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»

Виконала:
студентка групи ІМ-43:
Козаченко Софія Олександрівна

Перевірив:
Порєв В. М.

Київ 2025

Варіант завдання :
Номер студента (Ж): 14
Кількість елементів масиву: 114
“Гумовий” слід: суцільна синя лінія
Введення прямокутника: за двома протилежними кутами
Введення еліпса: від центру до кута
Прямокутник: чорний контур без заповнення
Еліпс: чорний контур із кольоровим заповненням
Колір заповнення еліпса: блакитний
Позначка поточного об’єкта: відображається в меню

Текст програми :
main.js

```
const { app, BrowserWindow } = require('electron');
const path = require('path');

let mainWindow;

function createWindow() {
 mainWindow = new BrowserWindow({
   width: 800,
   height: 600,
   webPreferences: {
     preload: path.join(__dirname, 'preload.js')
   }
 });

 mainWindow.loadFile('index.html');
}

app.whenReady().then(() => {
 createWindow();

 app.on('activate', () => {
   if (BrowserWindow.getAllWindows().length === 0) createWindow();
 });
});

app.on('window-all-closed', () => {
 if (process.platform !== 'darwin') app.quit();
});
```

renderer.js

```
import { ShapeFactory } from "./editors/ShapeFactory.js";

const canvas = document.getElementById("c1");
const ctx = canvas.getContext("2d");

let pcshape = new Array(114);
let count = 0;
let currentShape = "Point";
let preview = null;
let startPoint = null;

function redraw() {
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   for (let i = 0; i < count; i++) {
       pcshape[i].draw(ctx);
   }
   if (preview) {
       preview.preview(ctx);
   }
}

canvas.addEventListener("mousedown", (e) => {
   startPoint = { x: e.offsetX, y: e.offsetY };
});

canvas.addEventListener("mousemove", (e) => {
   if (!startPoint) return;
   preview = ShapeFactory.create(
       currentShape,
       startPoint.x,
       startPoint.y,
       e.offsetX,
       e.offsetY
   );
   redraw();
});

canvas.addEventListener("mouseup", (e) => {
   if (!startPoint) return;
   const shape = ShapeFactory.create(
       currentShape,
       startPoint.x,
       startPoint.y,
       e.offsetX,
       e.offsetY
```

```
    );
    pcshape[count++] = shape;
    startPoint = null;
    preview = null;
    redraw();
});

document.querySelectorAll(".menu-object").forEach(item => {
    item.addEventListener("click", (e) => {
        currentShape = e.target.dataset.shape;
        document.querySelectorAll(".menu-object").forEach(m =>
m.classList.remove("active"));
        e.target.classList.add("active");
    });
});

document.getElementById("clearCanvas").addEventListener("click", () => {
    ctx.clearRect(0, 0, canvas.width, canvas.height);
    pcshape = new Array(114);
    count = 0;
    preview = null;
    startPoint = null;
});

document.getElementById("about-btn").addEventListener("click", () => {
 alert(`Graphic Editor - Lab 2 (Variant 14)
Student number (Ж): 14
Number of shapes: 114
Rubber band: solid blue line
Rectangle input: by two opposite corners
Ellipse input: from center to corner
Rectangle: black outline, no fill
Ellipse: black outline, colored fill
Ellipse fill color: light blue
Current object indicator: shown in the menu.`);
});
```

index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
 <meta charset="UTF-8">
 <title>Lab 2 - Variant 14</title>
```

```
<style>
  body {
      font-family: Arial, sans-serif;
      margin: 10px;
  }
  canvas {
      border: 1px solid black;
      margin-top: 10px;
  }
  .menu {
      list-style: none;
      display: flex;
      gap: 20px;
      padding: 0;
      margin: 0 0 10px 0;
      background: #f1f1f1;
  }

  .menu > li {
      position: relative;
      padding: 10px;
      cursor: pointer;
  }

  .menu ul {
      display: none;
      list-style: none;
      position: absolute;
      left: 0;
      top: 100%;
      background: white;
      border: 1px solid #ccc;
      padding: 0;
      margin: 0;
      width: 150px;
  }

  .menu li:hover ul {
       display: block;
  }

  .menu ul li {
      padding: 8px 10px;
      cursor: pointer;
  }
```

```
   .menu ul li:hover {
       background-color: #e0e0e0;
   }

   .menu-object.active {
        font-weight: bold;
       color: blue;
   }
 </style>
</head>
<body>
 <ul class="menu">
   <li>
     File
     <ul>
       <li id="clearCanvas">Clear the field</li>
     </ul>
   </li>
   <li>
     Objects
     <ul>
       <li class="menu-object active" data-shape="Point">Point</li>
       <li class="menu-object" data-shape="Line">Line</li>
       <li class="menu-object" data-shape="Rectangle">Rectangle</li>
       <li class="menu-object" data-shape="Ellipse">Ellipse</li>
     </ul>
   </li>
   <li>
     Help
     <ul>
       <li id="about-btn">About</li>
     </ul>
   </li>
 </ul>

 <canvas id="c1" width="800" height="600"></canvas>

 <script type="module" src="renderer.js"></script>
</body>
</html>
```

shapes/Ellips.js

```
import { Shape } from "./Shape.js";
```

```
export class Ellipse extends Shape {
   constructor(cx, cy, rx, ry) {
       super();
       this.cx = cx;
       this.cy = cy;
       this.rx = rx;
       this.ry = ry;
   }

   draw(ctx) {
       ctx.beginPath();
       ctx.ellipse(this.cx, this.cy, this.rx, this.ry, 0, 0, Math.PI * 2);
       ctx.strokeStyle = "black";
       ctx.stroke();
       ctx.fillStyle = "lightgreen";
       ctx.fill();
   }

   preview(ctx) {
       ctx.beginPath();
       ctx.ellipse(this.cx, this.cy, this.rx, this.ry, 0, 0, Math.PI * 2);
       ctx.strokeStyle = "blue";
       ctx.stroke();
   }
}
```

shapes/Line.js

```
import { Shape } from "./Shape.js";

export class Line extends Shape {
   constructor(x1, y1, x2, y2) {
       super();
       this.x1 = x1;
       this.y1 = y1;
       this.x2 = x2;
       this.y2 = y2;
   }

   draw(ctx) {
       ctx.strokeStyle = "black";
       ctx.beginPath();
       ctx.moveTo(this.x1, this.y1);
       ctx.lineTo(this.x2, this.y2);
       ctx.stroke();
   }
```

```
    preview(ctx) {
        ctx.strokeStyle = "blue";
        ctx.beginPath();
        ctx.moveTo(this.x1, this.y1);
        ctx.lineTo(this.x2, this.y2);
        ctx.stroke();
    }
}
```

shapes/Point.js

```
import { Shape } from "./Shape.js";

export class Point extends Shape {
    constructor(x, y) {
        super();
        this.x = x;
        this.y = y;
    }

    draw(ctx) {
        ctx.fillStyle = "black";
        ctx.beginPath();
        ctx.arc(this.x, this.y, 2, 0, Math.PI * 2);
        ctx.fill();
    }

    preview(ctx) {
        ctx.strokeStyle = "blue";
        ctx.beginPath();
        ctx.arc(this.x, this.y, 2, 0, Math.PI * 2);
        ctx.stroke();
    }
}
```

shapes/Rectangle.js

```
import { Shape } from "./Shape.js";

export class Rectangle extends Shape {
    constructor(x1, y1, x2, y2) {
        super();
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
```

```
        this.y2 = y2;
    }

    draw(ctx) {
        const w = this.x2 - this.x1;
        const h = this.y2 - this.y1;
        ctx.strokeStyle = "black";
        ctx.strokeRect(this.x1, this.y1, w, h);
    }

    preview(ctx) {
        const w = this.x2 - this.x1;
        const h = this.y2 - this.y1;
        ctx.strokeStyle = "blue";
        ctx.strokeRect(this.x1, this.y1, w, h);
    }
}
```

shapes/Shape.js

```
export class Shape {
    constructor() {}

    draw(ctx) {
        throw new Error("draw() must be implemented in subclass");
    }

    preview(ctx) {
        throw new Error("preview() must be implemented in subclass");
    }
}
```

editors/EllipseEditor.js

```
import { ShapeEditor } from "./ShapeEditor.js";
import { Ellipse } from "../shapes/Ellipse.js";

export class EllipseEditor extends ShapeEditor {
    createShape(x1, y1, x2, y2) {
        const rx = Math.abs(x2 - x1);
        const ry = Math.abs(y2 - y1);
        return new Ellipse(x1, y1, rx, ry);
    }
}
```

editors/LineEditor.js

```
import { ShapeEditor } from "./ShapeEditor.js";
import { Line } from "../shapes/Line.js";

export class LineEditor extends ShapeEditor {
   createShape(x1, y1, x2, y2) {
       return new Line(x1, y1, x2, y2);
   }
}
```

editors/PointEditor.js

```
import { ShapeEditor } from "./ShapeEditor.js";
import { Point } from "../shapes/Point.js";

export class PointEditor extends ShapeEditor {
   createShape(x1, y1) {
       return new Point(x1, y1);
   }
}
```

editors/RectangleEditor.js

```
import { ShapeEditor } from "./ShapeEditor.js";
import { Rectangle } from "../shapes/Rectangle.js";

export class RectangleEditor extends ShapeEditor {
   createShape(x1, y1, x2, y2) {
       return new Rectangle(x1, y1, x2, y2);
   }
}
```

editors/ShapeEditor.js

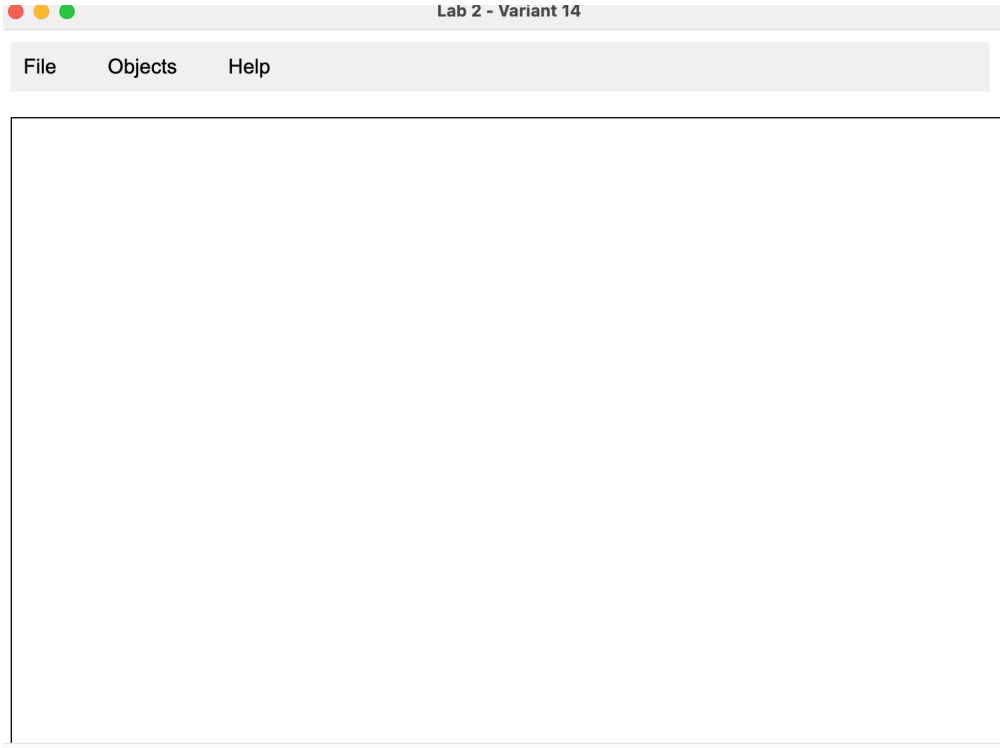
```
export class ShapeEditor {
   createShape(x1, y1, x2, y2) {
       throw new Error("createShape() must be implemented in subclass");
   }
}
```

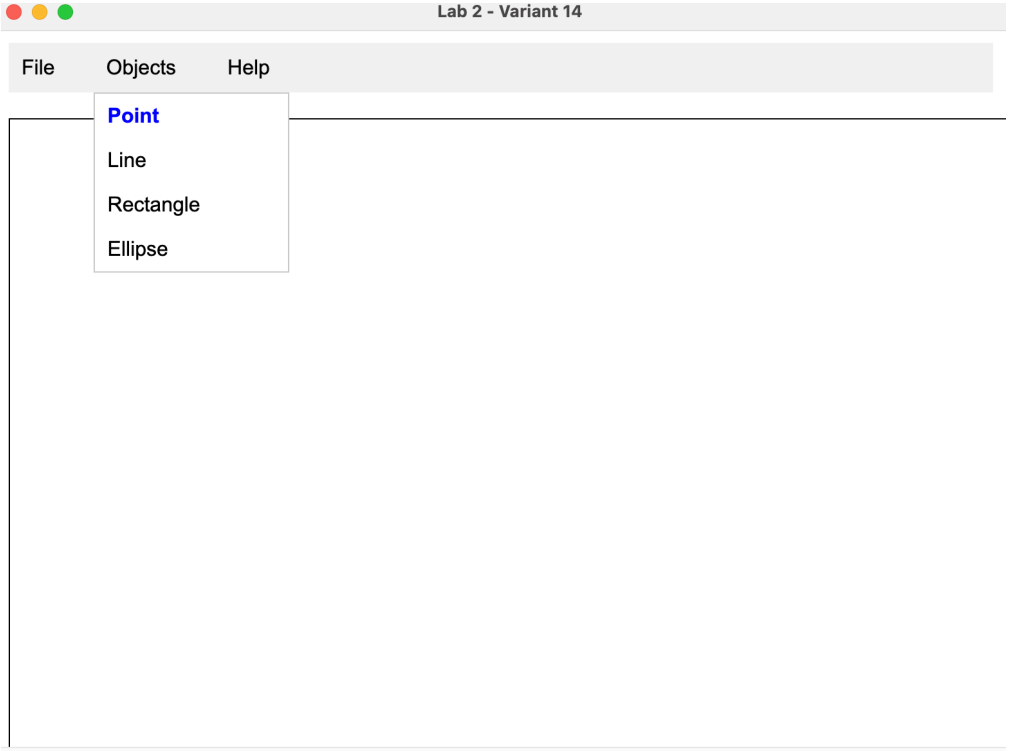
editors/ShapeFactory.js

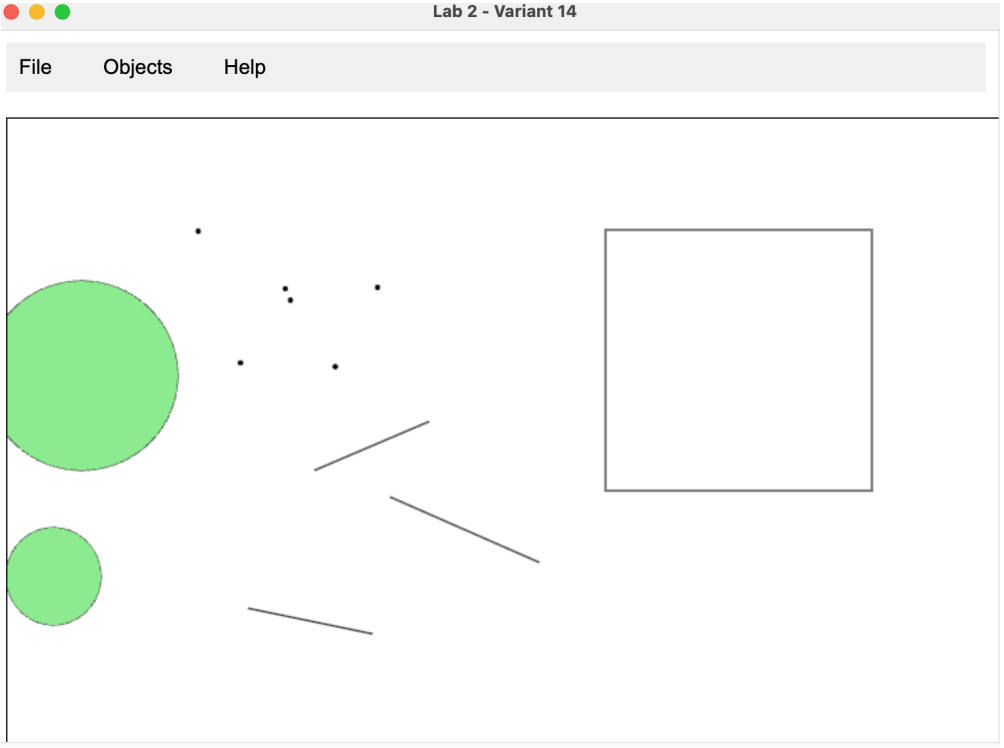
```
import { PointEditor } from "./PointEditor.js";
import { LineEditor } from "./LineEditor.js";
import { RectangleEditor } from "./RectangleEditor.js";
import { EllipseEditor } from "./EllipseEditor.js";
```

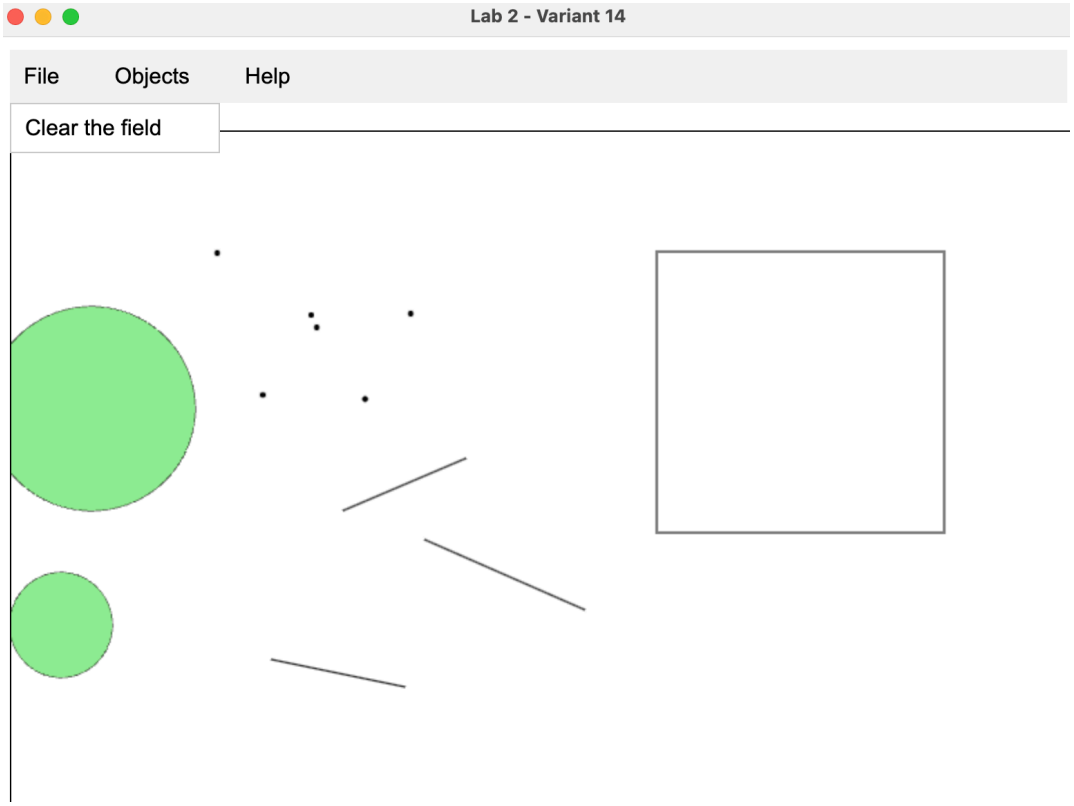
```
export class ShapeFactory {
   static editors = {
       Point: new PointEditor(),
       Line: new LineEditor(),
       Rectangle: new RectangleEditor(),
       Ellipse: new EllipseEditor()
   };

   static create(type, x1, y1, x2, y2) {
       const editor = this.editors[type];
       if (!editor) throw new Error(`Unknown shape type: ${type}`);
       return editor.createShape(x1, y1, x2, y2);
   }
}
```

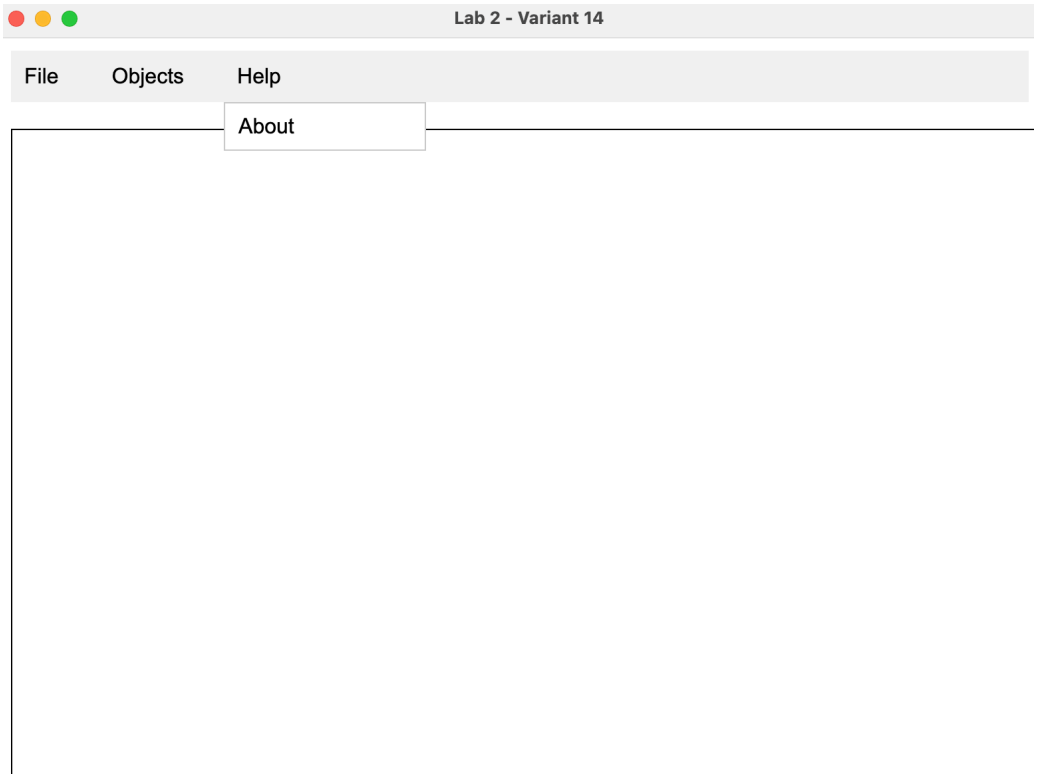
Скріншоти тестування програми :

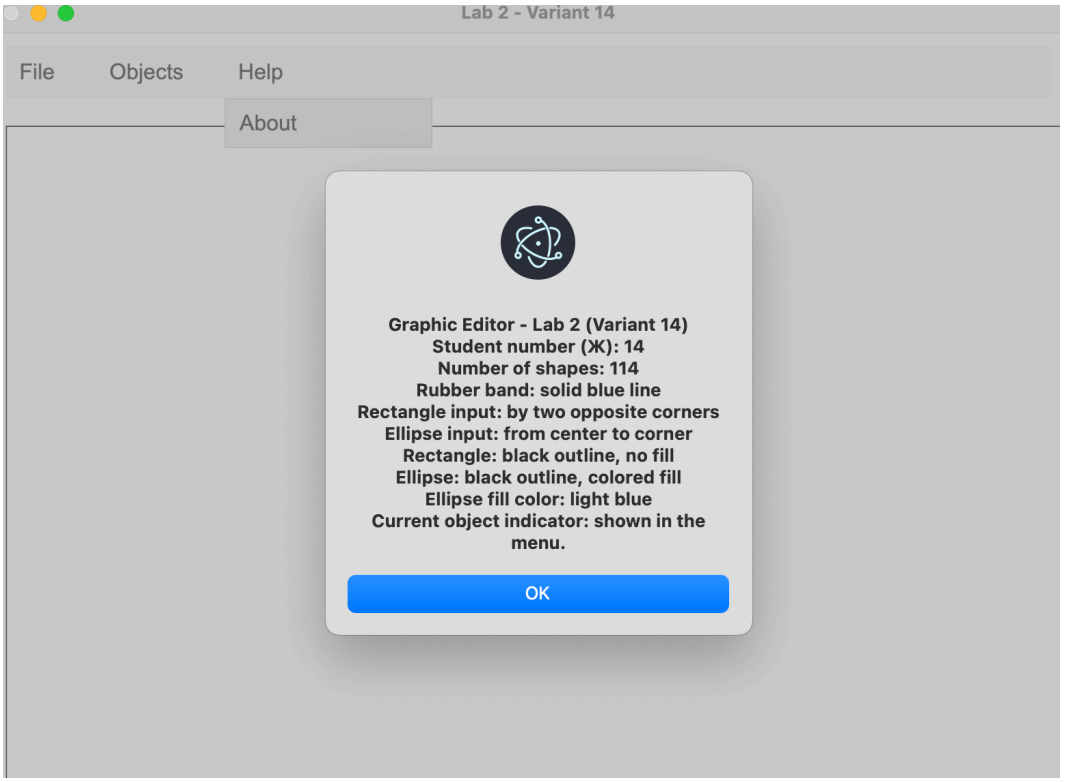



Lab 2 - Variant 14
File
Objects
Help
Point
Line
Rectangle
Ellipse

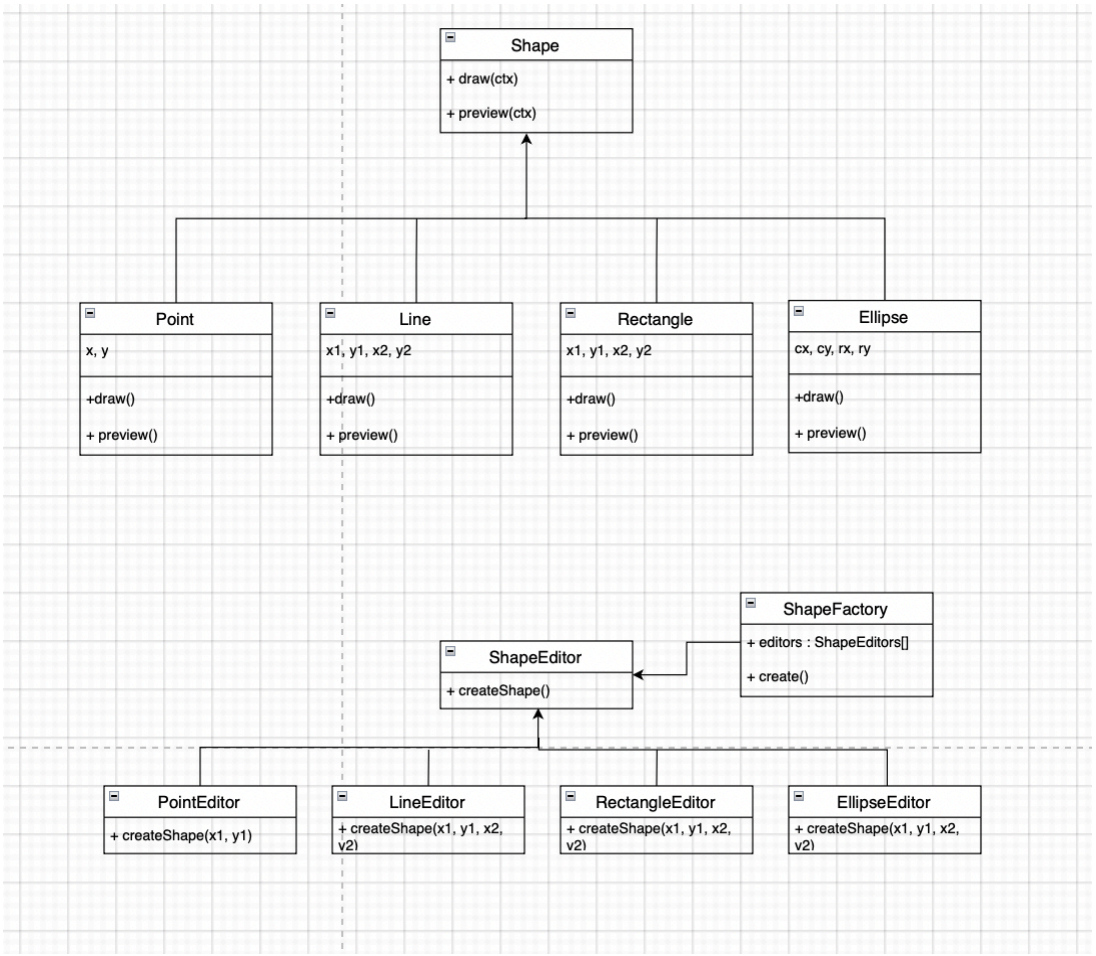

Lab 2 - Variant 14
File
Objects
Help


Lab 2 - Variant 14
File
Objects
Help
Clear the field


Lab 2 - Variant 14
File
Objects
Help
About



Діаграми класів :



Висновок :

У ході виконання лабораторної роботи №2 «Розробка графічного редактора об’єктів» було створено програму, яка дозволяє користувачеві взаємодіяти з графічним полотном та будувати геометричні фігури — крапку, лінію, прямокутник і еліпс — відповідно до умов варіанту №14.

Ієрархія ООП та Поліморфізм: Створено базовий клас Shape та його нащадків (Point, Line, Rectangle, Ellipse), які реалізують поліморфні методи draw() та preview(). Патерн Фабрика (Factory): Використано клас ShapeFactory та допоміжні *Editor класи для централізованого та гнучкого створення об'єктів-фігур.

Для реалізації графічного інтерфейсу було використано платформу Electron. Такий вибір зумовлений тим, що основна розробка велась на macOS. Electron дозволив відтворити необхідну логіку роботи графічного редактора за всіма умовами завдання.

Отже, у результаті роботи було створено функціональний графічний редактор, який відповідає вимогам варіанту №14, реалізує принципи ООП та демонструє використання сучасних інструментів розробки — Electron і JavaScript — для побудови настільних графічних застосунків.