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

Лабораторна робота №3

з дисципліни
«Об’єктно орієнтоване
програмування»на тему “Розробка
інтерфейсу користувача на C++”

Виконала:
Студентка групи ІМ-33
Пилипчук Вероніка Олексіївна
Номер у списку групи: 18

Перевірив:
Порєв В.М.

Київ 2024

Варіант завдання:

1. У звіті повинна бути схема успадкування класів – **діаграма класів**
2. 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.
3. Динамічний масив Shape**pcshape
4. Гумовий слід при вводі об’єктів: суцільна лінія синього кольору
5. Прямокутник: по двом протилежним кутам + чорний контур з жовтим заповненням
6. Еліпс: від центру до одного з кутів охоплюючого прямокутника + чорний контур з сірим заповненням
7. Позначка поточного об’єкту, що вводиться: в меню

Вихідний текст головного файлу (Lab3.tsx):

```
import React, { useEffect, useRef, useState } from "react";
import { Menu, MenuProps } from "antd";
import { Dot, Ellipse, Line, Rectangle, Shape } from "@/app/modules/Shape";
import { items } from "./constants";
import { Toolbar } from "./Toolbar";

export const Lab3: React.FC = () => {
  const canvasRef = useRef<HTMLCanvasElement | null>(null);
  const [shapes, setShapes] = useState<Shape[]>([]);
  const [currentTab, setCurrentTab] = useState("");
  const [isDrawing, setIsDrawing] = useState(false);
  const [lastPosition, setLastPosition] = useState<{
    x: number;
    y: number;
  } | null>(null);
  const [previewShape, setPreviewShape] = useState<Shape | null>(null);

  const onClick: MenuProps["onClick"] = (e) => {
    setCurrentTab(e.key);
  };

  const drawShape = (event: MouseEvent, preview = false) => {
    if (!canvasRef.current) return;

    const rect = canvasRef.current.getBoundingClientRect();
    const x = event.clientX - rect.left;
    const y = event.clientY - rect.top;

    let newShape: Shape | null = null;

    switch (currentTab) {
      case "dot":
        newShape = new Dot(x, y);
        break;
      case "line":
        if (lastPosition) {
          newShape = new Line(lastPosition.x, lastPosition.y, x, y);
        }
        break;
```

```
    case "rectangle":
      if (lastPosition) {
        const width = x - lastPosition.x;
        const height = y - lastPosition.y;
        newShape = new Rectangle(
          lastPosition.x,
          lastPosition.y,
          width,
          height
        );
      }
      break;
    case "ellipse":
      if (lastPosition) {
        const radiusX = Math.abs(x - lastPosition.x);
        const radiusY = Math.abs(y - lastPosition.y);
        newShape = new Ellipse(
          lastPosition.x,
          lastPosition.y,
          radiusX,
          radiusY
        );
      }
      break;
  }

  if (preview && newShape) {
    setPreviewShape(newShape);
  } else if (newShape) {
    setShapes((prev) => [...prev, newShape]);
  }
};

useEffect(() => {
  const canvas = canvasRef.current;
  if (canvas) {
    const ctx = canvas.getContext("2d");
    if (ctx) {
      ctx.clearRect(0, 0, canvas.width, canvas.height);

      shapes.forEach((shape) => shape.draw(ctx));

      if (
        previewShape &&
        "drawPreview" in previewShape &&
        typeof previewShape.drawPreview === "function"
      ) {
        previewShape.drawPreview(ctx);
      } else if (previewShape) {
        previewShape.draw(ctx);
      }
    }
  }
}, [shapes, previewShape]);

const startDrawing = (event: MouseEvent) => {
  const rect = canvasRef.current!.getBoundingClientRect();
  const x = event.clientX - rect.left;
  const y = event.clientY - rect.top;
  setLastPosition({ x, y });
  setIsDrawing(true);
};

const stopDrawing = () => {
  setIsDrawing(false);
```

```
    setLastPosition(null);
    setPreviewShape(null);
  };

  useEffect(() => {
    const canvas = canvasRef.current;

    const mouseDownHandler = (event: MouseEvent) => {
      startDrawing(event);
    };

    const mouseMoveHandler = (event: MouseEvent) => {
      if (isDrawing) {
        drawShape(event, true);
      }
    };

    const mouseUpHandler = (event: MouseEvent) => {
      if (isDrawing) {
        drawShape(event);
        stopDrawing();
      }
    };

    if (canvas) {
      canvas.addEventListener("mousedown", mouseDownHandler);
      canvas.addEventListener("mousemove", mouseMoveHandler);
      canvas.addEventListener("mouseup", mouseUpHandler);
      canvas.addEventListener("mouseleave", stopDrawing);

      return () => {
        canvas.removeEventListener("mousedown", mouseDownHandler);
        canvas.removeEventListener("mousemove", mouseMoveHandler);
        canvas.removeEventListener("mouseup", mouseUpHandler);
        canvas.removeEventListener("mouseleave", stopDrawing);
      };
    }
  }, [isDrawing]);

  return (
    <>
      <Menu
        onClick={onClick}
        selectedKeys={[currentTab]}
        mode="horizontal"
        items={items}
      />
      <Toolbar onClick={(key: string) => setCurrentTab(key)} />
      <canvas
        ref={canvasRef}
        style={{ margin: "10px", boxShadow: "0 4px 10px rgba(0, 0, 0, 0.2)" }}
        width={700}
        height={400}
      />
    </>
  );
};
```

Вихідні тексти модулів:

1. Файл з елементами меню constants.ts

```
import { MenuProps } from "antd";

type MenuItem = Required<MenuProps>["items"][number];
export const items: MenuItem[] = [
  {
    label: "Файл",
    key: "file",
    children: [
      { label: "Створити", key: "create" },
      { label: "Відкрити", key: "open" },
      { label: "Зберегти", key: "save" },
      { type: "divider" },
      { label: "Друк", key: "print" },
      { type: "divider" },
      { label: "Вихід", key: "exit" },
    ],
  },
  {
    label: "Об’єкти",
    key: "objects",
    children: [
      { label: "Крапка", key: "dot" },
      { label: "Лінія", key: "line" },
      { label: "Прямокутник", key: "rectangle" },
      { label: "Еліпс", key: "ellipse" },
    ],
  },
  {
    label: "Довідка",
    key: "note",
  },
];
```

2. Модулі компонентів з кастомними іконками для крапки та еліпса

```
export const DotIcon = () => (
  <svg
    version="1.1"
    id="Capa_1"
    xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 31.955 31.955"
    fill="currentColor"
  >
    <g>
      <path
        d="M27.25,4.655C20.996-1.571,10.88-1.546,4.656,4.706C-1.571,10.96-1.548,21.076,4.705,27.3
    c6.256,6.226,16.374,6.203,22.597-0.051C33.526,20.995,33.505,10.878,27.25,4.655z"
      />
      <path
        d="M13.288,23.896l-1.768,5.207c2.567,0.829,5.331,0.886,7.926,0.17l-0.665-5.416
    C17.01,24.487,15.067,24.5,13.288,23.896z
M8.12,13.122l-5.645-0.859c-0.741,2.666-0.666,5.514,0.225,8.143l5.491-1.375
    C7.452,17.138,7.426,15.029,8.12,13.122z
M28.763,11.333l-4.965,1.675c0.798,2.106,0.716,4.468-0.247,6.522l5.351,0.672
    C29.827,17.319,29.78,14.193,28.763,11.333z
M11.394,2.883l1.018,5.528c2.027-0.954,4.356-1.05,6.442-0.288l1.583-5.137
    C17.523,1.94,14.328,1.906,11.394,2.883z"
      />
      <circle cx="15.979" cy="15.977" r="6.117" />
    </g>
  </svg>
);

export const EllipseIcon = (props: React.SVGProps<SVGSVGElement>) => (
  <svg
    version="1.1"
    id="Capa_1"
    xmlns="http://www.w3.org/2000/svg"
    viewBox="0 0 88.332 88.333"
    fill="currentColor"
    {...props}
  >
    <g>
      <g>
        <path

d="M44.166,75.062C19.812,75.062,0,61.202,0,44.167C0,27.13,19.812,13.27,44.166,13.27c24.354,0,4
4.166,13.859,44.166,30.896
      C88.332,61.204,68.52,75.062,44.166,75.062z
M44.166,16.27C21.467,16.27,3,28.784,3,44.167c0,15.381,18.467,27.896,41.166,27.896
      s41.166-12.515,41.166-27.896C85.332,28.785,66.865,16.27,44.166,16.27z"
        stroke="currentColor"
        strokeWidth={4}
      />
      </g>
    </g>
  </svg>
);
```

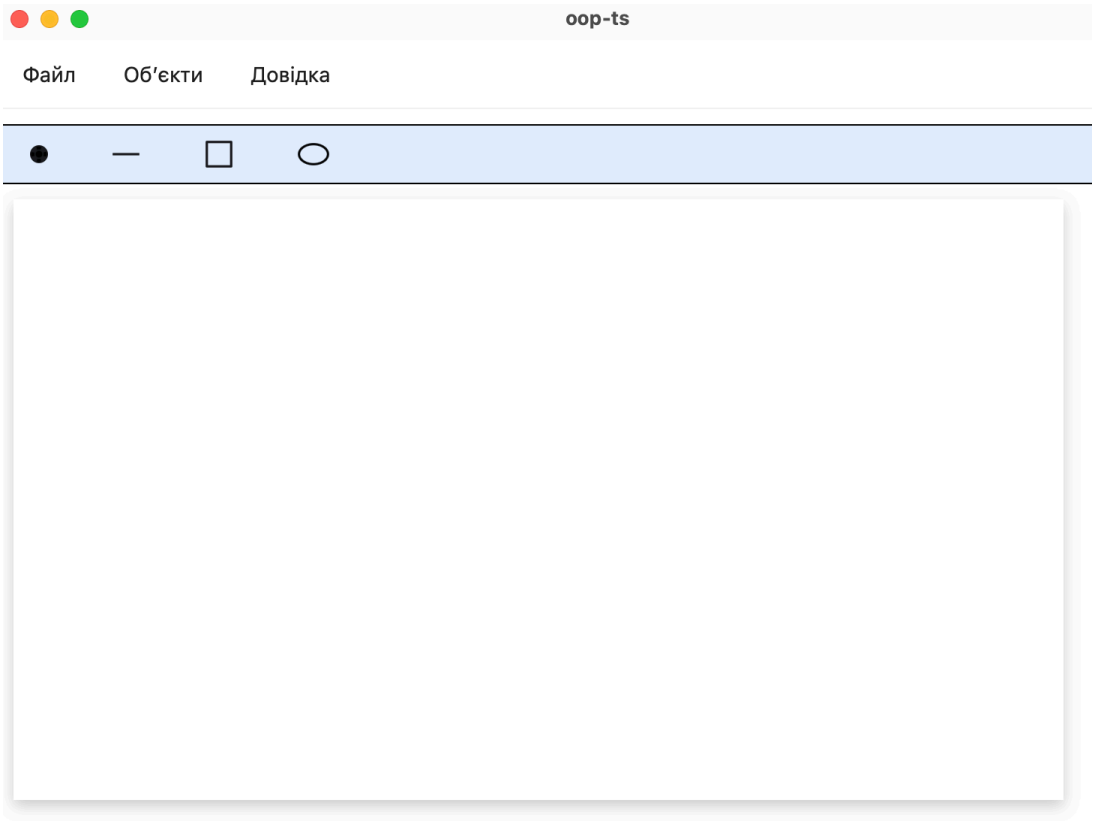
3. Модуль абстрактного класу Shape, додаткового інтерфейсу для реалізації гумового сліду та наслідуваними від нього класами

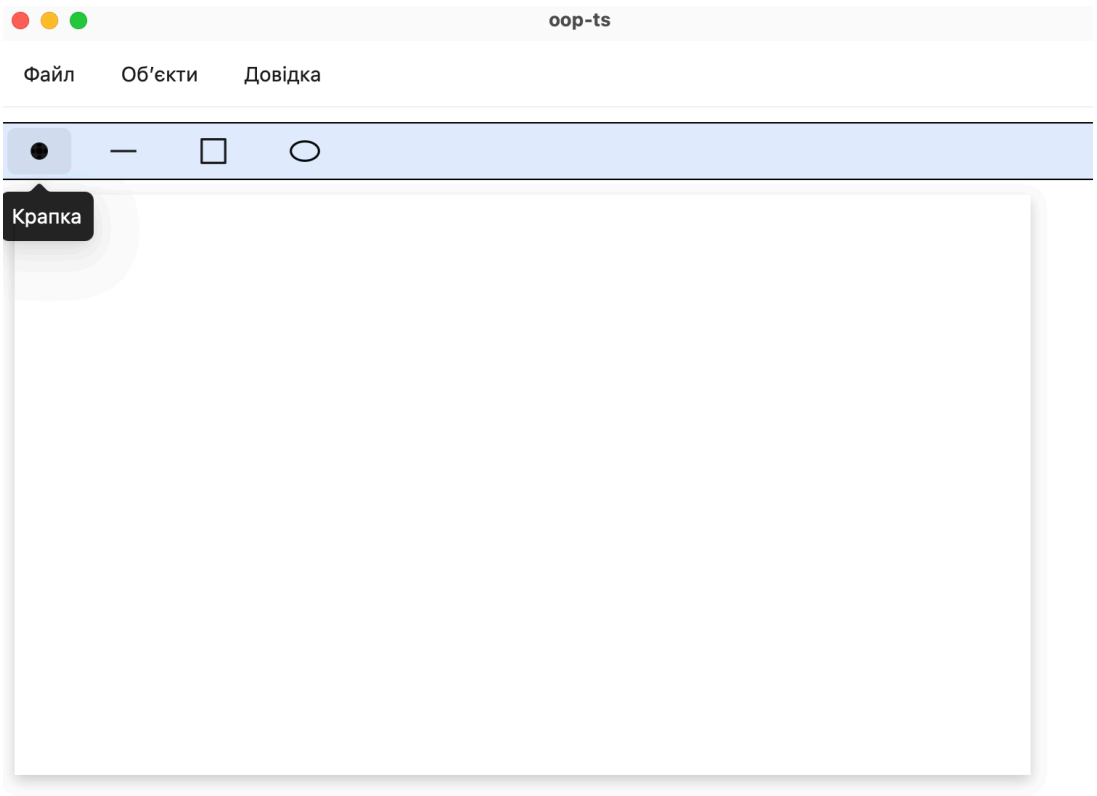
```
export abstract class Shape {
  protected color: string;
  constructor(color: string) {
    this.color = color;
  }

  abstract draw(ctx: CanvasRenderingContext2D): void;
}

export interface Previewable {
  drawPreview(ctx: CanvasRenderingContext2D): void;
}

export class Dot extends Shape {
  private x: number;
  private y: number;

  constructor(x: number, y: number, color: string = "blue") {
    super(color);
    this.x = x;
    this.y = y;
  }

  draw(ctx: CanvasRenderingContext2D): void {
    ctx.beginPath();
    ctx.arc(this.x, this.y, 5, 0, Math.PI * 2);
    ctx.fillStyle = this.color;
    ctx.fill();
  }
}

export class Line extends Shape {
  private startX: number;
  private startY: number;
  private endX: number;
  private endY: number;

  constructor(
    startX: number,
    startY: number,
    endX: number,
    endY: number,
    color: string = "blue"
  ) {
    super(color);
    this.startX = startX;
    this.startY = startY;
    this.endX = endX;
    this.endY = endY;
  }

  draw(ctx: CanvasRenderingContext2D): void {
    ctx.beginPath();
    ctx.moveTo(this.startX, this.startY);
    ctx.lineTo(this.endX, this.endY);
    ctx.strokeStyle = this.color;
    ctx.lineWidth = 2;
    ctx.stroke();
```

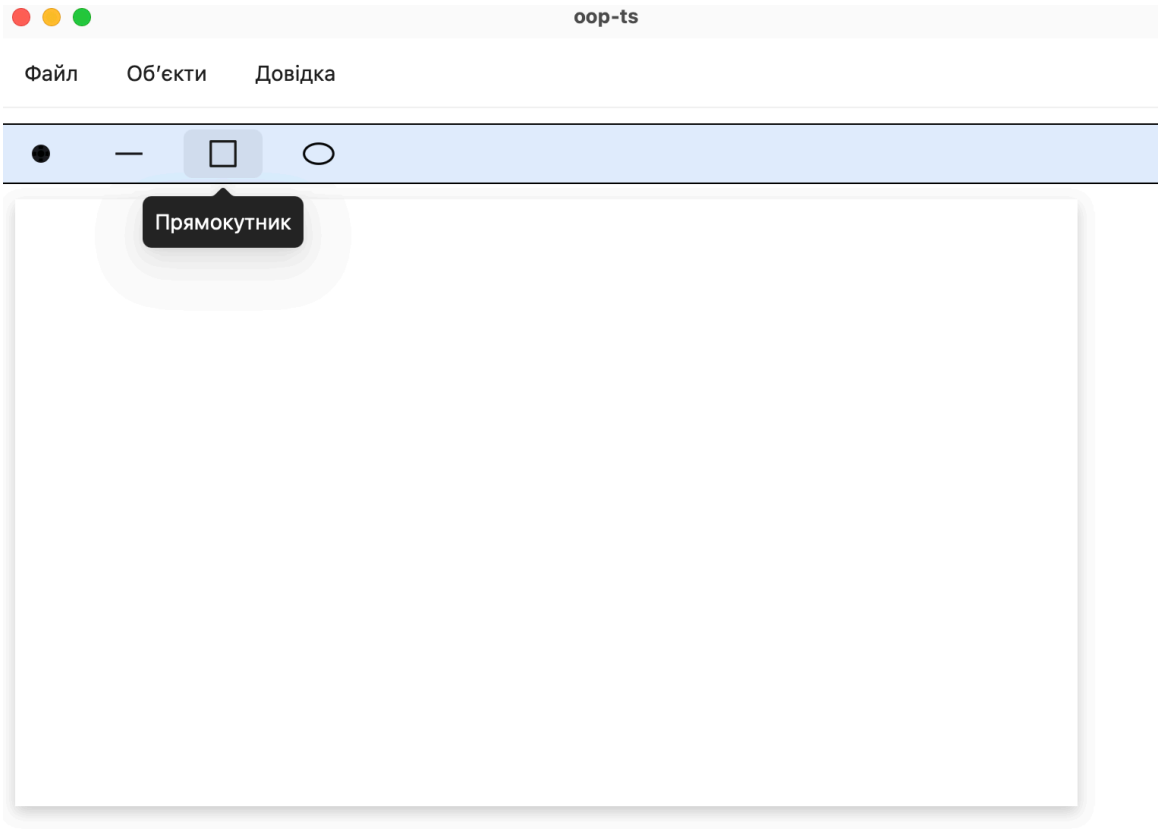
```
  }
}

export class Rectangle extends Shape implements Previewable {
  private x: number;
  private y: number;
  private width: number;
  private height: number;

  constructor(
    x: number,
    y: number,
    width: number,
    height: number,
    color: string = "yellow"
  ) {
    super(color);
    this.x = x;
    this.y = y;
    this.width = width;
    this.height = height;
  }

  draw(ctx: CanvasRenderingContext2D): void {
    ctx.fillStyle = this.color;
    ctx.fillRect(this.x, this.y, this.width, this.height);
    ctx.strokeStyle = "black";
    ctx.strokeRect(this.x, this.y, this.width, this.height);
  }

  drawPreview(ctx: CanvasRenderingContext2D): void {
    ctx.strokeStyle = "blue";
    ctx.strokeRect(this.x, this.y, this.width, this.height);
  }
}

export class Ellipse extends Shape implements Previewable {
  private x: number;
  private y: number;
  private radiusX: number;
  private radiusY: number;

  constructor(
    x: number,
    y: number,
    radiusX: number,
    radiusY: number,
    color: string = "grey"
  ) {
    super(color);
    this.x = x;
    this.y = y;
    this.radiusX = radiusX;
    this.radiusY = radiusY;
  }

  draw(ctx: CanvasRenderingContext2D): void {
    ctx.beginPath();
    ctx.ellipse(this.x, this.y, this.radiusX, this.radiusY, 0, 0, Math.PI * 2);
    ctx.fillStyle = this.color;
    ctx.fill();
    ctx.strokeStyle = "black";
    ctx.stroke();
  }
```

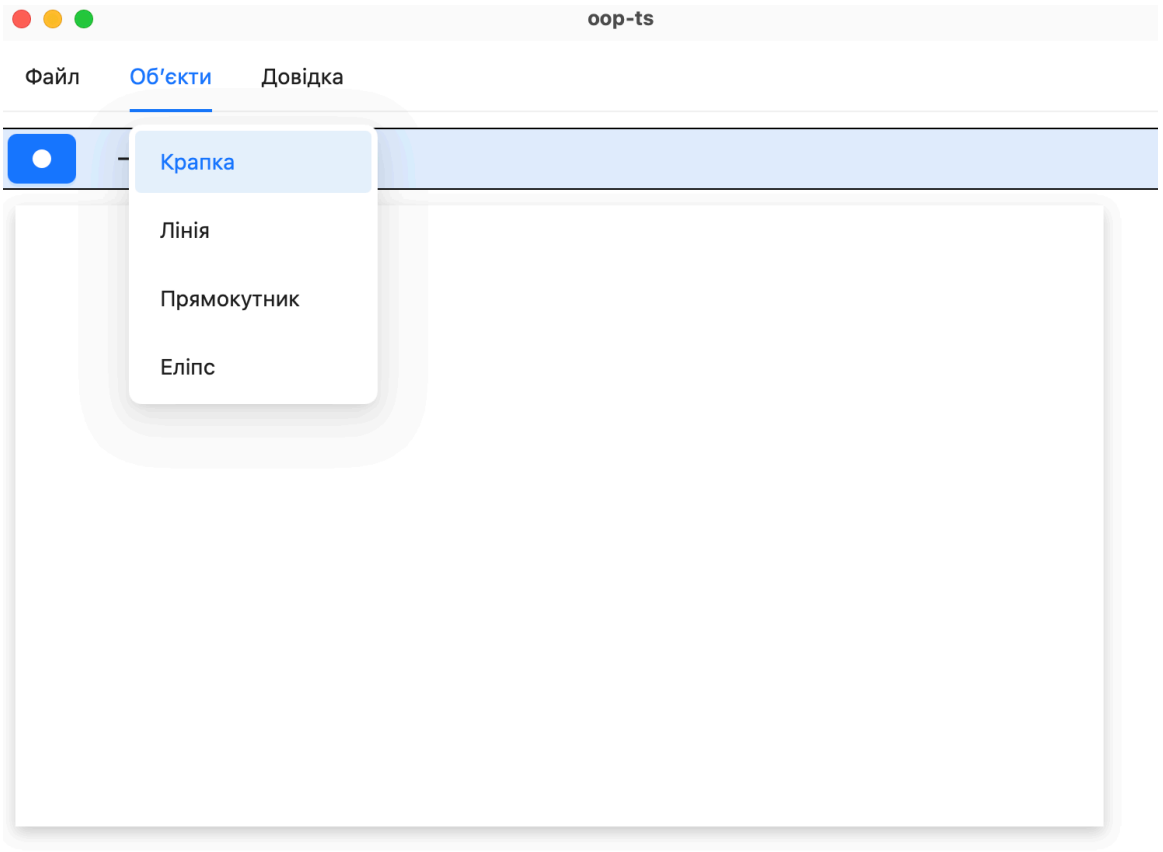
```
  drawPreview(ctx: CanvasRenderingContext2D): void {
    ctx.beginPath();
    ctx.ellipse(this.x, this.y, this.radiusX, this.radiusY, 0, 0, Math.PI * 2);
    ctx.strokeStyle = "blue";
    ctx.stroke();
  }
}
```

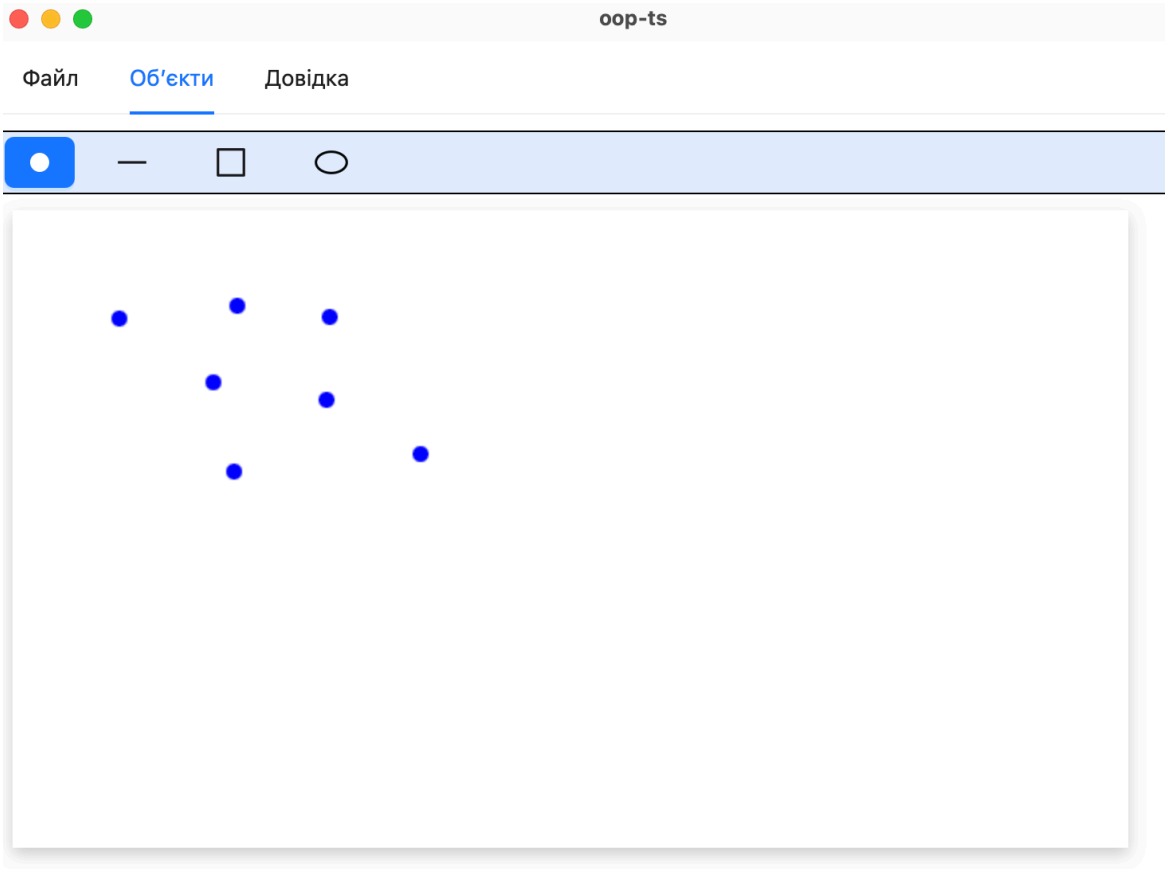
Скріншоти роботи виконаної програми:

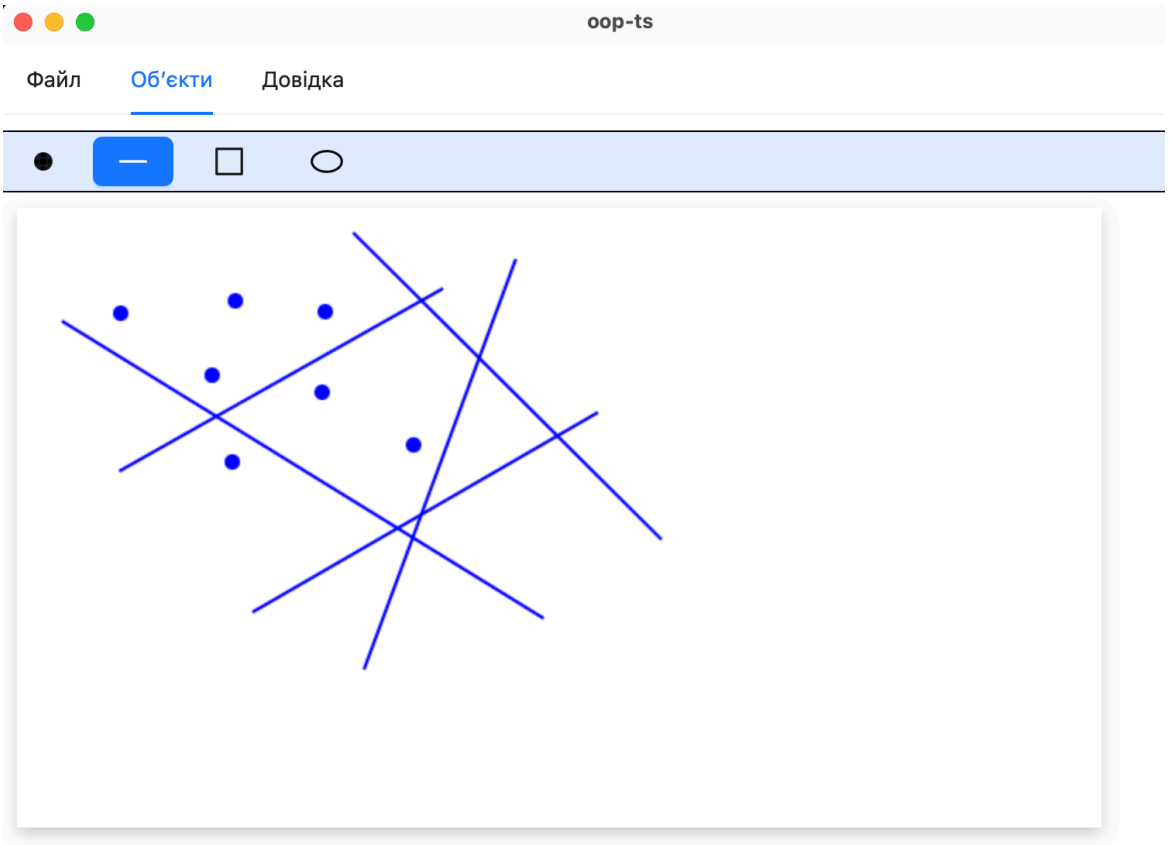


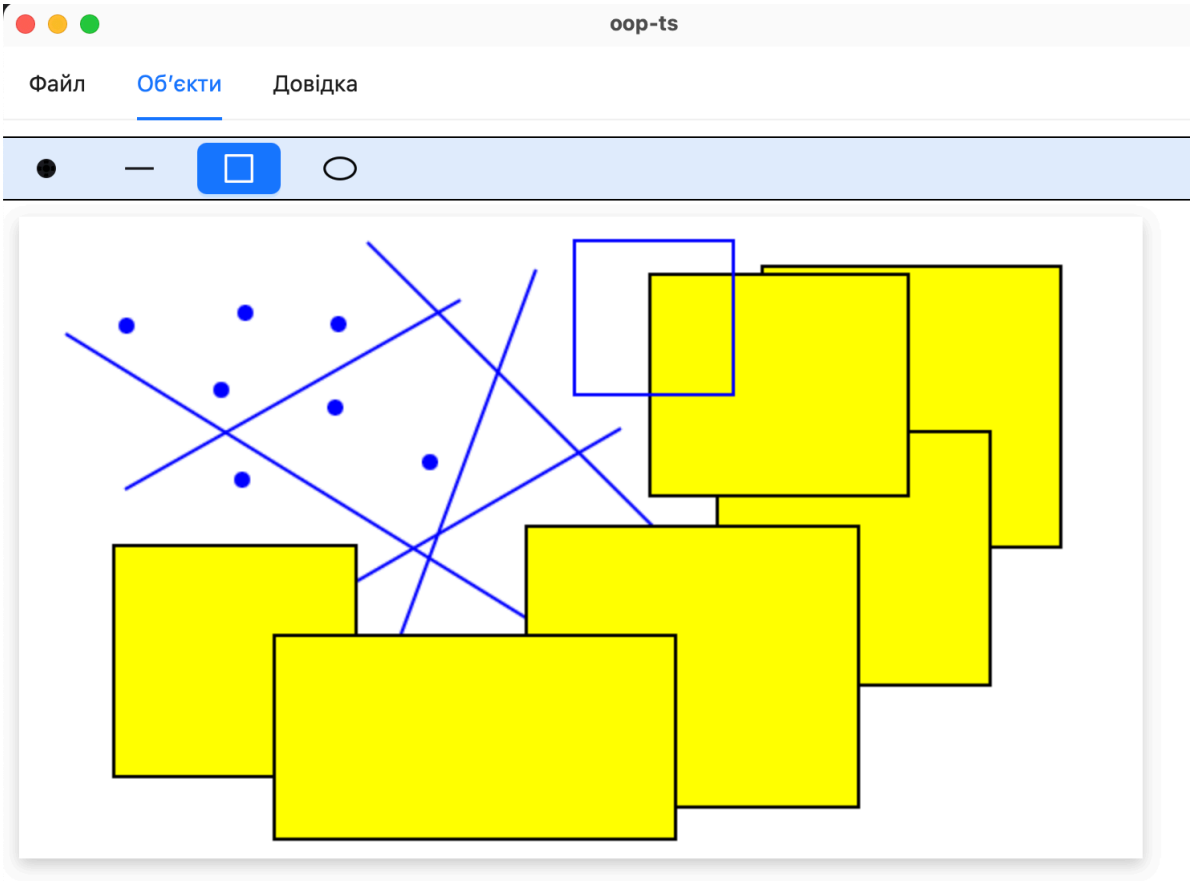


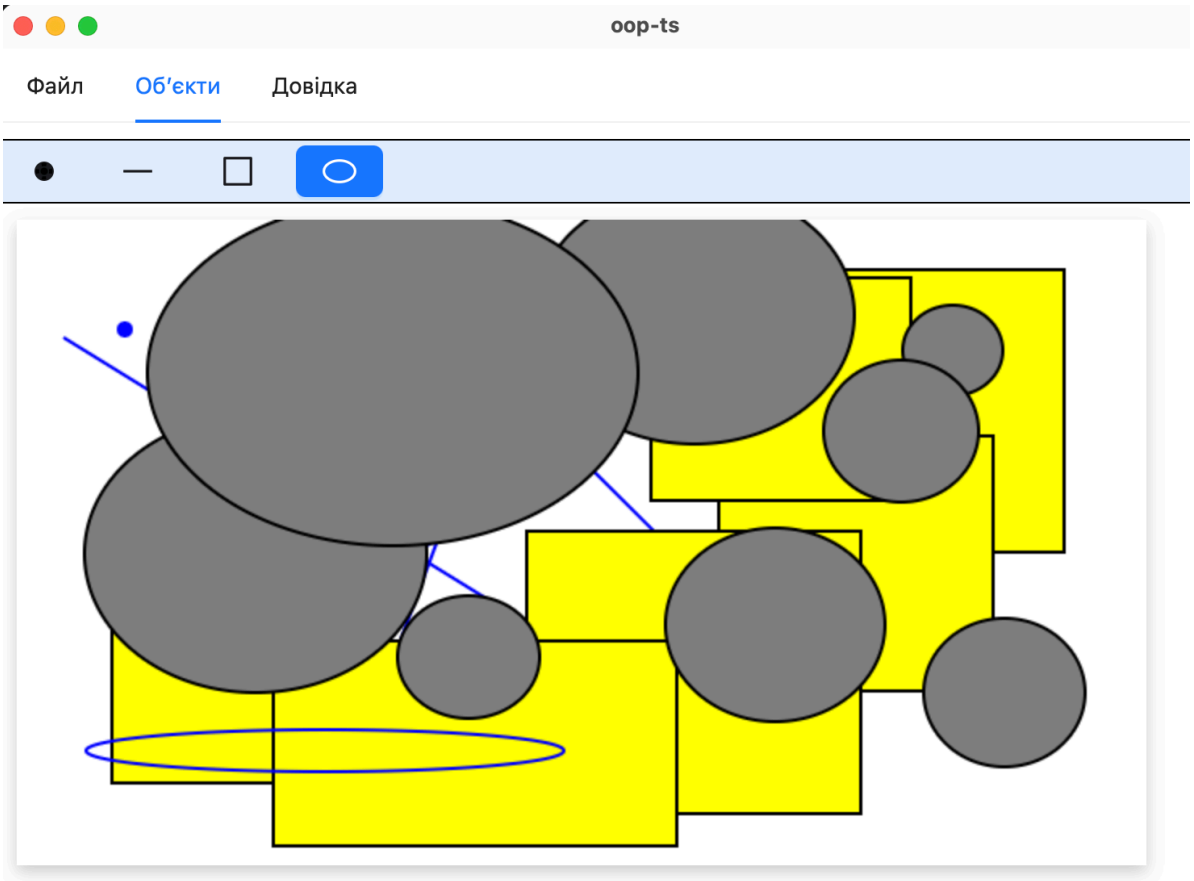

oop-ts
Файл
Об'єкти
Довідка
Прямокутник

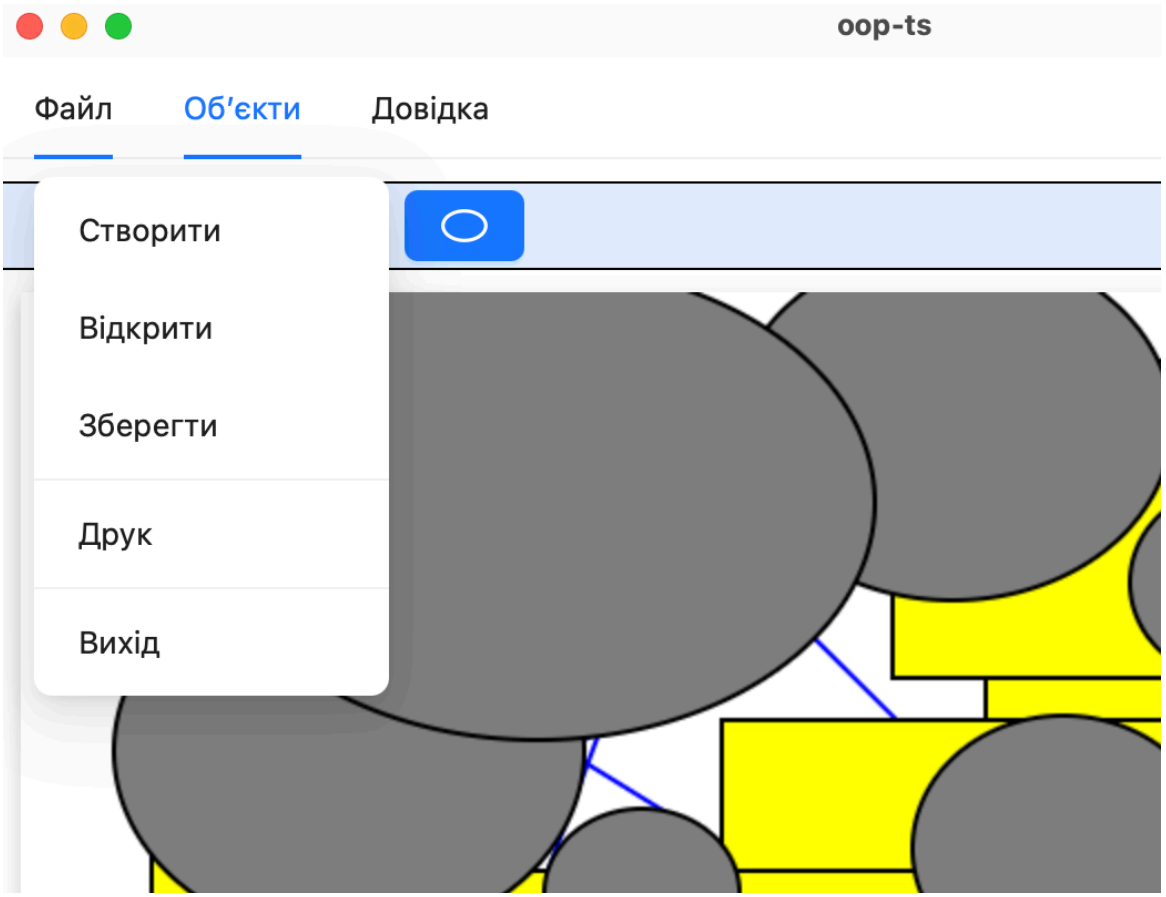

oop-ts
Файл
Об'єкти
Довідка
Крапка
Лінія
Прямокутник
Еліпс


oop-ts
Файл
Об'єкти
Довідка


oop-ts
Файл
Об'єкти
Довідка


oop-ts
Файл
Об'єкти
Довідка


oop-ts
Файл
Об'єкти
Довідка


oop-ts
Файл
Об'єкти
Довідка
Створити
Відкрити
Зберегти
Друк
Вихід


oop-ts
Файл
Об'єкти
Довідка
Крапка
Лінія
Прямокутник
Еліпс

Висновки:

Моя лабораторна робота виконана із використанням бібліотек для створення користувацьких інтерфейсів на Typescript із використанням об’єктно-орієнтованого підходу.

Спробувала виконати поставлену задачу згідно зі своїм варіантом. Вдосконалила вже наявний функціонал з минулої лабораторної роботи додавши додаткову панель з вибором елементів для малювання - або тулбар.

Також створила додатковий інтерфейс з методом для відображення гумового сліду в процесі малювання на холсті для прямокутника й еліпса. Такий підхід дав змогу лише розширити наявні методи дочірніх класів, не змінюючи батька, так як не для всіх таких елементів потрібен гумовий слід.