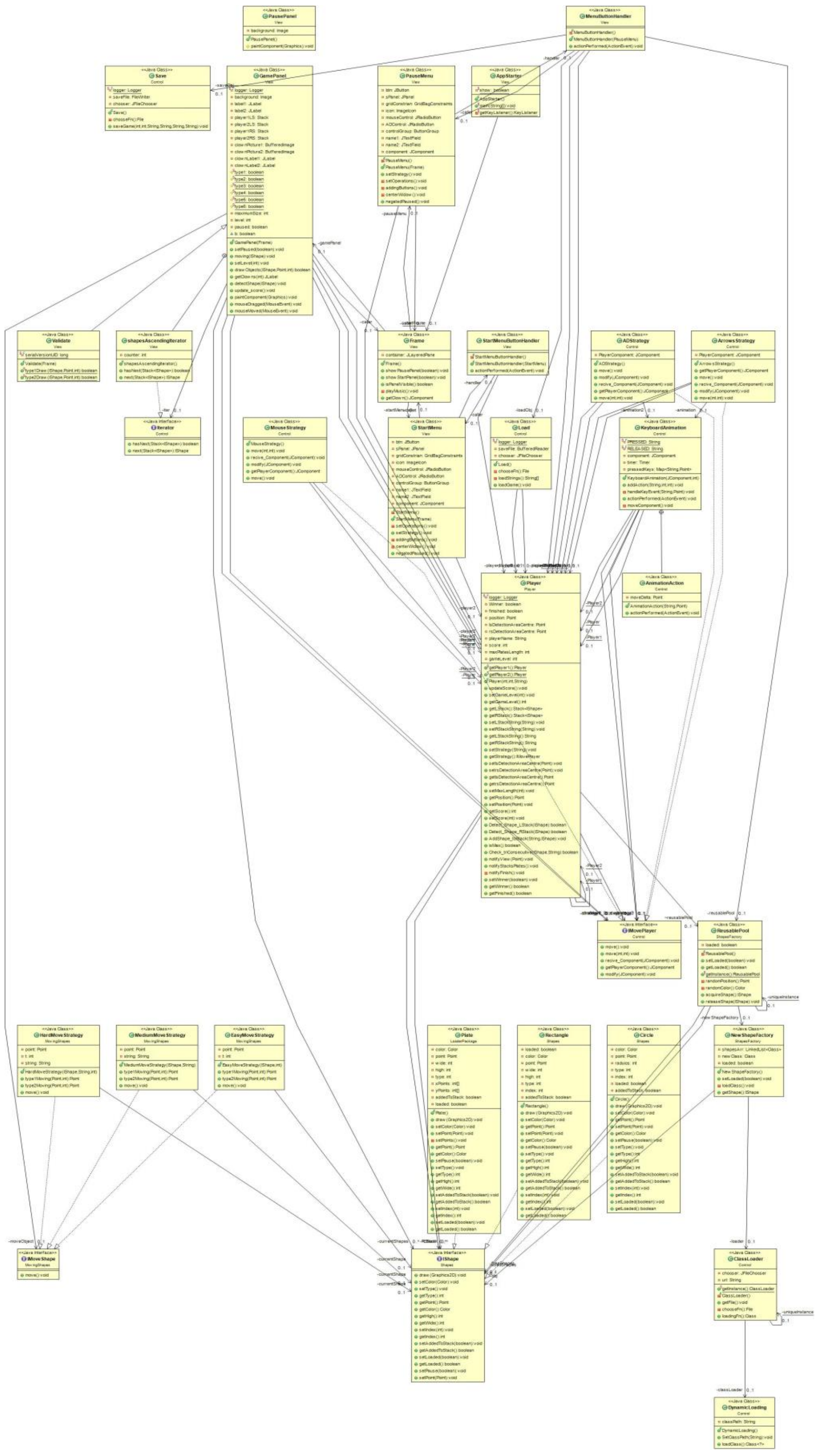
Circus of Plates - Game

Names:

1. Ahmed Mahmoud Rizk
2. Yahia Mohamed El Shahawy
3. Yousef Mohamed Ali Mohamed Zook

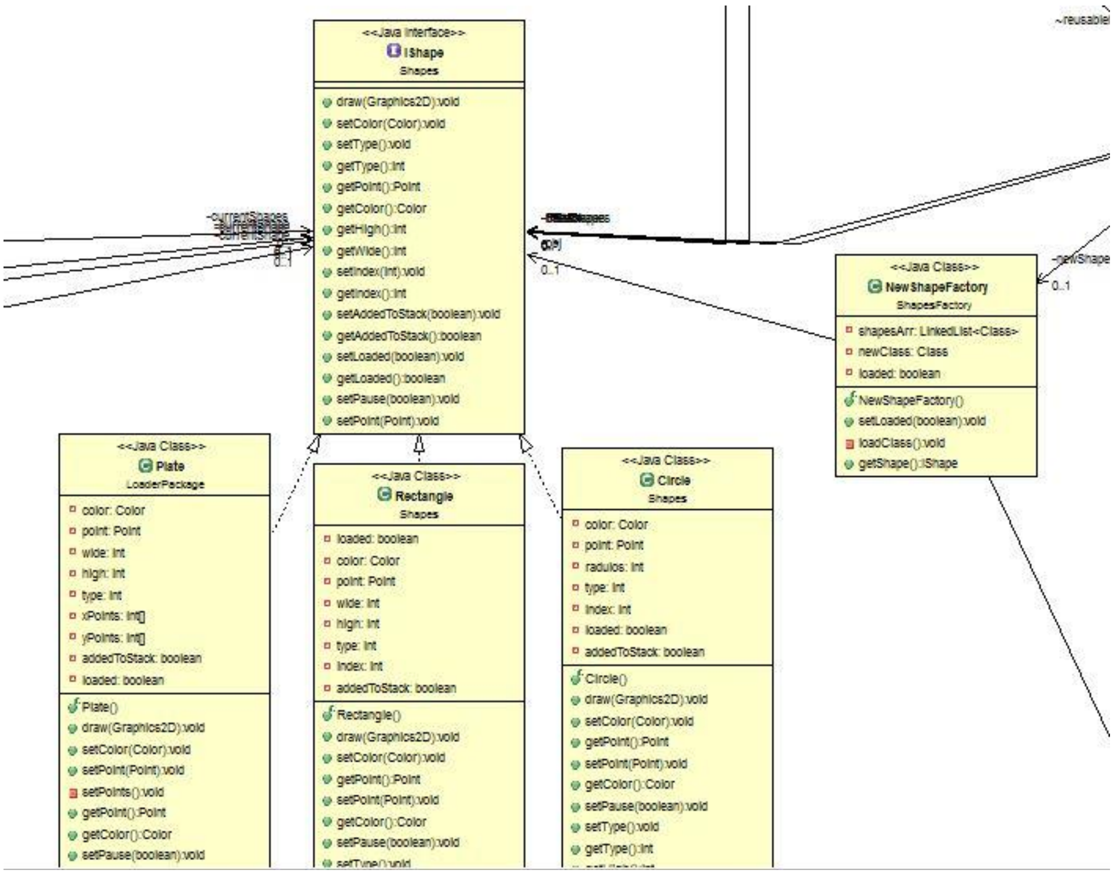
1. UML Class Diagram:



2. Design:

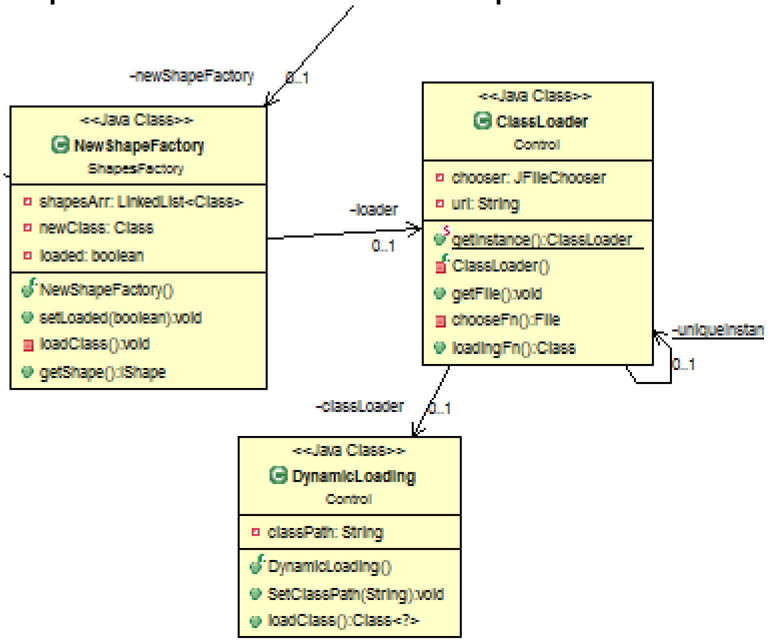
Factory DP:

we use this design Pattern to create the shapes in the game which have different sizes but they all have the same implementation. All the shapes implement the same interface with the same function but every class implement it in its way. the factory is one of the creational patterns which deals with different objects of shapes to create in ***a*** manner suitable to the situation**.**



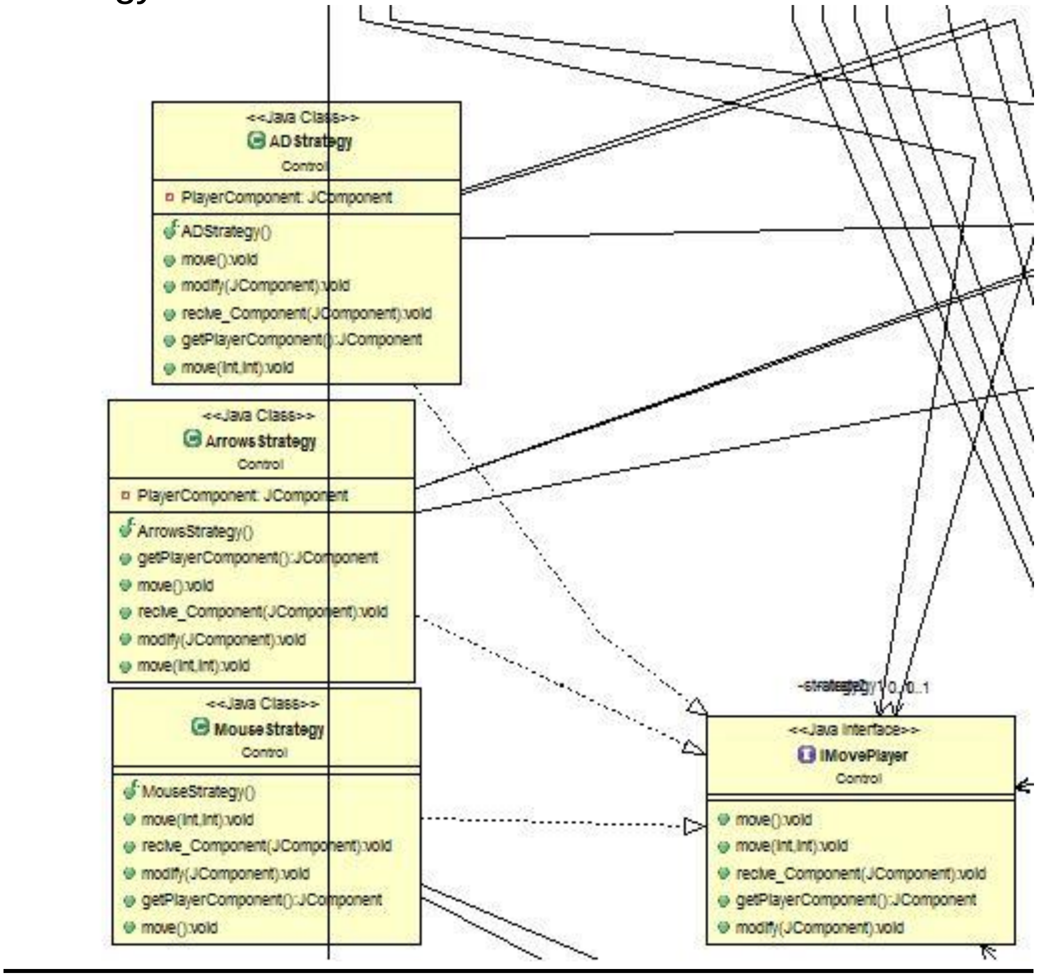
Dynamic linkage:

we use it to load any shape which have the same implementation of the shape.



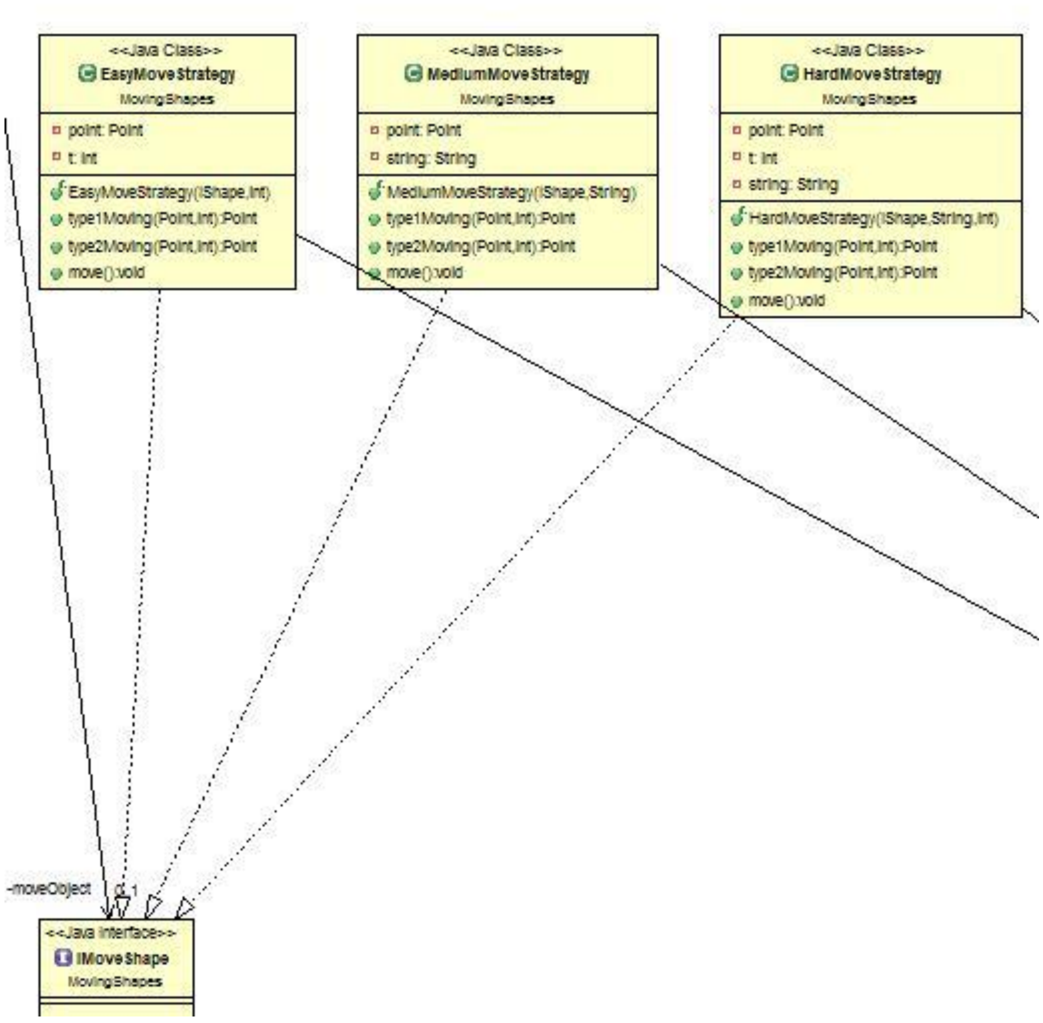
Strategy:

we use this design pattern to implement multi ways of control to the player for the clown. they implement the same interface. each strategy of control is independent of the rest of the project. it enables the user to play using key bindings, with mouse listener or with any Controlling strategy fit for him.



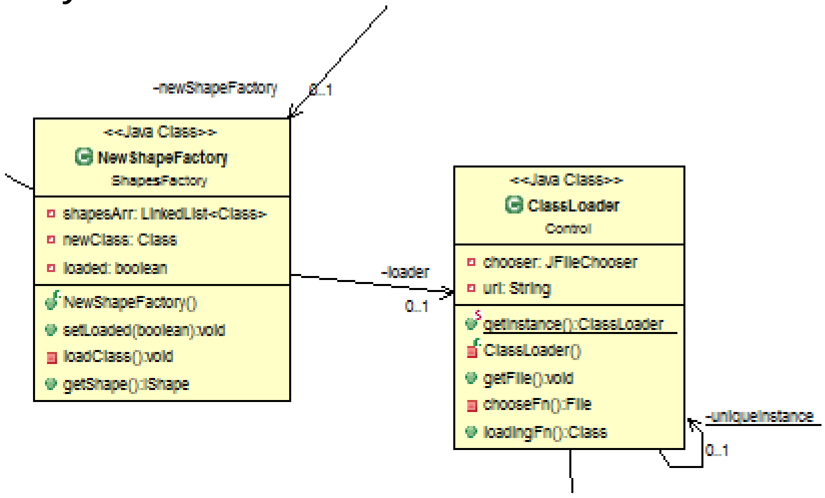
State:

we use this design pattern to set the Difficulty of the game. there are three levels in our game. we consider the three classes of the difficulty as states which change during the game. the three classes (Easy - Medium - Hard) all implement the same interface IMove. it represents various states and context object whose behavior varies as its state object changes.



singleton :

we use this design pattern in the reusable pool class because it's used one time in the whole project. creating the instance of this class and updating all the arrays and all data in the reused pool class. And we used it in the Class Loader class. Each player is a singleton in his own way.



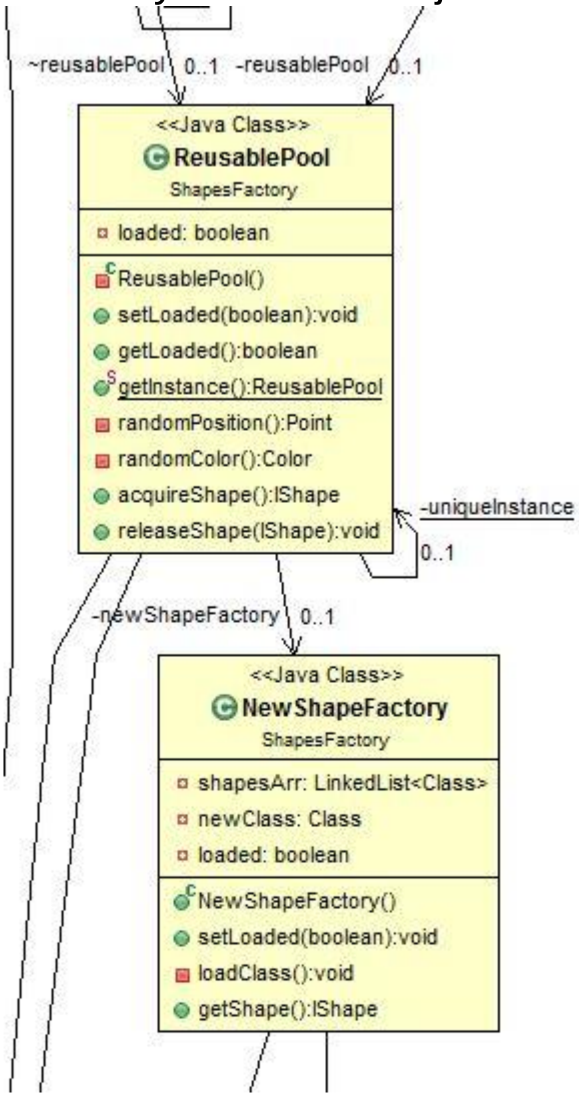
MVC :

The whole project depends on this design pattern. the player need to observe the shapes and detect their position to get them. and the frame detect the movement of the player and the shapes and detect the score of each play this design pattern is consists of two modules:
the first is the MVC of the <u>players</u> their score and positions
<u>model</u>: the player is the model and it have the notify function which update to the controller.
<u>controller</u>: the controller is the class controller which detect any change in the player object (score - position - attributes) and changes it through the Update function.
<u>view:</u> the frame of gui is the view in this project which each change in the player appear on it can implement the controller.
The second is the MVC of the shapes it only has position
<u>model:</u> the created shape can be any implemented shape which contains the main function of IShape interface and

this class have the notify function which detects any change in the model and send it to the controller to update
controller: the class controller which detects any changes in the shape object (position - attributes) and changes it through the Update function.
view: it also performs like the player. it shows any update in the shapes and updates it in the frame

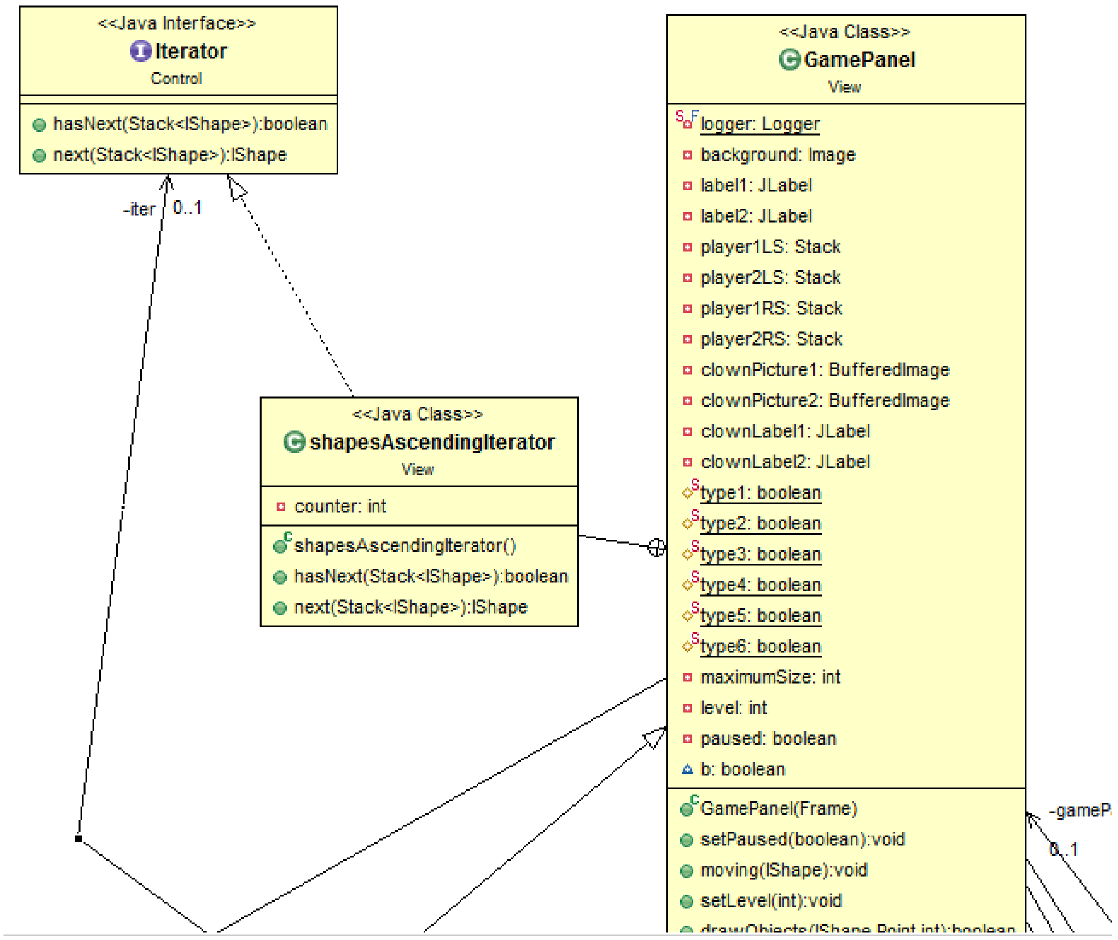
Object pool :

the main purpose from using this design pattern is to reduce the created shapes then lower the price of using it again. we implement it in the Reuse pool shapes. it uses the shapes which get out from the game and don't detected to any stack of each player. it can have two array one of the uses shapes and one of the free shapes which get out of the frame. not allowed to create new shape if the array of the free object is empty.



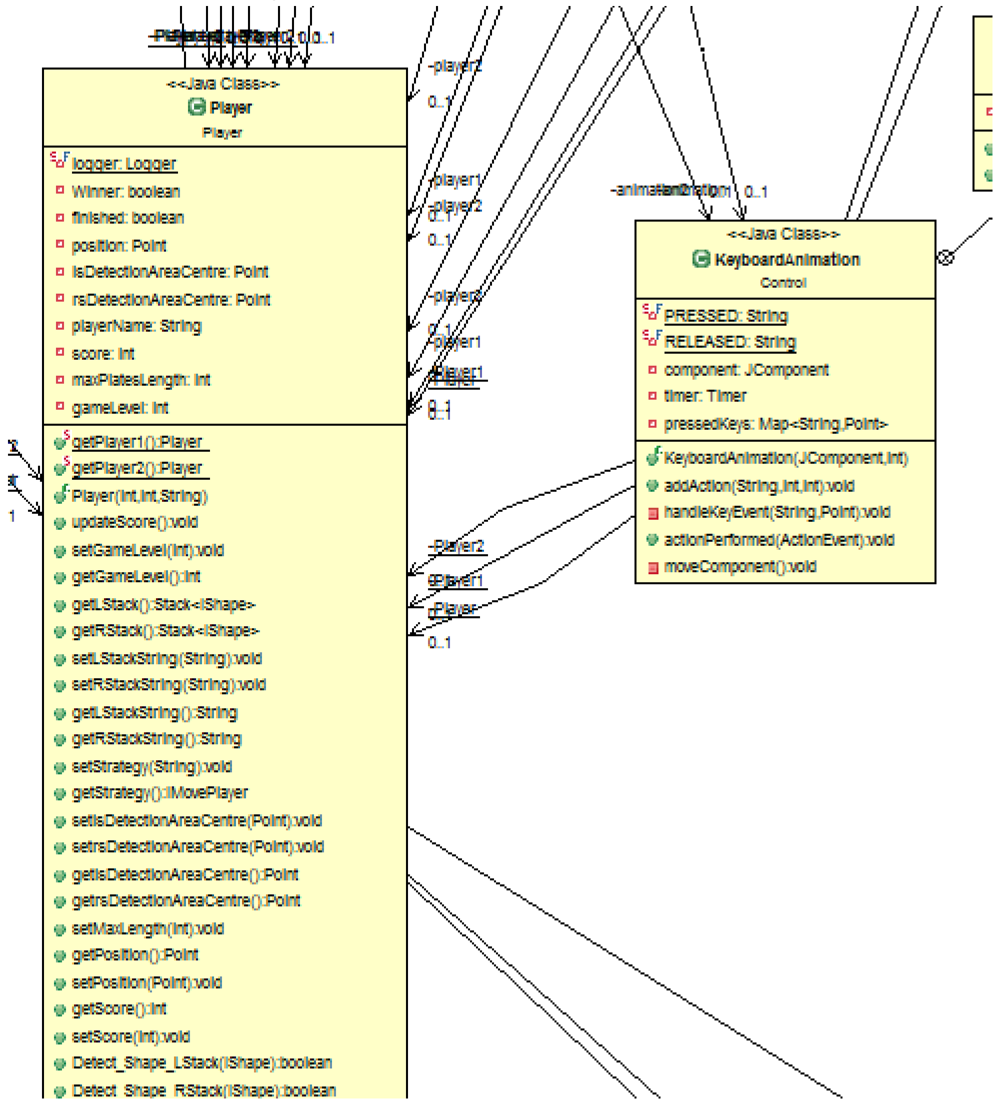
Iterator :

Is an interface to implement different iterating strategies, using It to iterate through object in a container object.



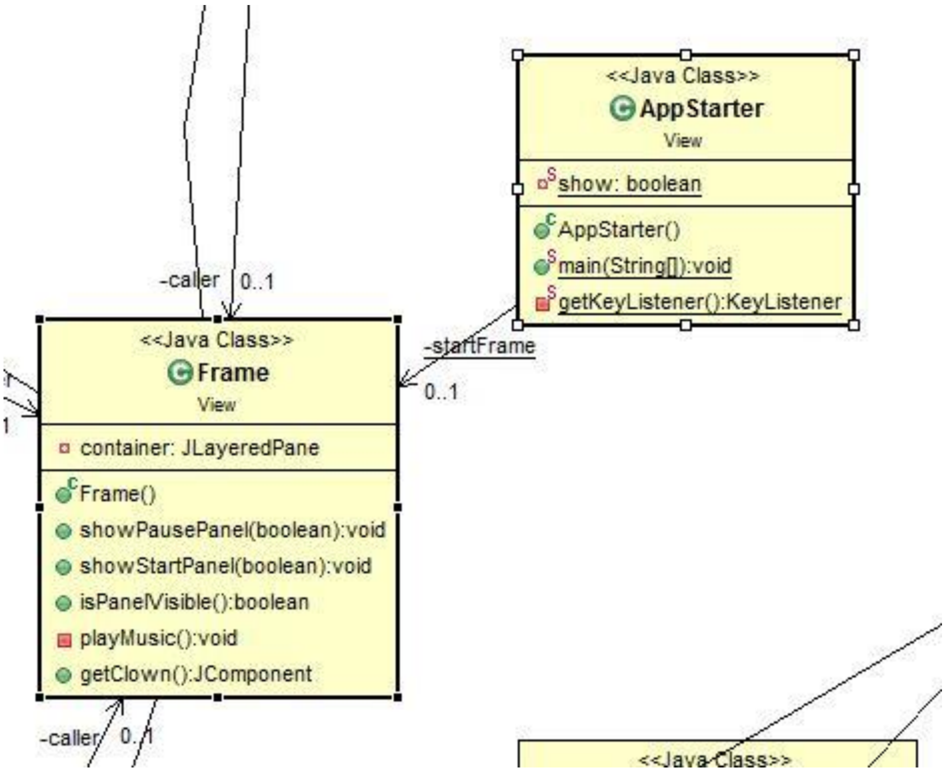
Observer :

Is an interface to implement different iterating strategies, using It to iterate through object in a container object.



Facade :

This design pattern appears between the AppStarter Class and Frame Class , as the first one use only one object of the second and Frame treats with all other classes.



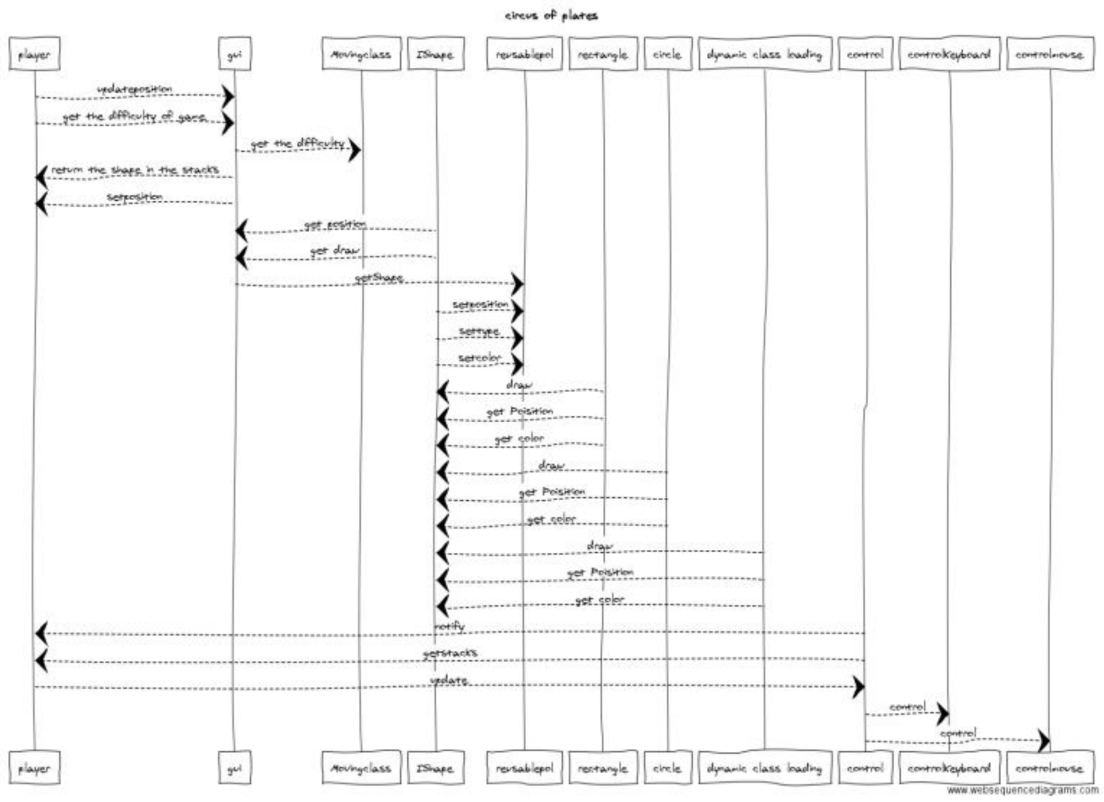
3. User Guides:

This is completely GUI game.at first, user can choose the difficulty of the game from three levels (Easy - Medium - Hard). The game consists of two players each one can move using controls of (keyboard or mouse) and each player has two stacks and can collect the shapes which are detected.

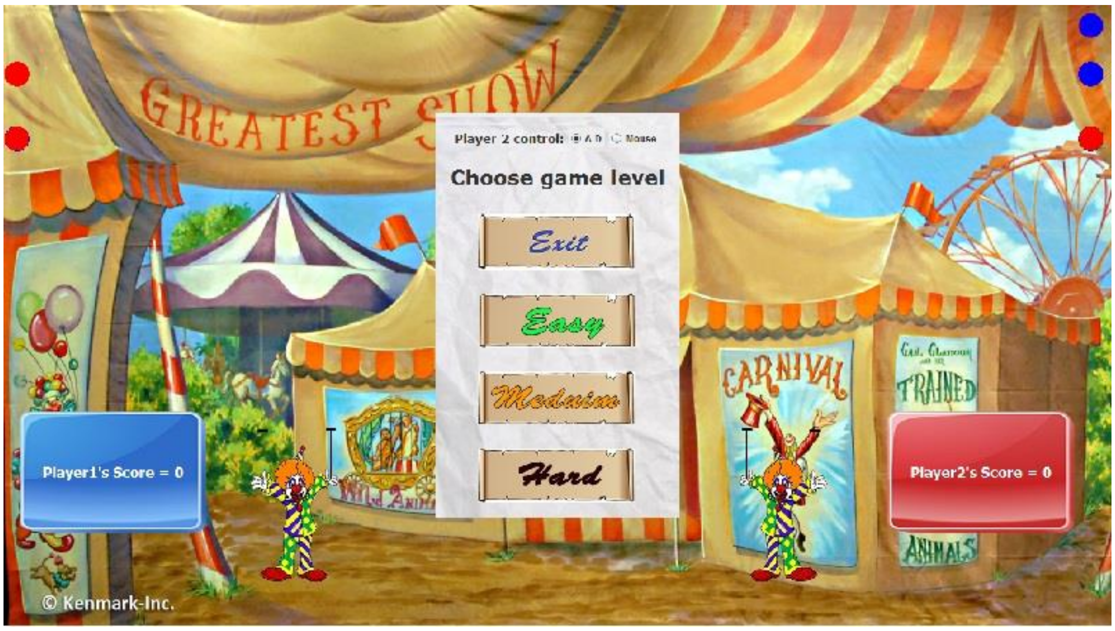
when the player collects three shapes from the same color they will disappear and increases the score of the player.

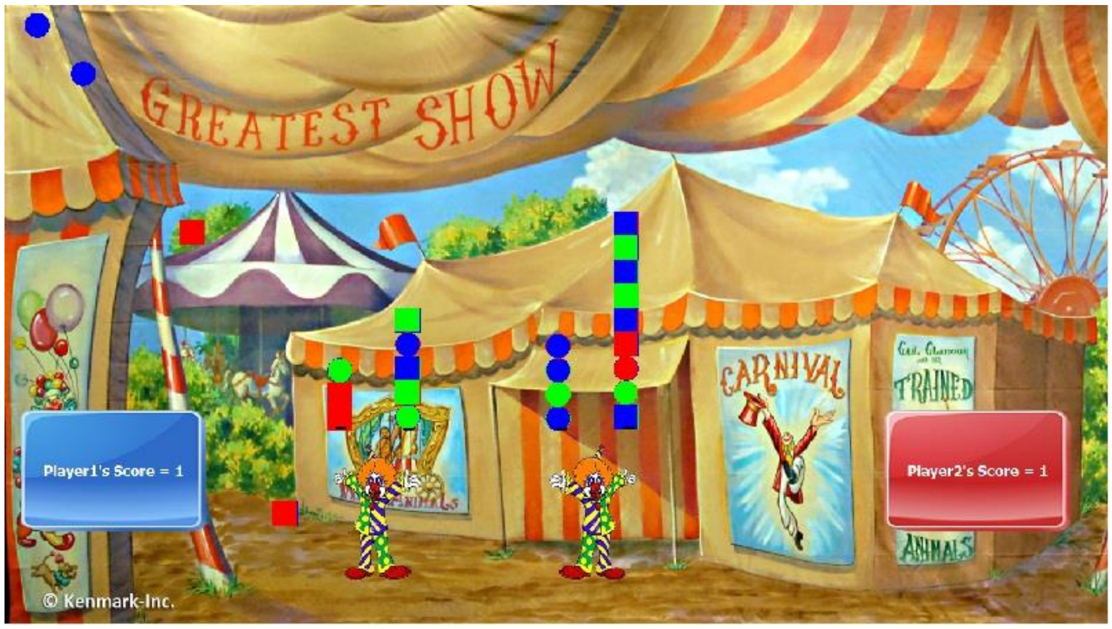
the game finishes when one of the player's shapes reaches the end of the frame of the game then the game ends and the winner is published.

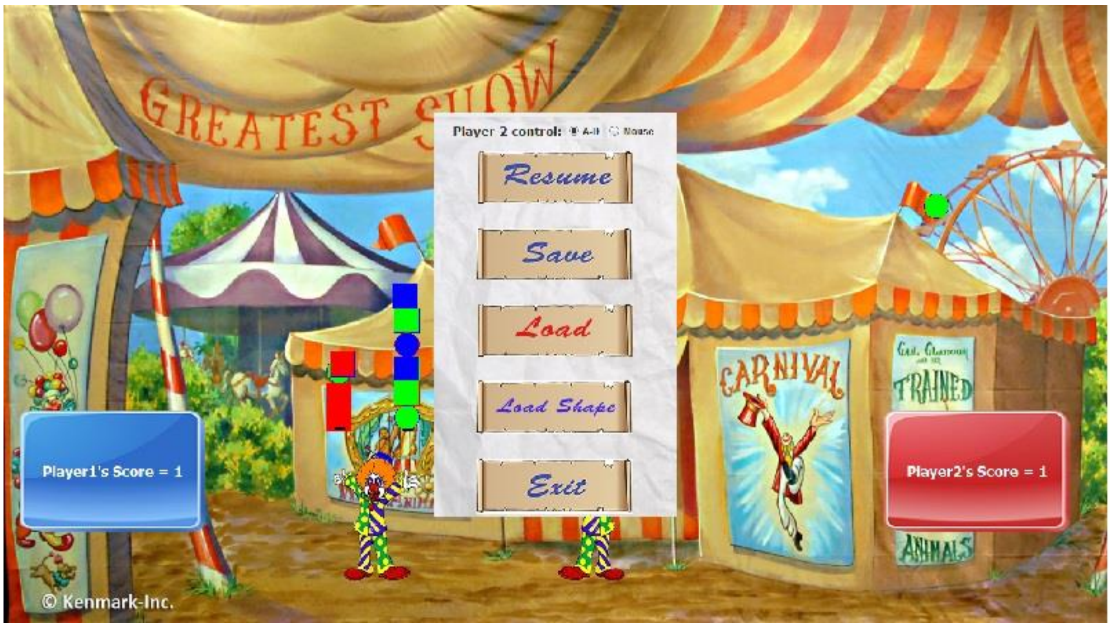
4. Sequence Diagram:



5. snape shot for GUI:


Player 2 control: [✓ A D] Mouse





6. Design Assumptions:

We assumed that the game finishes if a player reaches the top of the screen.