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 №3

з дисципліни

«Об’єктно-орієнтоване програмування»

Виконав:
студент групи ІМ-43
Олексійчук Станіслав Юрійович
номер у списку групи: 20

Перевірив:
Порєв В. М.

Київ 2025

Варіант завдання та основні вимоги

Варіант 20+1(21):

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

2. Усі методи-обробники повідомлень, зокрема, і метод OnNotify, повинні бути функціями-членами деякого класу (класів).

3. Для вибору типу об’єкту в графічному редакторі Lab3 повинно бути вікно Toolbar з кнопками відповідно типам об'єктів. Кнопки дублюють підпункти меню "Об’єкти". Кнопки мають бути з підказками (tooltips). Меню "Об’єкти" повинно бути праворуч меню "Файл" та ліворуч меню "Довідка".

Підпункти меню "Об’єкти" містять назви геометричних форм українською мовою. Геометричні форми згідно варіанту завдання.

4. Для вибору варіанту використовується значення Ж = Жлаб2 + 1, де

Жлаб2 – номер студента в журналі, який використовувався для попередньої лаб. роботи No2.

5. Масив вказівників для динамічних об’єктів типу Shape

- динамічний масив Shape **pcshape;

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

6. "Гумовий" слід при вводі об’єктів

- суцільна лінія червоного кольору для (Ж mod 4 = 1)

7. Чотири геометричні форми (крапка, лінія, прямокутник, еліпс) можуть мати наступні різновиди вводу та відображення.

7.1. Прямокутник

Увід прямокутника:

- від центру до одного з кутів для (Ж mod 2 = 1)

Відображення прямокутника:

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

Кольори заповнення прямокутника:

- рожевий для (Ж mod 6 = 3)

7.2. Еліпс

Ввід еліпсу:

- по двом протилежним кутам охоплюючого прямокутника для варіантів (Ж mod 2 = 1)

Відображення еліпсу:

- чорний контур з білим заповненням для (Ж mod 5 = 1)

8. Позначка поточного типу об’єкту, що вводиться

- в заголовку вікна для (Ж mod 2 = 1)

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
from toolbar import Toolbar

class Main:
    def __init__(self):
        self._main_window = None
        self._canvas = None
        self._editor_manager = None
        self._current_mode = None
        self._menu_bar = None
        self._toolbar = None

    def run(self):
        self._create_window()
        self._create_canvas()
        self._create_editor()
        self._create_menu()
        self._create_toolbar()
        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("Lab3")
        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_toolbar(self):
        self._toolbar = Toolbar(self._main_window, self._canvas, self,
self._current_mode)
        self._toolbar.create_toolbar()

    def _create_objects_menu(self):
        objects_menu = tk.Menu(self._menu_bar, tearoff=0)

        objects_menu.add_radiobutton(
            label="Крапка",
            command=self.switch_to_point_mode,
            variable=self._current_mode,
            value="Point"
        )
        objects_menu.add_radiobutton(
            label="Лінія",
            command=self.switch_to_line_mode,
            variable=self._current_mode,
            value="Line"
        )
        objects_menu.add_radiobutton(
            label="Прямокутник",
            command=self.switch_to_rect_mode,
            variable=self._current_mode,
            value="Rectangle"
        )
        objects_menu.add_radiobutton(
            label="Еліпс",
            command=self.switch_to_ellipse_mode,
            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 switch_to_point_mode(self):
        self._editor_manager.start_point_editor()
        self._current_mode.set("Point")
        self._main_window.title(f"Lab3 - {self._current_mode.get()} Mode")
        self._toolbar.update_button_states()

    def switch_to_line_mode(self):
        self._editor_manager.start_line_editor()
        self._current_mode.set("Line")
        self._main_window.title(f"Lab3 - {self._current_mode.get()} Mode")
        self._toolbar.update_button_states()

    def switch_to_rect_mode(self):
        self._editor_manager.start_rect_editor()
        self._current_mode.set("Rectangle")
        self._main_window.title(f"Lab3 - {self._current_mode.get()} Mode")
        self._toolbar.update_button_states()

    def switch_to_ellipse_mode(self):
        self._editor_manager.start_ellipse_editor()
        self._current_mode.set("Ellipse")
        self._main_window.title(f"Lab3 - {self._current_mode.get()} Mode")
        self._toolbar.update_button_states()

    def _show_lab_information(self):
        messagebox.showinfo("Info", "Lab 3 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.switch_to_point_mode()

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 = "pink"

    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_rectangle(x1, y1, x2, y2,
                                outline="black", fill=self._fill_color, width=1)

class EllipseShape(Shape):
    def __init__(self):
```

```
        super().__init__()
        self._fill_color = "white"

    def show(self, canvas):
        canvas.create_oval(self.x1, self.y1,
                           self.x2, self.y2,
                           outline="black", fill=self._fill_color, 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
```

```
    @abstractmethod
    def _draw_preview(self):
        pass

class PointEditor(ShapeEditor):
    def on_click(self, x, y):
        super().on_click(x, y)
        self._current_shape = PointShape()
        self._current_shape.set(x, y, x, y)
        if self._preview_id:
            self._canvas.delete(self._preview_id)
        self._preview_id = self._draw_preview()

    def _draw_preview(self):
        return self._canvas.create_oval(
            self._start_x-2, self._start_y-2,
            self._start_x+2, self._start_y+2,
            outline="red", width=1
        )

class LineEditor(ShapeEditor):
    def on_click(self, x, y):
        super().on_click(x, y)
        self._current_shape = LineShape()
        self._current_shape.set(x, y, x, y)
        if self._preview_id:
            self._canvas.delete(self._preview_id)
        self._preview_id = self._draw_preview()

    def _draw_preview(self):
        current_shape = self._current_shape
        return self._canvas.create_line(
            self._start_x, self._start_y,
            current_shape.x2, current_shape.y2,
            fill="red", width=1
        )

class RectEditor(ShapeEditor):
    def on_click(self, x, y):
        super().on_click(x, y)
        self._current_shape = RectShape()
        self._current_shape.set(x, y, x, y)
        if self._preview_id:
            self._canvas.delete(self._preview_id)
        self._preview_id = self._draw_preview()

    def _draw_preview(self):
        center_x = self._start_x
        center_y = self._start_y
        current_shape = self._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_rectangle(
            x1, y1, x2, y2,
            outline="red", width=1
        )

class EllipseEditor(ShapeEditor):
    def on_click(self, x, y):
        super().on_click(x, y)
        self._current_shape = EllipseShape()
        self._current_shape.set(x, y, x, y)
        if self._preview_id:
            self._canvas.delete(self._preview_id)
        self._preview_id = self._draw_preview()

    def _draw_preview(self):
        current_shape = self._current_shape
        return self._canvas.create_oval(
            self._start_x, self._start_y,
            current_shape.x2, current_shape.y2,
            outline="red", 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 = []
        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_drag(self, x, y):
        if self._current_editor:
            self._current_editor.on_drag(x, y)

    def on_drop(self, x, y):
        if self._current_editor:
            shape = self._current_editor.on_drop(x, y)
            if shape:
                self._shapes.append(shape)
                self._redraw()
        return None

    def _redraw(self):
        self._canvas.delete("all")
        for shape in self._shapes:
            if shape:
                shape.show(self._canvas)
```

5) bitmap_factory.py

```
from PIL import Image, ImageDraw, ImageTk

class BitmapFactory:
    def __init__(self):
        self._images = {}

    def create_toolbar_bitmaps(self, size=28):
        center = size // 2

        self._images["Крапка"] = self._create_point_bitmap(size, center)
        self._images["Лінія"] = self._create_line_bitmap(size, center)
        self._images["Прямокутник"] = self._create_rectangle_bitmap(size, center)
        self._images["Еліпс"] = self._create_ellipse_bitmap(size, center)

        return self._images

    def _create_point_bitmap(self, size, center):
        point_img = Image.new("RGB", (size, size), "white")
        point_draw = ImageDraw.Draw(point_img)

        point_draw.ellipse([center-4, center-4, center+4, center+4],
                           fill="black", outline="black", width=1)
        return ImageTk.PhotoImage(point_img)

    def _create_line_bitmap(self, size, center):
        line_img = Image.new("RGB", (size, size), "white")
        line_draw = ImageDraw.Draw(line_img)
```

```
        line_draw.line([center-7, center+7, center+7, center-7],
                       fill="black", width=2)
        return ImageTk.PhotoImage(line_img)

    def _create_rectangle_bitmap(self, size, center):
        rectangle_img = Image.new("RGB", (size, size), "white")
        rectangle_draw = ImageDraw.Draw(rectangle_img)

        rectangle_draw.rectangle([center-7, center-4, center+7, center+6],
                       fill="black", outline="black", width=1)
        return ImageTk.PhotoImage(rectangle_img)

    def _create_ellipse_bitmap(self, size, center):
        ellipse_img = Image.new("RGB", (size, size), "white")
        ellipse_draw = ImageDraw.Draw(ellipse_img)

        ellipse_draw.ellipse([center-7, center-4, center+7, center+4],
                       fill="black", outline="black", width=1)
        return ImageTk.PhotoImage(ellipse_img)

    def get_image(self, name):
        return self._images.get(name)

    def get_all_images(self):
        return self._images
```

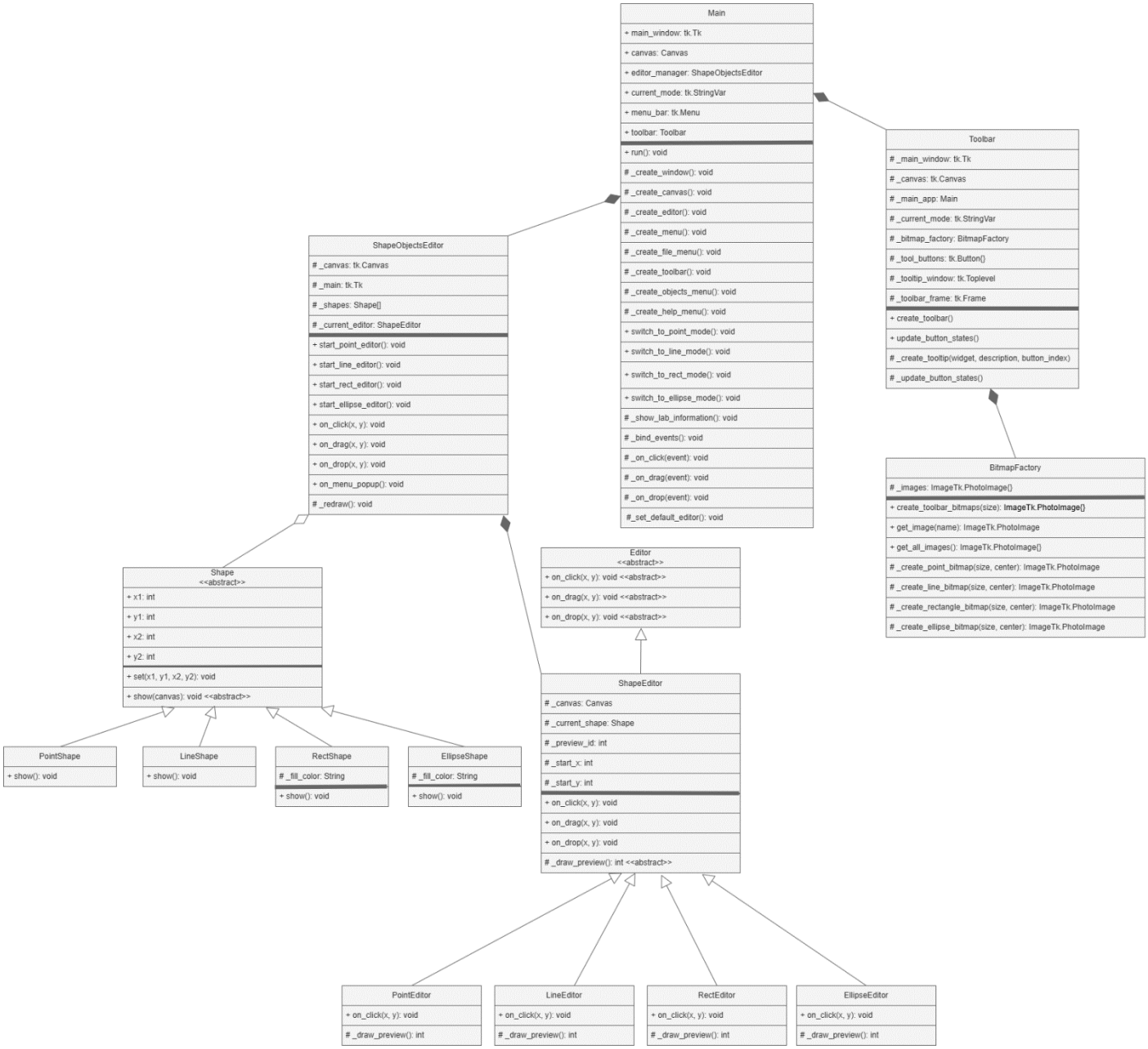
4) toolbar.py

```
import tkinter as tk
from bitmap_factory import BitmapFactory

class Toolbar:
    def __init__(self, main_window, canvas, main_app, current_mode):
        self._main_window = main_window
        self._canvas = canvas
        self._main_app = main_app
        self._current_mode = current_mode
        self._bitmap_factory = BitmapFactory()
        self._tool_buttons = {}
        self._tooltip_window = None
        self._toolbar_frame = None

    def create_toolbar(self):
        self._toolbar_frame = tk.Frame(self._main_window, bg="lightgray", height=42)
        self._toolbar_frame.pack(fill="x", side="top", before=self._canvas)
        self._toolbar_frame.pack_propagate(False)

        images = self._bitmap_factory.create_toolbar_bitmaps()

        tools = [
            ("Крапка", self._main_app.switch_to_point_mode, "Намалювати крапку"),
```

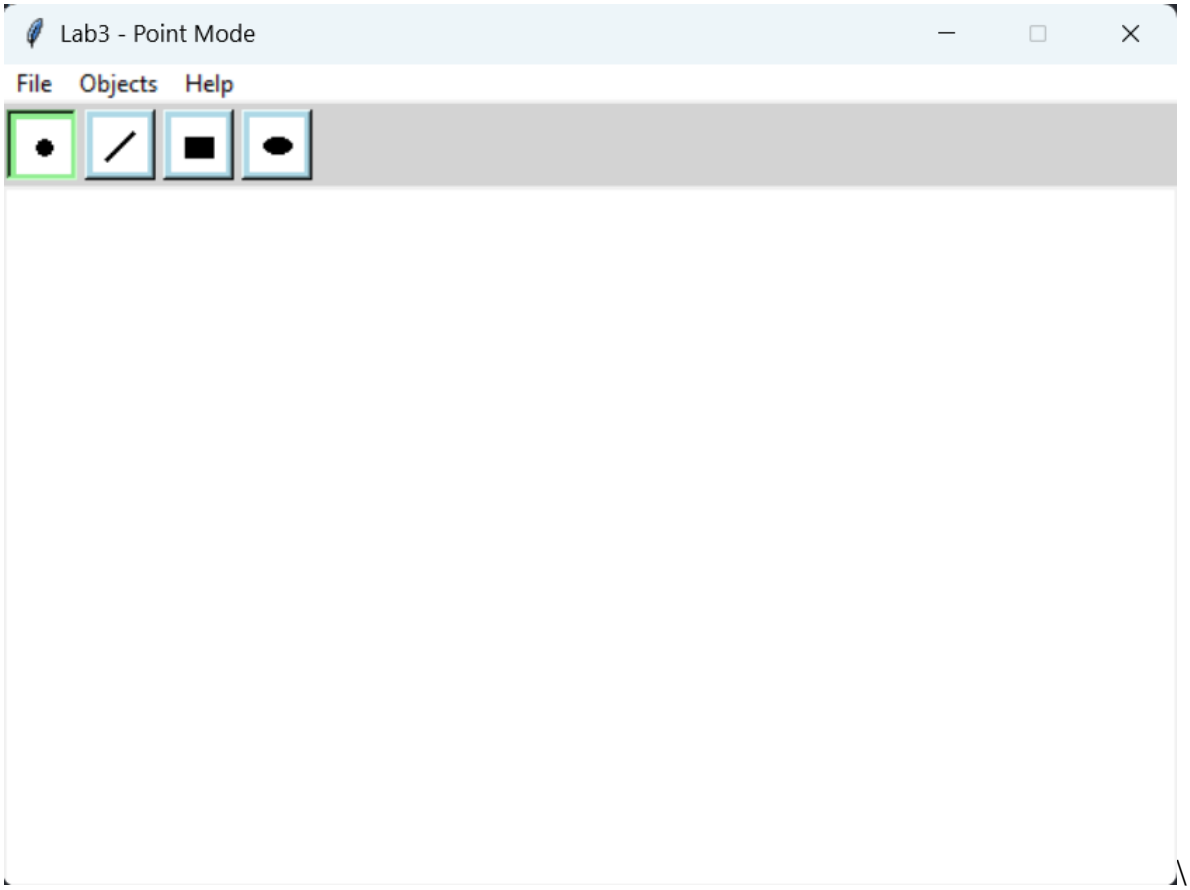
```
            ("Лінія", self._main_app.switch_to_line_mode, "Намалювати лінію"),
            ("Прямокутник", self._main_app.switch_to_rect_mode, "Намалювати
прямокутник"),
            ("Еліпс", self._main_app.switch_to_ellipse_mode, "Намалювати еліпс")
        ]

        for i, (name, command, tooltip) in enumerate(tools):
            button = tk.Button(
                self._toolbar_frame,
                image=images[name],
                command=command,
                relief="raised",
                bg="lightblue",
                width=30,
                height=30
            )

            button.pack(side="left", padx=2, pady=2)
            self._tool_buttons[name] = button

            self._create_tooltip(button, tooltip, i)

    def _create_tooltip(self, widget, description, button_index):
        def show_tooltip(event):
            if self._tooltip_window:
                self._tooltip_window.destroy()

            self._tooltip_window = tk.Toplevel(self._main_window)
            self._tooltip_window.wm_overrideredirect(True)

            base_x = self._main_window.winfo_rootx() + 10
            base_y = self._main_window.winfo_rooty() + 45

            screen_width = self._main_window.winfo_screenwidth()
            tooltip_x = min(base_x + (button_index * 35), screen_width - 200)
            tooltip_y = base_y

            self._tooltip_window.wm_geometry(f"+{tooltip_x}+{tooltip_y}")

            label = tk.Label(self._tooltip_window, text=description,
                             bg="lightyellow",
                             relief="solid",
                             borderwidth=1,
                             font=("Arial", 9),
                             padx=4, pady=2)
            label.pack()

        def hide_tooltip(event):
            if self._tooltip_window:
                self._tooltip_window.destroy()
```

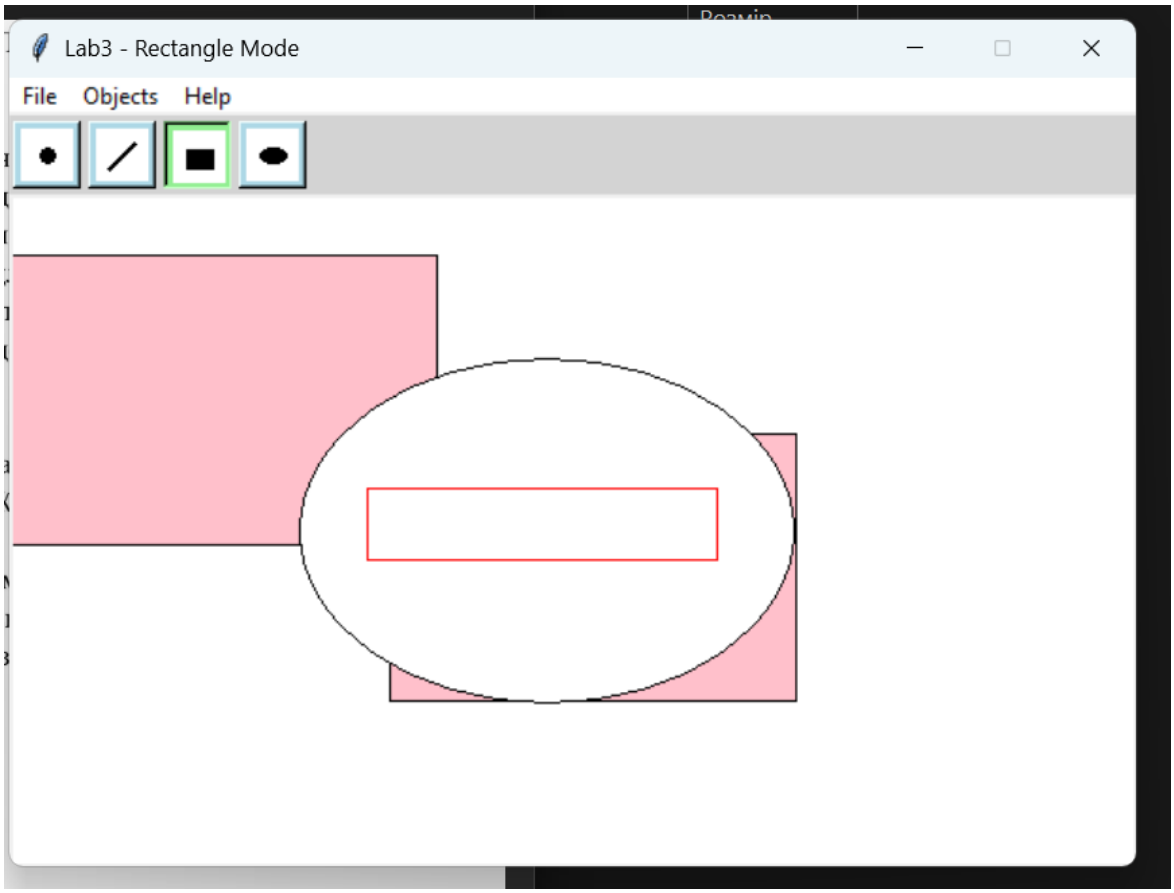
```
            self._tooltip_window = None

    widget.bind("<Enter>", show_tooltip)
    widget.bind("<Leave>", hide_tooltip)
    widget.bind("<ButtonPress>", hide_tooltip)

def _update_button_states(self):
    mode_mapping = {
        "Point": "Крапка",
        "Line": "Лінія",
        "Rectangle": "Прямокутник",
        "Ellipse": "Еліпс"
    }

    current_button_name = mode_mapping.get(self._current_mode.get())

    for name, button in self._tool_buttons.items():
        if name == current_button_name:
            button.config(relief="sunken", bg="lightgreen")
        else:
            button.config(relief="raised", bg="lightblue")

def update_button_states(self):
    self._update_button_states()
```

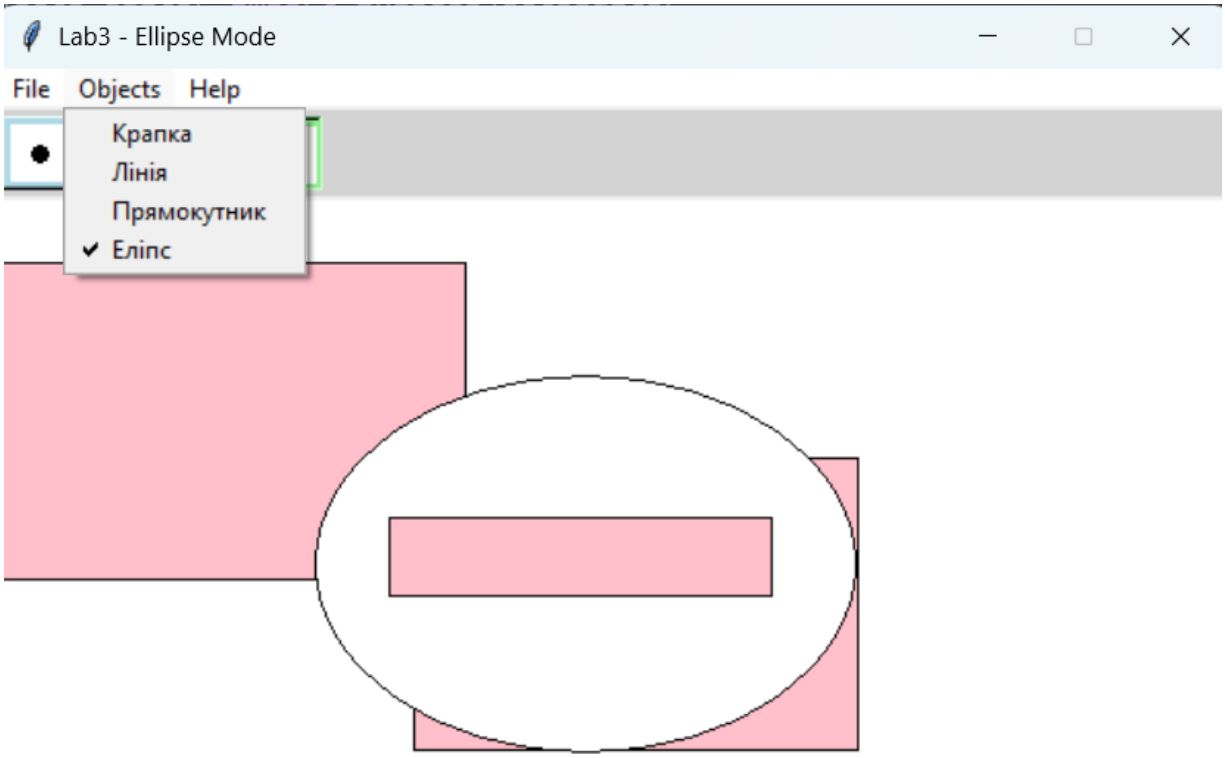
Ієрархія класів

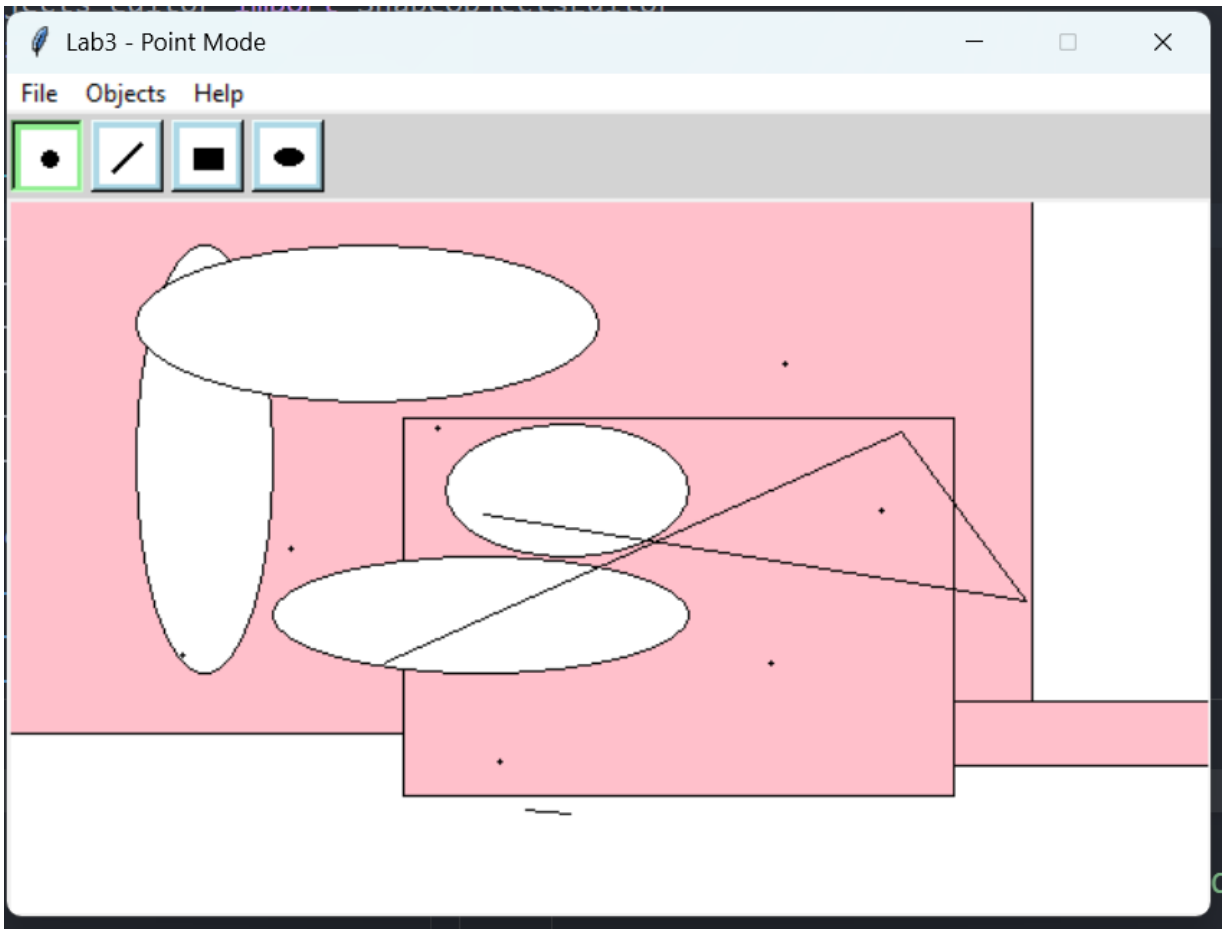


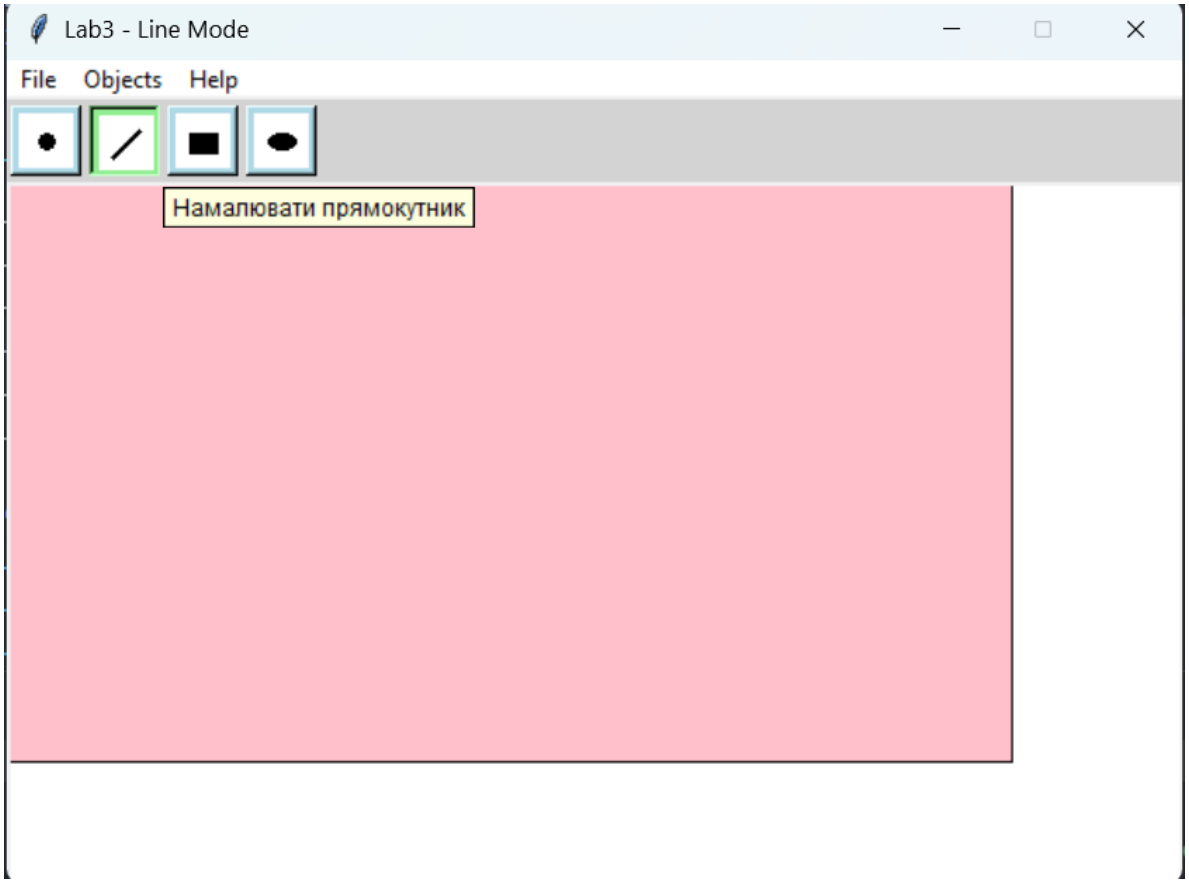
Результати тестування програми






Lab3 - Ellipse Mode
File
Objects
Help
Крапка
Лінія
Прямокутник
Еліпс


Lab3 - Point Mode
File
Objects
Help


Lab3 - Line Mode
File
Objects
Help
Намалювати прямокутник

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 або попередньо створений Toolbar, синхронізований з кнопкою меню Objects, можна створювати графічні примітиви: точку (при кліку мишею), лінію (перетягуванням миші), прямокутник (перетягуванням від центру з виділенням області) та еліпс (перетягування від протилежних кутів описаного прямокутника з виділенням області). Під час малювання реалізовано "гумовий слід" – червоний контур, що відображає поточну позицію створюваного об'єкта.

Реалізовано підказки у вигляді тексту при наведенні курсором миші на одну з кнопок.

У результаті виконання роботи було набуто практичні навички роботи з мовою програмування Python, а саме бібліотекою Tkinter для створення графічного інтерфейсу, PIL для створення бітмапів, використаних у Toolbar, 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.