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-22
Тимофеєв Даниіл Костянтинович
номер у списку групи: 23

Перевірив:

Порєв В.М

Київ 2023

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. Налагодити програму.

4. Проаналізувати та прокоментувати результати та вихідний текст програми.

5. Оформити звіт.

Варіанти :

- Статичний масив Shape *pcshape[N]; причому, кількість елементів масиву вказівників як для статичного, так і динамічного має бути N = 23+100 = 123.

- Гумовий" слід при вводі об’єкті - пунктирна лінія чорного кольору для варіантів (Ж mod 4 = 3)

- **Прямокутник**

- Увід прямокутника - від центру до одного з кутів для варіантів (Ж mod 2 = 1)
- Відображення прямокутника - чорний контур прямокутника без заповнення для (Ж mod 5 = 3 або 4)

- **Еліпс**

- Увід еліпсу - по двом протилежним кутам охоплюючого прямокутника для варіантів (Ж mod 2 = 1)
- Відображення еліпсу - чорний контур з кольоровим заповненням для (Ж mod 5 = 3 або 4)
- Кольори заповнення еліпсу - померанчевий для (Ж mod 6 = 5)

- Позначка поточного типу об’єкту, що вводиться - в заголовку вікна для (Ж mod 2 = 1)

Вихідний текст програм :

MainActivity.kt

```
package com.example.lab2

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import androidx.core.view.WindowCompat
import androidx.core.view.WindowInsetsCompat
import androidx.core.view.WindowInsetsControllerCompat
class MainActivity : AppCompatActivity() {

  private lateinit var drawingView: CustomDrawingView
  private var currentPrimitive: CustomDrawingView.PrimitivesSelection =
CustomDrawingView.PrimitivesSelection.DOT
  private lateinit var mainMenu: Menu
  private var menuItemMap: MutableMap<Int,
CustomDrawingView.PrimitivesSelection> = mutableMapOf()

  override fun onCreate (savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    drawingView = CustomDrawingView(this)
    drawingView.setShapePrimitiveEditor(currentPrimitive)
    setContentView(drawingView)
    showSystemBars()
  }

  private fun setCurrentPrimitive (primitive:
CustomDrawingView.PrimitivesSelection) {
```

```
    currentPrimitive = primitive
    drawingView.setShapePrimitiveEditor(currentPrimitive)
    updateMenuCheckState(currentPrimitive)
    supportActionBar?.title = getActionBarTitle(currentPrimitive)
  }

  override fun onCreateOptionsMenu (menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.main_menu, menu)
    mainMenu = menu!!

    menuItemMap[R.id.ellipseSelect] =
CustomDrawingView.PrimitivesSelection.ELLIPSE
    menuItemMap[R.id.lineSelect] =
CustomDrawingView.PrimitivesSelection.LINE
    menuItemMap[R.id.dotSelect] =
CustomDrawingView.PrimitivesSelection.DOT
    menuItemMap[R.id.rectSelect] =
CustomDrawingView.PrimitivesSelection.RECTANGLE
    updateMenuCheckState(currentPrimitive)
    return true
  }

  private fun getActionBarTitle (primitive:
CustomDrawingView.PrimitivesSelection): String {
    return when (primitive) {
      CustomDrawingView.PrimitivesSelection.ELLIPSE -> "Еліпс"
      CustomDrawingView.PrimitivesSelection.LINE -> "Лінія"
      CustomDrawingView.PrimitivesSelection.DOT -> "Крапка"
      CustomDrawingView.PrimitivesSelection.RECTANGLE -> "Прямокутник"
    }
  }
```

```
  override fun onOptionsItemSelected (item: MenuItem): Boolean {
    val selectedPrimitive = menuItemMap[item.itemId]
    if (selectedPrimitive != null) setCurrentPrimitive(selectedPrimitive)

    return super.onOptionsItemSelected(item)
  }

  private fun showSystemBars () {
    WindowCompat.setDecorFitsSystemWindows(window, true)
    WindowInsetsControllerCompat(window,
drawingView).show(WindowInsetsCompat.Type.systemBars())
  }

  private fun updateMenuCheckState (selectedOption:
CustomDrawingView.PrimitivesSelection) {
    menuItemMap.values.forEach
{ mainMenu.findItem(getMenuItemId(it))?.isChecked = false }
    mainMenu.findItem(getMenuItemId(selectedOption))?.isChecked = true
  }

  private fun getMenuItemId (option: CustomDrawingView.PrimitivesSelection):
Int {
    return when (option) {
      CustomDrawingView.PrimitivesSelection.ELLIPSE -> R.id.ellipseSelect
      CustomDrawingView.PrimitivesSelection.LINE -> R.id.lineSelect
      CustomDrawingView.PrimitivesSelection.DOT -> R.id.dotSelect
      CustomDrawingView.PrimitivesSelection.RECTANGLE -> R.id.rectSelect
    }
  }
```

}

CustomDrawingView.kt

```
package com.example.lab2

import android.content.Context
import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import android.view.MotionEvent
import android.view.View
import com.example.lab2.editor_primitives.DotEditor
import com.example.lab2.editor_primitives.EllipseEditor
import com.example.lab2.editor_primitives.LineEditor
import com.example.lab2.editor_primitives.RectangleEditor

class CustomDrawingView (context: Context) : View(context) {
  companion object {
    private const val BACKGROUND_COLOUR = Color.WHITE
    private var drawingCanvas = Canvas()
    private const val DRAWING_COLOR = Color.BLACK
    private const val STROKE_WIDTH = 20f
    private val shapeList = mutableListOf<Shape>()

    private var currentX = 0f
    private var currentY = 0f

    private val drawingSetting = Paint().apply {
      color = DRAWING_COLOR
      strokeWidth = STROKE_WIDTH
```

```
    style = Paint.Style.STROKE
    strokeCap = Paint.Cap.ROUND
    strokeJoin = Paint.Join.ROUND
    isAntiAlias = true
  }

  private var actualShape: ShapeEditor = DotEditor(drawingSetting, shapeList)
}

enum class PrimitivesSelection (val primitiveName: String) {
  DOT("Dot"),
  LINE("Line"),
  RECTANGLE("Rectangle"),
  ELLIPSE("Ellipse")
}

override fun onDraw (canvas: Canvas?) {
  super.onDraw(canvas)
  shapeList.forEach { it.draw(canvas!!) }
}

fun setShapePrimitiveEditor (primitiveName: PrimitivesSelection) {
  actualShape = when (primitiveName) {
    PrimitivesSelection.DOT -> DotEditor(drawingSetting, shapeList)
    PrimitivesSelection.LINE -> LineEditor(drawingSetting, shapeList)
    PrimitivesSelection.RECTANGLE -> RectangleEditor(drawingSetting,
shapeList)
    PrimitivesSelection.ELLIPSE -> EllipseEditor(drawingSetting, shapeList)
  }
```

```
}

private fun handleTouchUp () {
  invalidate()
  actualShape.onTouchUp()
}

private fun handleTouchMove () {
  invalidate()
  actualShape.handleMouseMovement(currentX, currentY)
}

private fun handleTouchStart () {
  invalidate()
  actualShape.onTouchDown(currentX, currentY)
}

override fun onTouchEvent (event: MotionEvent?): Boolean {
  currentX = event!!.x
  currentY = event.y

  when (event.action) {
    MotionEvent.ACTION_MOVE -> handleTouchMove()
    MotionEvent.ACTION_UP -> handleTouchUp()
    MotionEvent.ACTION_DOWN -> handleTouchStart()
  }
  return true
}
```

```
  override fun onSizeChanged (newWidth: Int, newHeight: Int, oldWidth: Int,
oldHeight: Int) {
    super.onSizeChanged(newWidth, newHeight, oldWidth, oldHeight)
    drawingCanvas = Canvas()
    drawingCanvas!!.drawColor(BACKGROUND_COLOUR)
  }
}
```

Editor.kt

```
package com.example.lab2

abstract class Editor {
  abstract fun handleMouseMovement (x: Float, y: Float)
  abstract fun onTouchUp ()
  abstract fun onTouchDown (x: Float, y: Float)

}
```

Shape.kt

```
package com.example.lab2

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

abstract class Shape (paintSettings : Paint) {
  private var isEraserMode: Boolean = true

  protected var startXCoordinate: Float = 0f
  protected var startYCoordinate: Float = 0f

  protected var endXCoordinate: Float = 0f
```

```
  protected var endYCoordinate: Float = 0f

  fun defineEraserMode (eraserMode: Boolean) {
    isEraserMode = eraserMode
  }

  fun defineStartCoordinates (x: Float, y: Float) {
    startXCoordinate = x
    startYCoordinate = y
  }

  fun defineEndCoordinates (x: Float, y: Float) {
    endXCoordinate  = x
    endYCoordinate  = y
  }

  abstract fun draw (canvas: Canvas)

  abstract fun configureDrawing ()
}
```

ShapeEditor.kt

```
package com.example.lab2

import android.graphics.Paint

abstract class ShapeEditor (paintSettings: Paint, shapesList:
MutableList<Shape>) : Editor() {
  val paint: Paint = paintSettings
  val shapes: MutableList<Shape> = shapesList
```

```
  private val shapesLimit: Int = 123
  protected fun addShapeToEditor (shape: Shape, shapes: MutableList<Shape>) {
    if (shapes.lastIndex == shapesLimit - 1) {
      shapes.removeAt(shapes.lastIndex)
    }
    shapes.add(shape)
  }
  override fun onTouchUp () {

  }

  override fun onTouchDown (x: Float, y: Float) {

  }

  override fun handleMouseMovement (x: Float, y: Float) {

  }
}
```

DotShape.kt

```
package com.example.lab2.shape_primitives

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.Paint
import com.example.lab2.Shape

class DotShape (paintSettings: Paint) : Shape(paintSettings) {
  private val paint: Paint = paintSettings
```

```
  override fun draw (canvas: Canvas) {
    configureDrawing()
    paint.strokeWidth = 20f
    canvas.drawPoint(startXCoordinate, startYCoordinate, paint)
  }

  override fun configureDrawing () {
    paint.apply { color = Color.BLACK }
  }

}
```

EllipseShape.kt

```
package com.example.lab2.shape_primitives

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.DashPathEffect
import android.graphics.Paint
import com.example.lab2.Shape

class EllipseShape (private val paintSettings: Paint) : Shape(paintSettings) {
  private val paint = Paint()

  override fun draw (canvas: Canvas) {
    if (!isEraserMode) {
      configureFillStyle()
      canvas.drawOval(
        2*startXCoordinate - endXCoordinate,
```

```
      2*startYCoordinate - endYCoordinate,
      endXCoordinate,
      endYCoordinate,
      paint,
    )
    resetDashedOutline()
  } else {
    configureDashedOutline()
  }

  configureDrawing()
  canvas.drawOval(
    2*startXCoordinate - endXCoordinate,
    2*startYCoordinate - endYCoordinate,
    endXCoordinate,
    endYCoordinate,
    paint,
  )
}

override fun configureDrawing () {
  paint.apply {
    this.color = Color.BLACK
    this.style = Paint.Style.STROKE
    this.strokeWidth = 20f
  }
}

private fun configureFillStyle () {
```

```
    applyDrawingStyle(Color.rgb(255, 165, 0), Paint.Style.FILL)
  }

  private fun applyDrawingStyle (color: Int, style: Paint.Style) {
    paint.apply {
      this.color = color
      this.style = style
    }
  }

  private fun configureDashedOutline() {
    val dashLength = 10f
    val dashGap = 10f
    val dashEffect = DashPathEffect(floatArrayOf(dashLength, dashGap), 0f)
    paint.pathEffect = dashEffect
  }

  private fun resetDashedOutline() {
    paint.pathEffect = null
  }
}
```

LineShape.kt

```
package com.example.lab2.shape_primitives

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.DashPathEffect
import android.graphics.Paint
import com.example.lab2.Shape
```

```
class LineShape(paintSettings: Paint) : Shape(paintSettings) {
  private var isDashed = true
  private val paint = Paint()

  fun toggleDashed () {
    isDashed = !isDashed
  }

  override fun draw (canvas: Canvas) {
    configureDrawing()
    if (isDashed) {
      paint.pathEffect = DashPathEffect(floatArrayOf(10f, 10f), 0f)
    } else {
      paint.pathEffect = null
    }
    canvas.drawLine(startXCoordinate, startYCoordinate, endXCoordinate,
endYCoordinate, paint)
  }

  override fun configureDrawing () {
    paint.apply {
      color = Color.BLACK
      style = Paint.Style.FILL_AND_STROKE
      strokeWidth = 20f
    }
  }
}
```

RectangleShape.kt

```
package com.example.lab2.shape_primitives

import android.graphics.Canvas
import android.graphics.Color
import android.graphics.DashPathEffect
import android.graphics.Paint
import android.graphics.RectF
import com.example.lab2.Shape


class RectangleShape (initialPaintSettings: Paint) : Shape(initialPaintSettings) {
  private var isDashed = true
  private val paint = Paint()

  fun toggleDashed () {
    isDashed = !isDashed
  }

  override fun draw (canvas: Canvas) {
    configureDrawing()
    val rect = RectF(startXCoordinate, startYCoordinate, endXCoordinate,
endYCoordinate)

    if (isDashed) {
      paint.pathEffect = DashPathEffect(floatArrayOf(10f, 10f), 0f)
    } else {
      paint.pathEffect = null
    }
```

```
    canvas.drawRect(rect, paint)
  }

  override fun configureDrawing () {
    paint.apply {
      color = Color.BLACK
      style = Paint.Style.STROKE
      strokeWidth = 20f
    }
  }
}
```

DotEditor.kt

```
package com.example.lab2.editor_primitives

import android.graphics.Paint
import com.example.lab2.Shape
import com.example.lab2.ShapeEditor
import com.example.lab2.shape_primitives.DotShape

class DotEditor (paintSettings: Paint, shapesList: MutableList<Shape>) :
ShapeEditor(paintSettings, shapesList) {
  private var currentDotShape: DotShape? = null

  override fun onTouchDown(x: Float, y: Float) {
    currentDotShape = DotShape(paint)
    currentDotShape?.defineStartCoordinates(x, y)
  }

  override fun onTouchUp() {
```

```
    currentDotShape?.let {
      addShapeToEditor(it, shapes)
      currentDotShape = null
    }
  }
}
```

EllipseEditor.kt

```
package com.example.lab2.editor_primitives

import android.graphics.Paint
import com.example.lab2.Shape
import com.example.lab2.ShapeEditor
import com.example.lab2.shape_primitives.EllipseShape

class EllipseEditor (private val paintSettings: Paint, private val shapesList:
MutableList<Shape>) : ShapeEditor(paintSettings, shapesList) {
  private var ellipseShape: EllipseShape? = null

  override fun onTouchDown (x: Float, y: Float) {
    ellipseShape = EllipseShape(paintSettings)
    ellipseShape?.defineStartCoordinates(x, y)
  }

  override fun onTouchUp () {
    ellipseShape?.let {
      if (shapesList.contains(it)) shapesList.remove(it)
      it.defineEraserMode(false)
      addShapeToEditor(it, shapesList)
    }
```

```
    ellipseShape = null
  }

  override fun handleMouseMovement (x: Float, y: Float) {
    ellipseShape?.let {
      if (shapesList.contains(it)) shapesList.remove(it)
      it.defineEndCoordinates(x, y)
      addShapeToEditor(it, shapesList)
    }
  }
}
```

LineEditor.kt

```
package com.example.lab2.editor_primitives

import android.graphics.Paint
import com.example.lab2.Shape
import com.example.lab2.ShapeEditor
import com.example.lab2.shape_primitives.LineShape

class LineEditor (private val paintSettings: Paint, private val shapesList:
MutableList<Shape>) : ShapeEditor(paintSettings, shapesList) {
  private var currentLine: LineShape? = null

  override fun onTouchDown(x: Float, y: Float) {
    currentLine = LineShape(paintSettings)
    currentLine?.defineStartCoordinates(x, y)
  }

  override fun onTouchUp() {
```

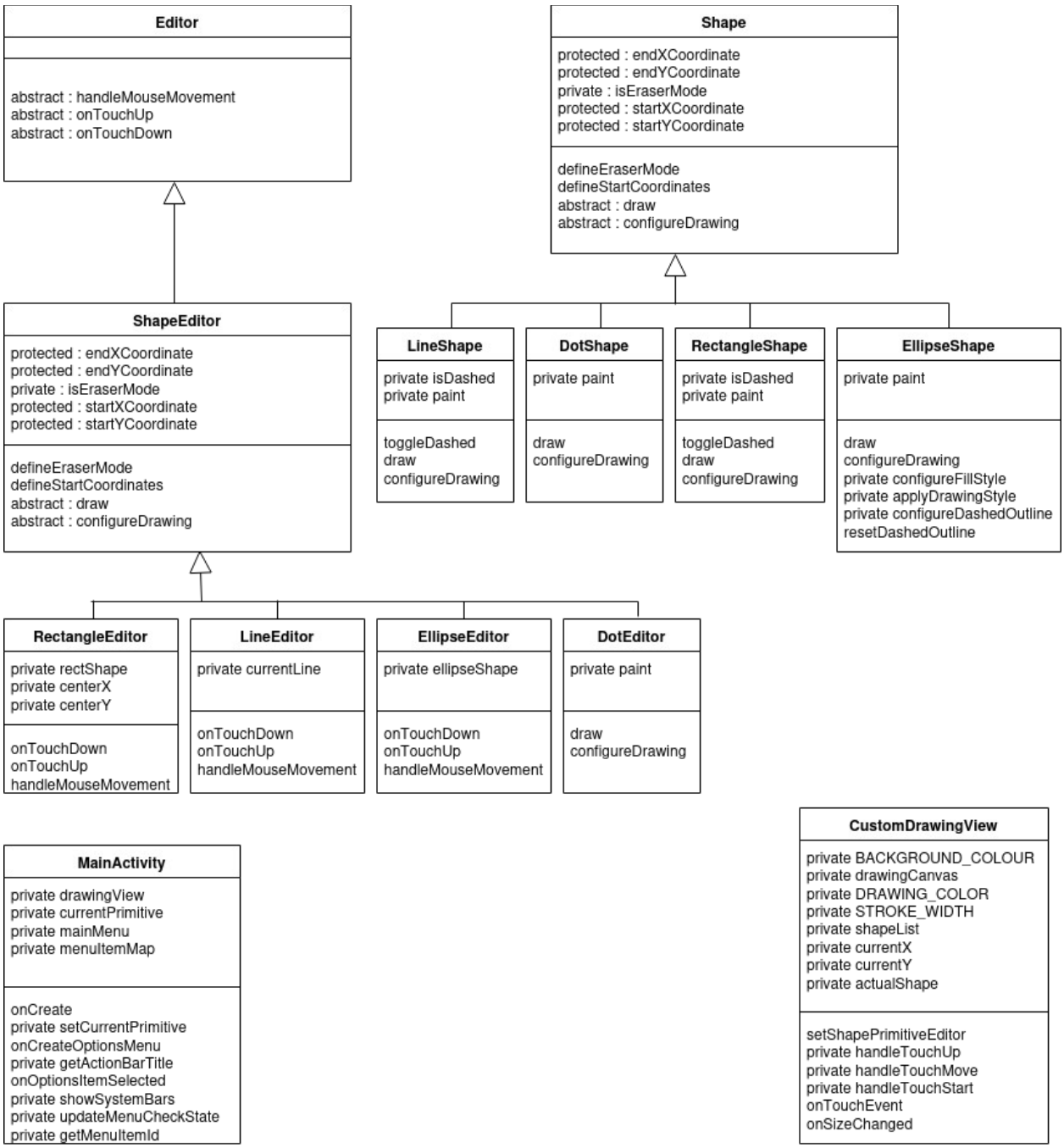
```
    currentLine?.let {
      it.toggleDashed()
      addShapeToEditor(it, shapesList)
    }
    currentLine = null
  }

  override fun handleMouseMovement(x: Float, y: Float) {
    currentLine?.let { lineShape ->
      shapesList.remove(lineShape)
      lineShape.defineEndCoordinates(x, y)
      addShapeToEditor(lineShape, shapesList)
    }
  }
}
```

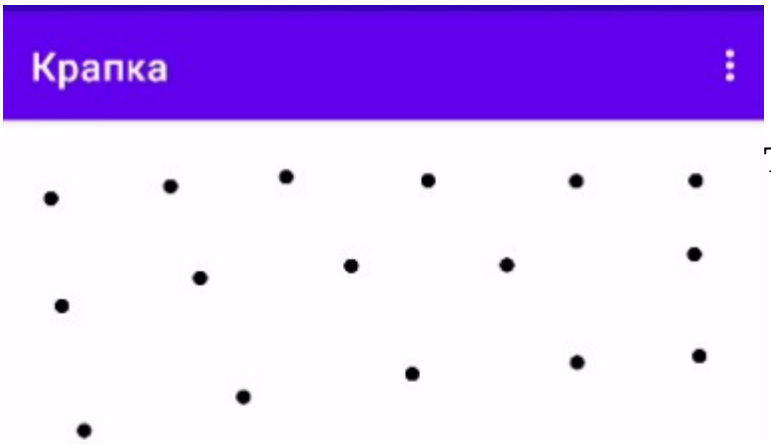
RectangleEditor.kt

```
package com.example.lab2.editor_primitives

import android.graphics.Paint
import com.example.lab2.Shape
import com.example.lab2.ShapeEditor
import com.example.lab2.shape_primitives.RectangleShape



class RectangleEditor (private val paintSettings: Paint, private val shapesList:
MutableList<Shape>) : ShapeEditor(paintSettings, shapesList) {
  private var rectShape: RectangleShape? = null
  private var centerX: Float = 0f
  private var centerY: Float = 0f
```

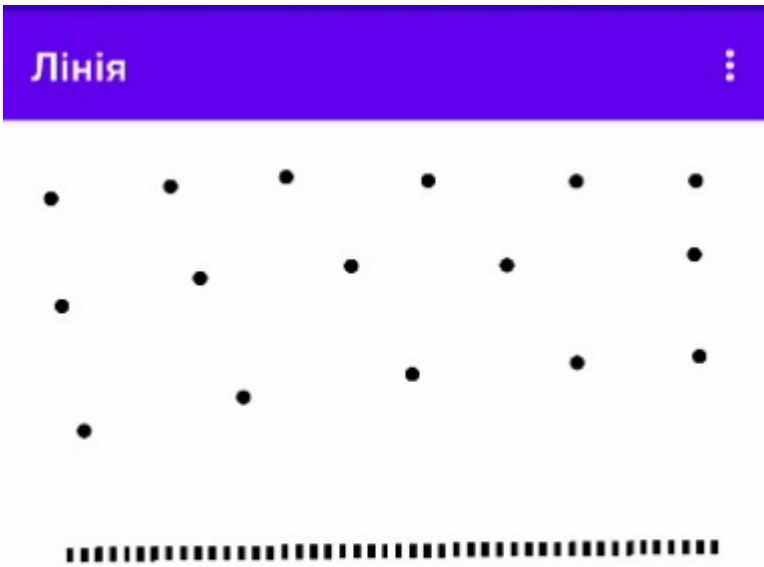
```
  override fun onTouchDown(x: Float, y: Float) {
    centerX = x
    centerY = y
    rectShape = RectangleShape(paintSettings)
  }

  override fun onTouchUp() {
    rectShape?.let {
      addShapeToEditor(it, shapesList)
      it.toggleDashed()
    }
    rectShape = null
  }

  override fun handleMouseMovement(x: Float, y: Float) {
    rectShape?.let {
      shapesList.remove(it)
      val left = centerX - (x - centerX)
      val top = centerY - (y - centerY)
      val right = centerX + (x - centerX)
      val bottom = centerY + (y - centerY)
      it.defineStartCoordinates(left, top)
      it.defineEndCoordinates(right, bottom)
      addShapeToEditor(it, shapesList)
    }
  }
}
```

Діаграма класів :

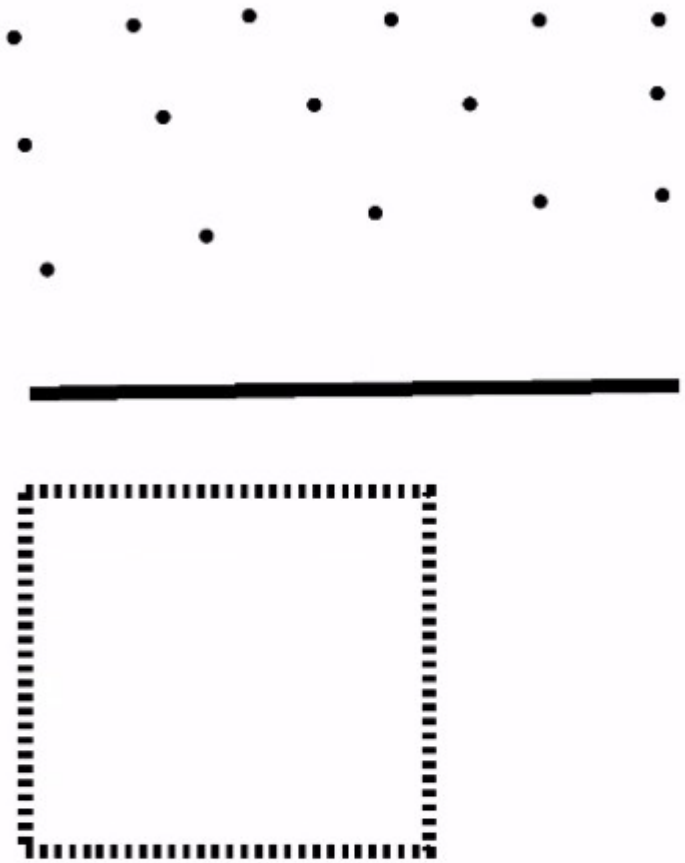




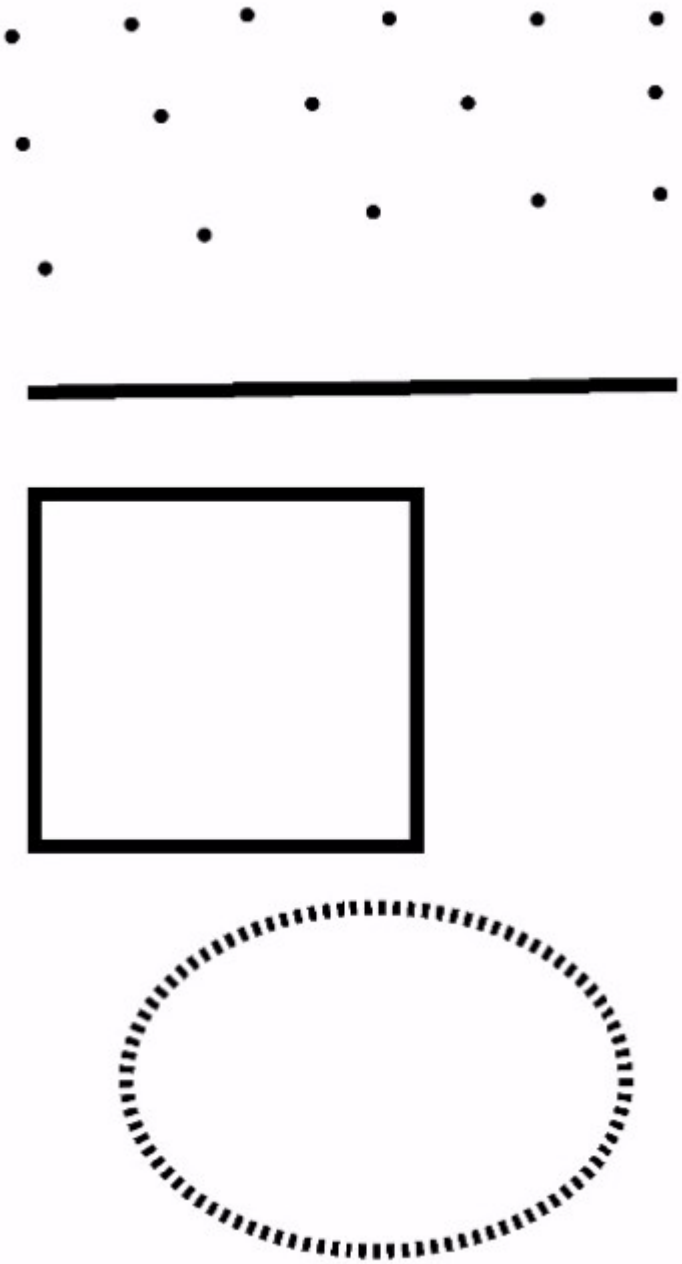
Тестування програми (скріншоти) :

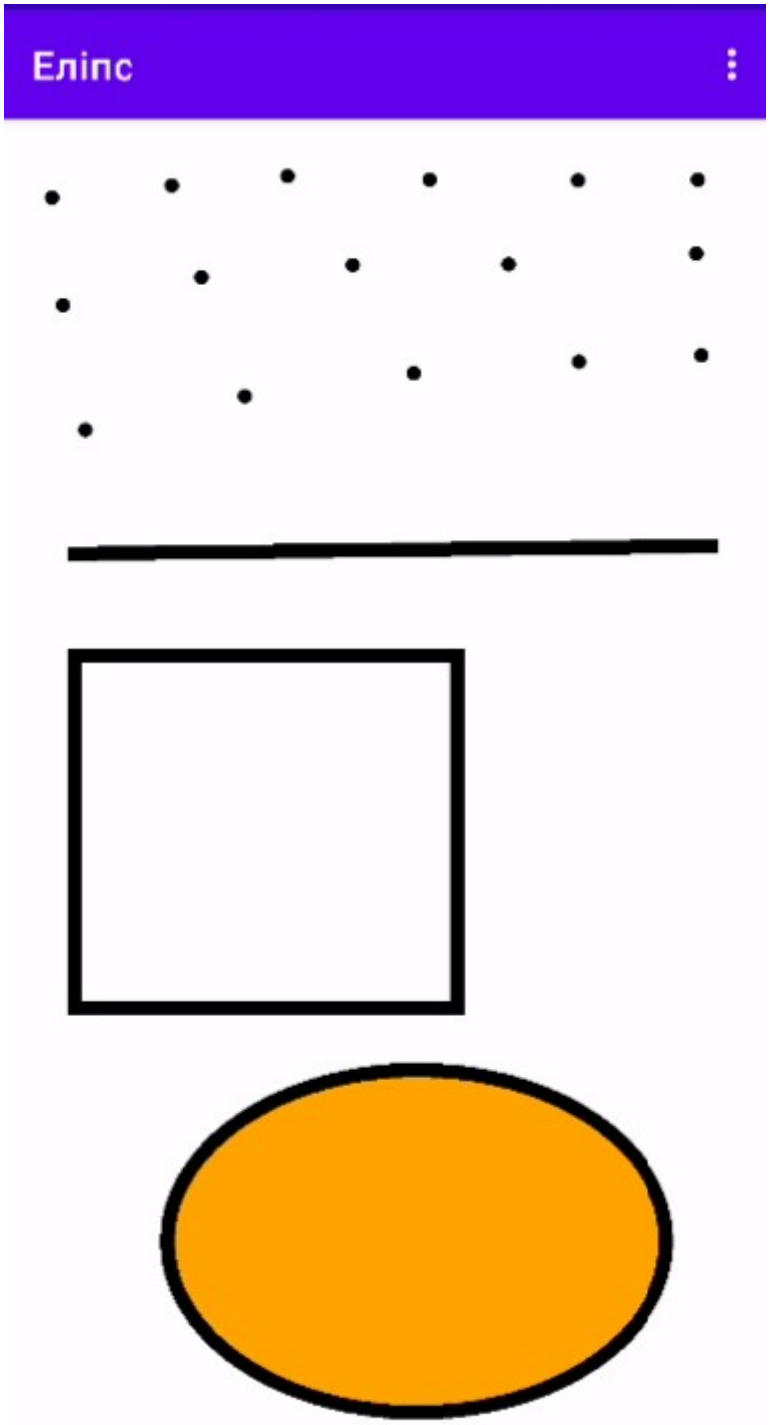

Лінія

Прямокутник



Еліпс





Висновки : У ході виконання лабораторної роботи № 2 я розробляв графічний додаток для малювання різних фігур на екрані. У цій програмі широко використовується поліморфізм, який є однією з ключових концепцій об'єктно-орієнтованого програмування. Поліморфізм дозволяє об'єктам різних класів виконувати однакові дії, а також надає гнучкість і можливість розширення програми. Клас Shape є абстрактним базовим класом для всіх графічних фігур, і він містить абстрактні методи, такі як draw і configureDrawing. Кожен конкретний клас фігури (наприклад, DotShape, LineShape, EllipseShape, RectangleShape) реалізує ці методи у власний спосіб.

При цьому всі ці класи можуть бути використані як Shape, і їхні методи можуть бути викликані поліморфно. Клас ShapeEditor також є абстрактним і містить методи, такі як onTouchDown, onTouchUp і handleMouseMovement, які реалізовані у конкретних редакторах (наприклад, DotEditor, LineEditor, EllipseEditor, RectangleEditor). Поліморфізм дозволяє використовувати об'єкти різних редакторів, які можуть обробляти події та взаємодіяти з фігурами, незалежно від їхнього конкретного типу. Усі ці приклади демонструють використання поліморфізму для забезпечення гнучкості та розширюваності програми, що дозволяє легко додавати нові фігури і редактори без необхідності змінювати вже існуючий код.