor(self):

self.\_\_current\_editor = PointEditor(self.\_canvas)

def start\_line\_editor(self):

self.\_\_current\_editor = LineEditor(self.\_canvas)

def start\_rect\_editor(self):

self.\_\_current\_editor = RectEditor(self.\_canvas)

def start\_ellipse\_editor(self):

self.\_\_current\_editor = EllipseEditor(self.\_canvas)

def on\_click(self, x, y):

if self.\_\_current\_editor:

self.\_\_current\_editor.on\_click(x, y)

def on\_drop(self, x, y):

if self.\_\_current\_editor:

shape = self.\_\_current\_editor.on\_drop(x, y)

if shape and self.\_\_shapes\_count < len(self.\_\_shapes) - 1:

self.\_\_shapes[self.\_\_shapes\_count] = shape

self.\_\_shapes\_count += 1

self.\_redraw()

return None

def on\_drag(self, x, y):

if self.\_\_current\_editor:

self.\_\_current\_editor.on\_drag(x, y)

def \_redraw(self):

self.\_canvas.delete("all")

for i in range(self.\_\_shapes\_count):

if self.\_\_shapes[i]:

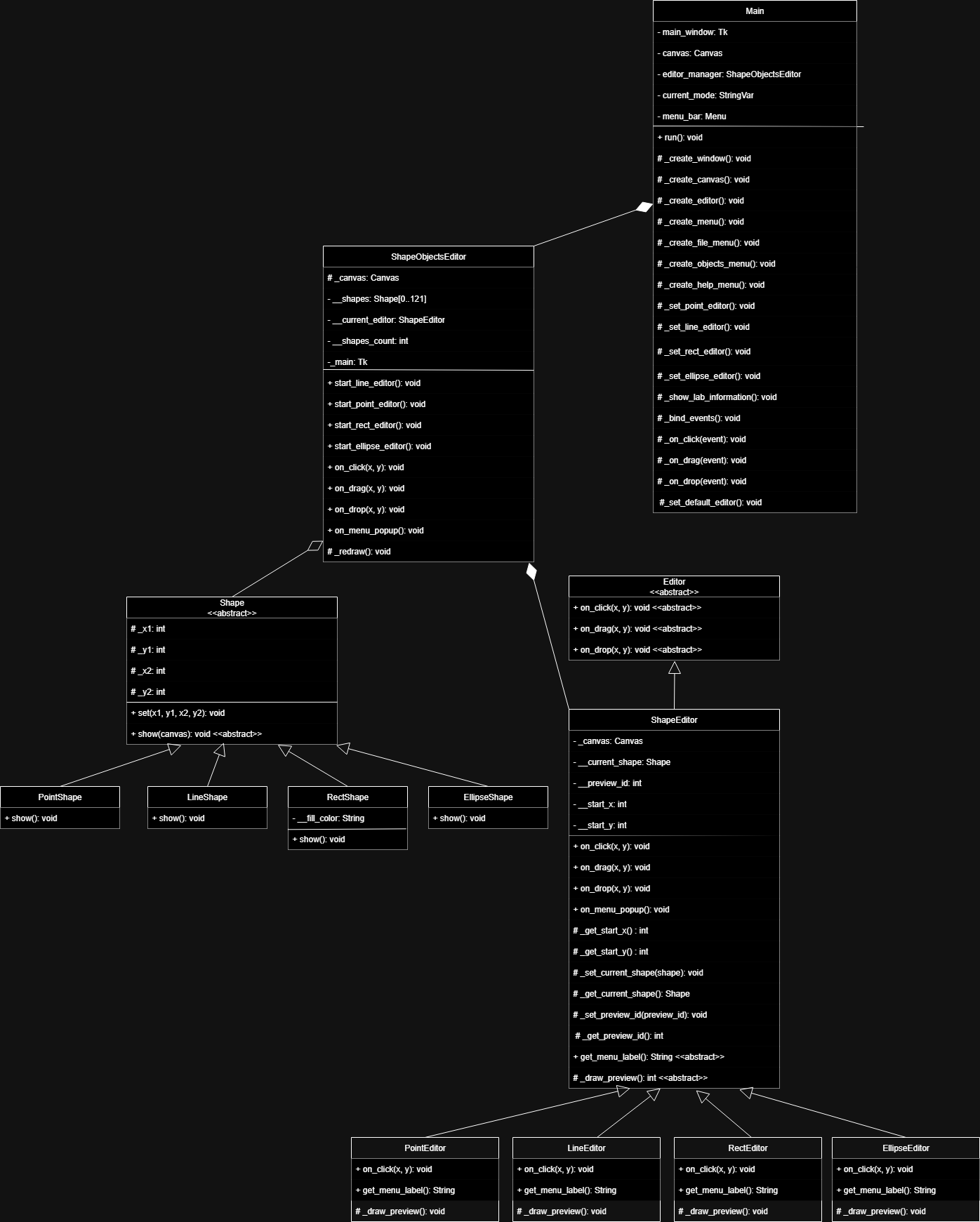
self.\_\_shapes[i].show(self.\_canvas)

def on\_menu\_popup(self):

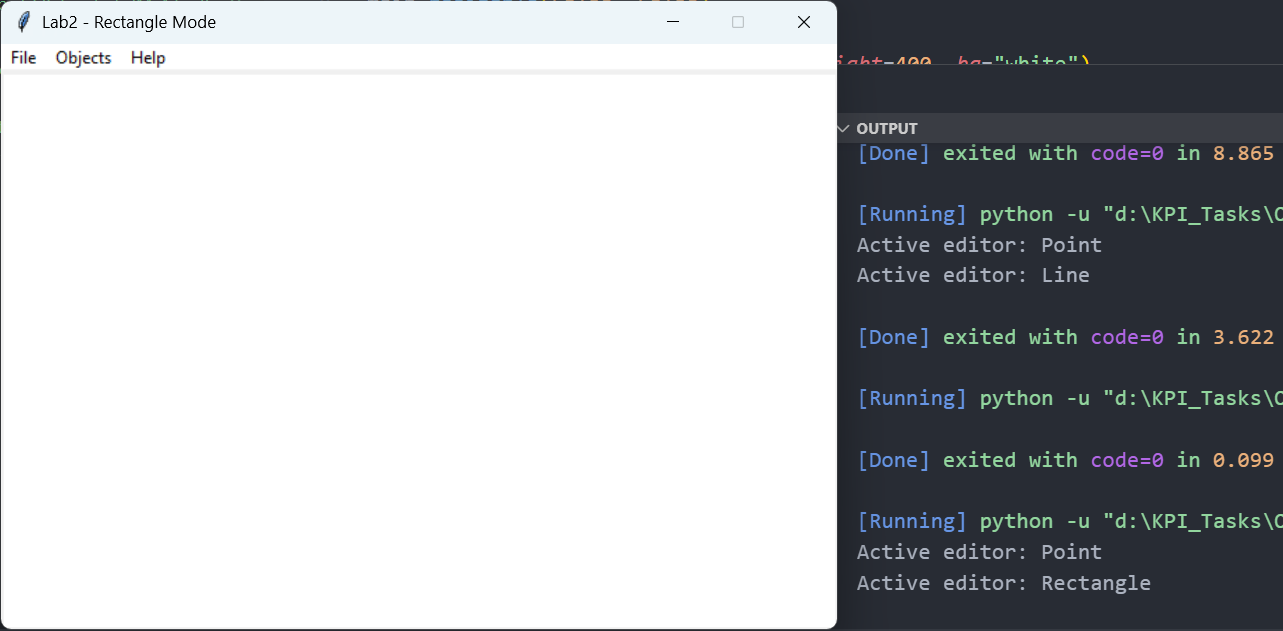
if self.\_\_current\_editor:

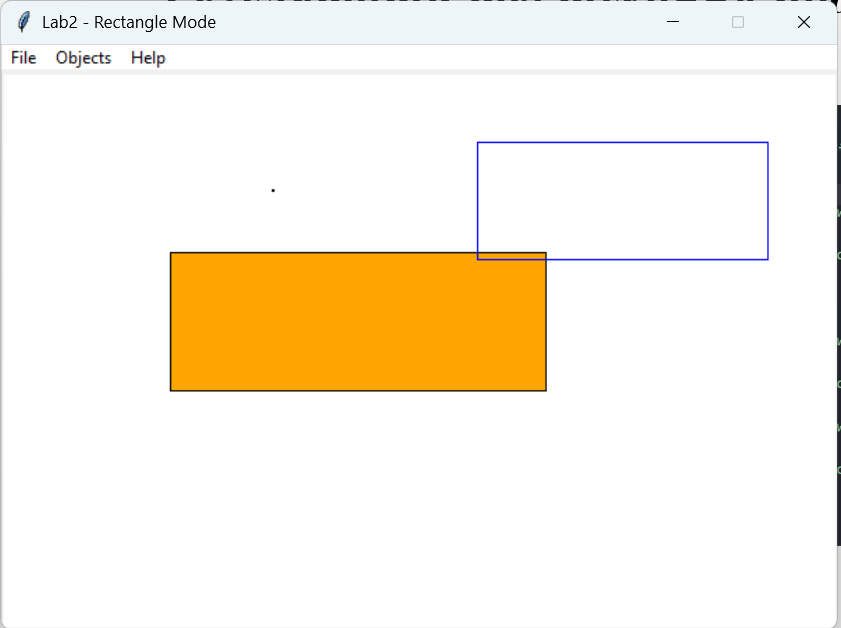
self.\_\_current\_editor.on\_menu\_popup()

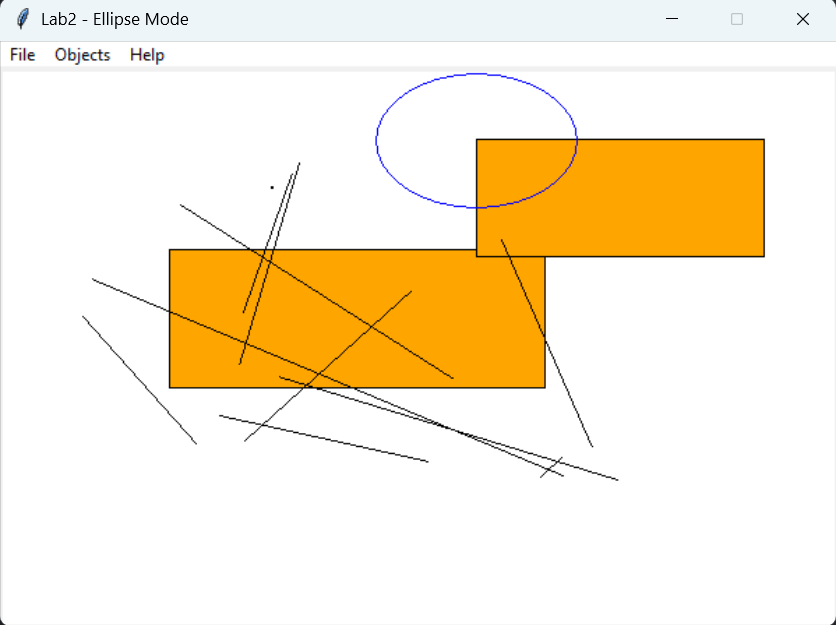
***Ієрархія класів***

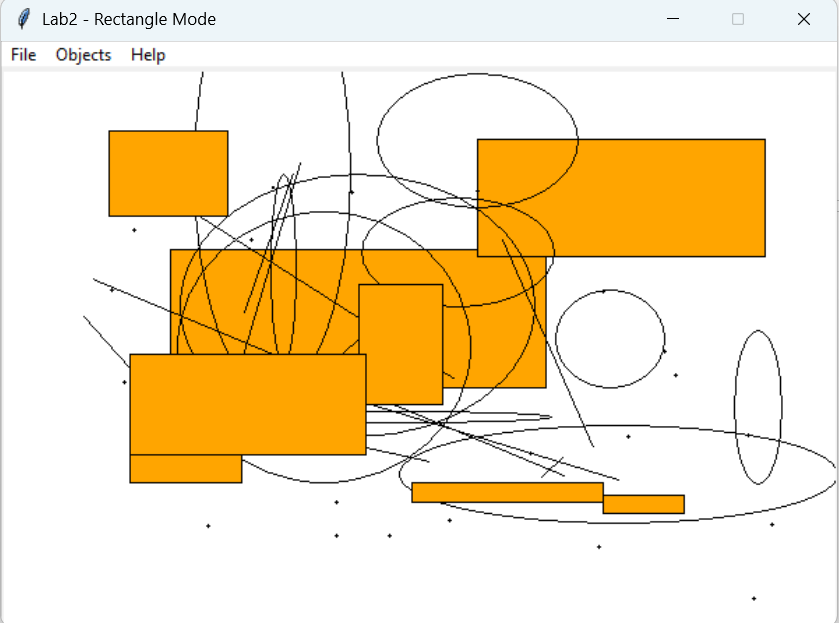
**

***Результати тестування програми***









***Висновки***

У ході виконання лабораторної роботи було реалізовано графічний редактор з панеллю меню, а саме кнопками File (кнопка Exit закриває програму), Objects (радіокнопки для вибору типу об'єкта: Крапка, Лінія, Прямокутник, Еліпс) та Help (кнопка About показує інформацію про роботу та авторські права).

Реалізовано інтерактивне полотно для малювання, на якому при виборі відповідного режиму через меню Objects можна створювати графічні примітиви: точку (при кліку мишею), лінію (перетягуванням миші), прямокутник (перетягуванням з виділенням області) та еліпс (перетягуванням з центральною точкою). Під час малювання реалізовано "гумовий слід" – синій контур, що відображає поточну позицію створюваного об'єкта.

У результаті виконання роботи було набуто практичні навички роботи з мовою програмування Python, а саме бібліотекою Tkinter для створення графічного інтерфейсу, відпрацьовано принципи об'єктно-орієнтованого програмування на практиці через створення ієрархії класів для редакторів та графічних фігур, реалізовано модульну архітектуру програми з чітким розподілом відповідальностей між компонентами та закріплено знання з проектування масштабованих та гнучких програмних рішень.