# Circus Of Plates

Ahmed Walid | Anas Harby | Mohamed Tolba | Salma Ahmed

## Design Description:

The project is divided into several packages containing several classes as stated below:

### 1. Model package:

responsible for storing the game data such as data related to players, shapes, timer and scoreboard.

- model:
  - Player: contains a player's name, score and a reference to its corresponding Character.
  - Timer.
- model.characters:
  - Character: An abstract class that contains the player's data related to its representation on view (x, y coordinates, shapes caught, etc), this class can be extended to support several kinds of characters (e.x: Clown).
- Model.save
  - ModelMemento: Acts as a data holder to the game's state, collects player's data, current timer, etc. this class is used to save and load data.
  - MementoOriginator: An interface for collecting data and setting it in a memento or loading data from a memento.
- model.shapes:
  - Shape Factory: A factory that is used to create shapes using a key string.
  - Shape: An abstract class that contains the shape's data, state, coordinates, color, etc.

### 2.Control package:

Responsible for controlling all of the game components where it includes the following sub controllers:

- MainController: has references to all other sub-controllers, acts as a middle-out shared resource for all these sub-controllers.
- ViewController: responsible for addition and removal from/to game view.
- InputController: responsible for handling keyboard events sent from view.
- ShapesController: responsible for creating shapes, binding them to their UI representations in game view, handling their various states, controlling pausing/resuming their transitions.

- LevelsController: responsible for setting the difficulty level user selects from main menu before starting a new game.
- PlayersController: responsible for controlling players' and characters' data manipulation, binds these characters to their UI representations in view.
- GameUtilController: responsible for controlling various game utilities (timer, score, etc).
- FileController: responsible for controlling the reading/writing process for game states, uses a predefined reader/writer for the process (Protocol buffer Reader/Writer in this case).

### 3.Behaviour package:

Which is responsible for the behaviour of different components within the game.For example: key bindings,shape states,shape transitions and different difficulty levels.

### 4.View.gui package:

Consists of the sub packages responsible for the main menu view, difficulty levels dialog view, main game view, pause menu view ,options dialog view and endgame view.

### 5.Plugins package:

Contains the plugin loader which is responsible for dynamic class loading of either supported shapes or supposedly supported characters.

### 6.Assets package:

Contains all the image resources used throughout the game and which represent the imageviews of the players,shapes,background.

### 7.Util package:

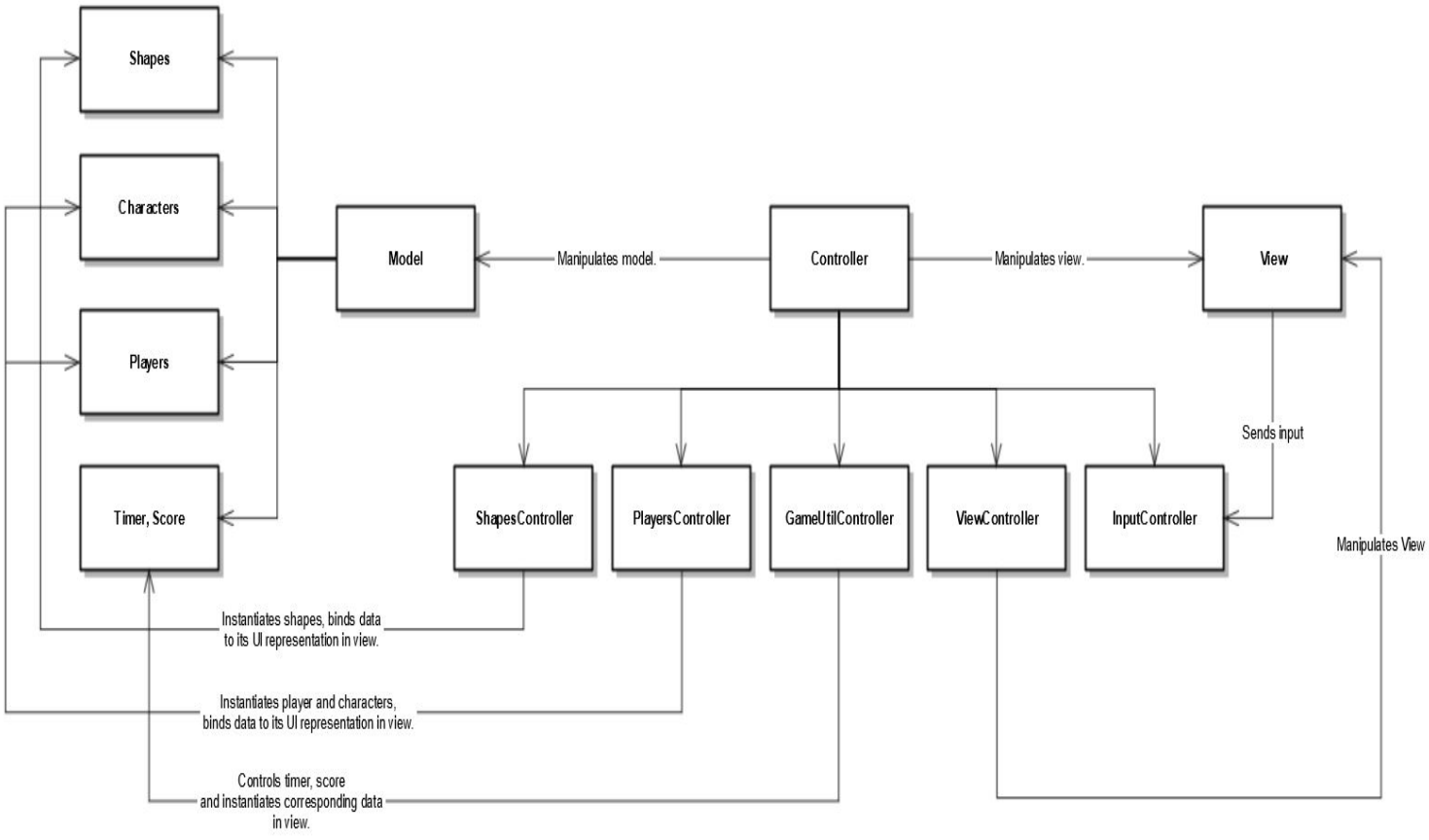
Contains the used utilities which are the timer, score, shelf, point, pausable thread, menuitem and menu box.

# Design Patterns Usage:

## 1.MVC:

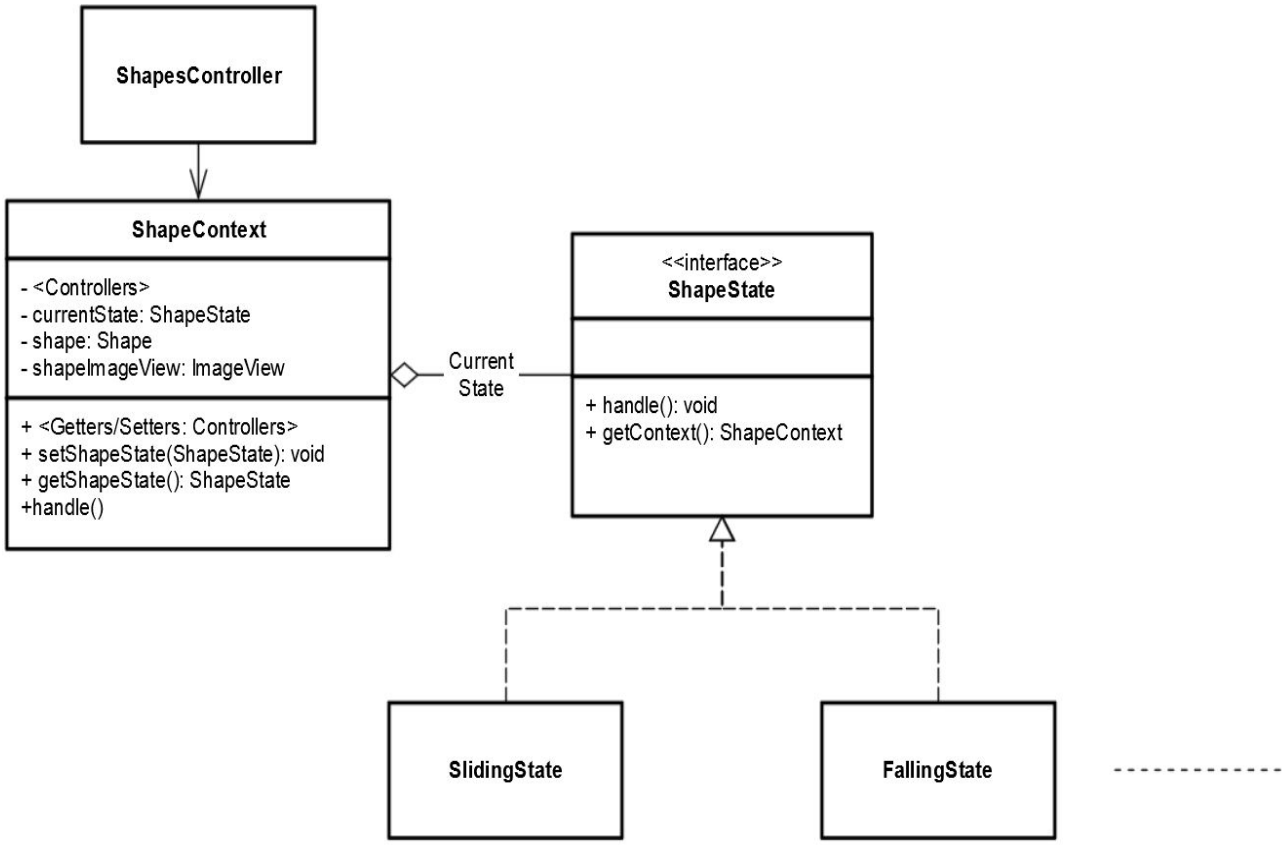
The Model View Controller design pattern is used to separate the logic of the game from the view using a controller that acts as a connection between the model which holds the game data and the passive view responsible for the game UI.

- The Model holds all the data of the game including data about the shapes, characters, players, game timer and score.
- The Controller which consists of several sub controllers including the input controller responsible for keyboard inputs acting on the players, the shapes controller, the game util controller responsible for the score and the timer and the view controller which manipulates the view.



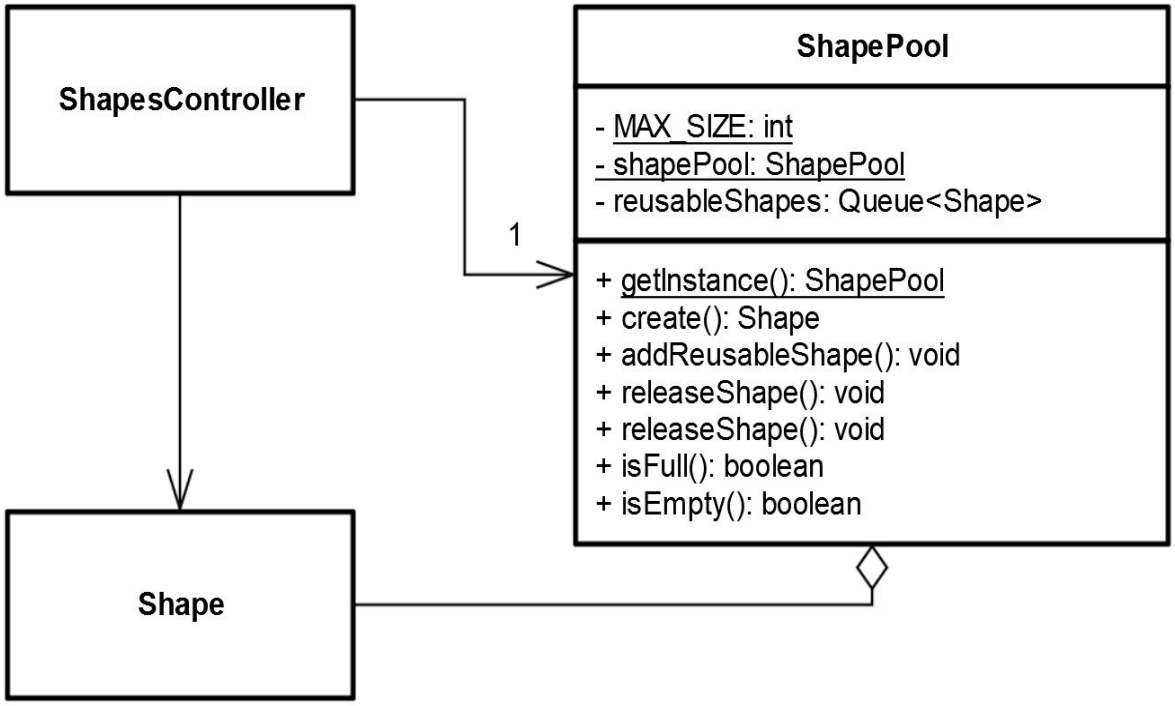
2.State:

State design pattern was used to express several shape states throughout the game where initially the state is sliding state when the shape is sliding on the shelf then the state handle moves the shape to the following state which is the falling state where the handle is responsible for transitions performed on the shape while falling, then the shape is moved to either the fetched state or added to the shape pool state, each state is handled by a separate handler that is part of a chain of handlers controlled by a ShapeContext.



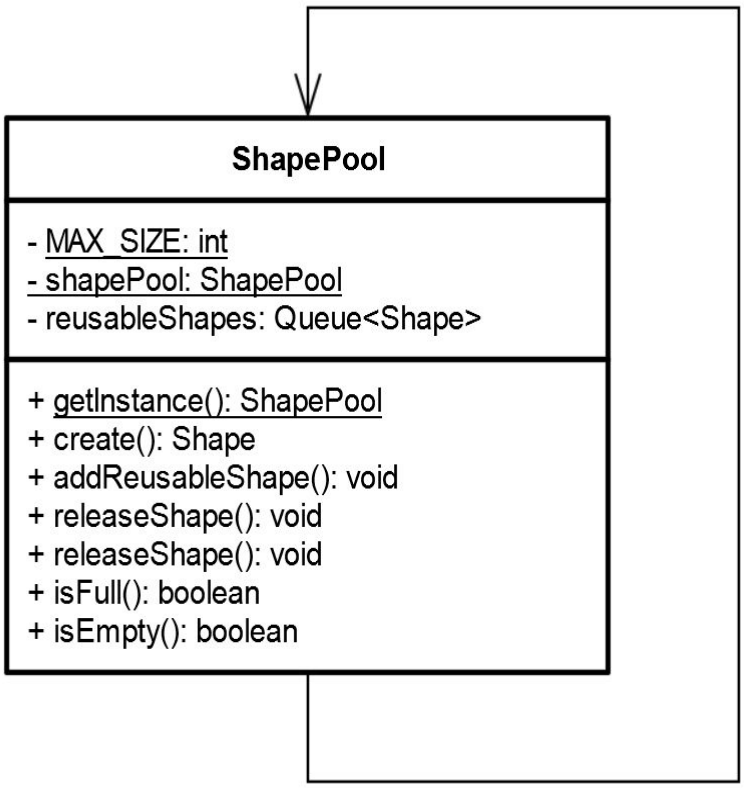
3.Object Pool:

Object pool design pattern is used for the creation of different random shapes if the pool does not have any shapes otherwise it reuses the shapes from the object pool where maximum pool size is 100 shapes. Whenever a shape falls on the ground without being fetched it is added back to the pool in order to be reused again.



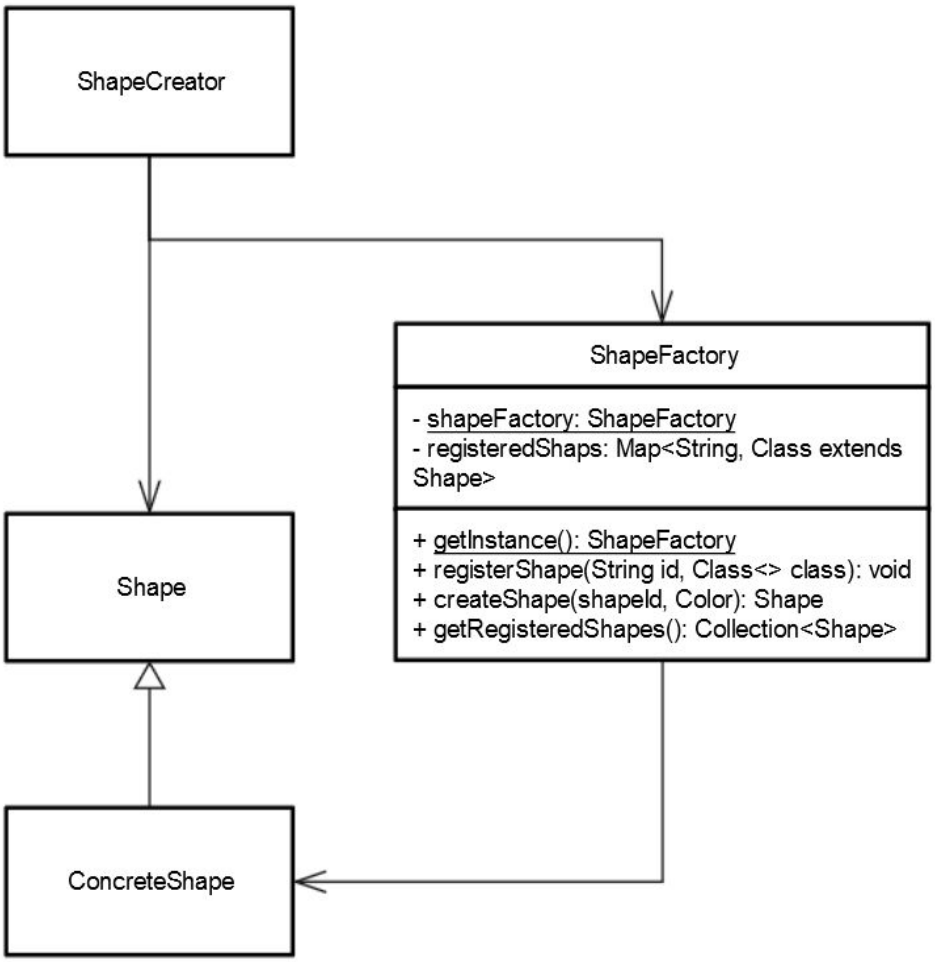
4.Singleton:

The Shape pool also acts as a singleton where the singleton restricts that only one object of the shape pool is being instantiated during the whole game.The shape pool was required to be a singleton because it is the class responsible for the creation of shapes during the game and there is no need for having several instance from it.



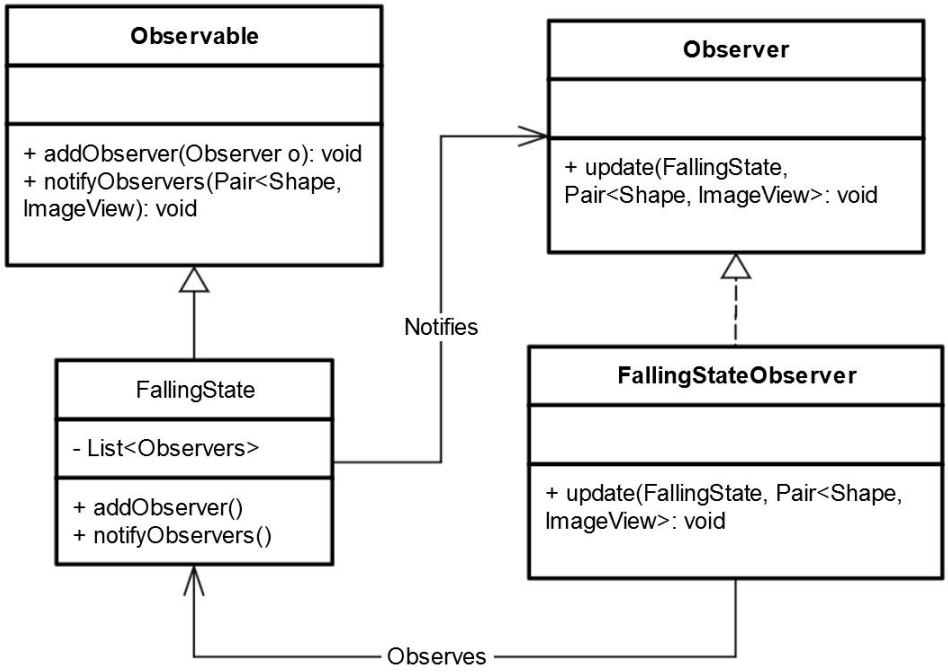
5.Factory:

The shape factory is used for shape registration where it registers different shapes and uses the shape creator class to create shapes with random features where the factory makes sure that the creation logic is unexposed.



## 6.Observer:

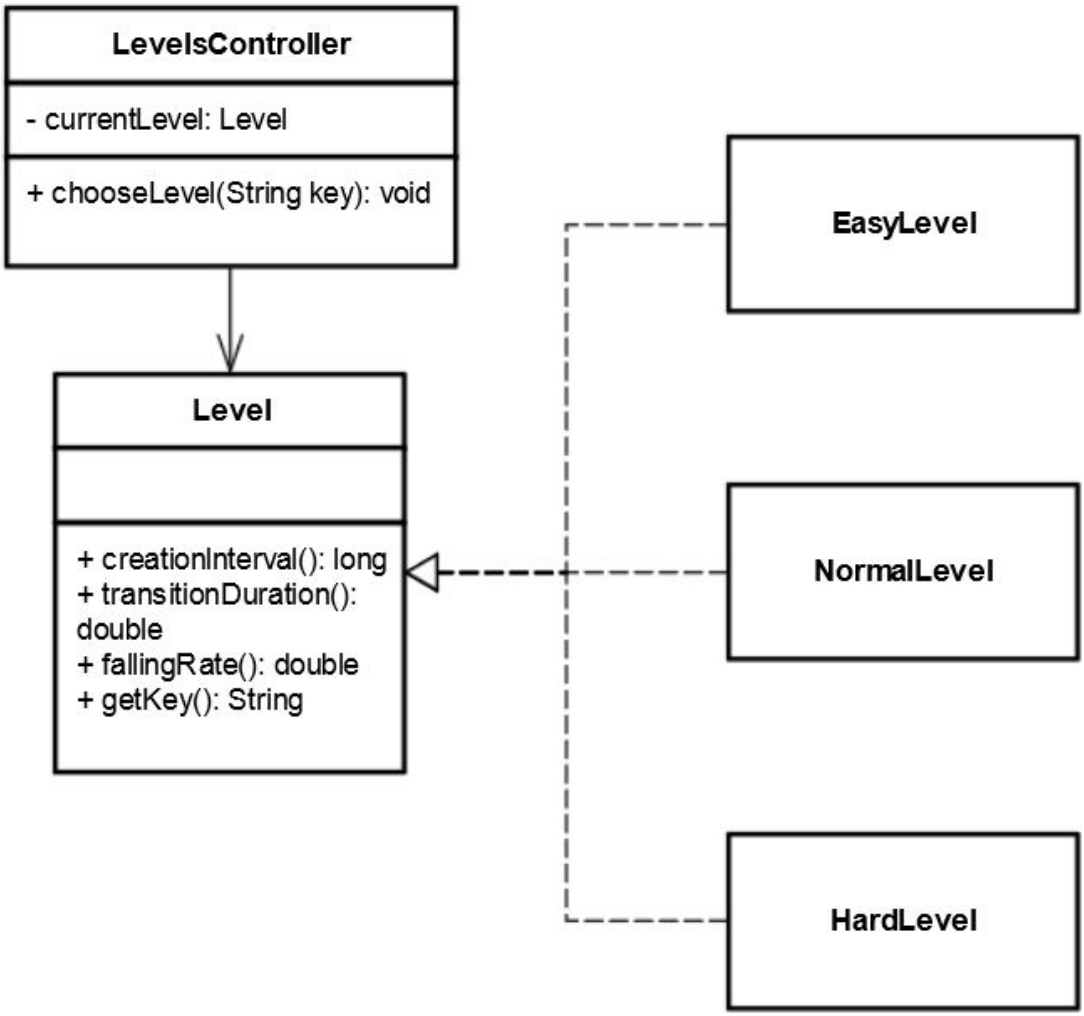
When a shape is in the falling state, whenever it changes its position it notifies an observer who checks whether the shape is in a state of which a player can catch it or not and acts accordingly as it updates the shape's state.



7. Strategy:

Strategy is used at the start of the game when the user chooses a certain difficulty, so the proper set of game configurations are loaded.

Also, it's used in the backend parser where a certain interface is defined to define an algorithm for reading and writing so any parser implementing this interface would parse the data.

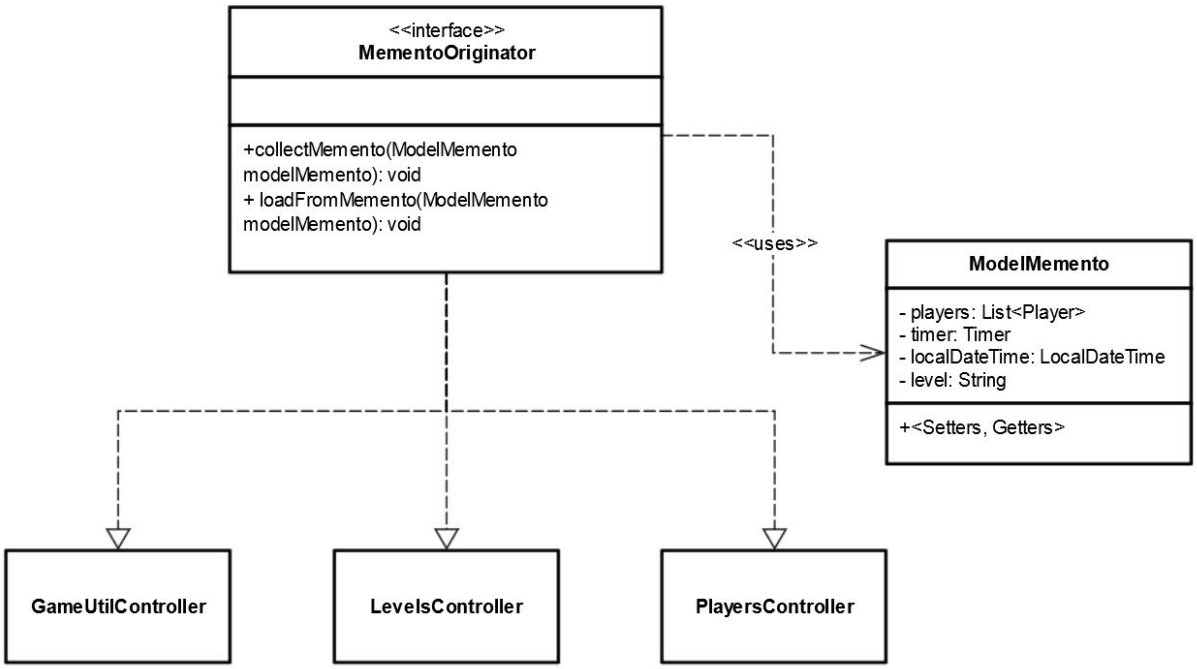



8.Dynamic Linkage:

The dynamic linkage usage appears in the dynamic class loading of different supported shapes and characters where you are allowed to load new shapes to the game.

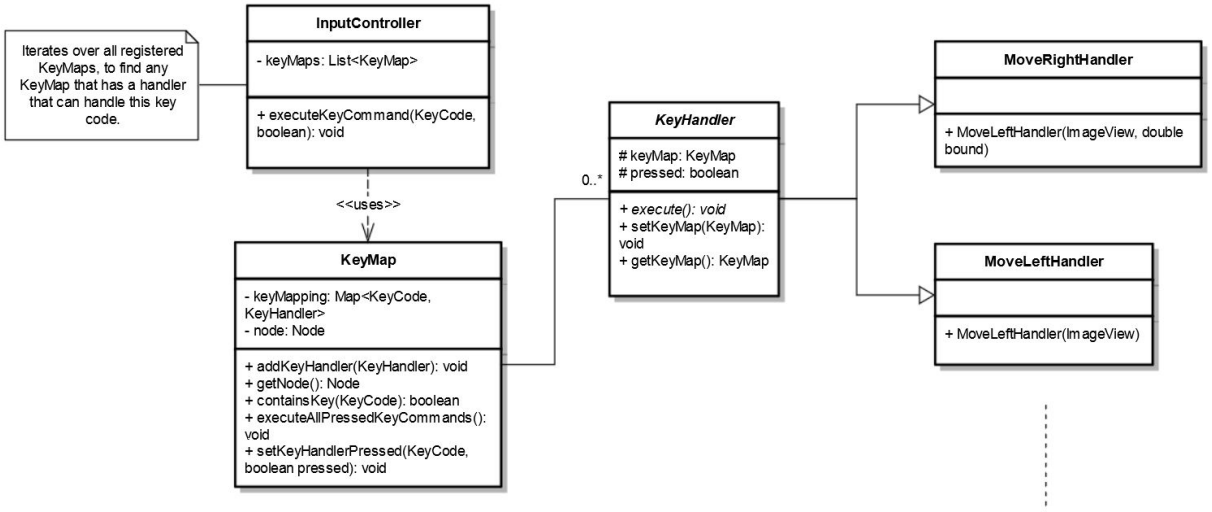
9.Snapshot (Memento):

Memento was used in the saving/loading technique for the game, where a class is used to collect the necessary information for the game when the user clicks on save, and, when the user loads the game, the class is used to read the data from the file and sends the info to the controllers to set the loaded info and load the game.



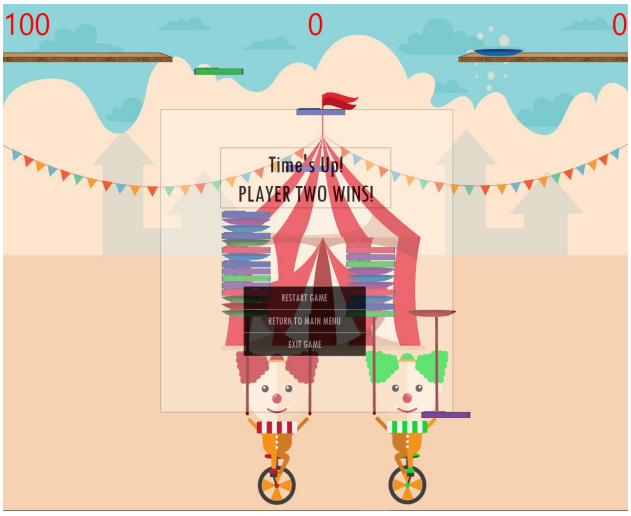
## 10.Chain of Responsibility:

A chain of responsibility is triggered whenever the input controller catches a pressed key, it iterates over every registered key map checking whether it has a handler that can handle this key, and calls it to execute this key handler if it's true.



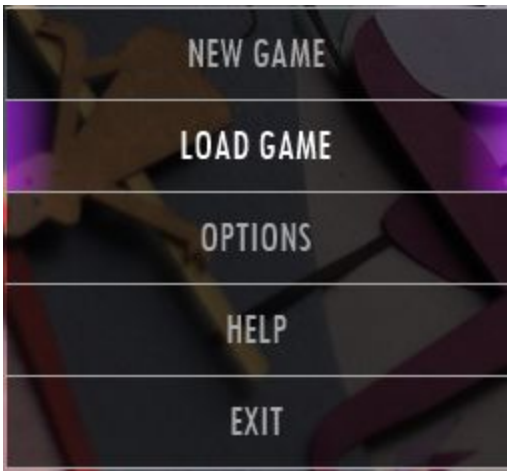
GUI Snapshots:






User Guide:

- The main controllers of the game are found in main menu where you can find five interactive buttons which control starting new game, loading new game, launching the capability of adding new shapes and characters at run time through the options button, the help button which supposedly shows text document as the README file which gives the user a brief overview about the game and all its features and finally ending the game using the exit button.




  - New Game:
    - New Game button allows the user to choose the difficulty that suits him most where the supported difficulties are Easy,Normal and Hard. Noting that they differ only in the speed of generation of plates as well as the speed of the players.

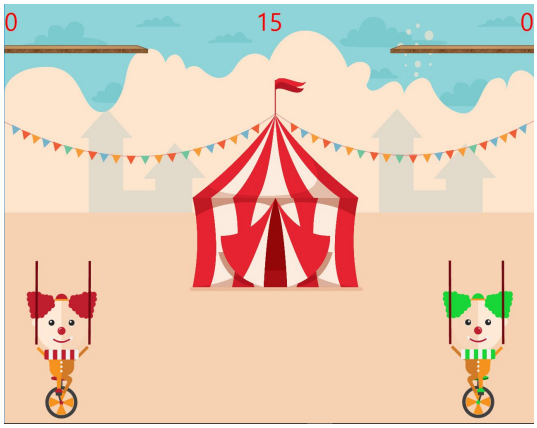

  - Load Game:
    - Load Game button allows the user to load a game so he can continue what was going on through the game by loading the game from protobuff file as the protocol buffers are the backend writer/reader we used.
  - Options:
    - Options button allows the user to add character or a shape at run-time so he can use it in the game. ***Note*** Adding characters dynamically is supported but it is still not integrated with the game at that point of development so the user can choose his favorite character ***while*** the shapes' usage is

automated during the random generation of shapes.

- Game :
  - Game View:

The game view is simple as that consists of some main parts:

- Score:




The score of the two players is shown on the 2 sides of the window as shown in figure the left zero is the score holder for the left player (Red clown, as shown) while the right zero is the score holder for the right player (Green clown, as shown).

- Timer:


The timer of the game is allocated at the middle in the upper part of the game view and it starts from 60 seconds and decreases till it reaches zero and then it will end the game showing the final result of the game deciding who's the winner depending on which player got the highest score ***Note*** The final result can be a draw if both players got the same score.

- Sliders:

  The sliders on the two sides of the game view are used in shapes' generation, as generated shapes first appear on these sliders moving with a constant speed horizontally (Depending on the selected difficulty level by the user).

- Characters:

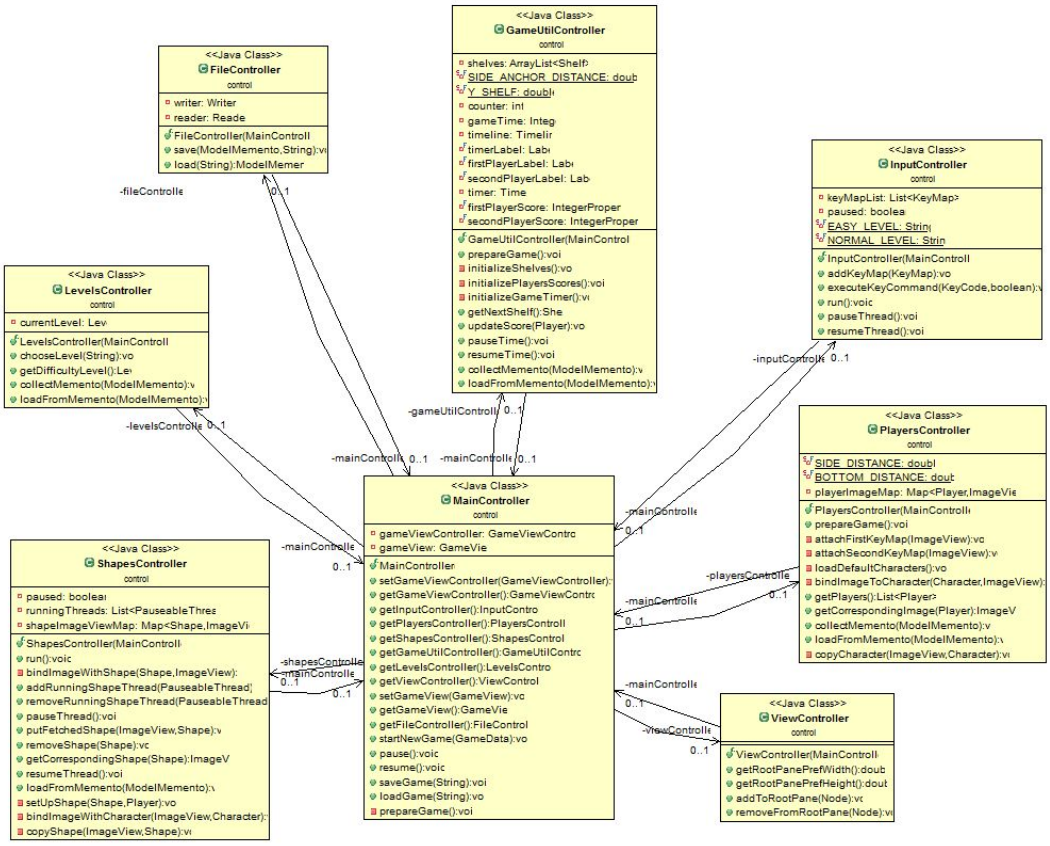
  Two characters are considered to be the main components of the game view (Two clowns, as shown in figure) as they are affected by the actions of the users and they are responsible for collecting the shapes on the sticks they are holding in their hands.

- Game controls:
  - Players:

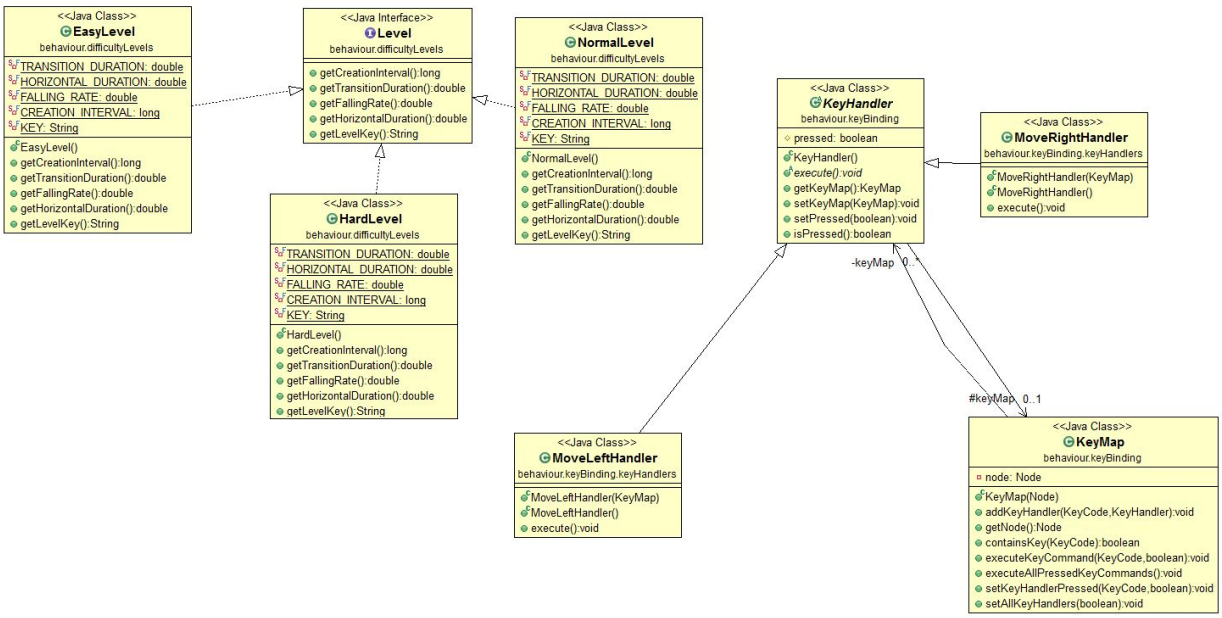
    Two players representing the two characters (Two clowns as shown in figure). Each player controls a character and can move it horizontally (One of them using the left and right arrows' buttons and the other using the A and D buttons in the keyboard).

## UML Class Diagrams:

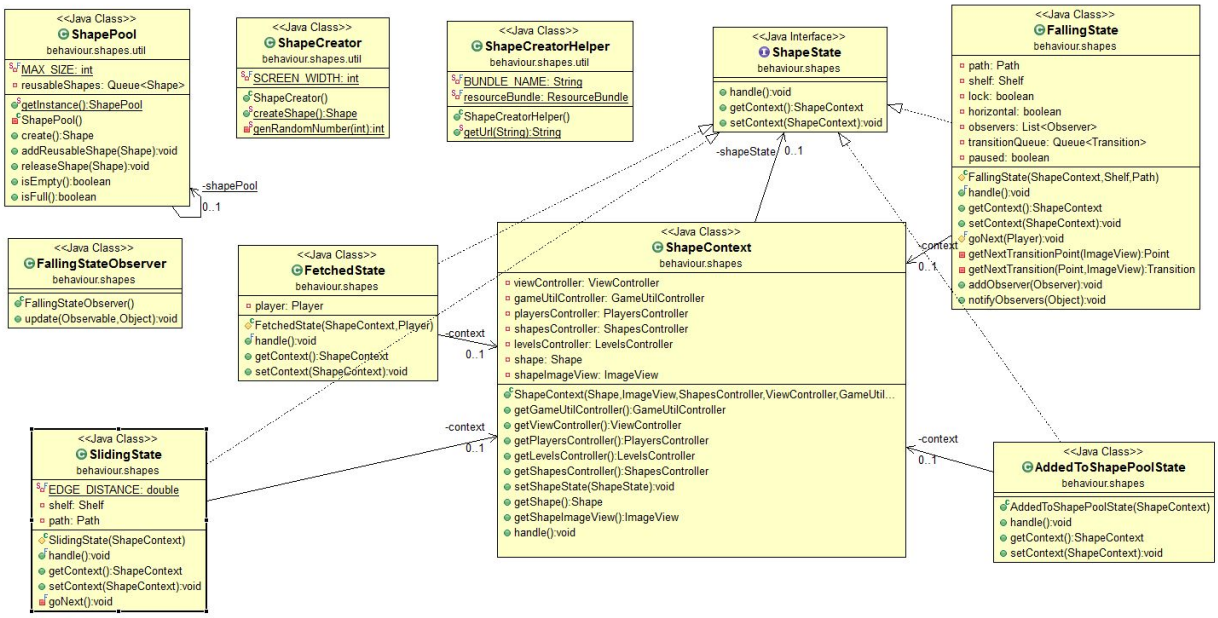
- *Controller class diagram:*



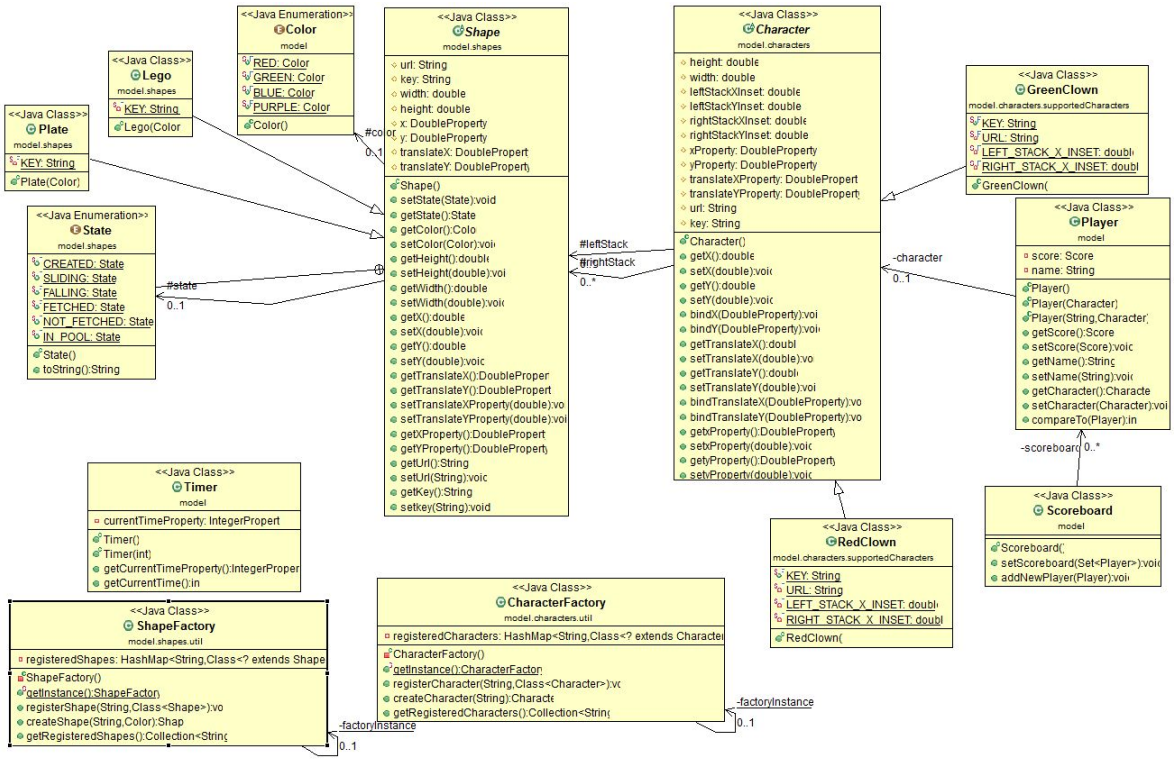
- *Difficulty levels and key handlers class diagram:*



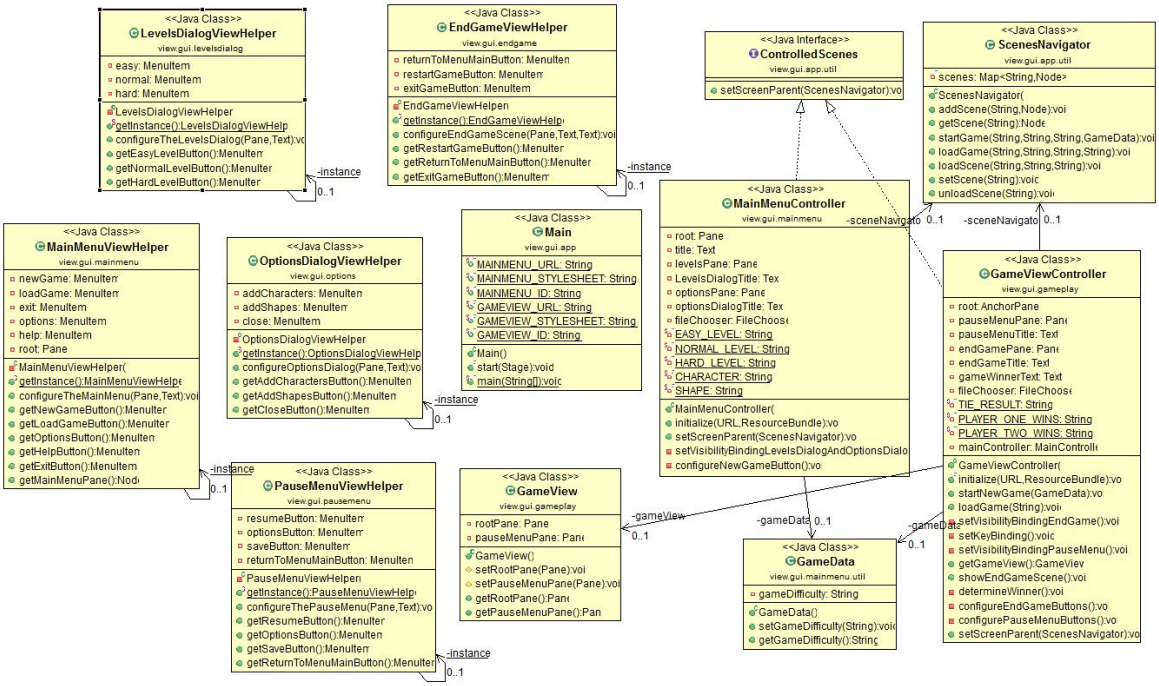
- *Shapes' states and controller:*



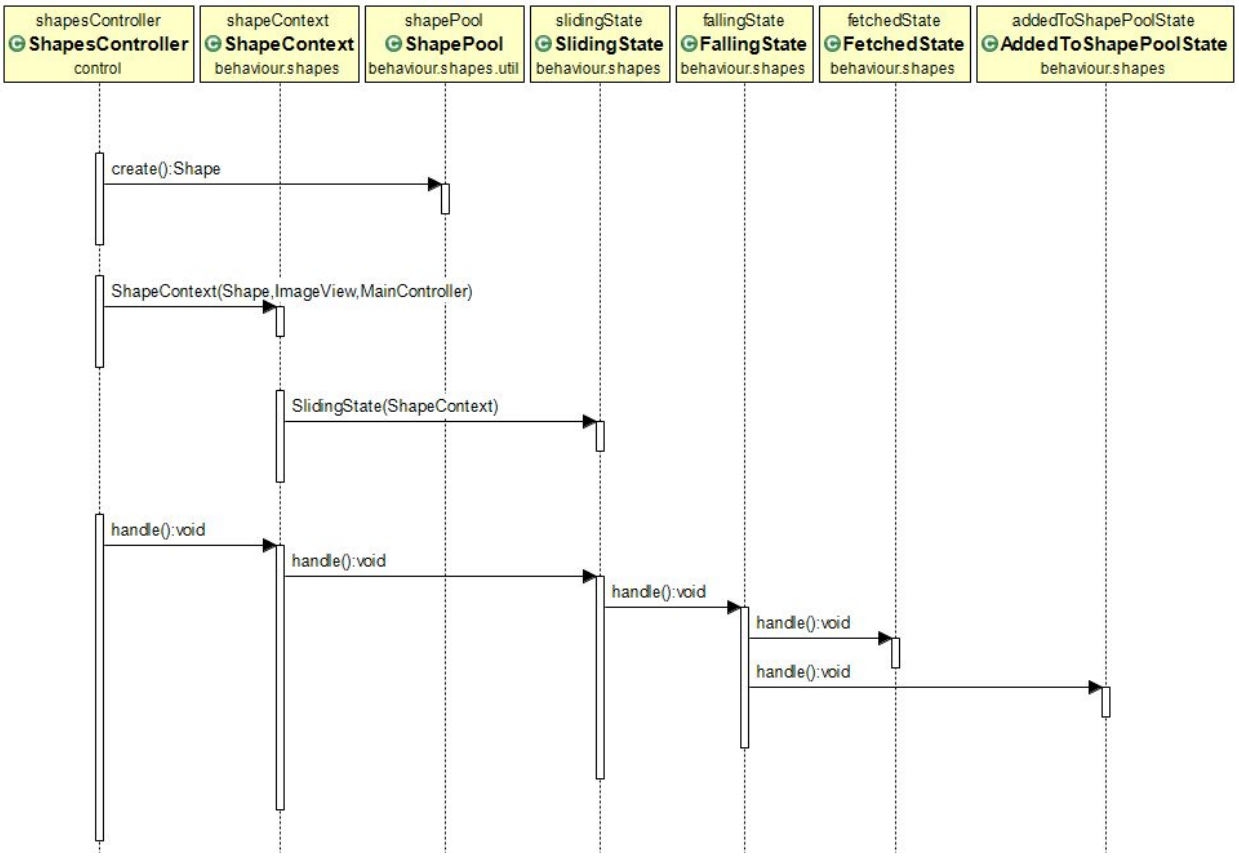
- *Shape and Character class in model:*



- *View class diagram:*



## Sequence Diagram:




## Design Decisions:

1- Characters can be chosen by the user and dynamically loaded before the start of the game, but the current characters are already preloaded in the game.

2- A timer is used and set for one minute and the player scoring more points during the game duration wins the game.

Credits:

UI artwork is motivated by a previous work done by Almas Baimagambetov on his github repository https://github.com/AlmasB/FXTutorials

Main menu background is adapted from project Paper Childhood on behance https://www.behance.net/gallery/18015159/Paper-Childhood

Game view background is adapted from vecteezy.com https://www.vecteezy.com/vector-art/83065-playing-retro-circus-vector