PROJECT REPORT: Simple Drawing Application

Student Name: İrem Carus

Student ID: 240201008

Course: CS2001 - Object-Oriented Programming (Java)

1. Project Overview

This project is a Java Swing-based application developed to demonstrate the core principles of Object-Oriented Programming (OOP). The application provides a digital canvas where users can interactively draw lines, rectangles, and ovals. It features a control panel for customizing shape properties and a status bar for real-time feedback.

2. Technical Architecture & OOP Principles

The application is designed with a clear separation between data (Model) and user interface (View).

A. Shape Hierarchy (Inheritance & Polymorphism)

- **MyShape (Abstract Base Class):** At the heart of the project is the MyShape abstract class. It encapsulates common attributes like coordinates (x1, y1, x2, y2), Color, and a boolean for filling. It defines an abstract draw(Graphics g) method, enforcing a contract for all subclasses.
- **Subclasses:** MyLine, MyRectangle, and MyOval extend MyShape. They implement the draw method using specific Java 2D Graphics API calls (e.g., drawLine, drawRect, fillOval).
- **Polymorphism:** All shapes are stored in a single ArrayList<MyShape>. During the paintComponent cycle, the program iterates through this list and calls draw() on each object. The JVM dynamically determines which specific shape's draw method to execute at runtime.

B. Encapsulation

All fields in the shape classes are marked as private. Access to these fields is provided through public getter and setter methods, ensuring data integrity and following Java Bean standards.

3. Event Handling and Logic

The application utilizes a multi-layered event-handling approach:

- **Mouse Event Handling:** A MouseHandler (extending MouseAdapter) is implemented within the DrawingPanel. It uses mousePressed to set the starting point, mouseDragged for real-time shape preview, and mouseReleased to permanently store the shape in the collection.
- **GUI Control Logic:** ActionListener interfaces (via Lambda expressions) are used for the JComboBox, JButton, and JCheckBox components in the MainFrame to update the drawing state (e.g., changing colors or shape types).

4. Persistent Drawing

To ensure that drawings are not lost when the window is resized or minimized, the application overrides the paintComponent(Graphics g) method. It clears the canvas and redraws every shape stored in the ArrayList every time a refresh is triggered.

5. Bonus Features Implemented

To go beyond the basic requirements, the following features were added:

- **Undo Functionality:** A dedicated button that removes the most recent shape from the ArrayList.
- **Shape Filling:** A "Filled" checkbox that toggles between drawing outlines and solid shapes for rectangles and ovals.
- **Status Bar:** A real-time display at the bottom of the window showing the current mouse coordinates (x, y) and the active drawing tool.
- **Professional Documentation:** Comprehensive **Javadoc** comments are provided for all major classes and methods.

6. Screenshot: