# Design Overview for Shape Duplicator Program

Name: Sovithyea Prach

Student ID: 105270743

# Summary of Program

This program is essentially a drawing tool that gives users an opportunity to create, manipulate, and play with different sorts of shapes, such as rectangles, circles, and lines, with all sorts of colours as well. Users can spawn these shapes by clicking the canvas, with the possibility of changing the type of shape depending on the key pressed.

Shapes can be selected, resized, deleted, and even saved onto a file for later use. With that, the program allows the creation of random shapes with random colours that also supports the saving/loading feature.

Include a sketch of sample output to illustrate your idea.

# Required Roles

Describe each of the classes, interfaces, and any enumerations you will create. Use a different table to describe each role you will have, using the following table templates.

Table 1: <<Shape.cs>> (abstract class) details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Stores state | Fields:  \_color: Color  \_x: float  \_y: float  \_selected: bool | Private fields to store shape's appearance and selection |
| Set/Get properties | Properties:  Color: Color  X: float  Y: float  Selected: bool | Public getters/setters for all private fields |
| Drawing logic | |  |  | | --- | --- | |  | Abstract methods:  Draw(): void  DrawOutline(): void  IsAt(Point2D pt): bool | | Must be overridden in all shape subclasses |
| File handling | Virtual methods: SaveTo(StreamWriter): void LoadFrom(StreamReader): void  Private fields to store shape's appearance and selection | Enables shapes to be saved to/loaded from text files |

Table 1: <<MyRectangle.cs>> details

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
| Stores size | Fields:  \_width: int  \_height: int | Rectangle-specific dimensions |
| Set/Get size | |  |  | | --- | --- | |  | Properties:  Width: int  Height: int | | |  |  | | --- | --- | |  | Used for resizing rectangles | |
| Drawing logic | Methods:  Draw(): void  DrawOutline(): void  IsAt(Point2D pt): bool | Implements required shape methods |
| File handling | |  |  | | --- | --- | |  | Overrides:  SaveTo(StreamWriter): void LoadFrom(StreamReader): void | | Writes/reads width and height to/from file |

Table 1: <<>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
|  | Field type, parameter and return types |  |

Table 1: <<role name>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
|  | Field type, parameter and return types |  |

Table 1: <<role name>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
|  | Field type, parameter and return types |  |

Table 1: <<role name>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
|  | Field type, parameter and return types |  |

Table 1: <<role name>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
|  | Field type, parameter and return types |  |

Table 1: <<role name>> details – duplicate

|  |  |  |
| --- | --- | --- |
| Responsibility | Type Details | Notes |
|  | Field type, parameter and return types |  |

Table 2: <<enumeration name>> details

|  |  |
| --- | --- |
| Value | Notes |
|  |  |

# Class Diagram

Provide an initial design for your program in the form of a class diagram.

# Sequence Diagram

Provide a sequence diagram showing how your proposed classes will interact to achieve a specific piece of functionality in your program.