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:
студент групи ІМ-43
Олексійчук Станіслав Юрійович
номер у списку групи: 22

Перевірив:
Порєв В. М.

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

main = tk.Tk()
main.geometry("600x400+400+150")
main.title("Lab2")
main.resizable(False, False)

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

editor_manager = ShapeObjectsEditor(canvas)

def on_click(event):
    editor_manager.on_click(event.x, event.y)

def on_drag(event):
    editor_manager.on_drag(event.x, event.y)

def on_drop(event):
    editor_manager.on_drop(event.x, event.y)

canvas.bind("<Button-1>", on_click)
canvas.bind("<B1-Motion>", on_drag)
canvas.bind("<ButtonRelease-1>", on_drop)

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

current_mode = tk.StringVar(value="Point")

def set_point_editor():
    editor_manager.start_point_editor()
    current_mode.set("Point")
    editor_manager.on_menu_popup()
    main.title("Lab2 - Point Mode")

def set_line_editor():
    editor_manager.start_line_editor()
    current_mode.set("Line")
    editor_manager.on_menu_popup()
```

```
    main.title("Lab2 - Line Mode")

def set_rect_editor():
    editor_manager.start_rect_editor()
    current_mode.set("Rectangle")
    editor_manager.on_menu_popup()
    main.title("Lab2 - Rectangle Mode")

def set_ellipse_editor():
    editor_manager.start_ellipse_editor()
    current_mode.set("Ellipse")
    editor_manager.on_menu_popup()
    main.title("Lab2 - Ellipse Mode")

menu_bar = tk.Menu(main)

file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="Exit", command=main.quit)
menu_bar.add_cascade(label="File", menu=file_menu)

objects_menu = tk.Menu(menu_bar, tearoff=0)
objects_menu.add_radiobutton(label="Крапка", command=set_point_editor,
variable=current_mode, value="Point")
objects_menu.add_radiobutton(label="Лінія", command=set_line_editor,
variable=current_mode, value="Line")
objects_menu.add_radiobutton(label="Прямокутник", command=set_rect_editor,
variable=current_mode, value="Rectangle")
objects_menu.add_radiobutton(label="Елипс", command=set_ellipse_editor,
variable=current_mode, value="Ellipse")
menu_bar.add_cascade(label="Objects", menu=objects_menu)

help_menu = tk.Menu(menu_bar, tearoff=0)
help_menu.add_command(label="About", command=show_lab_information)
menu_bar.add_cascade(label="Help", menu=help_menu)

main.config(menu=menu_bar)

set_point_editor()

main.mainloop()
```

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()
```

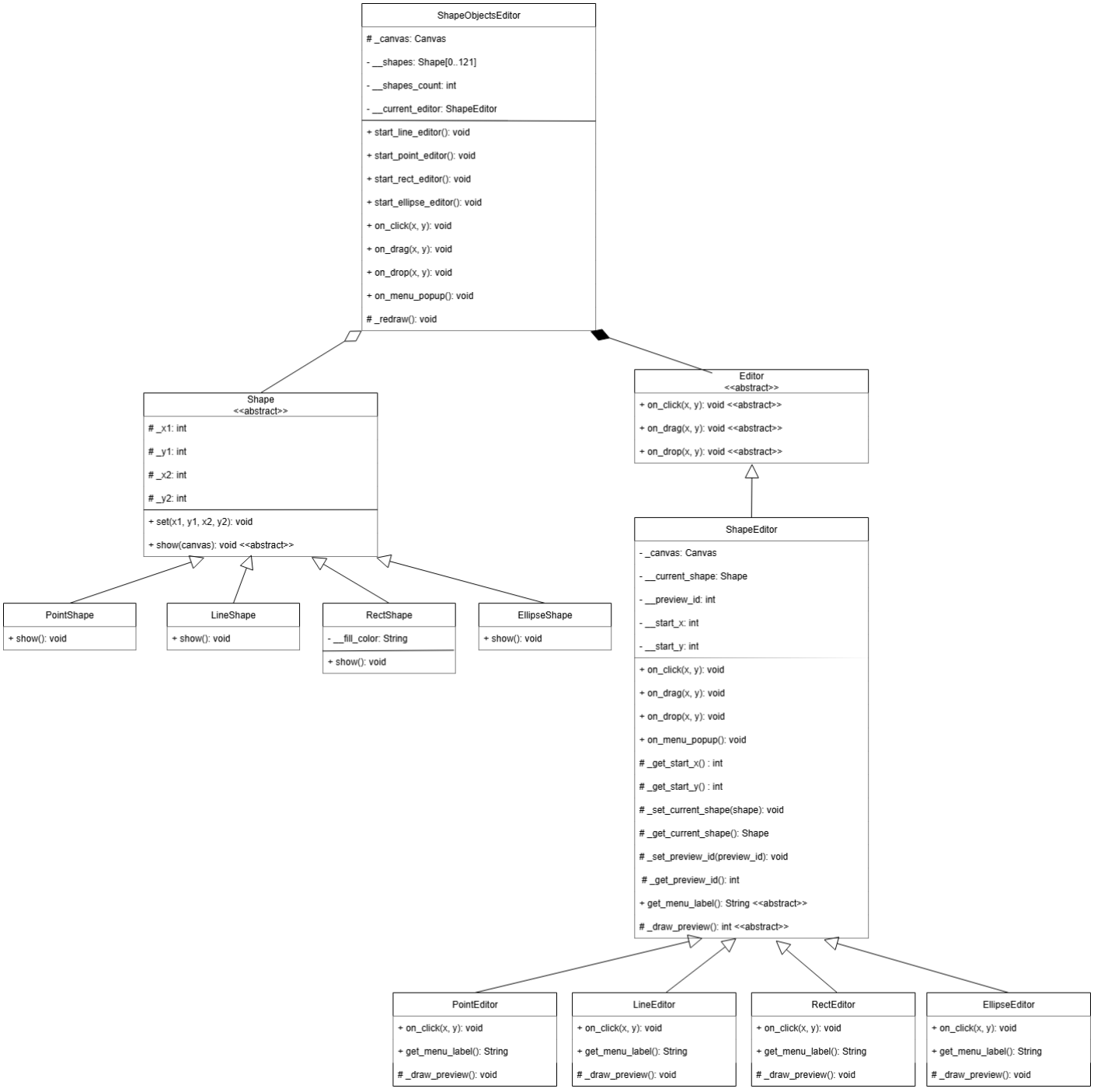
```
        return self._canvas.create_line(
            self._get_start_x(), self._get_start_y(),
            current_shape._x2, current_shape._y2,
            fill="blue", width=1
        )

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
        )
```

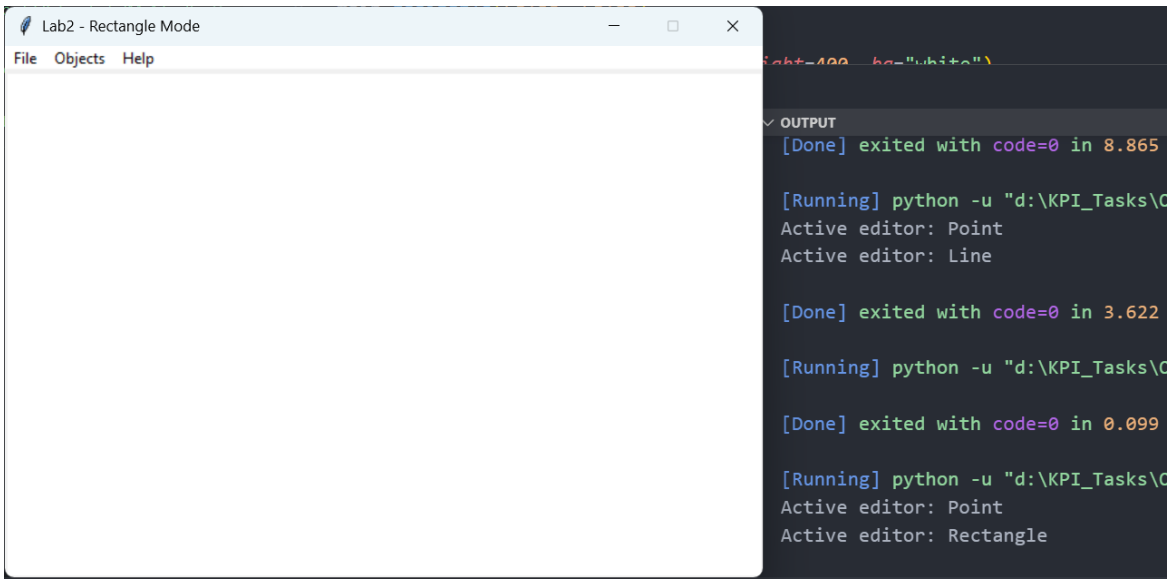
4) shape_objects_editor.py

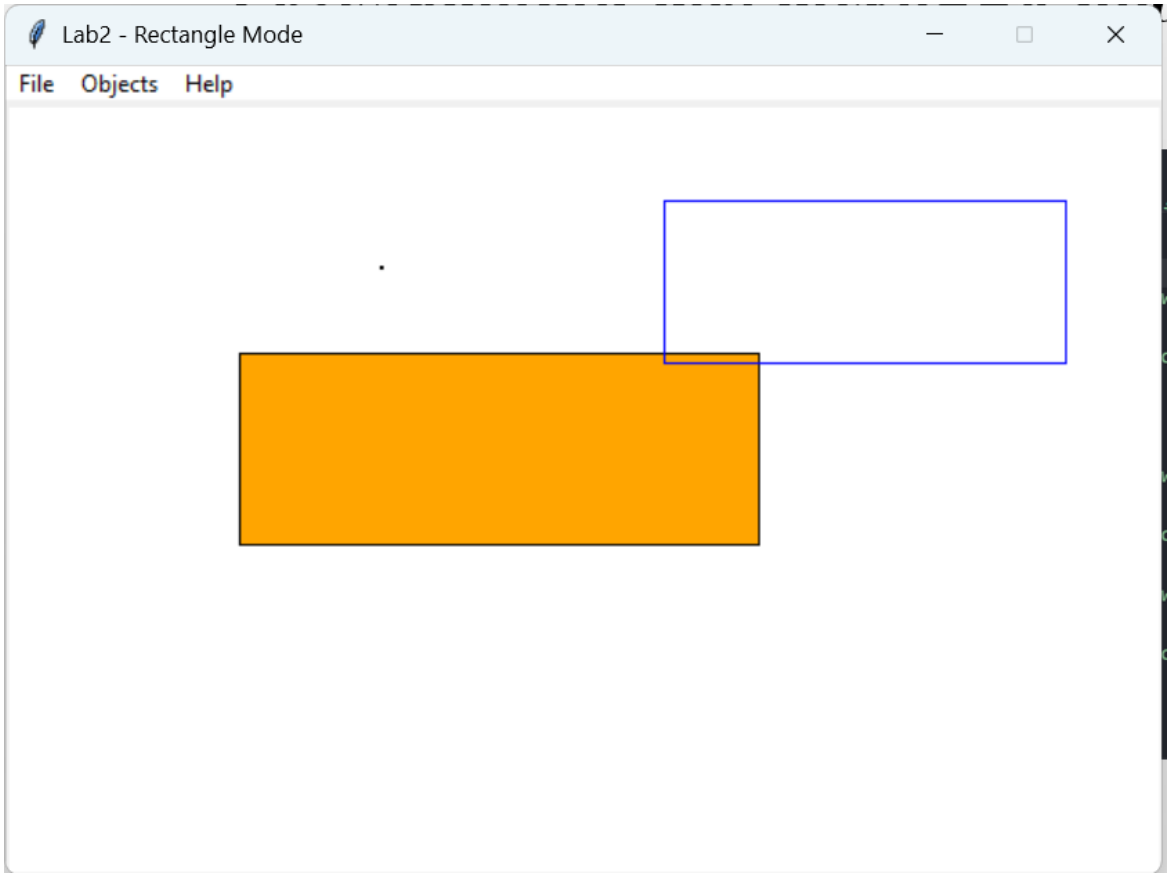
```
from editor import PointEditor, LineEditor, RectEditor, EllipseEditor

class ShapeObjectsEditor:
    def __init__(self, canvas):
        self._canvas = canvas
        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):
        for i in range(self.__shapes_count):
            if self.__shapes[i]:
                self.__shapes[i].show(self._canvas)

    def on_menu_popup(self):
        self.__current_editor.on_menu_popup()
```

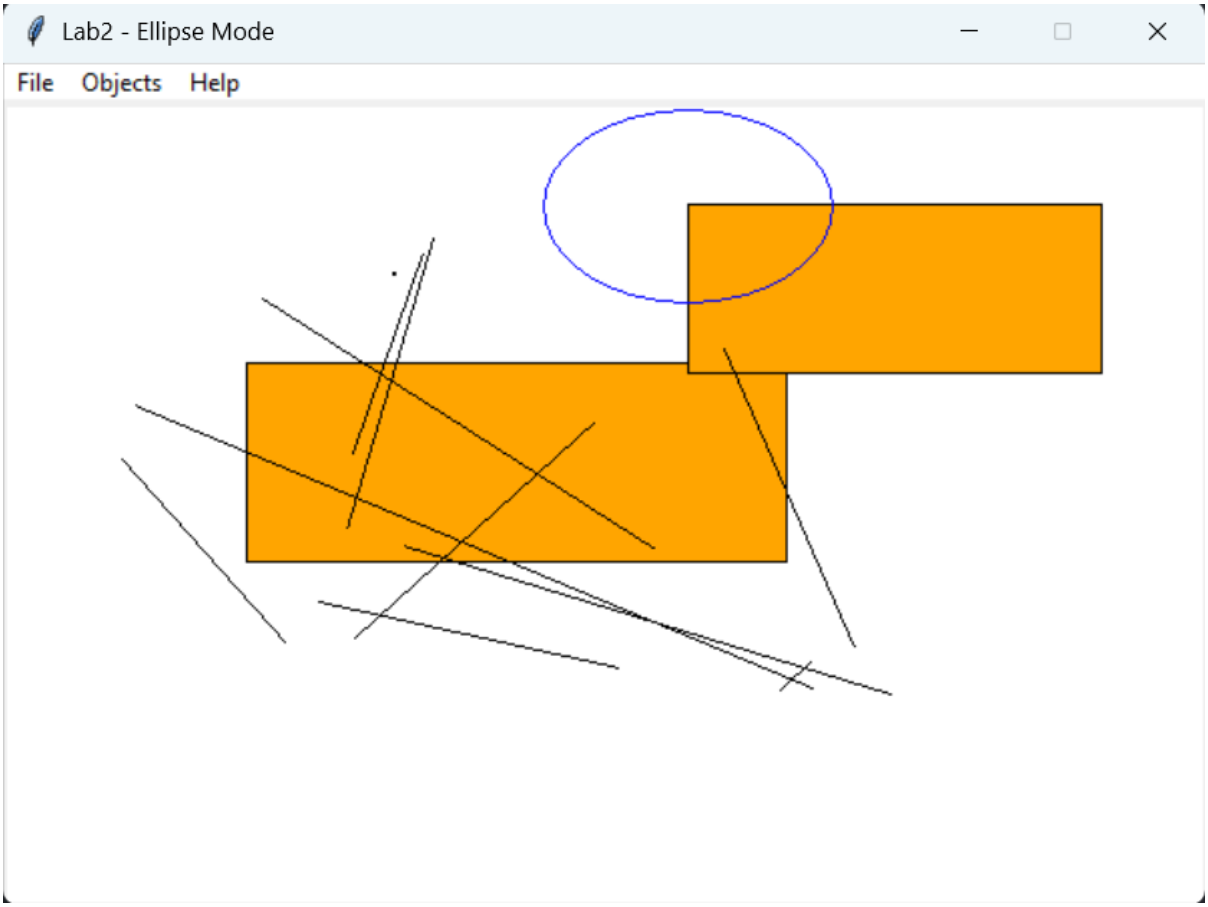
Ієрархія класів

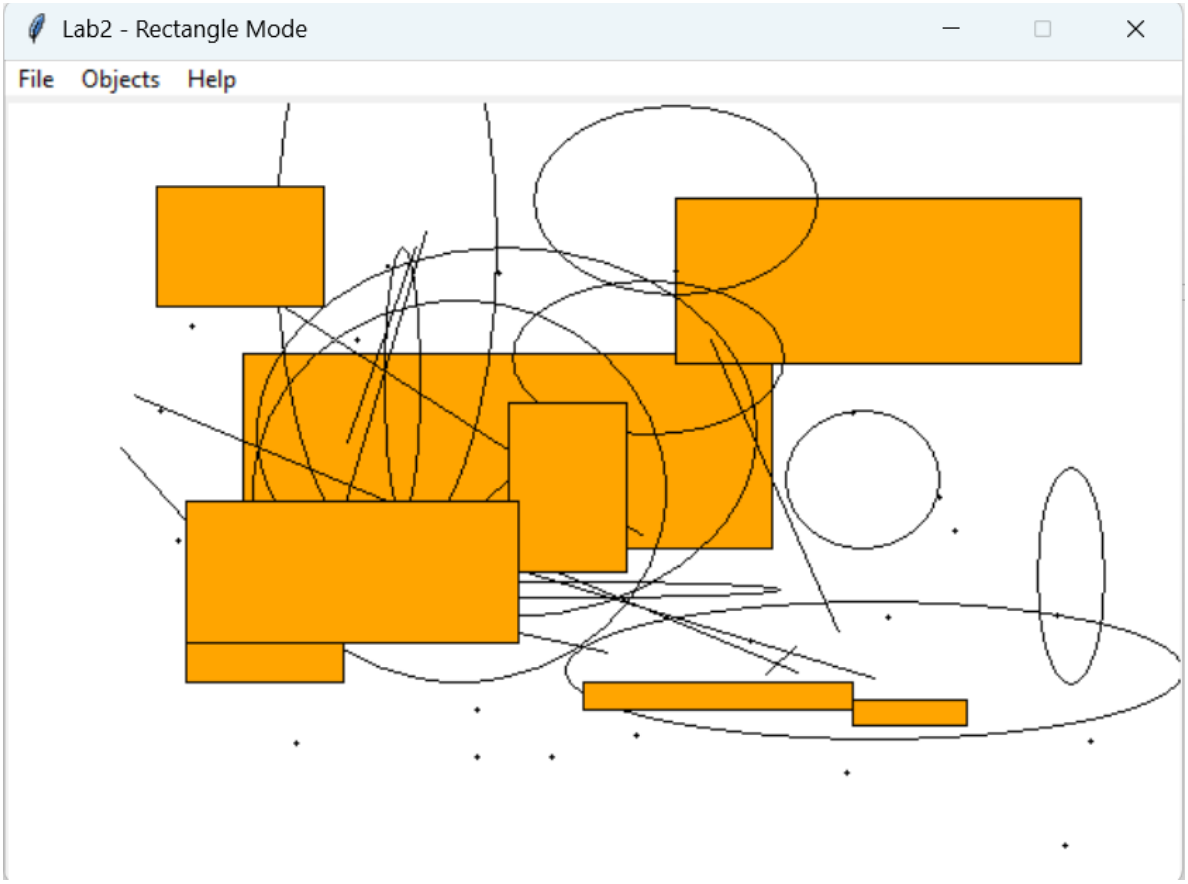


Результати тестування програми






Lab2 - Ellipse Mode
File
Objects
Help


Lab2 - Rectangle Mode
File
Objects
Help

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.