

**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-21  
Сірик Максим Олександрович  
номер у списку групи: 22

Перевірив:  
Порєв Віктор Миколайович

Київ 2023

# Зміст

<b>1</b>	<b>Мета:</b>	<b>2</b>
<b>2</b>	<b>Завдання:</b>	<b>2</b>
2.1	Варіант: . . . . .	2
<b>3</b>	<b>Текст програми:</b>	<b>3</b>
3.1	Module: com.github.erotourtes.view . . . . .	3
3.2	Module: com.github.erotourtes.app . . . . .	6
3.3	Module: com.github.erotourtes.drawing . . . . .	7
3.4	Module: com.github.erotourtes.utils . . . . .	7
3.5	Module: com.github.erotourtes.styles . . . . .	10
3.6	Module: com.github.erotourtes.drawing.editor . . . . .	11
3.7	Module: com.github.erotourtes.drawing.shape . . . . .	14
<b>4</b>	<b>Ілюстрації:</b>	<b>16</b>
4.1	Images . . . . .	16
4.2	UML . . . . .	18
<b>5</b>	<b>Висновки:</b>	<b>19</b>

# 1 Мета:

Мета роботи – отримати вміння та навички використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм на основі класів C++, запрограмувавши графічний інтерфейс користувача.

## 2 Завдання:

1. Створити у середовищі MS Visual Studio C++ проект Win32 з ім'ям Lab3.
2. Написати вихідний текст програми згідно варіанту завдання.
3. Скомпілювати вихідний текст і отримати виконуваний файл програми.
4. Перевірити роботу програми. Налаштувати програму.
5. Проаналізувати та прокоментувати результати та вихідний текст програми.
6. Оформити звіт.

### 2.1 Варіант:

Варіанти завдань та основні вимоги

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  $J = J_{\text{лаб2}} + 1$ , де  $J_{\text{лаб2}}$  – номер студента в журналі, який використовувався для попередньої лаб. роботи No2.

$$J = 22 + 1 \quad (1)$$

$$J = 23 \quad (2)$$

5. Масив вказівників для динамічних об'єктів типу Shape

- динамічний масив Shape `**pcshape;`
- статичний масив Shape `*pcshape[N];`

причому кількість елементів масиву вказівників як для статичного, так і динамічного має бути  $N = J + 100$ .  $N = 23 + 100$

Динамічний масив обирають студенти, у яких варіант ( $J \bmod 3 = 0$ ).

Решта студентів роблять статичний масив. Примітка. Позначка `mod` означає залишок від ділення.  $23 \bmod 3 = 2$

6. "Гумовий" слід при вводі об'єктів

- суцільна лінія чорного кольору для варіантів ( $J \bmod 4 = 0$ )
- суцільна лінія червоного кольору для ( $J \bmod 4 = 1$ )
- суцільна лінія синього кольору для ( $J \bmod 4 = 2$ )
- пунктирна лінія чорного кольору для ( $J \bmod 4 = 3$ )  $23 \bmod 4 = 3$

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

#### 7.1. Прямокутник

Увід прямокутника:

- по двом протилежним кутам для варіантів ( $J \bmod 2 = 0$ )
- від центру до одного з кутів для ( $J \bmod 2 = 1$ )  $23 \bmod 2 = 1$

Відображення прямокутника:

- чорний контур з білим заповненням для ( $\mathcal{K} \bmod 5 = 0$ )
- чорний контур з кольоровим заповненням для ( $\mathcal{K} \bmod 5 = 1$  або  $2$ )
- чорний контур прямокутника без заповнення для ( $\mathcal{K} \bmod 5 = 3$  або  $4$ )  $23 \bmod 5 = 3$

Кольори заповнення прямокутника:

- жовтий для ( $\mathcal{K} \bmod 6 = 0$ )
- світло-зелений для ( $\mathcal{K} \bmod 6 = 1$ )
- блакитний для ( $\mathcal{K} \bmod 6 = 2$ )
- рожевий для ( $\mathcal{K} \bmod 6 = 3$ )
- померанчевий для ( $\mathcal{K} \bmod 6 = 4$ )
- сірий для ( $\mathcal{K} \bmod 6 = 5$ )  $23 \bmod 6 = 5$

## 7.2. Еліпс

Ввід еліпсу:

- по двом протилежним кутам охоплюючого прямокутника для варіантів ( $\mathcal{K} \bmod 2 = 1$ )  $23 \bmod 2 = 1$
- від центру до одного з кутів охоплюючого прямокутника для варіантів ( $\mathcal{K} \bmod 2 = 0$ )

Відображення еліпсу:

- чорний контур з білим заповненням для ( $\mathcal{K} \bmod 5 = 1$ )
- чорний контур з кольоровим заповненням для ( $\mathcal{K} \bmod 5 = 3$  або  $4$ )  $23 \bmod 5 = 3$
- чорний контур еліпсу без заповнення для ( $\mathcal{K} \bmod 5 = 0$  або  $2$ )

Кольори заповнення еліпсу:

- жовтий для ( $\mathcal{K} \bmod 6 = 1$ )
- світло-зелений для ( $\mathcal{K} \bmod 6 = 2$ )
- блакитний для ( $\mathcal{K} \bmod 6 = 3$ )
- рожевий для ( $\mathcal{K} \bmod 6 = 4$ )
- померанчевий для ( $\mathcal{K} \bmod 6 = 5$ )  $23 \bmod 6 = 5$
- сірий для ( $\mathcal{K} \bmod 6 = 0$ )

## 8. Позначка поточного типу об'єкту, що вводиться

- в меню (метод `OnInitMenuPopup`) для варіантів ( $\mathcal{K} \bmod 2 = 0$ )
- в заголовку вікна для ( $\mathcal{K} \bmod 2 = 1$ )  $23 \bmod 2 = 1$

Примітка. Визначення кольорів та інші параметри варіантів можуть бути змінені викладачем шляхом оголошення студентам відповідного повідомлення завчасно перед постановкою завдань.

## 3 Текст програми:

### 3.1 Module: `com.github.erotourtes.view`

Лістинг 1: `MainView.kt`

```
package com.github.erotourtes.view

import com.github.erotourtes.drawing.EditorHandler
import com.github.erotourtes.drawing.CanvasPane
import com.github.erotourtes.drawing.editor.*
import com.github.erotourtes.utils.MenuItemInfo
import com.github.erotourtes.utils.ToolbarItemInfo
import de.jensd.fx.glyphs.fontawesome.FontAwesomeIcon
import javafx.scene.canvas.Canvas
import javafx.scene.control.ToggleGroup
import tornadofx.*

class MainView : View("Lab3") {
    private val canvas = Canvas()
    private val editorHandler = EditorHandler(canvas)
```

```

override val root = borderpane {
    top = MenuBar(createMenu())
    center = borderpane {
        top = ToolBar(createToolBar(), editorHandler)
        center = CanvasPane(canvas)
    }
}

// [icons](https://fontawesome.com/v4/icons/)
private fun createToolBar(): List<ToolBarItemInfo> = listOf(
    ToolBarItemInfo(
        tooltip = "Dot", icon = FontAwesomeIcon.DOT_CIRCLE_ALT, kotlinClass = PointEditor::class.java
    ),
    ToolBarItemInfo(
        tooltip = "Line", icon = FontAwesomeIcon.MINUS, kotlinClass = LineEditor::class.java
    ),
    ToolBarItemInfo(
        tooltip = "Rectangle", icon = FontAwesomeIcon.SQUARE, kotlinClass = RectEditor::class.java
    ),
    ToolBarItemInfo(
        tooltip = "Ellipse", icon = FontAwesomeIcon.CIRCLE_ALT, kotlinClass = EllipseEditor::class.java
    )
)

private fun createMenu(): List<MenuItemInfo> {
    val group = ToggleGroup()
    val menuList = listOf(
        MenuItemInfo("Dot", PointEditor::class.java),
        MenuItemInfo("Line", LineEditor::class.java),
        MenuItemInfo("Rectangle", RectEditor::class.java),
        MenuItemInfo("Ellipse", EllipseEditor::class.java)
    )

    menuList.forEach { it.group = group; it.action = { editorHandler.useEditor(it.kotlinClass) } }

    editorHandler.listenToChanges { _, _, newValue ->
        group.toggles.forEach {
            val userData = it.userData as MenuItemInfo
            it.isSelected = userData.kotlinClass == newValue.javaClass
        }
    }

    return menuList
}
}

```

Лістинг 2: MenuBar.kt

```

package com.github.erotourtes.view

import com.github.erotourtes.utils.MenuItemInfo
import com.github.erotourtes.utils.PopupView
import com.github.erotourtes.utils.Gui
import com.github.erotourtes.utils.Nb
import javafx.scene.control.MenuBar

```

```

import javafx.scene.control.MenuItem
import javafx.stage.StageStyle
import tornadofx.*

class MenuBar(shapes: List<MenuItemInfo>) : MenuBar() {
    init {
        menu("File") {
            val invoke: MenuItem.() -> Unit = {
                action { find<PopupView>(PopupView.ScopeInfo(text)).openModal(StageStyle.U
            }

            item("New...") { invoke() }
            item("Open...") { invoke() }
            item("Save_as...") { invoke() }
            separator()
            item("Print") { invoke() }
            separator()
            item("Exit") { invoke() }
        }
        menu("Objects") {
            shapes.forEach {
                radiomenuitem(it.name, it.group) {
                    action(it.action)
                    isSelected = false
                    userData = it
                }
            }
        }
        menu("Help") {
            item(
                """
                0) Ж_=$g
                1) Статичний масив (Ж_mod_3!=0) обсягом $g+100=$n.
                2) "Гумовий" слід при вводит'єктів - пунктирна лінія чорного кольору
                3) Прямокутник:
                Увід_прямокутника:
                - від центру до одного з кутів для (Ж_mod_2=1) g_%2= ${g_%2}
                Відображення_прямокутника:
                - чорний контур_прямокутника без заповнення для (Ж_mod_5=3 або
                Кольори_заповнення_прямокутника:
                - сірий для (Ж_mod_6=5) g_%6= ${g_%6}
                4) Еліпс:
                Ввід_еліпсу:
                - по двом протилежним кутам охоплюючого_прямокутника для варіан
                Відображення_еліпсу:
                - чорний контур з кольоровим заповненням для (Ж_mod_5=3 або 4
                Кольори_заповнення_еліпсу:
                - померанчевий для (Ж_mod_6=5) g_%6= ${g_%6}
                5) Позначка поточного типу об'єкту, що вводиться
                - в заголовку вікна для (Ж_mod_2=1) g_%2= ${g_%2}
                """).trimIndent()
            )
        }
    }
}

```

```
}
```

Лістинг 3: ToolBar.kt

```
package com.github.erotourtes.view

import com.github.erotourtes.drawing.EditorHandler
import com.github.erotourtes.drawing.editor.Editor
import com.github.erotourtes.drawing.editor.EmptyEditor
import com.github.erotourtes.styles.ToolbarStyles
import com.github.erotourtes.utils.ToolbarItemInfo
import de.jensd.fx.glyphs.fontawesome.FontAwesomeIconView
import javafx.beans.value.ChangeListener
import javafx.scene.control.ToggleGroup
import javafx.scene.layout.HBox
import tornadofx.*

class ToolBar(items: List<ToolbarItemInfo>, editorHandler: EditorHandler) : HBox() {
    private val group = ToggleGroup()
    private val listener: ChangeListener<Editor> = ChangeListener { _, _, newValue -> listener

    init {
        addClass(ToolbarStyles.toolbar)

        items.forEach {
            togglebutton {
                tooltip(it.tooltip)
                addClass(ToolbarStyles.iconButton)
                add(FontAwesomeIconView(it.icon).apply { addClass(ToolbarStyles.icon) })
                toggleGroup = group
                isSelected = false
                userData = it
                action { editorHandler.useEditor(if (isSelected) it.kotlinClass else EmptyEditor) }
            }
        }

        editorHandler.listenToChanges(listener)
    }

    private fun listenToChanges(newValue: Editor) {
        group.toggles.forEach {
            val userData = it.userData as ToolbarItemInfo
            it.isSelected = userData.kotlinClass == newValue.javaClass
        }
    }
}
```

### 3.2 Module: com.github.erotourtes.app

Лістинг 4: MyApp.kt

```
package com.github.erotourtes.app

import com.github.erotourtes.styles.ToolbarStyles
import com.github.erotourtes.view.MainView
import tornadofx.App
```

```
class MyApp: App(MainView:: class, ToolbarStyles:: class)
```

### 3.3 Module: com.github.erotourtes.drawing

Лістинг 5: CanvasPane.kt

```
package com.github.erotourtes.drawing

import javafx.scene.canvas.Canvas
import javafx.scene.layout.Pane

class CanvasPane(canvas: Canvas) : Pane() {
    init {
        children.add(canvas)

        canvas.widthProperty().bind(this.widthProperty())
        canvas.heightProperty().bind(this.heightProperty())
    }
}
```

Лістинг 6: EditorHandler.kt

```
package com.github.erotourtes.drawing

import com.github.erotourtes.utils.n
import com.github.erotourtes.drawing.editor.Editor
import com.github.erotourtes.drawing.editor.ShapesList
import javafx.beans.property.SimpleObjectProperty
import javafx.beans.value.ChangeListener
import javafx.scene.canvas.Canvas
import javafx.scene.canvas.GraphicsContext

class EditorHandler(private val canvas: Canvas) {
    private val shapes = ShapesList(n)
    private var map: MutableMap<Class<out Editor>, Editor> = mutableMapOf()
    private val curEditor = SimpleObjectProperty<Editor>()

    fun useEditor(editorClass: Class<out Editor>) {
        val editor = getOrCreateEditor(editorClass)
        editor.listenToEvents()
        curEditor.set(editor)
    }

    fun listenToChanges(subscriber: ChangeListener<Editor>) = curEditor.addListener(subscriber)

    private fun getOrCreateEditor(editorClass: Class<out Editor>): Editor = map.getOrPut(editorClass, {
        editorClass.getConstructor(ShapesList:: class.java, GraphicsContext:: class.java)
            .newInstance(shapes, canvas.graphicsContext2D)
    })
}
```

### 3.4 Module: com.github.erotourtes.utils

Лістинг 7: Dimension.kt

```
package com.github.erotourtes.utils
```



```

import kotlin.math.abs

class Dimension {
    private var x1: Double = 0.0
    private var y1: Double = 0.0
    private var x2: Double = 0.0
    private var y2: Double = 0.0

    val width: Double
        get() = abs(x2 - x1)
    val height: Double
        get() = abs(y2 - y1)

    fun setStart(x: Double, y: Double): Dimension {
        x1 = x
        y1 = y
        return this
    }

    fun setEnd(x: Double, y: Double): Dimension {
        x2 = x
        y2 = y
        return this
    }

    fun copyTo(dst: Dimension) {
        dst.x1 = x1
        dst.y1 = y1
        dst.x2 = x2
        dst.y2 = y2
    }

    fun copyFrom(src: Dimension) = src.copyTo(this)

    fun getBoundaries(): Pair<Point, Point> {
        return Pair(
            Point(x1.coerceAtMost(x2), y1.coerceAtMost(y2)),
            Point(x1.coerceAtLeast(x2), y1.coerceAtLeast(y2))
        )
    }

    fun getRaw(): Pair<Point, Point> {
        return Pair(
            Point(x1, y1),
            Point(x2, y2)
        )
    }

    data class Point(val x: Double, val y: Double)

    override fun toString(): String = "Dimension(x1=$x1, y1=$y1, x2=$x2, y2=$y2)"
}

```

Лістинг 8: ExtensionFunctions.kt

```
package com.github.erotourtes.utils

import javafx.scene.canvas.GraphicsContext

fun GraphicsContext.fillRect(dm: Dimension) {
    val (s, e) = dm.getBoundaries()
    fillRect(s.x, s.y, e.x - s.x, e.y - s.y)
}

fun GraphicsContext.strokeRect(dm: Dimension) {
    val (s, e) = dm.getBoundaries()
    strokeRect(s.x, s.y, e.x - s.x, e.y - s.y)
}

fun GraphicsContext.fillOval(dm: Dimension) {
    val (s, e) = dm.getBoundaries()
    fillOval(s.x, s.y, e.x - s.x, e.y - s.y)
}

fun GraphicsContext.strokeOval(dm: Dimension) {
    val (s, e) = dm.getBoundaries()
    strokeOval(s.x, s.y, e.x - s.x, e.y - s.y)
}

fun GraphicsContext.strokeLine(dm: Dimension) {
    val (s, e) = dm.getRaw()
    strokeLine(s.x, s.y, e.x, e.y)
}

inline fun GraphicsContext.drawOnce(lambda: GraphicsContext.() -> Unit) {
    save()
    lambda(this)
    restore()
}
```

Лістинг 9: Utils.kt

```
package com.github.erotourtes.utils

import com.github.erotourtes.drawing.editor.Editor
import de.jensd.fx.glyphs.fontawesome.FontAwesomeIcon
import javafx.scene.control.ToggleGroup
import tornadofx.*

data class MenuItemInfo(
    val name: String,
    val kotlinClass : Class<out Editor>,
    var group: ToggleGroup? = null,
    var action: () -> Unit = {},
)

data class ToolbarItemInfo(
    val tooltip: String,
```

```

        val kotlinClass : Class<out Editor>,
        val icon: FontAwesomeIcon? = null,
    )

fun getToCornerDimension(dm: Dimension): Dimension {
    val w = dm.width
    val h = dm.height
    val (cx, cy) = dm.getRow().first

    val sX = cx - w
    val sY = cy - h

    return Dimension().setStart(sX, sY).setEnd(sX + w * 2, sY + h * 2)
}

const val g = 22 + 1
const val n = 100 + g

class PopupView : Fragment("Mode_selection_check") {
    private val action = super.scope as ScopeInfo

    override val root = vbox {
        style { prefWidth = 250.px }
        label("You_clicked_on_${action.name}")
        button("Close").action { close() }
    }

    data class ScopeInfo(val name: String) : Scope()
}

```

### 3.5 Module: com.github.erotourtes.styles

ЛІСТИНГ 10: ToolbarStyles.kt

```

package com.github.erotourtes.styles

import tornadofx.*
import javafx.scene.paint.Color

class ToolbarStyles : Stylesheet() {

    companion object {
        val toolbar by cssclass()
        val iconButton by cssclass()
        val icon by cssclass()
        val dark = c("#555")
        val light = Color.LIGHTSTEELBLUE!!
    }

    init {
        val toolbarHeight = 40.px
        toolbar {
            padding = box(5.px)
            spacing = 5.px
            minHeight = toolbarHeight
        }
    }
}

```

```

        maxHeight = toolbarHeight
        alignment = javafx.geometry.Pos.CENTER_LEFT
        borderWidth += box(0.px, 0.px, 1.px, 0.px)
        borderColor += box(dark)
    }

    iconButton {
        backgroundColor += Color.TRANSPARENT

        and(selected) {
            backgroundColor += dark
            icon { fill = light }
        }

        minHeight = toolbarHeight / 1.5
        maxHeight = toolbarHeight / 1.5
    }

    icon { fill = dark }
}
}

```

### 3.6 Module: com.github.erotourtes.drawing.editor

Лістинг 11: Editor.kt

```

package com.github.erotourtes.drawing.editor

import com.github.erotourtes.drawing.shape.*
import javafx.scene.paint.Color
import com.github.erotourtes.utils.*
import javafx.scene.canvas.GraphicsContext
import javafx.scene.input.MouseEvent

abstract class Editor(protected val shapes: ShapesList, protected val gc: GraphicsContext) {
    protected val dm = Dimension()
    protected abstract val shape: Shape

    open fun listenToEvents() {
        val c = gc.canvas
        c.setOnMousePressed(this::onMousePressed)
        c.setOnMouseDragged(this::onMouseDragged)
        c.setOnMouseReleased(this::onMouseReleased)
    }

    protected open fun onMousePressed(e: MouseEvent) {
        redraw()
        dm.setStart(e.x, e.y)
    }

    protected open fun onMouseDragged(e: MouseEvent) {
        redraw()

        dm.setEnd(e.x, e.y)
        previewLine()
    }
}

```

```

    }

    protected open fun onMouseReleased(e: MouseEvent) {
        if (e.isDragDetect) return // returns if mouse was not dragged
        shape.setDm(dm)
        shapes.add(shape.copy())
        redraw()
    }

    private fun drawAll() {
        for (shape in shapes) shape.draw()
    }

    private fun clear() = gc.clearRect(0.0, 0.0, gc.canvas.width, gc.canvas.height)

    protected fun redraw() {
        clear()
        drawAll()
    }

    abstract fun previewLine()

    protected fun setPreviewProperties() {
        gc.setLineDashes(5.0)
        gc.stroke = Color.BLACK
    }
}

```

#### Лістинг 12: Editors.kt

```

package com.github.erotourtes.drawing.editor

import com.github.erotourtes.drawing.shape.*
import com.github.erotourtes.utils.*
import javafx.scene.canvas.GraphicsContext
import javafx.scene.input.MouseEvent

class PointEditor(shapes: ShapesList, gc: GraphicsContext) : Editor(shapes, gc) {
    override val shape = Point(gc)

    override fun onMouseDragged(e: MouseEvent) {}

    override fun onMousePressed(e: MouseEvent) {
        dm.setEnd(e.x, e.y)
        super.onMousePressed(e)
        shape.setDm(dm)
    }

    override fun onMouseReleased(e: MouseEvent) {
        shapes.add(shape.copy())
        redraw()
    }

    override fun previewLine() {}
}

```

```

class LineEditor(shapes: ShapesList, gc: GraphicsContext) : Editor(shapes, gc) {
    override val shape = Line(gc)

    override fun previewLine() = gc.drawOnce {
        setPreviewProperties()
        strokeLine(dm)
    }
}

class RectEditor(shapes: ShapesList, gc: GraphicsContext) : Editor(shapes, gc) {
    override val shape = Rect(gc)
    override fun previewLine() = gc.drawOnce {
        setPreviewProperties()
        strokeRect(getToCornerDimension(dm))
    }
}

class EllipseEditor(shapes: ShapesList, gc: GraphicsContext) : Editor(shapes, gc) {
    override val shape = Ellipse(gc)
    override fun previewLine() = gc.drawOnce {
        setPreviewProperties()
        strokeOval(getToCornerDimension(dm))
        strokeRect(getToCornerDimension(dm))
    }
}

class EmptyEditor(shapes: ShapesList, gc: GraphicsContext) : Editor(shapes, gc) {
    override val shape = object : Shape(gc) {
        override fun draw() {}
    }

    override fun onMouseDragged(e: MouseEvent) {}
    override fun onMousePressed(e: MouseEvent) {}
    override fun onMouseReleased(e: MouseEvent) {}
    override fun previewLine() {}
}

```

Лістинг 13: ShapesList.kt

```

package com.github.erotourtes.drawing.editor

import com.github.erotourtes.drawing.shape.Shape

class ShapesList(n: Int) : Iterable<Shape> {
    private val shapeArr = Array<Shape?>(n) { null }
    private var shapeIndex = 0

    val size: Int
        get() = shapeIndex

    fun add(sh: Shape) {
        if (shapeIndex == shapeArr.size) throw IllegalArgumentException("History is overflowed")
        shapeArr[shapeIndex++] = sh
    }
}

```

```

    override fun iterator(): Iterator<Shape> = ShapeIterator()

    inner class ShapeIterator : Iterator<Shape> {
        private var curIndex = 0
        override fun hasNext(): Boolean = curIndex < size
        override fun next(): Shape = if (hasNext()) shapeArr[curIndex++]!! else throw Illegal
    }
    override fun toString(): String = "ShapesList(index=$shapeIndex)"
}

```

### 3.7 Module: com.github.erotourtes.drawing.shape

Лістинг 14: Shape.kt

```

package com.github.erotourtes.drawing.shape

import com.github.erotourtes.utils.Dimension
import javafx.scene.canvas.GraphicsContext
import javafx.scene.paint.Color
import java.lang.RuntimeException

abstract class Shape(val gc: GraphicsContext) {
    protected val dm = Dimension()

    var colorFill: Color = Color.BLACK
    var colorStroke: Color = Color.BLACK

    abstract fun draw()

    open fun setDm(curDm: Dimension) = curDm.copyTo(dm)

    fun setProperties() {
        with(gc) {
            fill = colorFill
            stroke = colorStroke
        }
    }

    fun copy(): Shape {
        try {
            val shape = this::class.java.getConstructor(GraphicsContext::class.java).newInstance()
            shape.dm.copyFrom(dm)
            shape.colorFill = colorFill
            shape.colorStroke = colorStroke
            return shape
        } catch (e: Exception) {
            throw RuntimeException("Can't copy a shape")
        }
    }
}

```

Лістинг 15: Shapes.kt

```

package com.github.erotourtes.drawing.shape

```

```

import com.github.erotourtes.utils.*
import javafx.scene.canvas.GraphicsContext
import javafx.scene.paint.Color

class Point(gc: GraphicsContext) : Shape(gc) {
    override fun draw() {
        val radius = 12.0
        gc.drawOnce {
            setProperties()
            val (x, y) = dm.getBoundaries().first
            fillOval(x, y, radius, radius)
        }
    }
}

class Line(gc: GraphicsContext) : Shape(gc) {
    override fun draw() {
        gc.drawOnce {
            setProperties()
            strokeLine(dm)
        }
    }
}

class Rect(gc: GraphicsContext) : Shape(gc) {
    init {
        colorFill = Color.TRANSPARENT
        colorStroke = Color.BLACK
    }

    override fun setDm(curDm: Dimension) = dm.copyFrom(getToCornerDimension(curDm))

    override fun draw() {
        gc.drawOnce {
            setProperties()
            fillRect(dm)
            strokeRect(dm)
        }
    }
}

class Ellipse(gc: GraphicsContext) : Shape(gc) {
    init {
        colorFill = Color.ORANGE
        colorStroke = Color.BLACK
    }

    override fun draw() {
        gc.drawOnce {
            setProperties();
            fillOval(dm)
            strokeOval(dm)
        }
    }
}

```



```

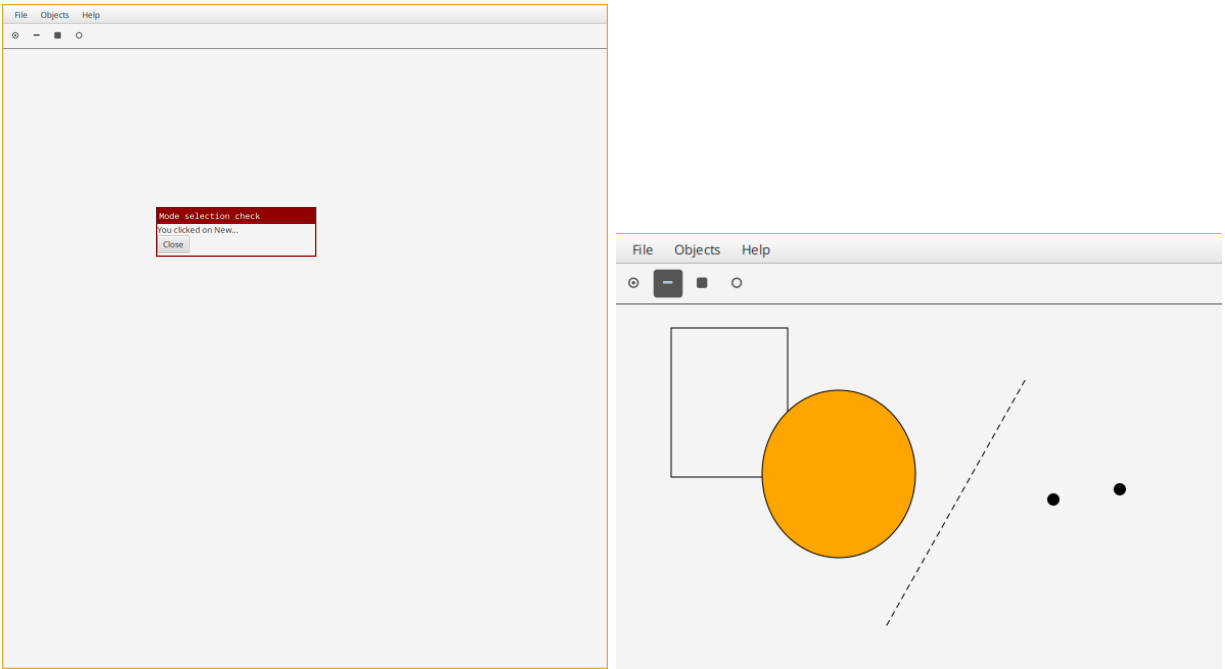
    }

    override fun setDm(curDm: Dimension) = dm.copyFrom(getToCornerDimension(curDm))
}

```

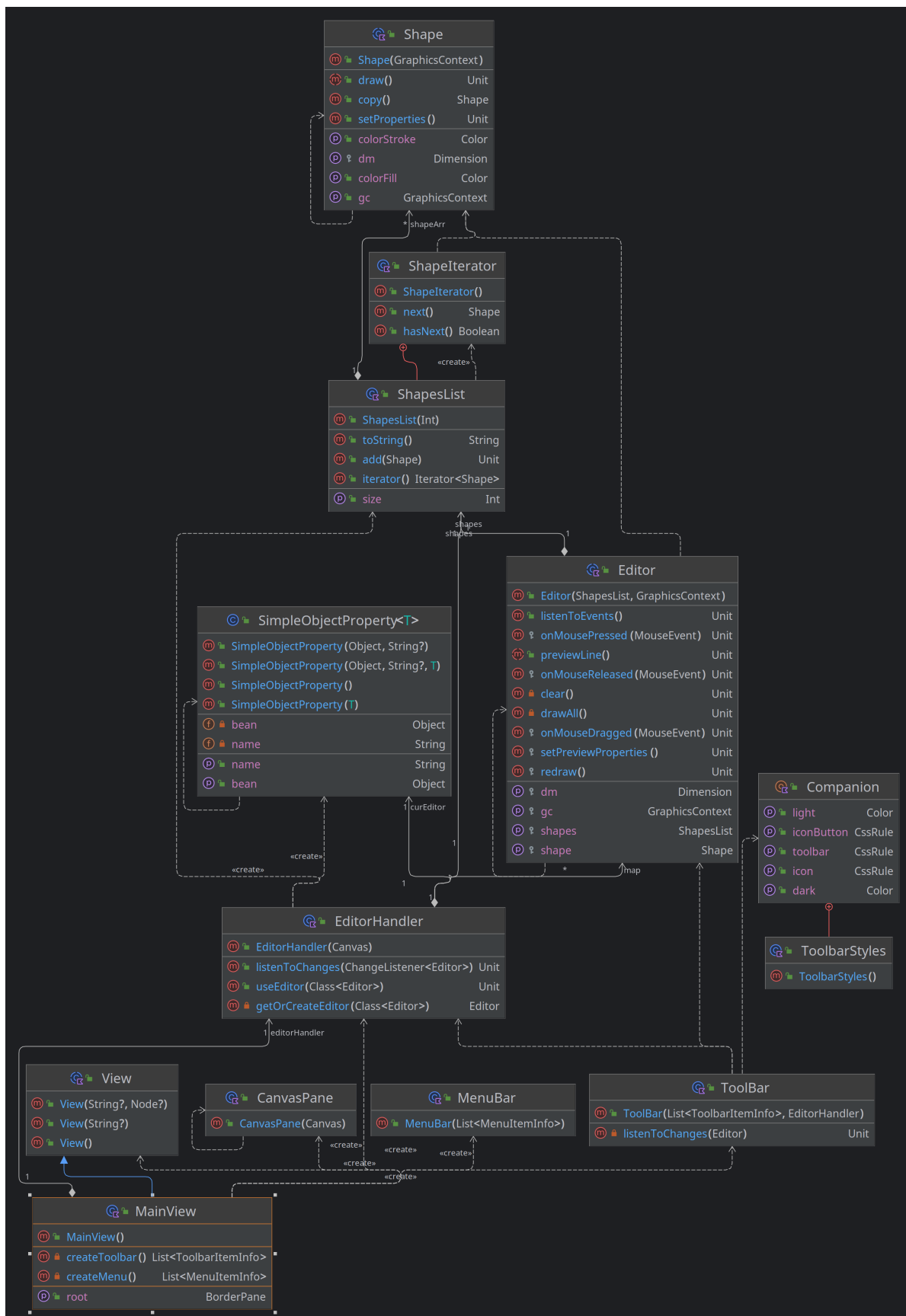
## 4 Ілюстрації:

### 4.1 Images



## 4.2 UML





## 5 Висновки:

Отже, я отримав вміння та навички використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм на основі класів Kotlin, запрограмувавши графічний інтерфейс користувача.