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

«ООП»

Виконав:

Студент групи ІМ-12

Пастушок Вадим Андрійович

Номер у списку групи: 19

Перевірив: Порєв Віктор Миколайович

Київ 2022

**Варіант завдання**

**"Гумовий" слід** суцільна лінія чорного кольору **Прямокутник** - по двом протилежним кутам, чорний контур з білим заповненням

**Еліпс** - від центру до одного з кутів, чорний контур без заповнення

MainActivity

class MainActivity : AppCompatActivity() {  
  
 private lateinit var binding: ActivityMainBinding  
 private lateinit var drawingBoard: DrawingBoard  
 private lateinit var popupMenu: PopupMenu  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(LayoutInflater.from(this))  
 setContentView(binding.root)  
 initDrawingBoard()  
 initPopupMenu()  
 initListeners()  
 }  
  
 private fun initDrawingBoard() {  
 drawingBoard = DrawingBoard(this)  
 drawingBoard.layoutParams = LinearLayout.LayoutParams(  
 LinearLayout.LayoutParams.MATCH\_PARENT,  
 LinearLayout.LayoutParams.MATCH\_PARENT  
 )  
 binding.boardLayout.addView(drawingBoard)  
 }  
  
 private fun initPopupMenu() {  
 popupMenu = PopupMenu(this, binding.moreBtn)  
 popupMenu.menu.add(getString(R.string.point))  
 popupMenu.menu.add(getString(R.string.line))  
 popupMenu.menu.add(getString(R.string.rect))  
 popupMenu.menu.add(getString(R.string.ellipse))  
 popupMenu.setOnMenuItemClickListener **{** when (it.title) {  
 getString(R.string.point) -> setSelectedShape(ShapeType.POINT)  
 getString(R.string.line) -> setSelectedShape(ShapeType.LINE)  
 getString(R.string.rect) -> setSelectedShape(ShapeType.RECTANGLE)  
 getString(R.string.ellipse) -> setSelectedShape(ShapeType.ELLIPSE)  
 }  
 true  
 **}** }  
  
 private fun setSelectedShape(shapeType: ShapeType) {  
 drawingBoard.setDrawingShape(shapeType)  
 binding.pointBtn.setBackgroundColor(  
 if (shapeType == ShapeType.POINT) Color.WHITE else Color.TRANSPARENT)  
 binding.lineBtn.setBackgroundColor(  
 if (shapeType == ShapeType.LINE) Color.WHITE else Color.TRANSPARENT)  
 binding.rectBtn.setBackgroundColor(  
 if (shapeType == ShapeType.RECTANGLE) Color.WHITE else Color.TRANSPARENT)  
 binding.ellipseBtn.setBackgroundColor(  
 if (shapeType == ShapeType.ELLIPSE) Color.WHITE else Color.TRANSPARENT)  
 }  
  
 private fun initListeners() {  
 binding.moreBtn.setOnClickListener **{** popupMenu.show()  
 **}** binding.pointBtn.setOnClickListener **{** setSelectedShape(ShapeType.POINT)  
 **}** binding.lineBtn.setOnClickListener **{** setSelectedShape(ShapeType.LINE)  
 **}** binding.rectBtn.setOnClickListener **{** setSelectedShape(ShapeType.RECTANGLE)  
 **}** binding.ellipseBtn.setOnClickListener **{** setSelectedShape(ShapeType.ELLIPSE)  
 **}** }  
}

DrawingBoard

class DrawingBoard(context: Context) : SurfaceView(context) {  
  
 private var selectedShapeType = ShapeType.POINT  
 private var selectedShape: Shape? = null  
 fun setDrawingShape(shape: ShapeType) {  
 selectedShapeType = shape  
 }  
  
 fun clearPreviousFigures() {  
 shapes.clear()  
 val canvas = holder.lockCanvas()  
 drawPreviousShapes(canvas)  
 holder.unlockCanvasAndPost(canvas)  
 }  
  
 private val shapes = mutableListOf<Shape>()  
 private fun drawPreviousShapes(canvas: Canvas) {  
 canvas.drawColor(Color.WHITE)  
 for(shape in shapes) { shape.draw(canvas) }  
 }  
  
 @SuppressLint("ClickableViewAccessibility")  
 override fun onTouchEvent(event: MotionEvent): Boolean {  
 val canvas = holder.lockCanvas()  
 drawPreviousShapes(canvas)  
 when (event.action) {  
 MotionEvent.ACTION\_DOWN -> {  
 onActionDown(canvas, event)  
 }  
 MotionEvent.ACTION\_MOVE -> {  
 onActionMove(canvas, event)  
 }  
 MotionEvent.ACTION\_UP -> {  
 onActionUp(canvas)  
 }  
 }  
 holder.unlockCanvasAndPost(canvas)  
 return true  
 }  
  
 private fun onActionDown(canvas: Canvas, event: MotionEvent) {  
 selectedShape = Shape.createShape(selectedShapeType, event.x, event.y)  
 selectedShape?.preDraw(canvas)  
 }  
  
 private fun onActionMove(canvas: Canvas, event: MotionEvent) {  
 selectedShape?.moveX(event.x)  
 selectedShape?.moveY(event.y)  
 selectedShape?.preDraw(canvas)  
 }  
  
 private fun onActionUp(canvas: Canvas) {  
 selectedShape?.let {  
 it.draw(canvas)  
 shapes.add(it)  
 }  
 selectedShape = null  
 }  
}

Shape

package com.example.lab3.shapes  
  
import android.graphics.Canvas  
import android.graphics.Color  
import android.graphics.Paint  
  
  
abstract class Shape {  
  
 val preDrawPaint = Paint(Paint.*ANTI\_ALIAS\_FLAG*)  
 val strokePaint = Paint(Paint.*ANTI\_ALIAS\_FLAG*)  
 val fillPaint = Paint(Paint.*ANTI\_ALIAS\_FLAG*)  
 init {  
 preDrawPaint.*color* = Color.*BLACK* preDrawPaint.*style* = Paint.Style.*STROKE* }  
  
 abstract fun moveX(x: Float)  
  
 abstract fun moveY(y: Float)  
  
 abstract fun preDraw(canvas: Canvas)  
  
 abstract fun draw(canvas: Canvas)  
  
 companion object {  
 fun createShape(shapeType: ShapeType, startX: Float, startY: Float): Shape {  
 return when (shapeType) {  
 ShapeType.*POINT* -> {  
 Point(startX, startY)  
 }  
 ShapeType.*LINE* -> {  
 Line(startX, startY)  
 }  
 ShapeType.*RECTANGLE* -> {  
 Rectangle(startX, startY)  
 }  
 ShapeType.*ELLIPSE* -> {  
 Ellipse(startX, startY)  
 }  
 }  
 }  
 }  
}

Point

package com.example.lab3.shapes  
  
import android.graphics.Canvas  
import android.graphics.Color  
import android.graphics.Paint  
  
class Point(private var x: Float, private var y: Float) : Shape() {  
  
 companion object {  
 private const val radius = 16f  
 }  
  
 private var paint = Paint(Paint.*ANTI\_ALIAS\_FLAG*)  
 init {  
 paint.*color* = Color.*BLACK* }  
  
 override fun moveX(x: Float) {  
 this.x = x  
 }  
  
 override fun moveY(y: Float) {  
 this.y = y  
 }  
  
 override fun preDraw(canvas: Canvas) {  
 canvas.drawCircle(x, y, radius, preDrawPaint)  
 }  
  
 override fun draw(canvas: Canvas) {  
 canvas.drawCircle(x, y, radius, paint)  
 }  
}

Line

package com.example.lab3.shapes  
  
import android.graphics.Canvas  
import android.graphics.Color  
import android.graphics.Paint  
  
class Line(private val startX: Float, private val startY: Float) : Shape() {  
  
 companion object {  
 private const val strokeWidth = 4f  
 }  
  
 private var endX = startX  
 private var endY = startY  
  
 private var paint = Paint(Paint.ANTI\_ALIAS\_FLAG)  
 init {  
 paint.color = Color.BLACK  
 paint.strokeWidth = strokeWidth  
 }  
  
 override fun moveX(x: Float) {  
 endX = x  
 }  
  
 override fun moveY(y: Float) {  
 endY = y  
 }  
  
 override fun preDraw(canvas: Canvas) {  
 canvas.drawLine(startX, startY, endX, endY, preDrawPaint)  
 }  
  
 override fun draw(canvas: Canvas) {  
 canvas.drawLine(startX, startY, endX, endY, paint)  
 }  
}

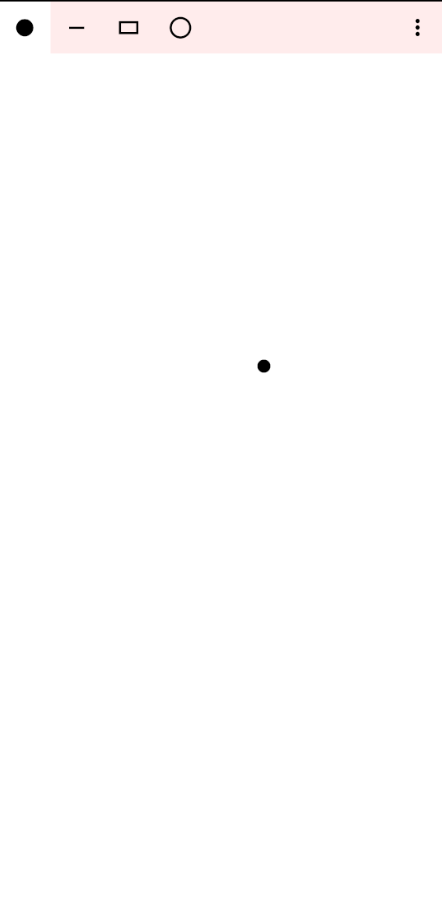
Rectangle

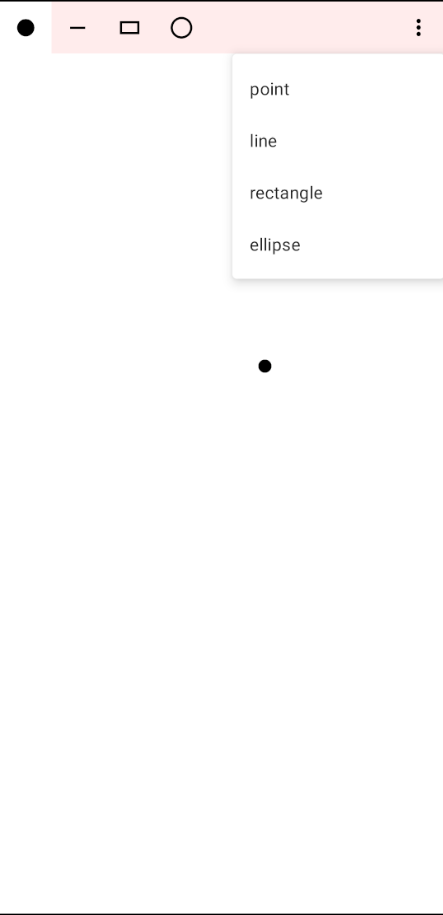
package com.example.lab3.shapes  
  
import android.graphics.Canvas  
import android.graphics.Color  
import android.graphics.Paint  
  
class Rectangle(private val centerX: Float, private val centerY: Float): Shape() {  
  
 companion object {  
 private const val strokeWidth = 8f  
 }  
  
 private var startX = centerX  
 private var startY = centerY  
  
 private var endX = centerX  
 private var endY = centerY  
  
 init {  
 strokePaint.*style* = Paint.Style.*STROKE* strokePaint.*color* = Color.*BLACK* strokePaint.*strokeWidth* = strokeWidth  
  
 fillPaint.*style* = Paint.Style.*FILL* fillPaint.*color* = Color.*WHITE* }  
  
 override fun moveX(x: Float) {  
 endX = x  
 }  
  
 override fun moveY(y: Float) {  
 endY = y  
 }  
  
 override fun preDraw(canvas: Canvas) {  
 canvas.drawRect(startX, startY, endX, endY, preDrawPaint)  
 }  
  
 override fun draw(canvas: Canvas) {  
 canvas.drawRect(startX, startY, endX, endY, strokePaint)  
 canvas.drawRect(startX, startY, endX, endY, fillPaint)  
 }  
}

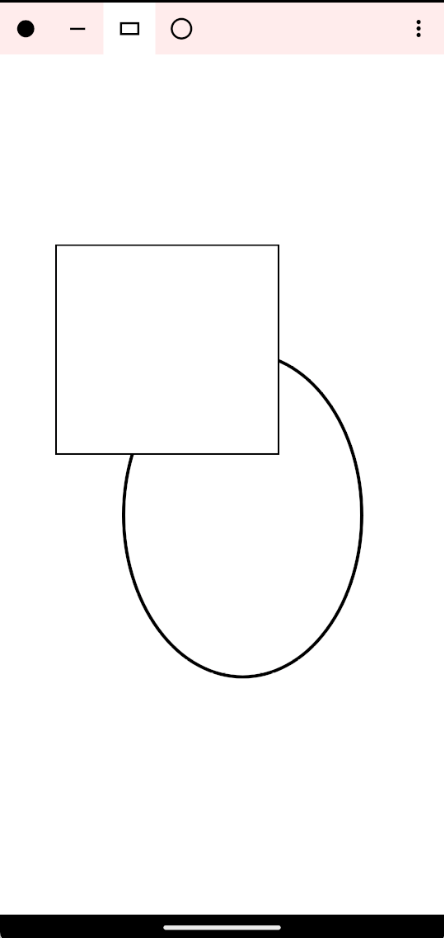
Ellipse

package com.example.lab3.shapes  
  
import android.graphics.Canvas  
import android.graphics.Color  
import android.graphics.Paint  
  
class Ellipse(private val startX: Float, private val startY: Float): Shape() {  
  
 companion object {  
 private const val strokeWidth = 8f  
 }  
  
 private var endX = startX  
 private var endY = startY  
  
 init {  
 strokePaint.*style* = Paint.Style.*STROKE* strokePaint.*color* = Color.*BLACK* strokePaint.*strokeWidth* = strokeWidth  
 }  
  
 override fun moveX(x: Float) {  
 endX = x  
 }  
  
 override fun moveY(y: Float) {  
 endY = y  
 }  
  
 override fun preDraw(canvas: Canvas) {  
 canvas.drawOval(startX, startY, endX, endY, preDrawPaint)  
 }  
  
 override fun draw(canvas: Canvas) {  
 canvas.drawOval(startX, startY, endX, endY, strokePaint)  
 }  
}

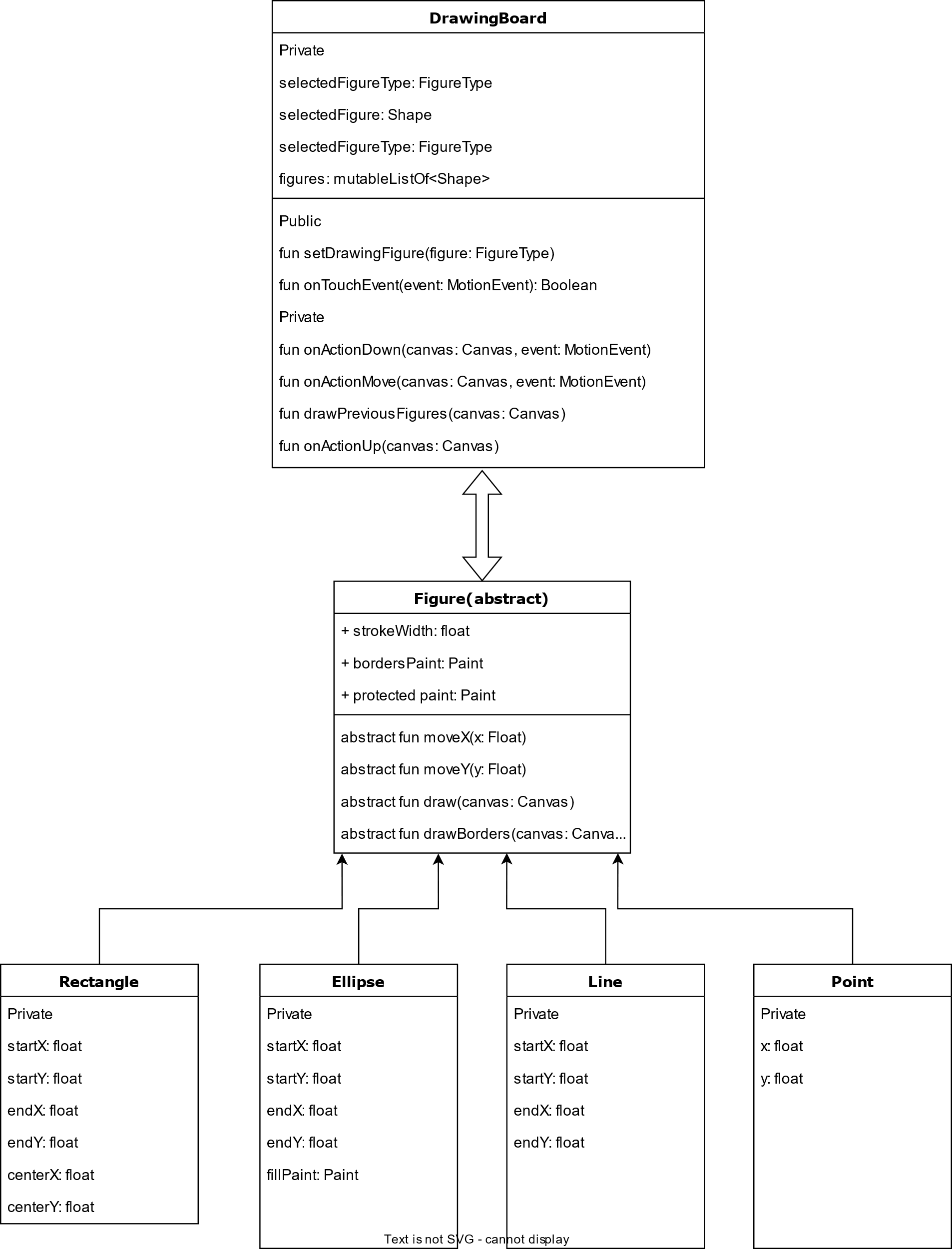
Скріншоти







Діаграма



**Висновки:**

У цій лабораторній роботі ми навчились застосовувати поліморфізм, наслідування та інкапсуляцію, працювали з розробкою графічного інтерфейсу для Android та відображенням геометричних фігур на канвасі.