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-34
Нетяга Максим Костянтинович
номер у списку групи: 14

Перевірив:
Порєв В. М.

Київ 2024

Вихідний код

application.py

```
import tkinter as tk

from .editor_canvas import EditorCanvas, ShapeNames


class Application(tk.Tk):
    def __init__(self, screenName=None, baseName=None,
                 className="oop_lab2", useTk=True, sync=False,
                 use=None):
        super().__init__(screenName, baseName, className, useTk, sync, use)

        self.resizable(False, False)
        self.geometry("500x500")

        self._editor_canvas = EditorCanvas(self)
        self._menubar = _MenuBar(self)

        # Application config
        self.config(menu=self._menubar, border=5)

        # Placing widgets
        self._editor_canvas.pack(fill=tk.BOTH, expand=True)
        # self.menubar.grid(row=0, column=0)

    def select_shape(self, shape: ShapeNames):
        self._editor_canvas.select_shape(shape)

    def clear_canvas(self):
        self._editor_canvas.clear_canvas()


class _MenuBar(tk.Menu):
    def __init__(self, parent: Application):
        super().__init__(parent)

        file_menu = tk.Menu(self, tearoff=0)
        file_menu.add_command(
            label="Очистити",
            command=parent.clear_canvas
        )

        object_menu = tk.Menu(self, tearoff=0)
        object_menu.add_radiobutton(
            label="Крапка",
            command=(lambda: parent.select_shape("Dot"))
        )
```

```
        object_menu.add_radiobutton(
            label="Лінія",
            command=(lambda: parent.select_shape("Line"))
        )
        object_menu.add_radiobutton(
            label="Прямокутник",
            command=(lambda: parent.select_shape("Rectangle"))
        )
        object_menu.add_radiobutton(
            label="Еліпс",
            command=(lambda: parent.select_shape("Ellipse"))
        )

        info_menu = tk.Menu(self, tearoff=0)

        self.add_cascade(label="Файл", menu=file_menu)
        self.add_cascade(label="Об’єкти", menu=object_menu)
        self.add_cascade(label="Довідка", menu=info_menu)
```

editor_canvas.py

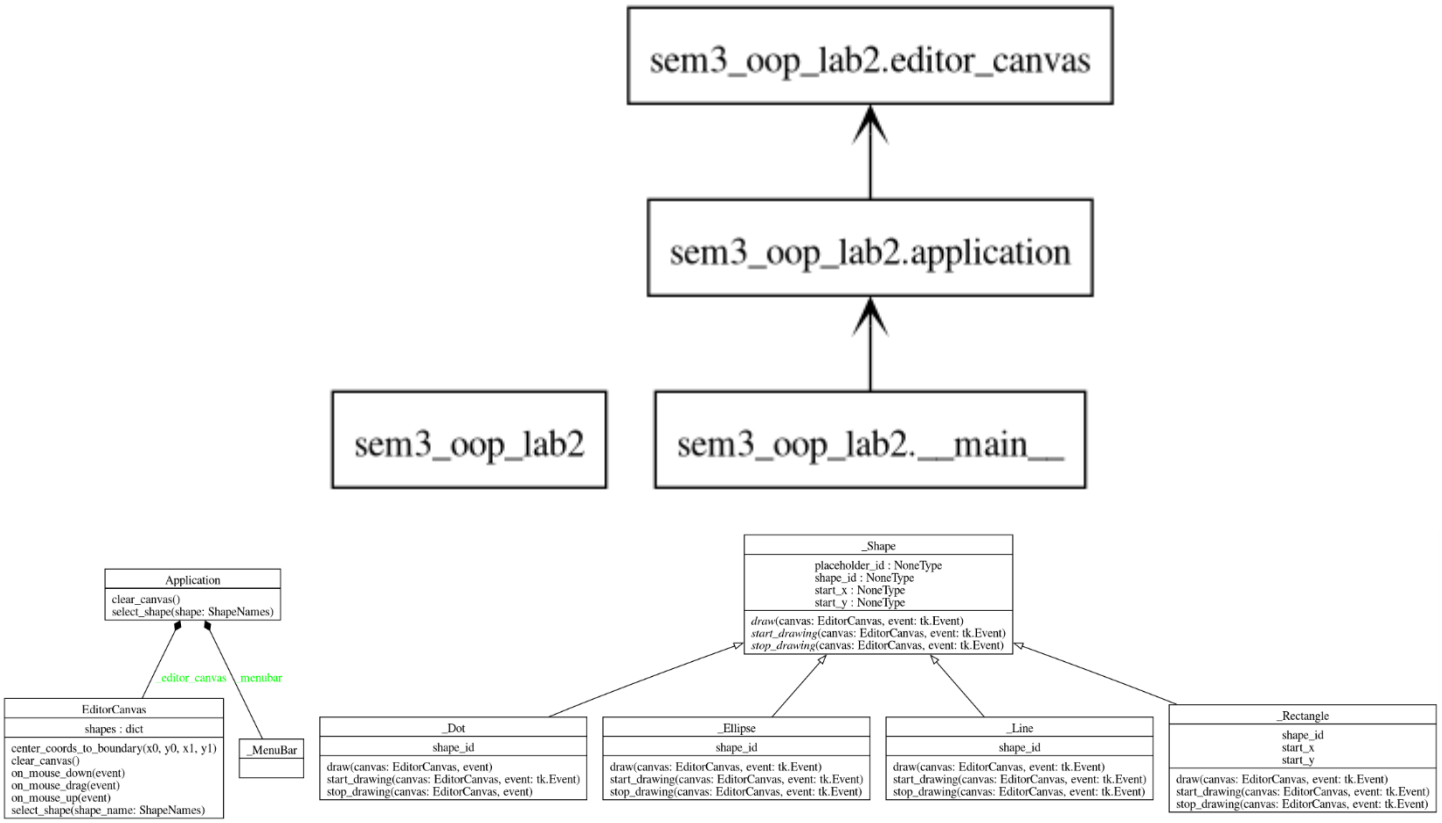
```
from typing import Literal
from abc import ABC, abstractmethod
import tkinter as tk

ShapeNames = Literal["Dot", "Line", "Rectangle", "Ellipse"]


class EditorCanvas(tk.Canvas):
    @staticmethod
    def center_coords_to_boundary(x0, y0, x1, y1):
        return (x0 - (x1 - x0)/2, y0 - (y1 - y0)/2, x1, y1)

    def __init__(self, parent):
        super().__init__(parent, bg="white")

        self._shape_to_draw: ShapeNames = None
        self._editable_shape: _Shape = None
        self.shapes = {}

        # Bind mouse events
        self.bind("<ButtonPress-1>", self.on_mouse_down)
        self.bind("<B1-Motion>", self.on_mouse_drag)
        self.bind("<ButtonRelease-1>", self.on_mouse_up)

    def select_shape(self, shape_name: ShapeNames):
        self._shape_to_draw = shape_name

    def on_mouse_down(self, event):
        if not self._shape_to_draw:
```

```
            return

        self._editable_shape = _SHAPES[self._shape_to_draw]()
        self._editable_shape.start_drawing(self, event)

    def on_mouse_drag(self, event):
        if not self._editable_shape:
            return
        self._editable_shape.draw(self, event)

    def on_mouse_up(self, event):
        if not self._editable_shape:
            return
        self._editable_shape.stop_drawing(self, event)
        self._editable_shape = None

    def clear_canvas(self):
        self.delete("all")
        self.shapes.clear()


class _Shape(ABC):
    def __init__(self):
        super().__init__()

        self.shape_id = None
        self.placeholder_id = None
        self.start_x = None
        self.start_y = None

    @abstractmethod
    def start_drawing(self, canvas: EditorCanvas, event: tk.Event):
        self.start_x, self.start_y = event.x, event.y

        self.placeholder_id = canvas.create_rectangle(
            -1, -1, -1, -1,
            outline="blue"
        )

    @abstractmethod
    def draw(self, canvas: EditorCanvas, event: tk.Event):
        pass

    @abstractmethod
    def stop_drawing(self, canvas: EditorCanvas, event: tk.Event):
        canvas.delete(self.placeholder_id)
        self.placeholder_id = None
        canvas.shapes[self.shape_id] = self
```

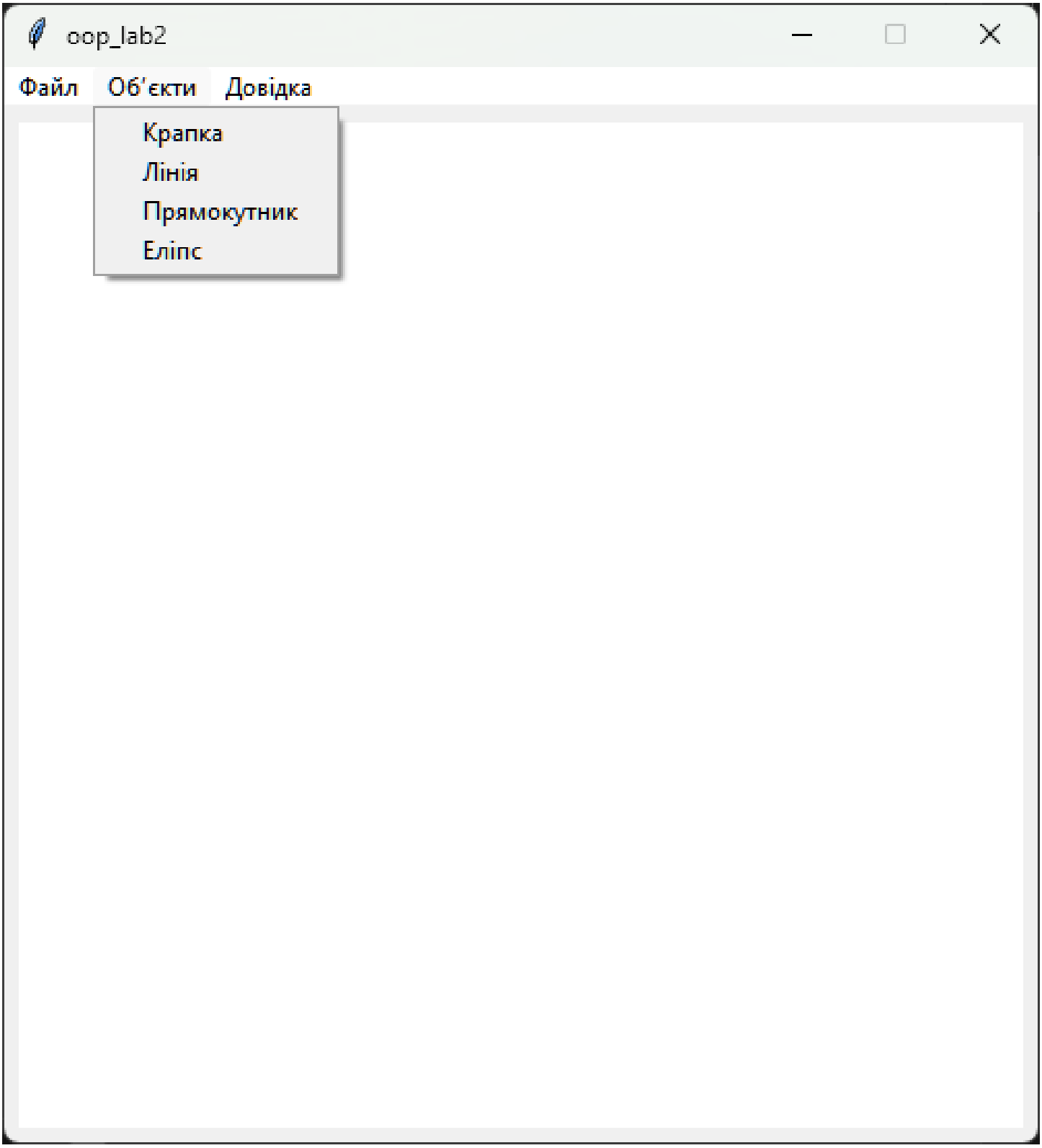
```
class _Dot(_Shape):
    def start_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().start_drawing(canvas, event)

        self.shape_id = canvas.create_oval(
            event.x - 1, event.y - 1,
            event.x + 1, event.y + 1,
            fill="black"
        )

    def draw(self, canvas: EditorCanvas, event):
        super().draw(canvas, event)

    def stop_drawing(self, canvas: EditorCanvas, event):
        super().stop_drawing(canvas, event)


class _Line(_Shape):
    def start_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().start_drawing(canvas, event)

        self.shape_id = canvas.create_line(-1, -1, -1, -1)

    def draw(self, canvas: EditorCanvas, event: tk.Event):
        super().draw(canvas, event)

        canvas.coords(
            self.placeholder_id, self.start_x, self.start_y,
            event.x, event.y
        )

    def stop_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().stop_drawing(canvas, event)

        canvas.coords(
            self.shape_id, self.start_x, self.start_y,
            event.x, event.y
        )


class _Rectangle(_Shape):
    def start_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().start_drawing(canvas, event)

        self.start_x, self.start_y = event.x, event.y
        self.shape_id = canvas.create_rectangle(-1, -1, -1, -1)

    def draw(self, canvas: EditorCanvas, event: tk.Event):
        super().draw(canvas, event)
```

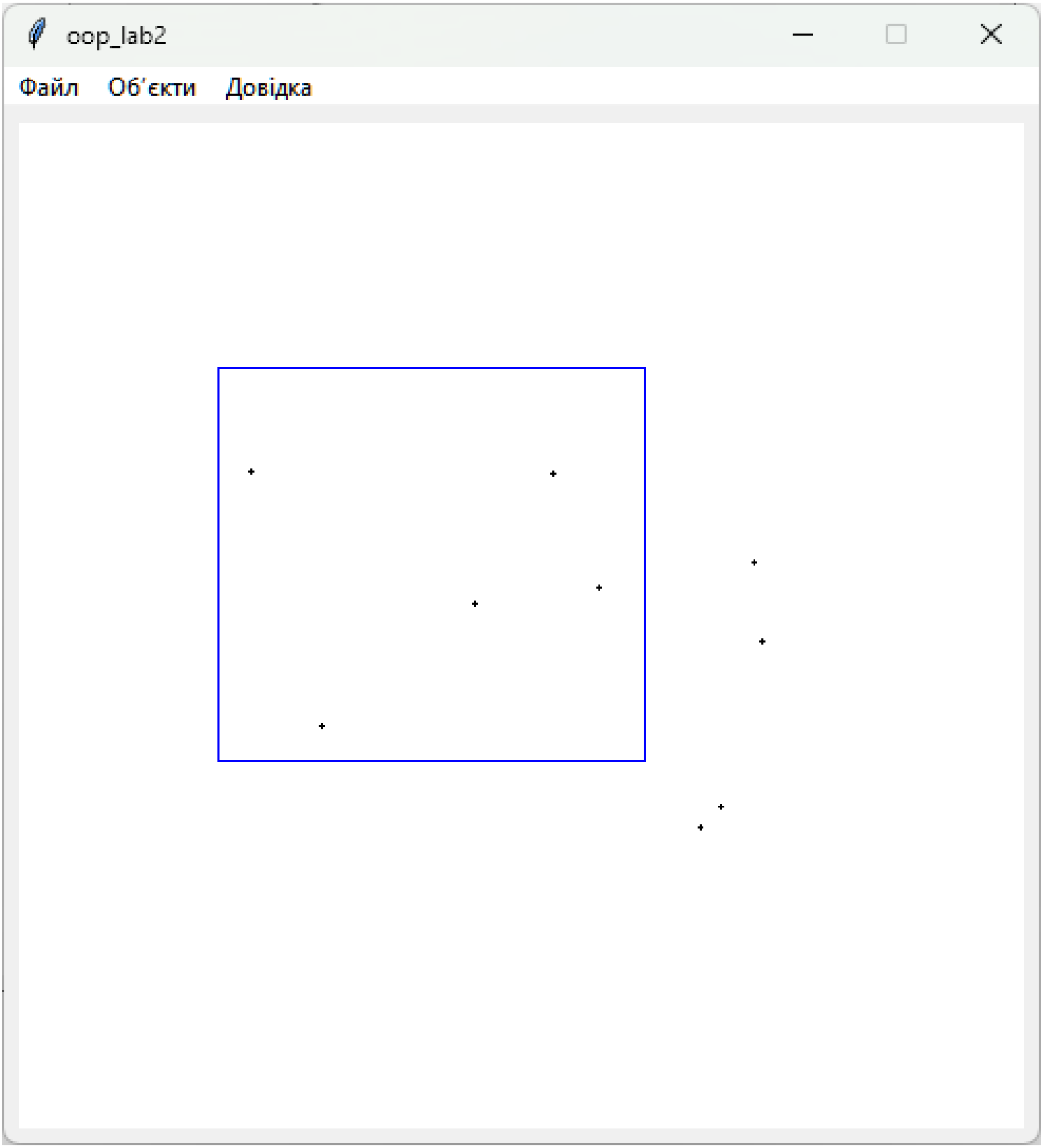
```
        canvas.coords(
            self.placeholder_id, self.start_x, self.start_y,
            event.x, event.y
        )

    def stop_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().stop_drawing(canvas, event)

        canvas.coords(
            self.shape_id, self.start_x, self.start_y,
            event.x, event.y
        )


class _Ellipse(_Shape):
    def start_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().start_drawing(canvas, event)

        self.shape_id = canvas.create_oval(
            -1, -1, -1, -1,
            fill="lightgreen"
        )

    def draw(self, canvas: EditorCanvas, event: tk.Event):
        super().draw(canvas, event)

        canvas.coords(
            self.placeholder_id,
            *EditorCanvas.center_coords_to_boundary(
                self.start_x, self.start_y, event.x, event.y
            )
        )

    def stop_drawing(self, canvas: EditorCanvas, event: tk.Event):
        super().stop_drawing(canvas, event)

        canvas.coords(
            self.shape_id,
            *EditorCanvas.center_coords_to_boundary(
                self.start_x, self.start_y, event.x, event.y
            )
        )


_SHAPES = {
    "Dot": _Dot,
    "Line": _Line,
    "Rectangle": _Rectangle,
    "Ellipse": _Ellipse,
}
```

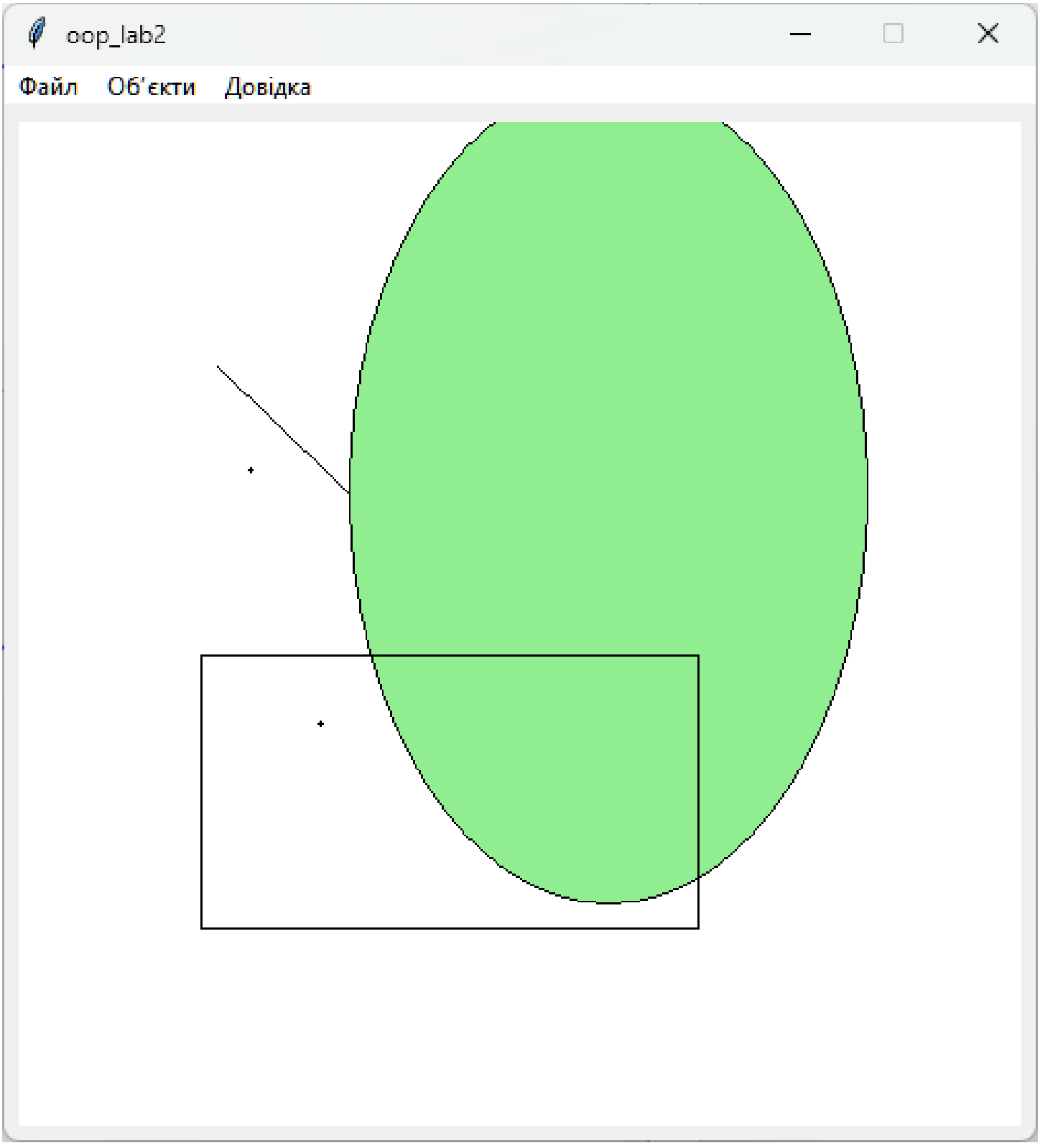
Діаграми залежностей



Скріншоти


oop_lab2
Файл
Об'єкти
Довідка
Крапка
Лінія
Прямокутник
Еліпс


oop_lab2
Файл
Об'єкти
Довідка



Висновки

В ході виконання лабораторної роботи було розроблено програму для реалізації механізму малювання графічних фігур (точок, ліній, прямокутників, еліпсів) з використанням патерну "Стратегія". Було детально вивчено та застосовано концепції об’єктно-орієнтованого програмування, такі як інкапсуляція, поліморфізм та

успадкування. Використання патерну "Стратегія" дозволило гнучко змінювати алгоритми малювання фігур, не змінюючи основний код програми. Також було продемонстровано можливість легко додавати нові стратегії малювання, що забезпечує масштабованість і гнучкість розширення функціоналу програми.

Таким чином, виконана робота дала змогу закріпити теоретичні знання про патерни проектування та застосувати їх на практиці для вирішення задач графічного інтерфейсу в Python за допомогою бібліотеки Tkinter.