

**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»  
на тему  
“Розробка графічного редактора об'єктів на C++”

Виконав:  
Студент групи ІМ-31  
Максимовський Назар Русланович  
номер у списку групи: 13

Перевірів:  
Порєв В.М.

Київ 2024

## Мета

Отримати вміння та навички використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм на основі класів C++, запрограмувавши простий графічний редактор в об'єктно-орієнтованому стилі.

## Завдання

1. Створити у середовищі MS Visual Studio C++ проект типу Windows Desktop Application з ім'ям **Lab2**.
2. Скомпілювати проект і отримати виконуваний файл програми.
3. Перевірити роботу програми. Налаштувати програму.
4. Проаналізувати та прокоментувати результати та вихідний текст програми.
5. Оформити звіт.

### Умови завдання за варіантом (Ж = 13):

- Масив вказівників для динамічних об'єктів типу Shape: статичний масив для Shape обсягом 113 елементів ( $13 \bmod 3 = 1$ )
- "Гумовий" слід при вводі об'єктів: суцільна лінія червоного кольору ( $13 \bmod 4 = 1$ )
- Увід прямокутника: від центру до одного з кутів ( $13 \bmod 2 = 1$ )
- Відображення прямокутника: чорний контур прямокутника без заповнення ( $13 \bmod 5 = 3$ )
- Увід еліпса: по двом протилежним кутам охоплюючого прямокутника ( $13 \bmod 2 = 1$ )
- Відображення еліпса: чорний контур з кольоровим заповненням ( $13 \bmod 5 = 3$ )
- Колір заповнення еліпса: жовтий ( $13 \bmod 6 = 1$ )
- Позначка поточного типу об'єкту, що вводиться: в заголовку вікна ( $13 \bmod 2 = 1$ )

## Вихідні тексти файлів програми

### Lab2.kt

```
package com.oop.lab2

import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.view.View
import android.widget.PopupMenu
import android.widget.Toast
```

```

import androidx.appcompat.app.AppCompatActivity
import androidx.core.view.forEach
import com.oop.lab2.shape_editor.ShapeObjectsEditor
import com.oop.lab2.shape_editor.ShapeObjectsEditorInterface

class Lab2 : AppCompatActivity() {
    private var objectsPopupMenu: PopupMenu? = null
    private val shapeObjectsEditor: ShapeObjectsEditorInterface =
        ShapeObjectsEditor()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.main_activity)

        findViewById<PaintView>(R.id.paint_view).apply {
            shapeObjectsEditor = this@Lab2.shapeObjectsEditor
            this@Lab2.shapeObjectsEditor.paintView = this
        }
    }

    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        menuInflater.inflate(R.menu.actionbar_menu, menu)
        return true
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when (item.itemId) {
            R.id.btn_file -> showToast("Ви натиснули кнопку \"Файл\"")
            R.id.btn_objects -> handleObjectsMenu()
            R.id.btn_info -> showToast("Ви натиснули кнопку \"Довідка\"")
            else -> return super.onOptionsItemSelected(item)
        }
        return true
    }

    private fun handleObjectsMenu() {
        if (objectsPopupMenu == null) {
            objectsPopupMenu =
                createObjectsPopupMenu(findViewById(R.id.btn_objects))
        }
        objectsPopupMenu?.show()
    }

    private fun createObjectsPopupMenu(anchor: View): PopupMenu {
        return PopupMenu(this, anchor).apply {
            menuInflater.inflate(R.menu.objects_popup_menu, menu)
            setOnMenuItemClickListener { menuItem ->
                menu.forEach { it.isChecked = false }
                menuItem.isChecked = true
                handleObjectSelection(menuItem.itemId)
            }
        }
    }

    private fun handleObjectSelection(itemId: Int): Boolean {
        val titleResId = when (itemId) {
            R.id.btn_point -> R.string.point.also {
                shapeObjectsEditor.startPointEditor()
            }
            R.id.btn_line -> R.string.line.also {
                shapeObjectsEditor.startLineEditor()
            }
            R.id.btn_rectangle -> R.string.rectangle.also {
                shapeObjectsEditor.startRectEditor()
            }
            R.id.btn_ellipse -> R.string.ellipse.also {
                shapeObjectsEditor.startEllipseEditor()
            }
        }
    }

```

```

        else -> return false
    }
    supportActionBar?.title = getString(titleResId)
    return true
}

private fun showToast(message: String) {
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show()
}
}

```

## PaintViewInterface.kt

```

package com.oop.lab2

import android.graphics.Canvas
import android.graphics.Paint
import com.oop.lab2.shape_editor.ShapeObjectsEditorInterface

interface PaintViewInterface {
    val paint: Paint
    var shapeObjectsEditor: ShapeObjectsEditorInterface
    val drawnShapesCanvas: Canvas
    val rubberTraceCanvas: Canvas

    fun repaint()

    fun clearCanvas(canvas: Canvas)
}

```

## PaintView.kt

```

package com.oop.lab2

import android.content.Context
import android.graphics.Bitmap
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.graphics.PorterDuff
import android.util.AttributeSet
import android.view.MotionEvent
import android.view.View
import com.oop.lab2.shape_editor.ShapeObjectsEditorInterface

class PaintView(context: Context, attrs: AttributeSet?) :
    View(context, attrs),
    PaintViewInterface {

    override val paint = Paint().apply {
        isAntiAlias = true
        strokeWidth = 7f
    }

    override lateinit var shapeObjectsEditor: ShapeObjectsEditorInterface
    override lateinit var drawnShapesCanvas: Canvas
    override lateinit var rubberTraceCanvas: Canvas
}

```

```

private var drawnShapesBitmap: Bitmap? = null
private var rubberTraceBitmap: Bitmap? = null

override fun onMeasure(widthMeasureSpec: Int, heightMeasureSpec: Int) {
    super.onMeasure(widthMeasureSpec, heightMeasureSpec)
    val width = MeasureSpec.getSize(widthMeasureSpec)
    val height = MeasureSpec.getSize(heightMeasureSpec)

    if (drawnShapesBitmap == null || rubberTraceBitmap == null) {
        initializeBitmaps(width, height)
    }
}

private fun initializeBitmaps(width: Int, height: Int) {
    drawnShapesBitmap = Bitmap.createBitmap(width, height,
        Bitmap.Config.ARGB_8888).apply {
        drawnShapesCanvas = Canvas(this)
    }
    rubberTraceBitmap = Bitmap.createBitmap(width, height,
        Bitmap.Config.ARGB_8888).apply {
        rubberTraceCanvas = Canvas(this)
    }
}

override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    canvas.drawBitmap(drawnShapesBitmap!!, 0f, 0f, null)

    if (shapeObjectsEditor.isRubberTraceModeOn) {
        canvas.drawBitmap(rubberTraceBitmap!!, 0f, 0f, null)
    } else {
        shapeObjectsEditor.onPaint()
    }
}

override fun onTouchEvent(event: MotionEvent): Boolean {
    val (x, y) = event.x to event.y
    when (event.action) {
        MotionEvent.ACTION_DOWN -> shapeObjectsEditor.onFingerTouch(x, y)
        MotionEvent.ACTION_MOVE -> shapeObjectsEditor.onFingerMove(x, y)
        MotionEvent.ACTION_UP -> shapeObjectsEditor.onFingerRelease()
    }
    return true
}

override fun repaint() {
    invalidate()
}

override fun clearCanvas(canvas: Canvas) {
    canvas.drawColor(Color.TRANSPARENT, PorterDuff.Mode.CLEAR)
}
}

```

## ShapeObjectsEditorInterface.kt

```

package com.oop.lab2.shape_editor

import com.oop.lab2.PaintViewInterface

interface ShapeObjectsEditorInterface {
    var isRubberTraceModeOn: Boolean
}

```

```

var paintView: PaintViewInterface

fun startPointEditor()

fun startLineEditor()

fun startRectEditor()

fun startEllipseEditor()

fun onFingerTouch(x: Float, y: Float)

fun onFingerMove(x: Float, y: Float)

fun onFingerRelease()

fun onPaint()
}

```

## ShapeObjectsEditor.kt

```

package com.oop.lab2.shape_editor

import com.oop.lab2.PaintViewInterface
import com.oop.lab2.editor.ShapeEditor
import com.oop.lab2.editor.PointShapeEditor
import com.oop.lab2.editor.LineShapeEditor
import com.oop.lab2.editor.RectShapeEditor
import com.oop.lab2.editor.EllipseShapeEditor
import com.oop.lab2.shape.Shape
import com.oop.lab2.shape.PointShape
import com.oop.lab2.shape.LineShape
import com.oop.lab2.shape.RectShape
import com.oop.lab2.shape.EllipseShape

class ShapeObjectsEditor: ShapeObjectsEditorInterface {
    override var isRubberTraceModeOn = false
    override lateinit var paintView: PaintViewInterface

    private val drawnShapes = mutableListOf<Shape>()
    private lateinit var activeEditor: ShapeEditor
    private lateinit var currentShape: Shape

    private fun startShapeEditor(editor: ShapeEditor, shape: Shape) {
        activeEditor = editor
        currentShape = shape
    }

    override fun startPointEditor() {
        startShapeEditor(PointShapeEditor(), PointShape())
    }

    override fun startLineEditor() {
        startShapeEditor(LineShapeEditor(), LineShape())
    }

    override fun startRectEditor() {
        startShapeEditor(RectShapeEditor(), RectShape())
    }

    override fun startEllipseEditor() {
        startShapeEditor(EllipseShapeEditor(), EllipseShape())
    }
}

```

```

    }

    override fun onFingerTouch(x: Float, y: Float) {
        activeEditor.shape = currentShape
        activeEditor.onFingerTouch(x, y)
        isRubberTraceModeOn = true
    }

    override fun onFingerMove(x: Float, y: Float) {
        clearCanvas()
        activeEditor.onFingerMove(paintView.rubberTraceCanvas,
        paintView.paint, x, y)
        paintView.repaint()
    }

    override fun onFingerRelease() {
        activeEditor.onFingerRelease(drawnShapes)
        isRubberTraceModeOn = false
        paintView.repaint()
    }

    override fun onPaint() {
        clearCanvas()
        drawnShapes.forEach { it.show(paintView.drawnShapesCanvas,
        paintView.paint) }
    }

    private fun clearCanvas() {
        paintView.clearCanvas(paintView.rubberTraceCanvas)
        paintView.clearCanvas(paintView.drawnShapesCanvas)
    }
}

```

## Shape.kt

```

package com.oop.lab2.shape

import android.graphics.Canvas
import android.graphics.Paint

abstract class Shape {
    protected var startX = 0F
    protected var startY = 0F
    protected var endX = 0F
    protected var endY = 0F

    protected abstract val defaultOutlineColor: Int
    protected abstract val defaultFillingColor: Int
    protected var outlineColor: Int? = null
    protected var fillingColor: Int? = null

    fun setStart(x: Float, y: Float) {
        startX = x
        startY = y
    }

    fun setEnd(x: Float, y: Float) {
        endX = x
        endY = y
    }

    fun setColors(outlineColor: Int, fillingColor: Int) {

```

```

        this.outlineColor = outlineColor
        this.fillingColor = fillingColor
    }

    fun getOutlineColor(): Int {
        return outlineColor ?: defaultOutlineColor
    }

    fun getFillingColor(): Int {
        return fillingColor ?: defaultFillingColor
    }

    abstract fun show(canvas: Canvas, paint: Paint)
}

```

## PointShape.kt

```

package com.oop.lab2.shape

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint

class PointShape: Shape() {
    override val defaultOutlineColor = Color.argb(255, 0, 0, 0)
    override val defaultFillingColor = Color.TRANSPARENT

    override fun show(canvas: Canvas, paint: Paint) {
        paint.color = outlineColor ?: defaultOutlineColor
        paint.style = Paint.Style.STROKE
        canvas.drawPoint(startX, startY, paint)
    }
}

```

## LineShape.kt

```

package com.oop.lab2.shape

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint

class LineShape: Shape() {
    override val defaultOutlineColor = Color.argb(255, 0, 0, 0)
    override val defaultFillingColor = Color.TRANSPARENT

    override fun show(canvas: Canvas, paint: Paint) {
        paint.color = outlineColor ?: defaultOutlineColor
        paint.style = Paint.Style.STROKE
        canvas.drawLine(startX, startY, endX, endY, paint)
    }
}

```

## RectShape.kt



```

package com.oop.lab2.shape

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.graphics.Rect

class RectShape: Shape() {
    override val defaultOutlineColor = Color.argb(255, 0, 0, 0)
    override val defaultFillingColor = Color.TRANSPARENT

    override fun show(canvas: Canvas, paint: Paint) {
        val rect = Rect(startX.toInt(), startY.toInt(), endX.toInt(),
endY.toInt())
        paint.color = outlineColor ?: defaultOutlineColor
        paint.style = Paint.Style.STROKE
        canvas.drawRect(rect, paint)

        paint.color = fillingColor ?: defaultFillingColor
        paint.style = Paint.Style.FILL
        canvas.drawRect(rect, paint)
    }
}

```

## EllipseShape.kt

```

package com.oop.lab2.shape

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.graphics.RectF

class EllipseShape: Shape() {
    override val defaultOutlineColor = Color.argb(255, 0, 0, 0)
    override val defaultFillingColor = Color.argb(255, 255, 255, 0)

    override fun show(canvas: Canvas, paint: Paint) {
        val rect = RectF(startX, startY, endX, endY)
        paint.color = outlineColor ?: defaultOutlineColor
        paint.style = Paint.Style.STROKE
        canvas.drawOval(rect, paint)

        paint.color = fillingColor ?: defaultFillingColor
        paint.style = Paint.Style.FILL
        canvas.drawOval(rect, paint)
    }
}

```

## Editor.kt

```

package com.oop.lab2.editor

import android.graphics.Canvas
import android.graphics.Paint
import com.oop.lab2.shape.Shape

abstract class Editor {

```

```

        abstract fun onFingerTouch(x: Float, y: Float)
        abstract fun onFingerMove(canvas: Canvas, paint: Paint, x: Float, y:
Float)
        abstract fun onFingerRelease(drawnShapes: MutableList<Shape>)
    }

```

## ShapeEditor.kt

```

package com.oop.lab2.editor

import android.graphics.Color
import com.oop.lab2.shape.Shape

abstract class ShapeEditor: Editor() {
    lateinit var shape: Shape

    override fun onFingerRelease(drawnShapes: MutableList<Shape>) {
        drawnShapes.add(shape)
    }

    protected fun setRubberTraceMode() {
        shape.setColors(Color.argb(255, 255, 0, 0), Color.TRANSPARENT)
    }
}

```

## PointShapeEditor.kt

```

package com.oop.lab2.editor

import android.graphics.Canvas
import android.graphics.Paint

class PointShapeEditor: ShapeEditor() {
    override fun onFingerTouch(x: Float, y: Float) {
        shape.setStart(x, y)
    }

    override fun onFingerMove(canvas: Canvas, paint: Paint, x: Float, y:
Float) {
        setRubberTraceMode()
        shape.show(canvas, paint)
    }
}

```

## LineShapeEditor.kt

```

package com.oop.lab2.editor

import android.graphics.Canvas
import android.graphics.Paint

class LineShapeEditor: ShapeEditor() {
    override fun onFingerTouch(x: Float, y: Float) {
        shape.setStart(x, y)
    }
}

```

```

        override fun onFingerMove(canvas: Canvas, paint: Paint, x: Float, y:
Float) {
            shape.setEnd(x, y)
            setRubberTraceMode()
            shape.show(canvas, paint)
        }
    }
}

```

## RectShapeEditor.kt

```

package com.oop.lab2.editor

import android.graphics.Canvas
import android.graphics.Paint
import android.graphics.PointF

class RectShapeEditor: ShapeEditor() {
    private val shapeCenterPoint = PointF()

    override fun onFingerTouch(x: Float, y: Float) {
        shapeCenterPoint.set(x, y)
    }

    override fun onFingerMove(canvas: Canvas, paint: Paint, x: Float, y:
Float) {
        val dx = x - shapeCenterPoint.x
        val oppositeX = shapeCenterPoint.x - dx
        val shapeStartX = if (x < shapeCenterPoint.x) x else oppositeX
        val shapeEndX = if (x < shapeCenterPoint.x) oppositeX else x

        val dy = y - shapeCenterPoint.y
        val oppositeY = shapeCenterPoint.y - dy
        val shapeStartY = if (y < shapeCenterPoint.y) y else oppositeY
        val shapeEndY = if (y < shapeCenterPoint.y) oppositeY else y

        shape.setStart(shapeStartX, shapeStartY)
        shape.setEnd(shapeEndX, shapeEndY)
        setRubberTraceMode()
        shape.show(canvas, paint)
    }
}

```

## EllipseShapeEditor.kt

```

package com.oop.lab2.editor

import android.graphics.Canvas
import android.graphics.Paint

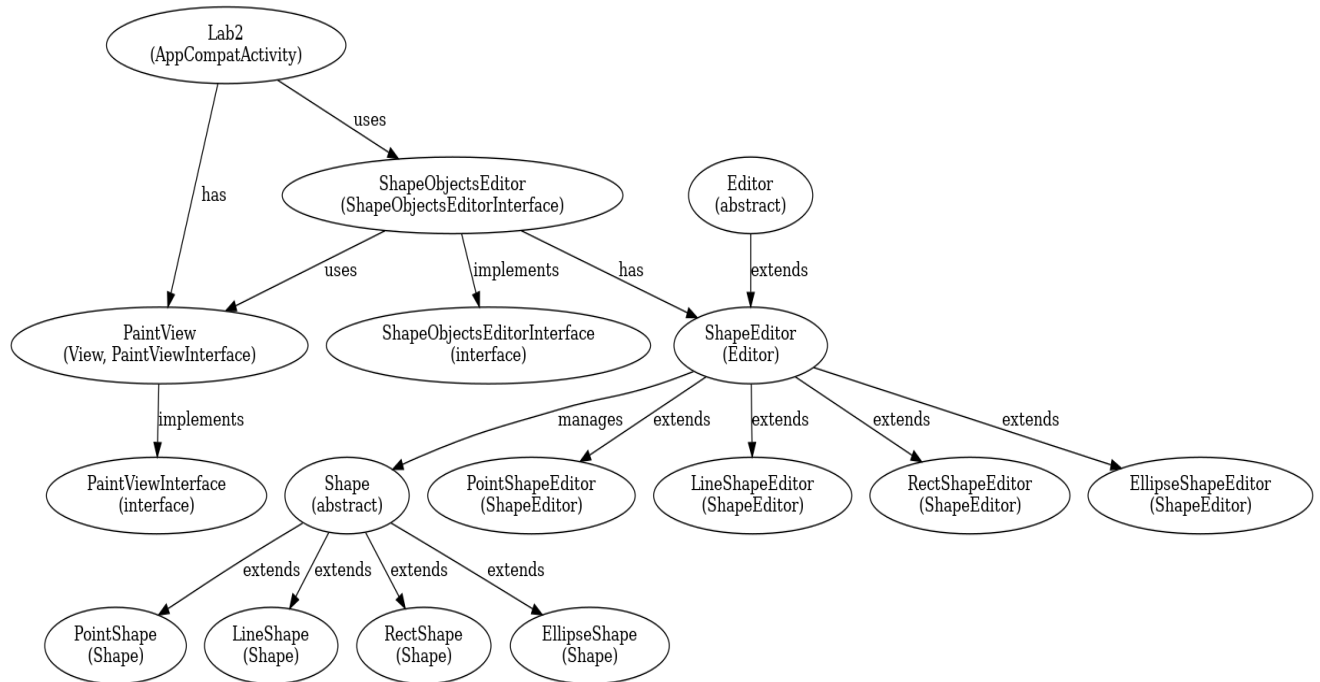
class EllipseShapeEditor : ShapeEditor() {
    override fun onFingerTouch(x: Float, y: Float) {
        shape.setStart(x, y)
    }

    override fun onFingerMove(canvas: Canvas, paint: Paint, x: Float, y:
Float) {
        shape.setEnd(x, y)
        setRubberTraceMode()
    }
}

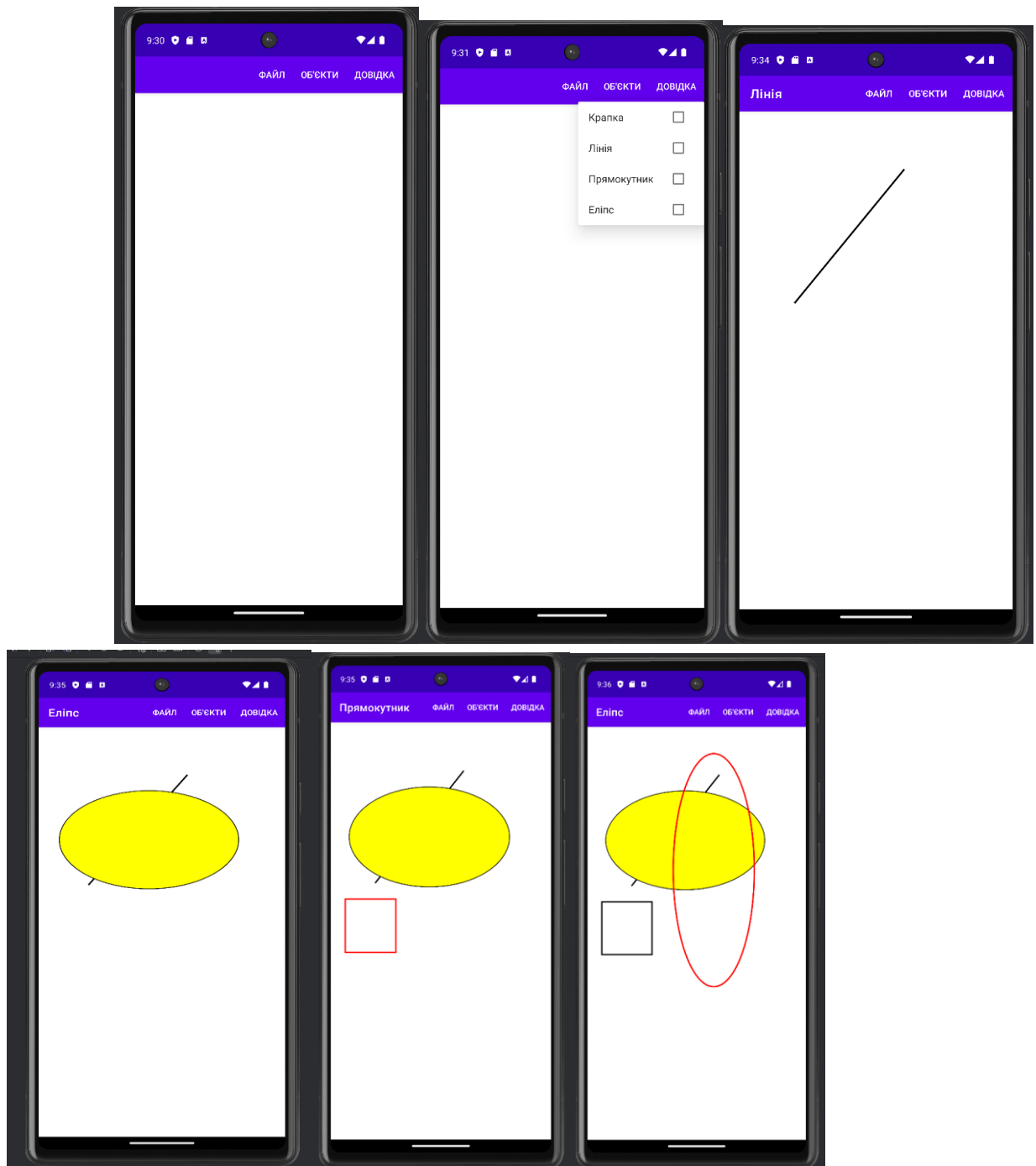
```

```
    shape.show(canvas, paint)
}
}
```

## Діаграма класів програми



## Ілюстрації виконання програми



## Висновки

Під час цієї лабораторної роботи я опанував застосування інкапсуляції, абстракції типів, успадкування та поліморфізму в програмуванні на мові Kotlin. Як результат, мені вдалося створити простий графічний редактор для платформи Android, використовуючи об'єктно-орієнтований підхід.