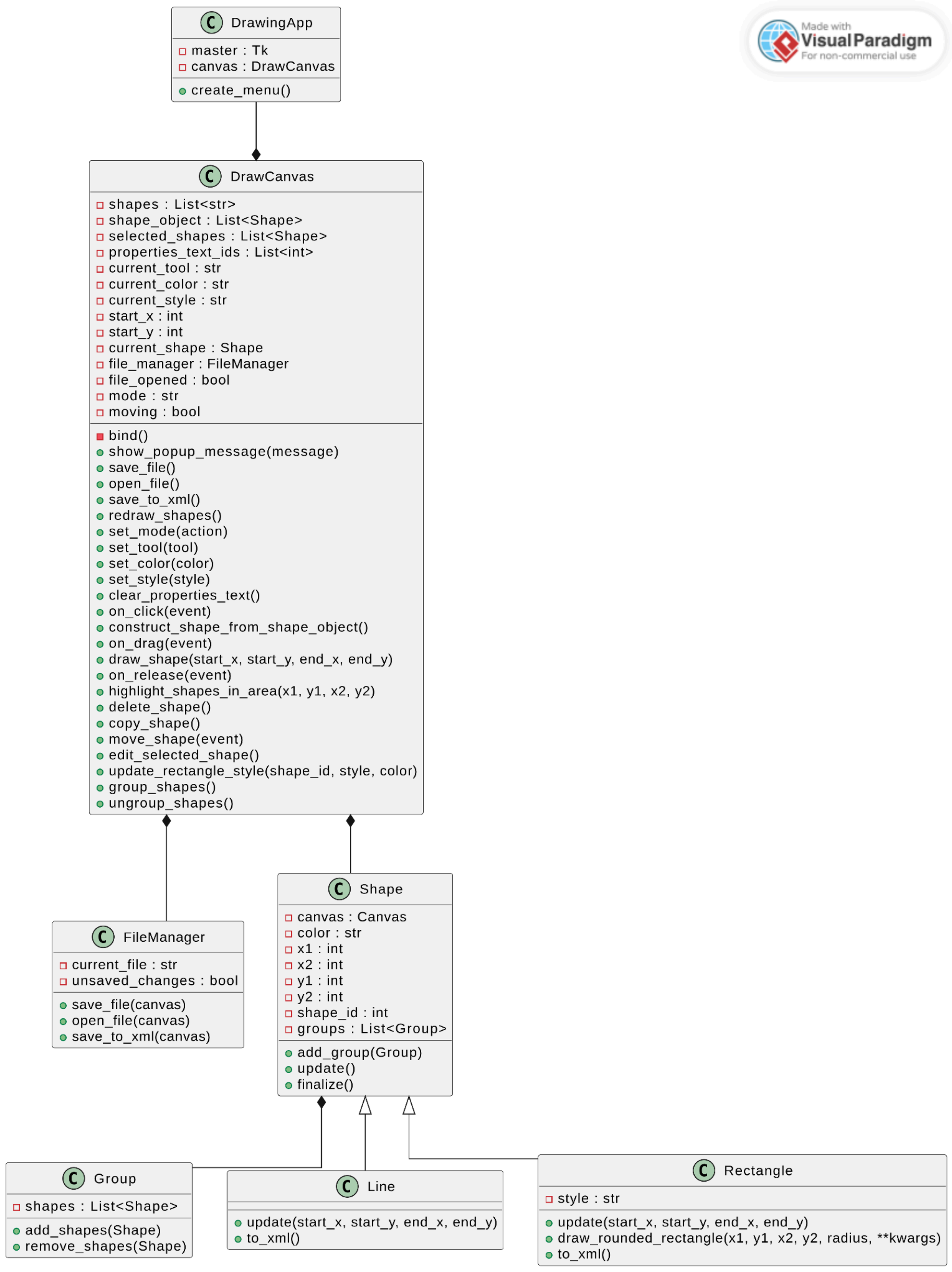
Date:6/05/2024

Design Activity - Drawing Editor
Bonus Assignment

Team Members:
Archisha Panda-2022111019
Gargi Shroff-2022114009
Ketaki Shetye-2022114013
Paridhi Jain-2022101119

SketchShapes

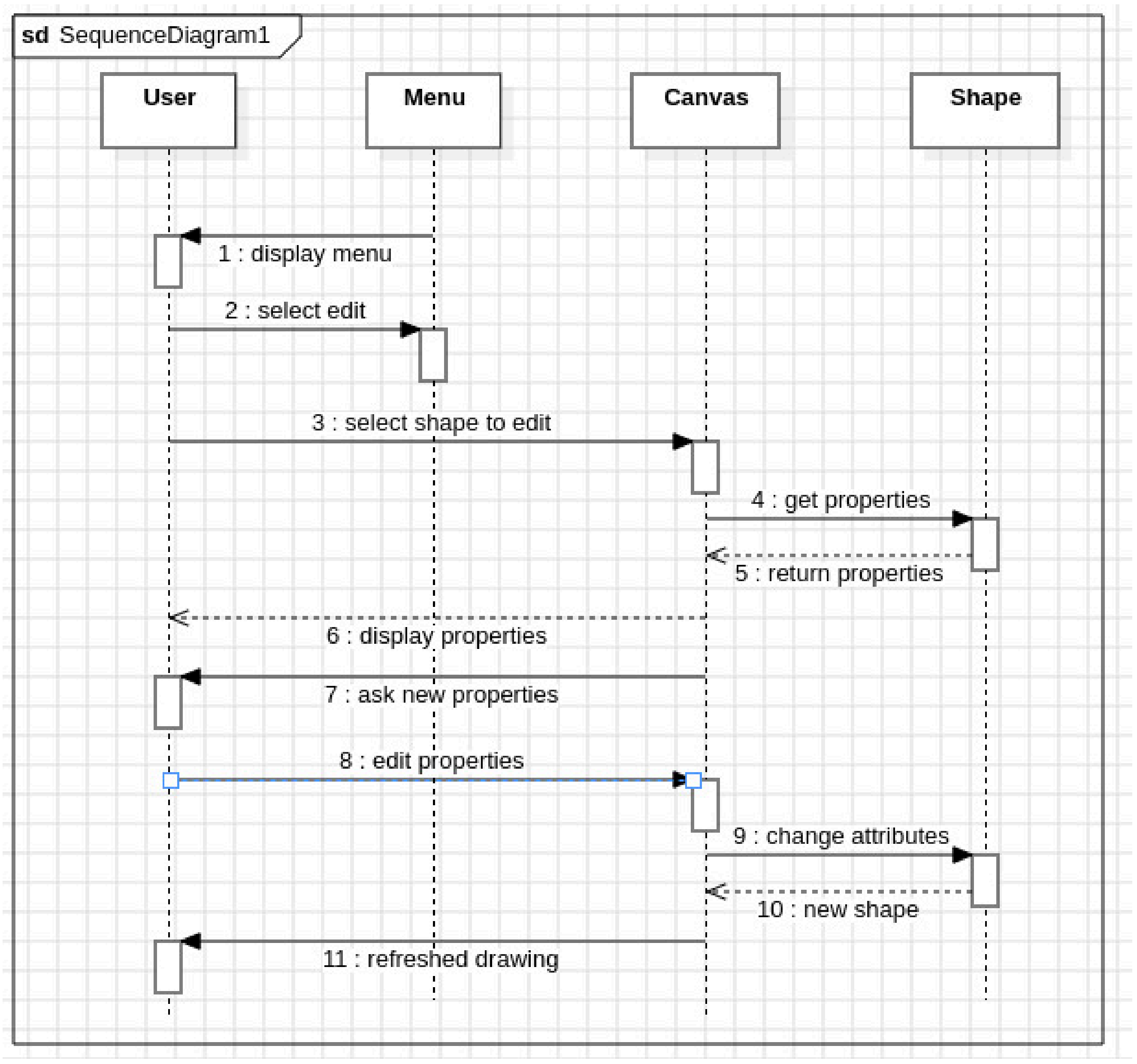
UML Diagram



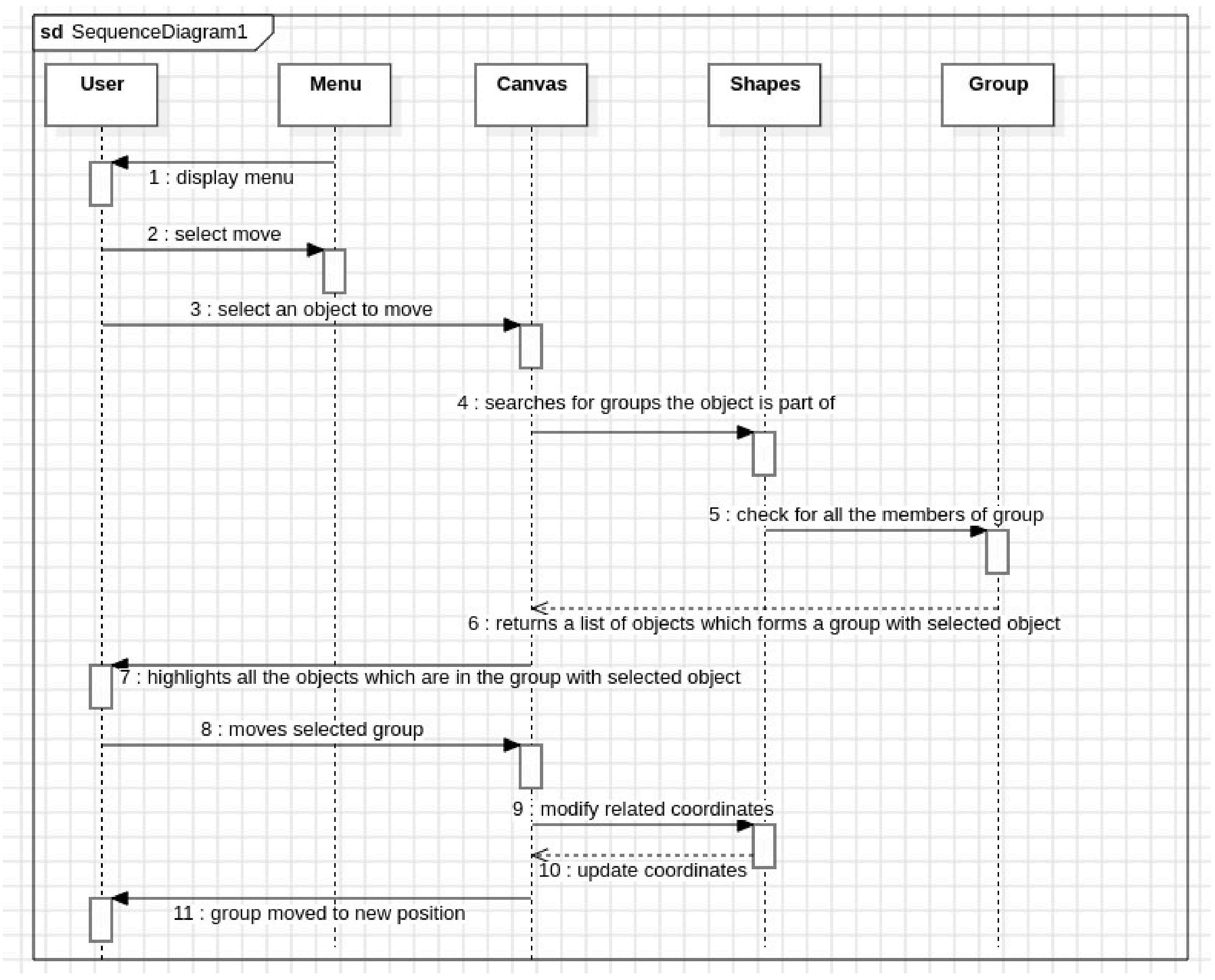
DrawingApp
master : Tk
canvas : DrawCanvas
create_menu()
DrawCanvas
shapes : List<str>
shape_object : List<Shape>
selected_shapes : List<Shape>
properties_text_ids : List<int>
current_tool : str
current_color : str
current_style : str
start_x : int
start_y : int
current_shape : Shape
file_manager : FileManager
file_opened : bool
mode : str
moving : bool
bind()
show_popup_message(message)
save_file()
open_file()
save_to_xml()
redraw_shapes()
set_mode(action)
set_tool(tool)
set_color(color)
set_style(style)
clear_properties_text()
on_click(event)
construct_shape_from_shape_object()
on_drag(event)
draw_shape(start_x, start_y, end_x, end_y)
on_release(event)
highlight_shapes_in_area(x1, y1, x2, y2)
delete_shape()
copy_shape()
move_shape(event)
edit_selected_shape()
update_rectangle_style(shape_id, style, color)
group_shapes()
ungroup_shapes()
FileManager
current_file : str
unsaved_changes : bool
save_file(canvas)
open_file(canvas)
save_to_xml(canvas)
Shape
canvas : Canvas
color : str
x1 : int
x2 : int
y1 : int
y2 : int
shape_id : int
groups : List<Group>
add_group(Group)
update()
finalize()
Group
shapes : List<Shape>
add_shapes(Shape)
remove_shapes(Shape)
Line
update(start_x, start_y, end_x, end_y)
to_xml()
Rectangle
style : str
update(start_x, start_y, end_x, end_y)
draw_rounded_rectangle(x1, y1, x2, y2, radius, **kwargs)
to_xml()

Sequence Diagrams

1. Mouse selection of an element that is in a group, drag to a new position, update of the model, refresh of the drawing.



2. Menu choice to edit the current object, user change of attributes, user confirmation of change, update of element in the model, refresh of the drawing.


sd SequenceDiagram1
User
Menu
Canvas
Shapes
Group
1 : display menu
2 : select move
3 : select an object to move
4 : searches for groups the object is part of
5 : check for all the members of group
6 : returns a list of objects which forms a group with selected object
7 : highlights all the objects which are in the group with selected object
8 : moves selected group
9 : modify related coordinates
10 : update coordinates
11 : group moved to new position

Class Diagram

Class Name	**Responsibility**
DrawingApp	- Represents the main application window - Contains the drawing canvas.
DrawCanvas	- Handles all drawing-related operations, such as setting tools, colours, styles, drawing shapes, selecting/deselecting shapes, grouping/ungrouping shapes, saving/opening files, and managing the file manager.
FileManager	- Responsible for saving and opening files, as well as saving the canvas to an XML format.
Group	- Represents a group of shapes. - Can add or remove shapes from the group.
Shape	- Base class for different types of shapes. - Stores properties like canvas, colour, coordinates, shape ID, and associated groups. - Provides methods for updating and finalising the shape.
Line	- Represents a line shape. - Can update its coordinates and convert to XML format.
	- Represents a rectangle shape. - Can update its coordinate, draw rounded/straight-edged rectangles and convert to XML format

Analysing Design

The design phase stands as a crucial cornerstone in the development lifecycle, setting the stage for the entire process. It's during this phase that pivotal decisions regarding system architecture, data structures, interfaces, and algorithms are made. A meticulously crafted design ensures that the final product aligns with specified requirements, possesses scalability, maintainability, and operates efficiently. Moreover, this phase serves to unearth potential risks and challenges early on, paving the way for preemptive mitigation strategies before significant resources are committed. Through comprehensive planning and design, errors can be minimised, the implementation process can be streamlined, ultimately resulting in the delivery of high-calibre software that fulfils all expectations.

Defining what constitutes a good design can often delve into subjective and immeasurable territory. Nonetheless, design models within the software market establish sound design principles. These principles encompass abstraction, separation of concerns, modularity, information hiding, stepwise refinement, class design, and functional independence, among others.

Separation Of Concerns: This principle encourages splitting a complex problem into simpler problems that are easier and can be solved/optimised independently. The above design aligns itself with this principle as each class is ensured to have only one responsibility. For example DrawCanvas handles all canvas related operations like selecting and Shape handles shape related operations like setting of properties, copying, moving and so on.

Modularity: Modularity has been achieved by compartmentalisation of data and functions. A standard practice we followed throughout our design was “operate only with local data” which means that the functions modified only with data from the same class adding simplicity to our design.

High Cohesion: As mentioned in separation of concerns and modularity, our classes are very specific and thus highly cohesive. All operations of a class are closely related and work to achieve a common goal of the larger system they are a part of. For example for achieving the functionality of editing, the user first selects edit from the menu (create_menu). This is conveyed to Canvas which enables selecting of objects to edit (highlight_selected_shapes). Once these objects are selected, user input is asked for new properties following which all the data is sent to Shapes to draw the new shapes and update the canvas. All the functions are in the DrawCanvas class which in turn calls external class functions.

Law of Demeter: Also called the ‘principle of least knowledge’, the fundamental goal is to enhance modularity and promote a more decoupled system by limiting the number of classes with which an object interacts. It promotes only talking to immediate friends and not strangers. The drawing editor has varied functionalities like FileHandling which are not linked with drawing of objects directly. Hence they are handled by separate classes and communicate only when save_file or open_file is called that too through the Canvas Class.

Extensibility and Reusability : Some aspects of the system have been designed specifically keeping in mind reusability and extensibility. For example, the system can be modified to add new shapes, editing functionalities like resize, rotate, other file formats for export like JPEG and so on.

Information Hiding: It refers to the principle of building a controlled interface for accessing the internal details like data structures, algorithms, resource allocation policy that have been hidden. For example, draw_shape simply assumes the correct shape is drawn without having to worry about whether to call create_line or create_rectangle. Similarly export_to_xml automatically calls functions from line or rectangle without the FileManager class having to worry about the same.