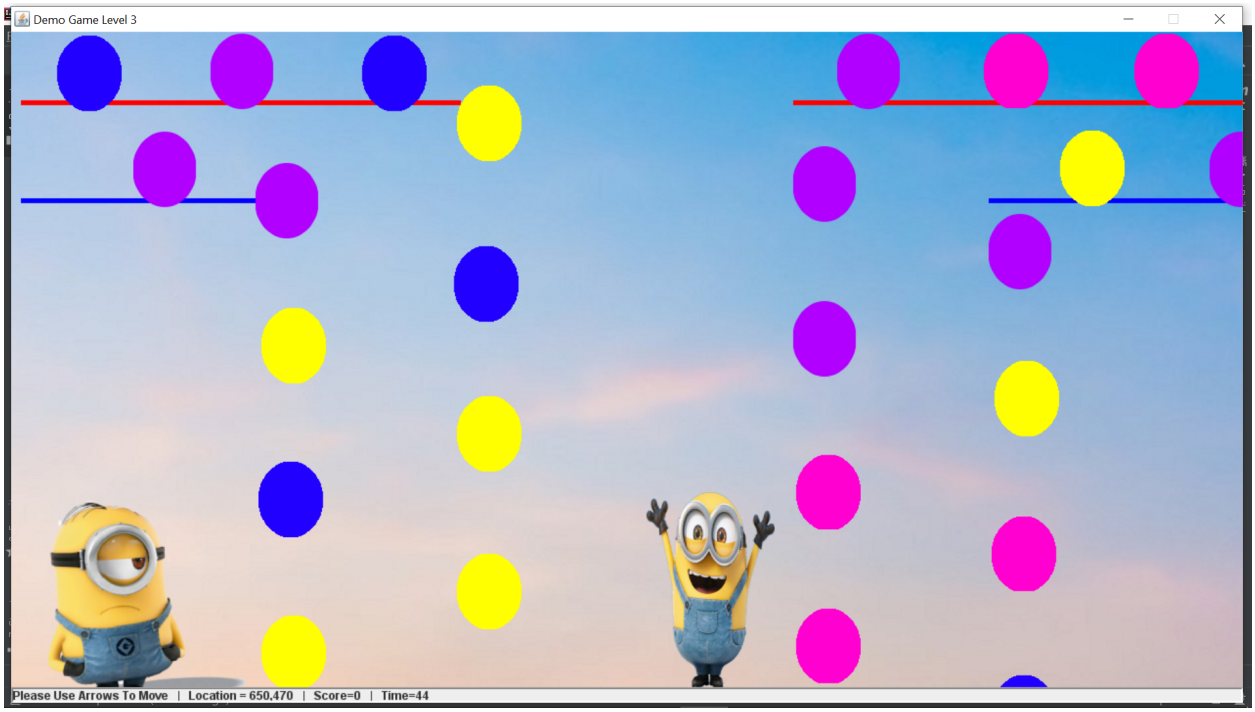
Programming2

Circus of Plates

project report



- Khadija assem (27)
- Abdelaziz Elsayed (40)
- Essam Mohamed (41)
- Norhan magdi (69)

Content

- Overview.
- Design description.
- Class diagram.
- Sequence diagram.
- Used design pattern.
- User guide & Snapshots of GUI.

Overview:

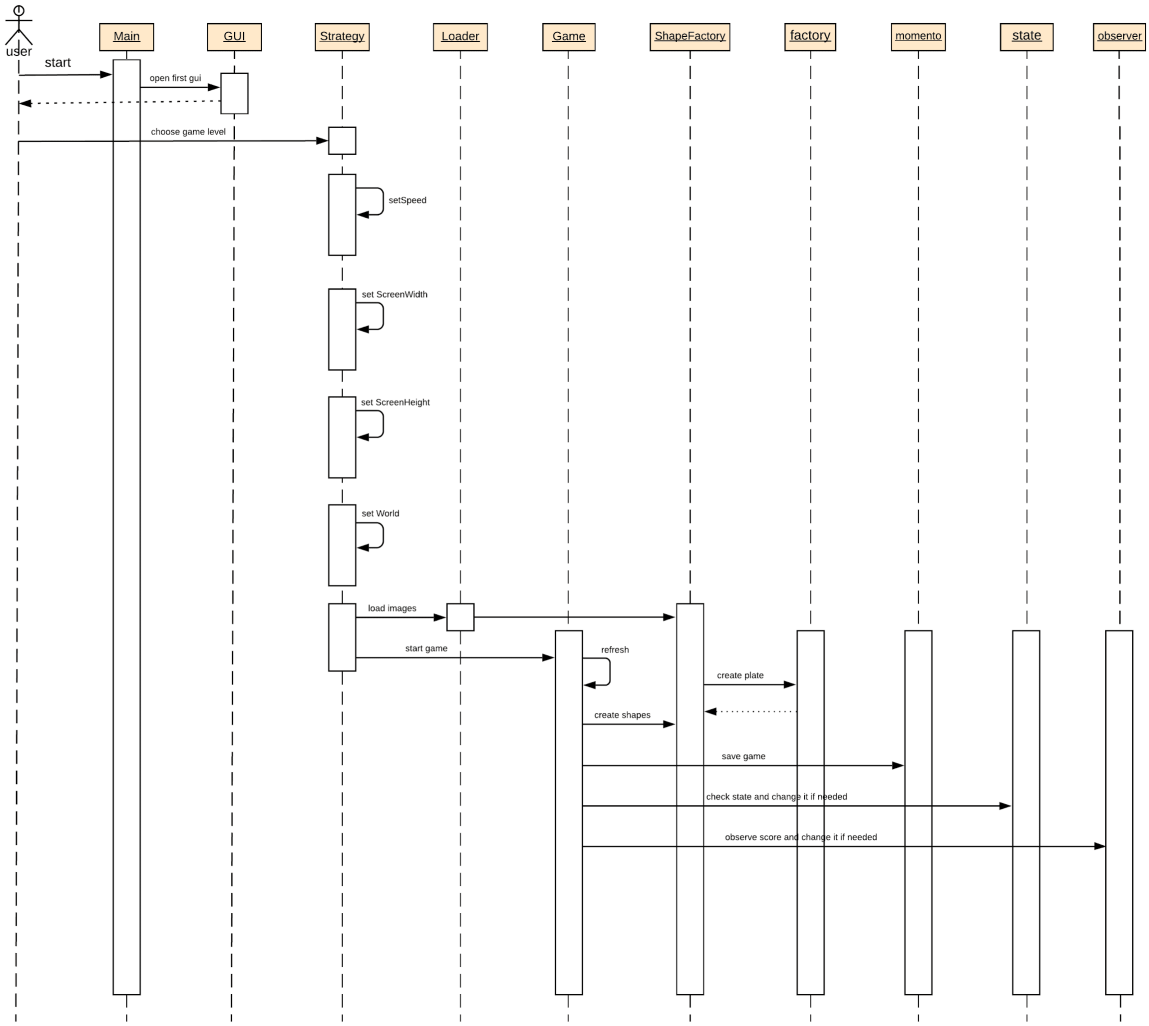
A single player-game in which a minion carries two stacks of balls or bananas which fall and he tries to catch them, if he manages to collect three consecutive shapes of the same color, then they are vanished and his score increases.

The game has Three levels which varies the speed of the falling shapes, the score you get and the time given to the User.

Design description and decisions:

- ➔ The Game design was mainly divided into 3 main components through the usage of the MVC design pattern : Model, View, and Controller.
- ➔ First ,the user choose a level from three which varies the speed of the falling shapes using strategy design pattern .
- ➔ The main class is **Game** class which implements world, it is passed to the game engine after the user chooses a level.
- ➔ The Shapes are dynamically loaded through dynamic linkage. Any class that extends the GameObject class will be enlisted as a shape to be used eg:balls,clown and shelves.
- ➔ Factory create a shape randomly and returns it to flyweight which manage creating them
- ➔ Flyweight starts the game with a limit of falling shapes and when the shape become in floor state flyweight use it again in order to save memory
- ➔ Playing music during game is applied
- ➔ In each refresh:
    - Memento design pattern is used to record state of all objects
    - State design pattern is used to switch between different states of objects as Shelf state,falling state or floor state
    - Observer is used to update time and score if user succeeded to collect three consecutive shapes of the same color
- ➔ In order to catch shapes , user wants to get the shapes into minion's hands , but as the minion catches the shapes, the contrable shapes won't be straight so we handled it to be able to catch the falling shapes that exists bit left or right but still inside this interval

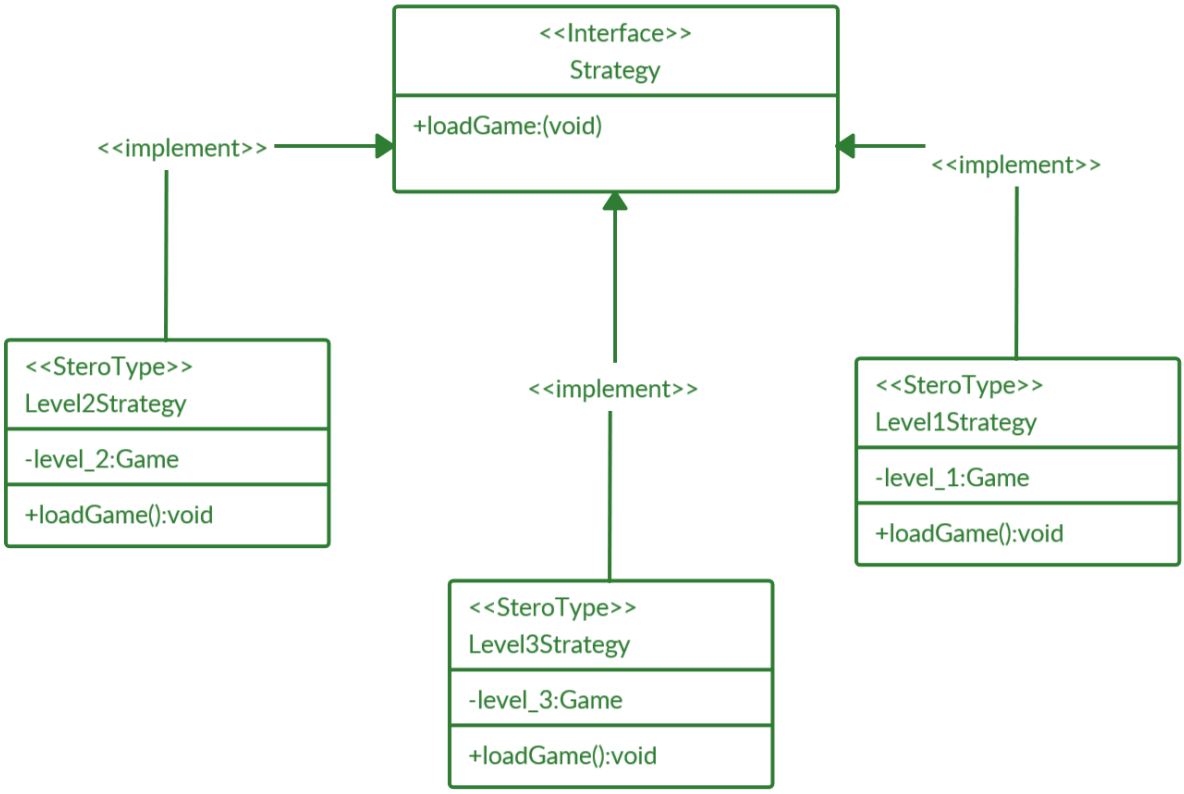
Sequence diagram:



Used Design patterns:

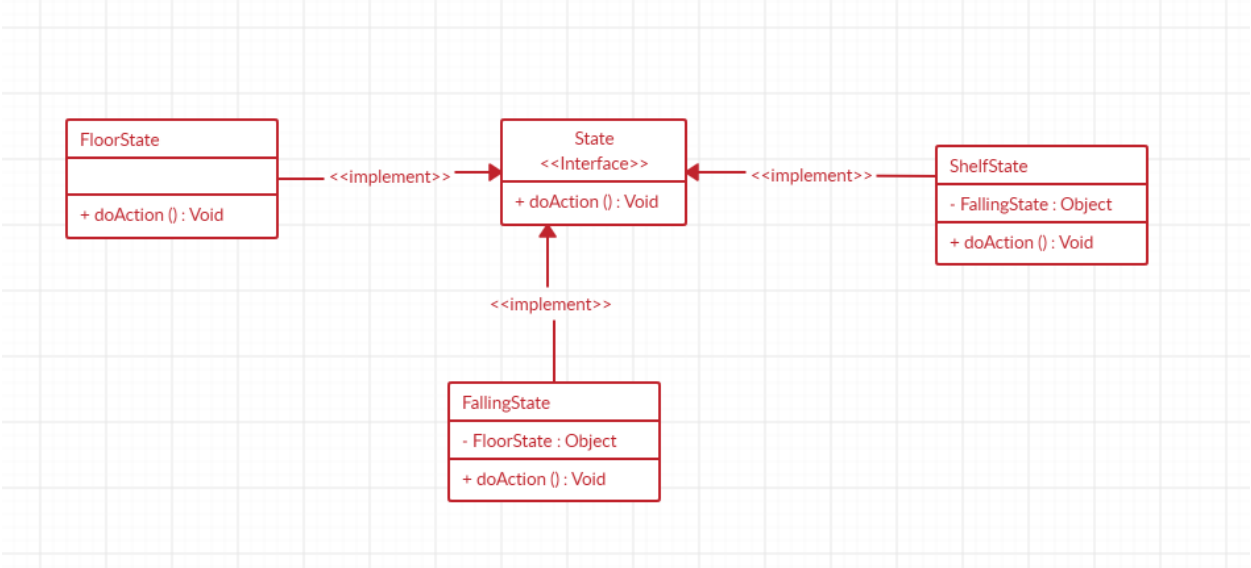
➔Strategy:

To enable multiple game levels which varies the speed of the falling shapes , strategy design pattern was used.



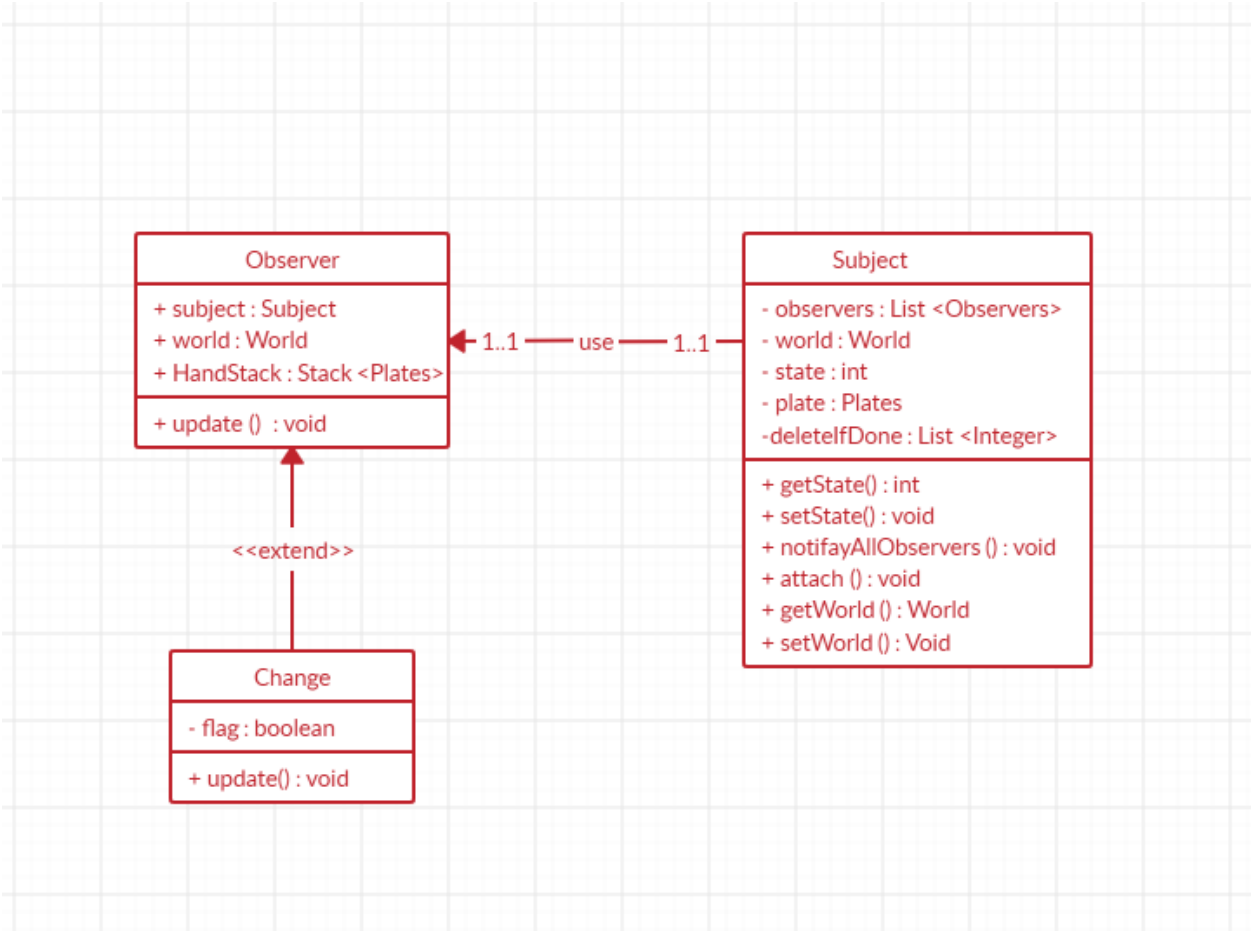
→State:

We use it to Know if the Plate in FloorState , FallingState , CaughtState or ShelfState , We use it to make Flyweight easier by Changing visibility of Plate , Overall it Classes extend Big Class ”State” .



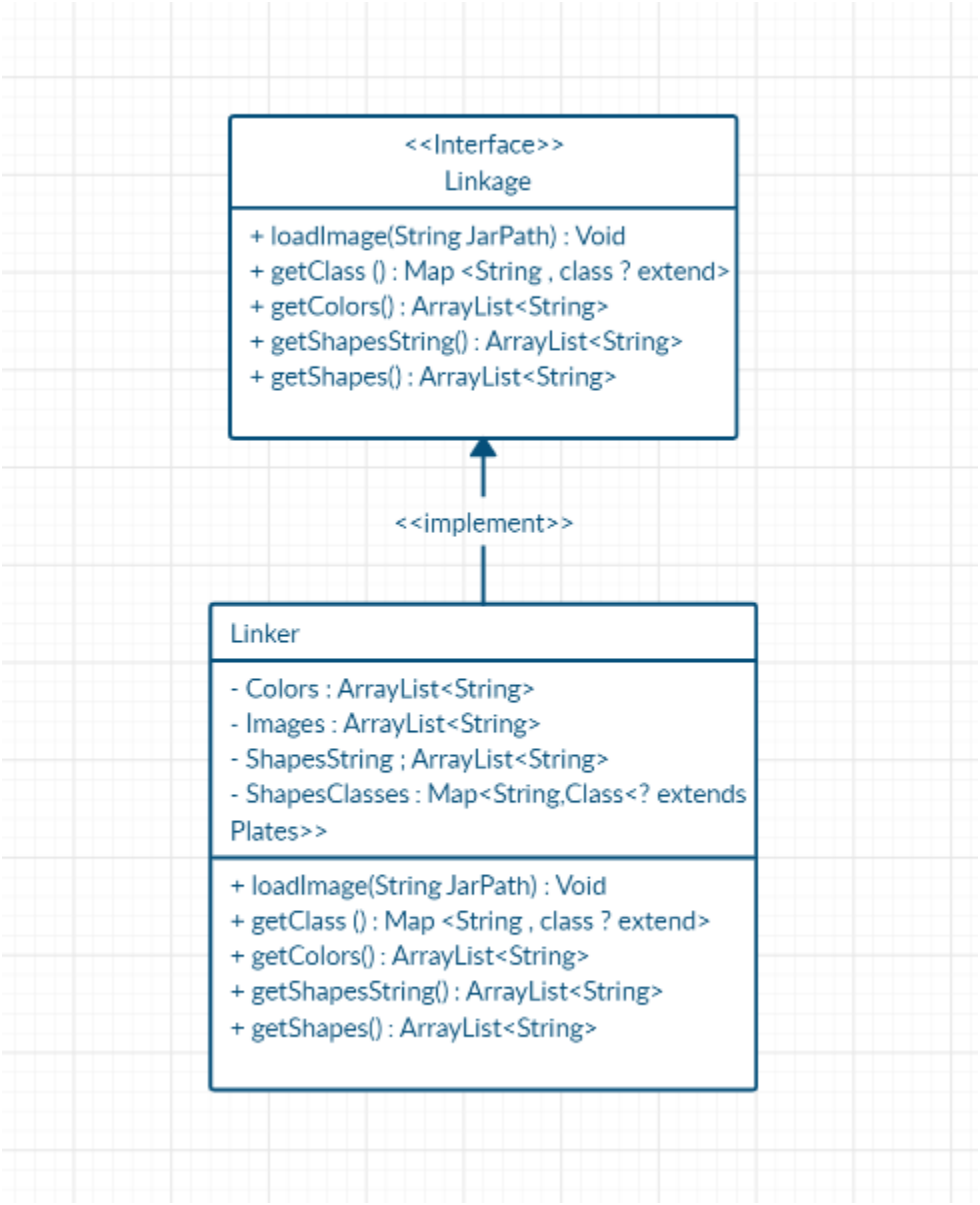
➔Observer:

To ge Score and Save all of scores in the game in Subject Class We Can get new Score From Change Class extend Observer Class .



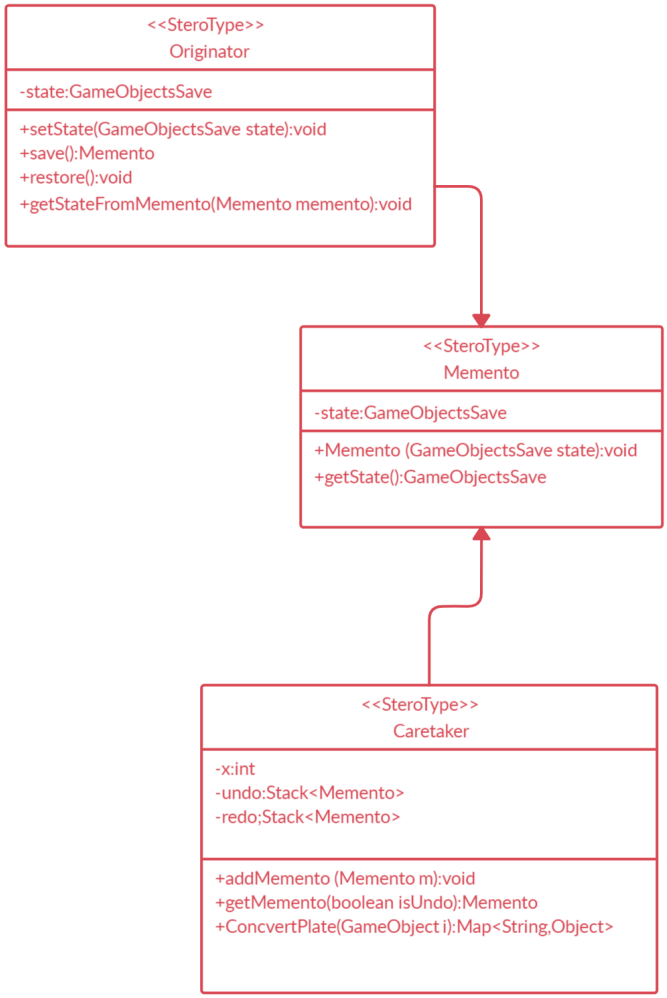
➔Dynamic linkage:

It use to Load images To the game By interface call “Linkage” implements By Class called “ Linker ” Linker could upload images to the Game .



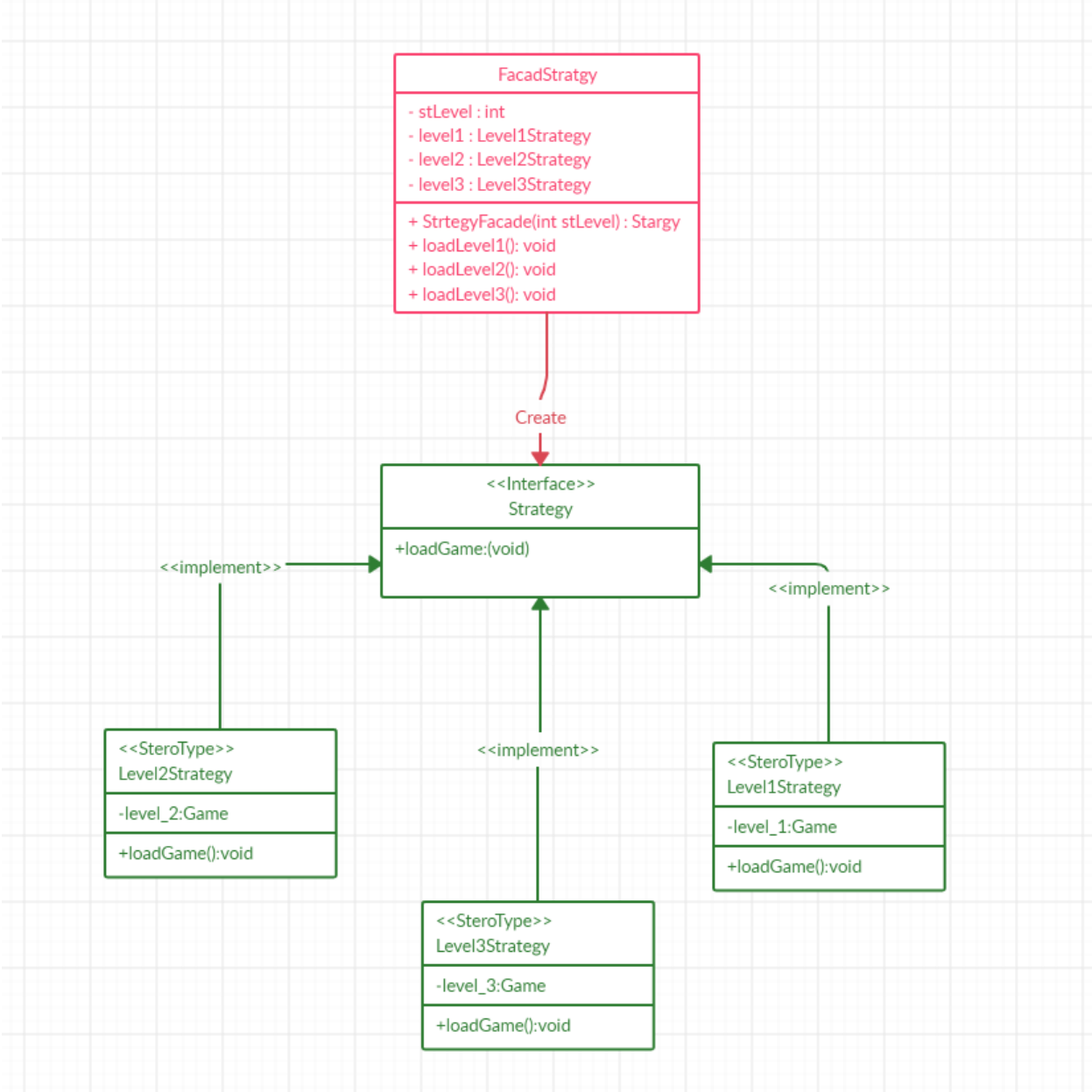
➔Momento:

We are using it to make Undo By saving every state by all of its details , In the game we have Save function to save all of the world in the momento and its contain three Classe one to save in the momento State and other to save all momentos and last one to know the state



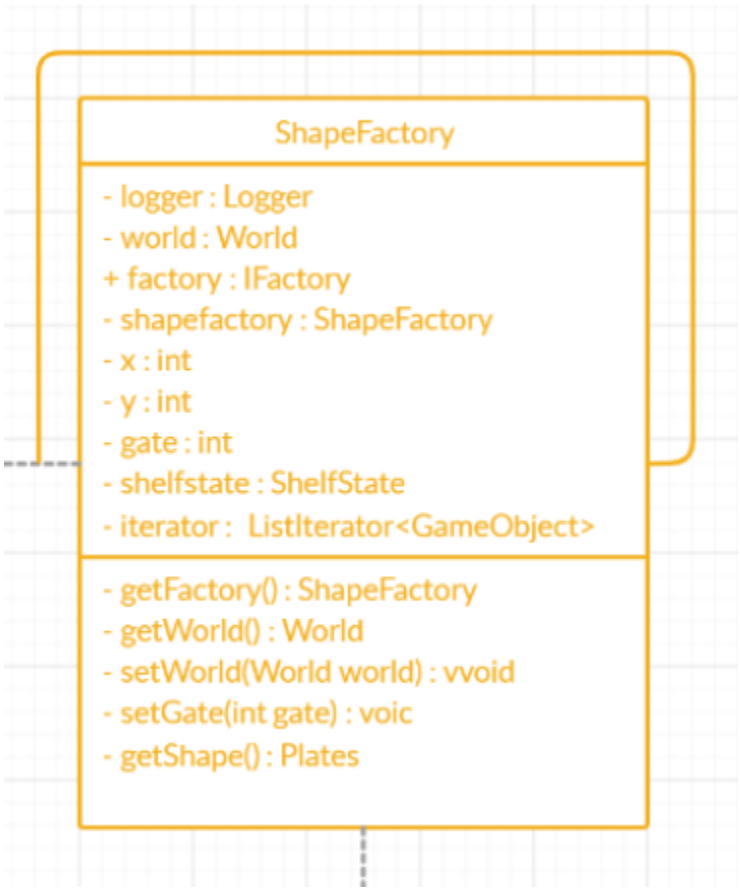
➔Facade:

We are using Facade to hide the complexity of the Class by choosing which level in the Strategy will be Called .



➔Singleton:

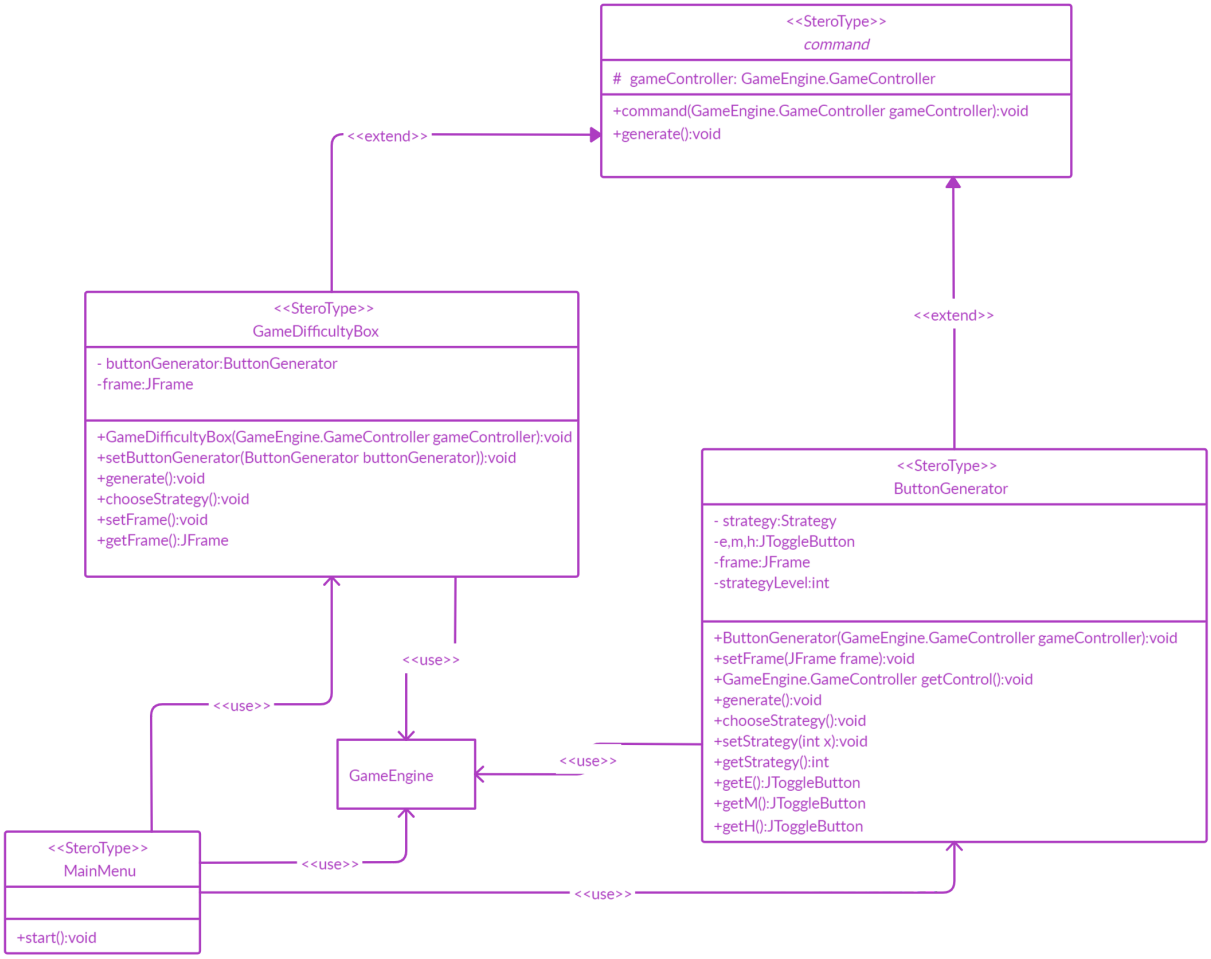
We use it in the Shape Factory to make sure to have only one of it and can’t create another one if he have exciting one .



➔Command:

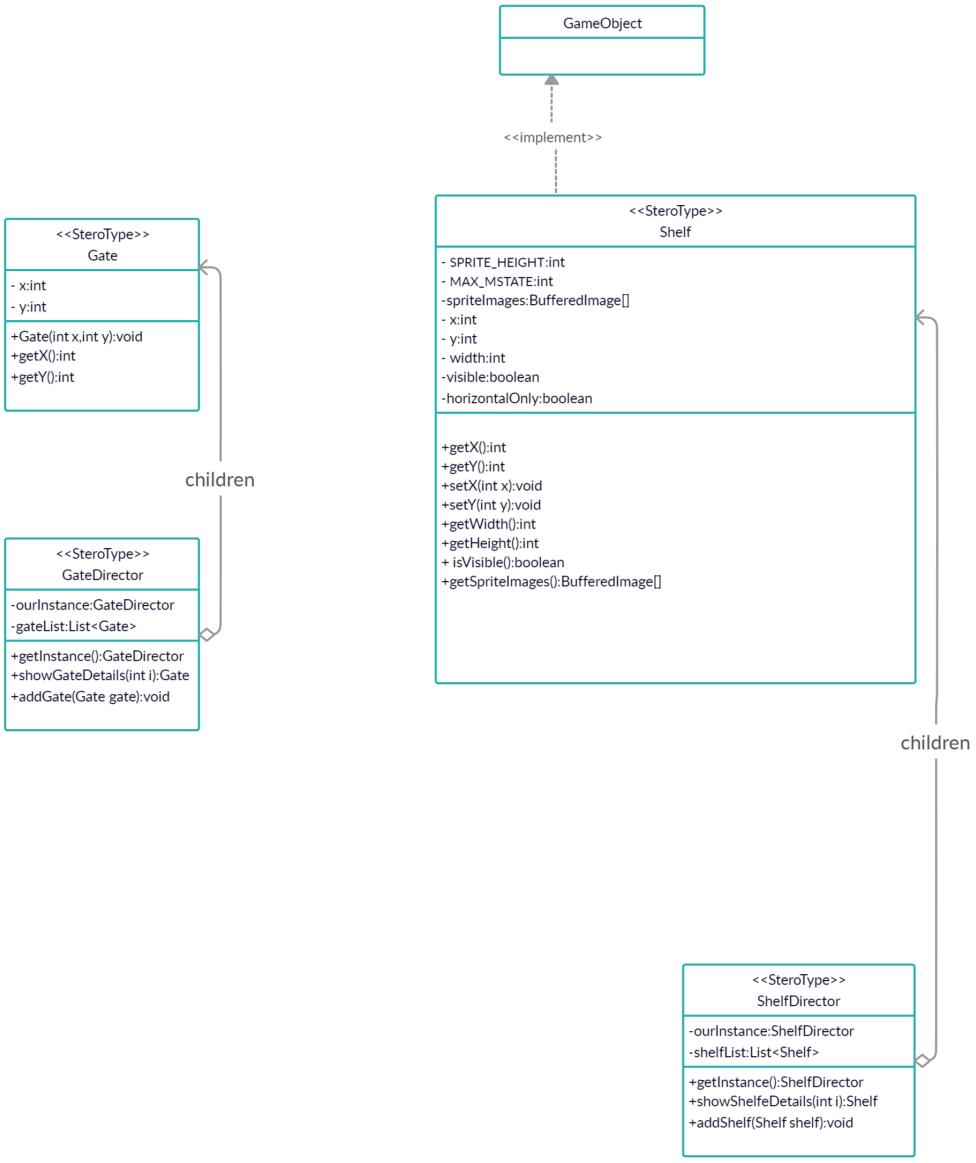
We use Command to choose which button the user pushed difficulty buttons or pause game we have Abstract Class extend by two class .

We also use it for moving backward and forward and for resuming and starting game



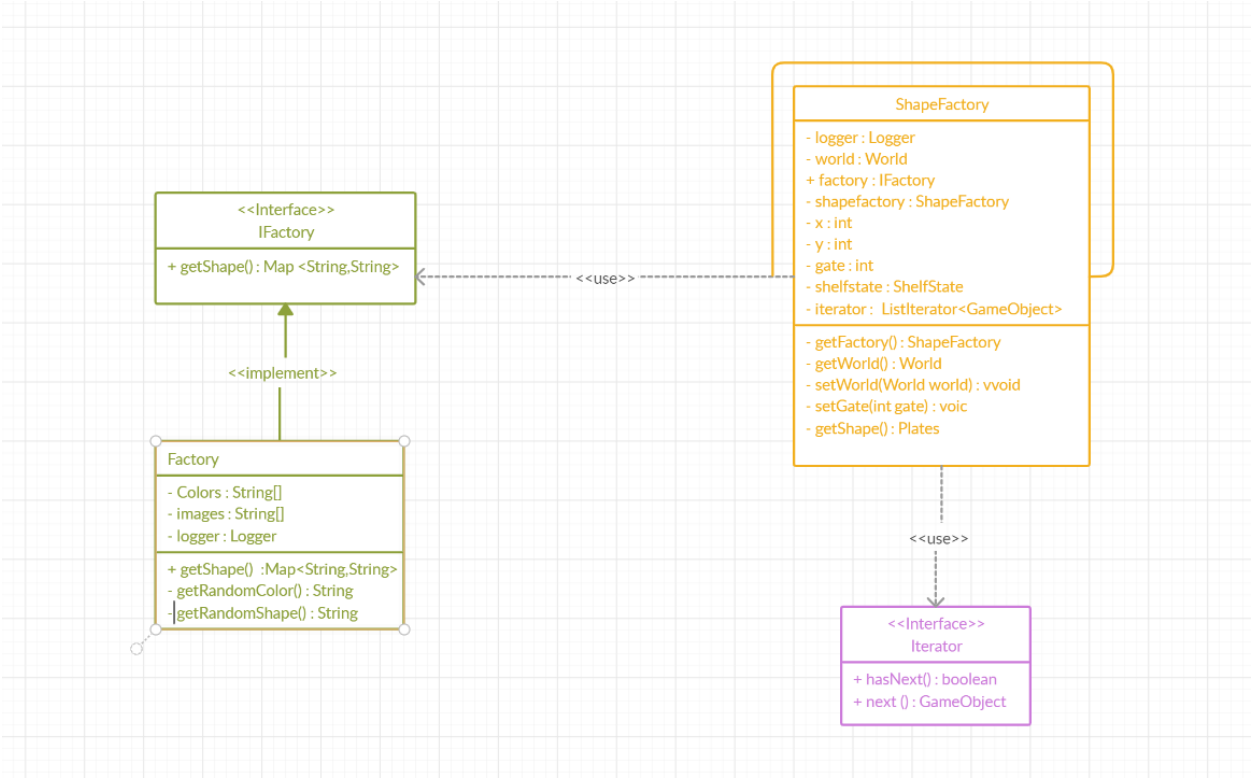
➔Composite:

We are using composite to make an array of the class like parent and child and We are using it twice it's like to be Singleton but not with only one execution .



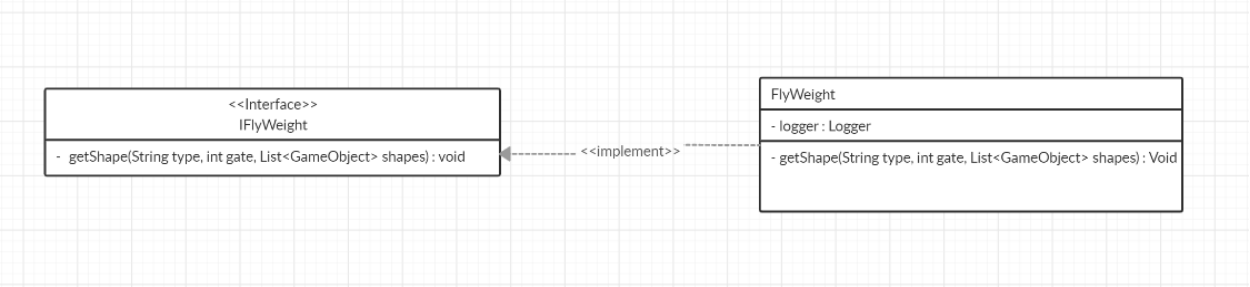
➔Factory:

We use Factory to make class easier for client to use it and make it easier to programmer to edit it We have interface called “ IFactory “ and class implement called “ Factory ” with all methods .



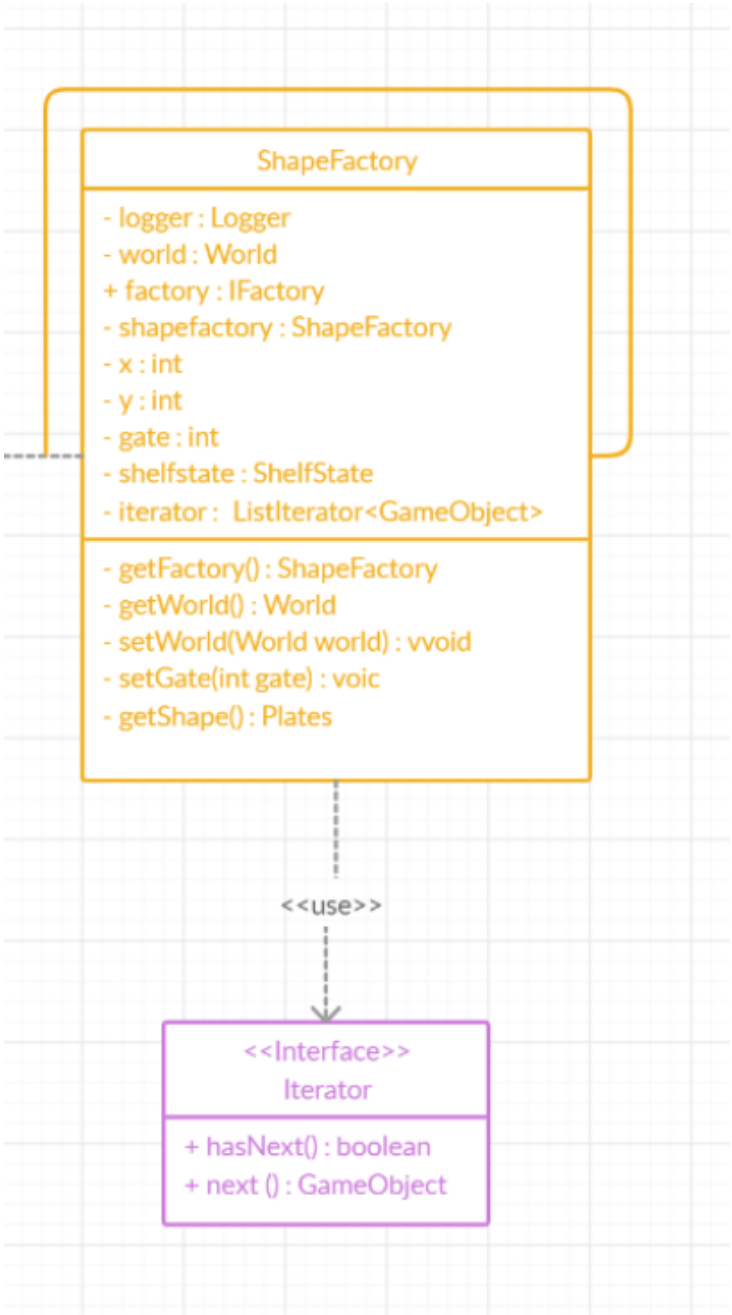
➔Flyweight:

We use Flyweight to check if plate is not visible if it out of the window bound or any Other way will make this plate come from the gates again to reduce the number of plates .



➔Iterator:

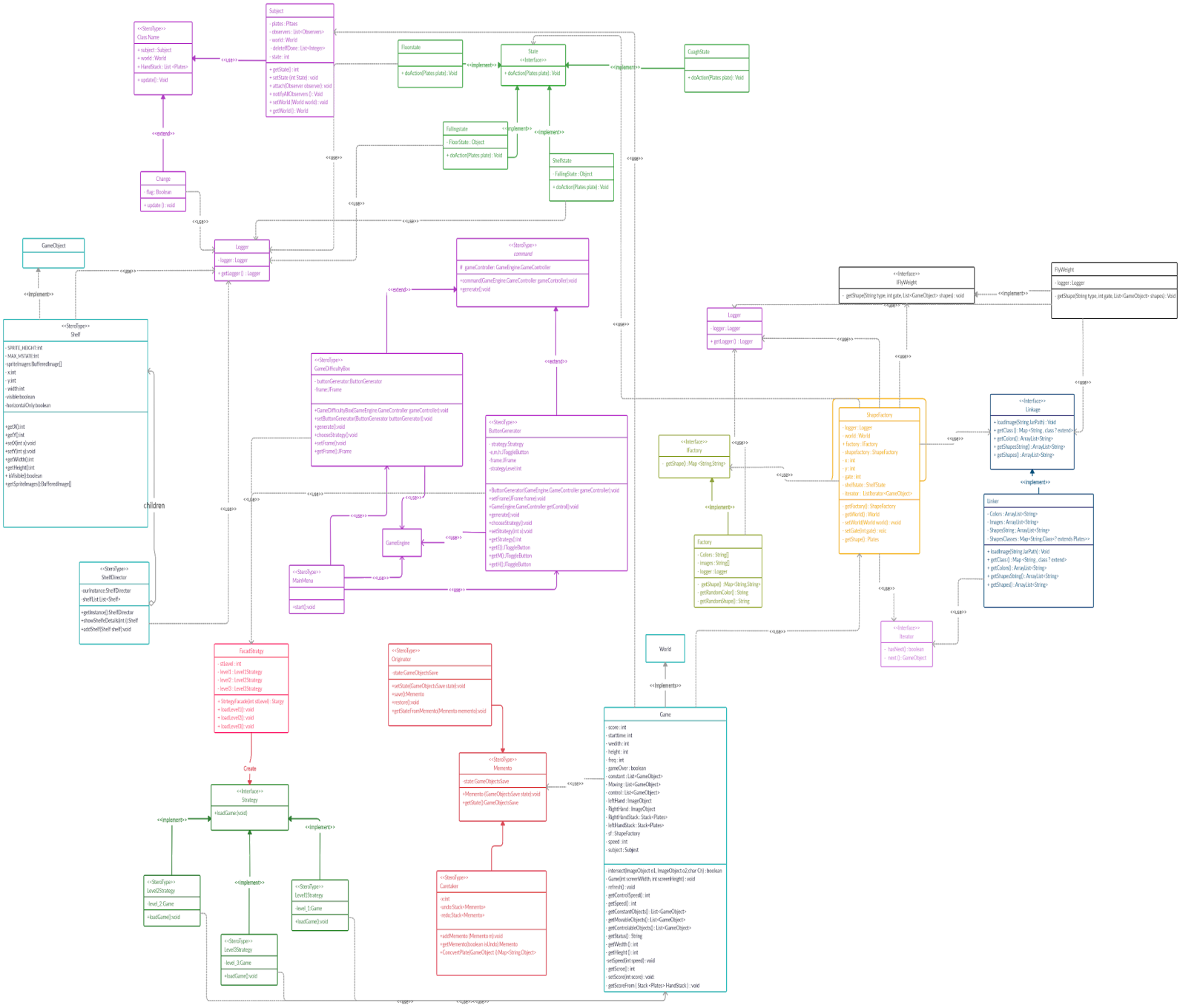
This pattern is used to get a way to access the elements of a collection object in sequential manner without any need to know its underlying representation and we are using built in .



➔MVC:

➔ MVC is applied it is divided into 3 main components : Model, View, and Controller.

Class diagram:

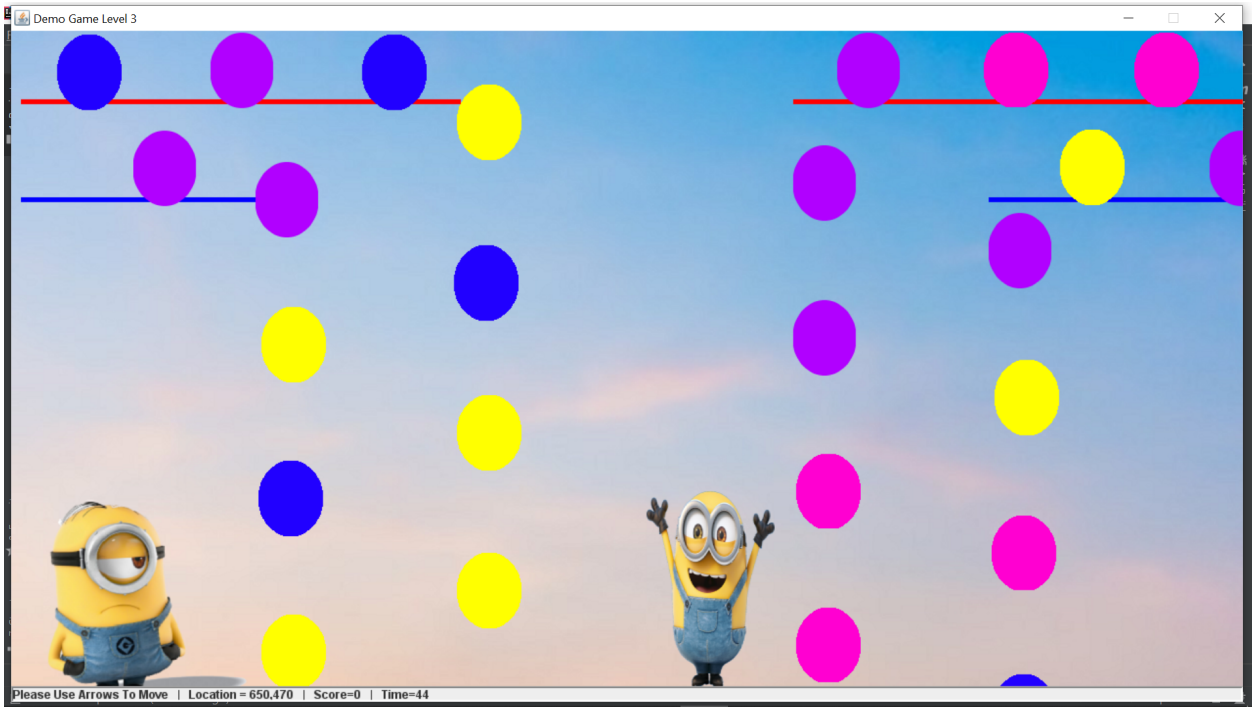


User guide & GUI Snapps:

➔ When the user open the game

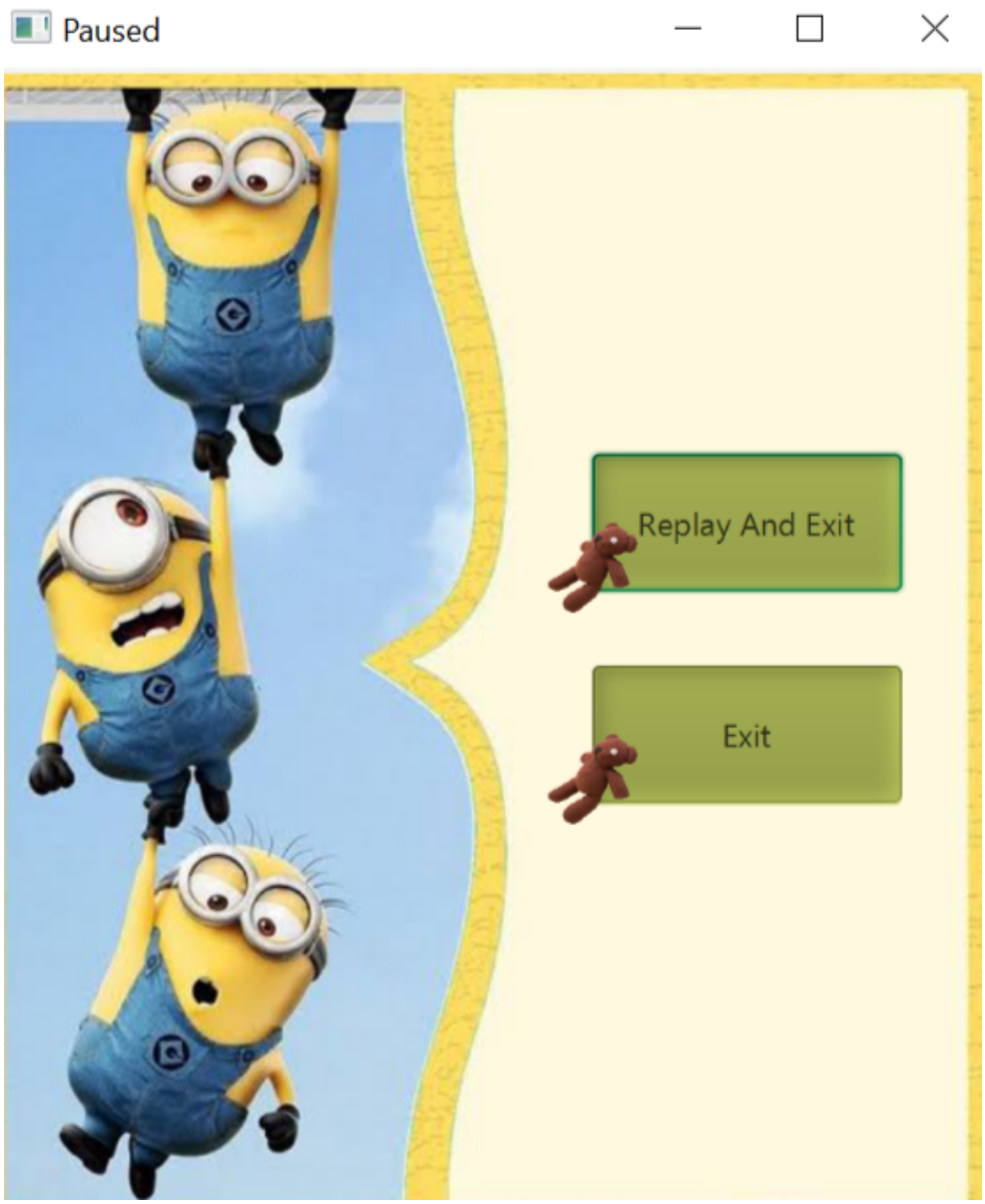
The user must choose level

➔ After choosing a level the game started



➔ To backward and forward the game click on F or B

➔ Now, you are able to reply game or exit



➔ Time and score is shown below

➔ Game over is applied when time is out or stack is