**Circus of Plates**

**Prepared by:**

* Eman Rafik Abd El-Kader ID: 11
* Toka Alaa Elgindy ID: 14
* Nada Salama Mohamed ID: 55
* Yomna Gamal El-Din Mahmoud ID: 60
* **Design Description:**

It is single player-game in which clown carries two stacks, and there are a set of colored shapes that 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.

**Implementation details:**

1 - Game class implements the world interface which represents the game flow its objects. Game class has an instance of the singleton class GameInfo in which all the game data is being initialized, modified throughout the game.

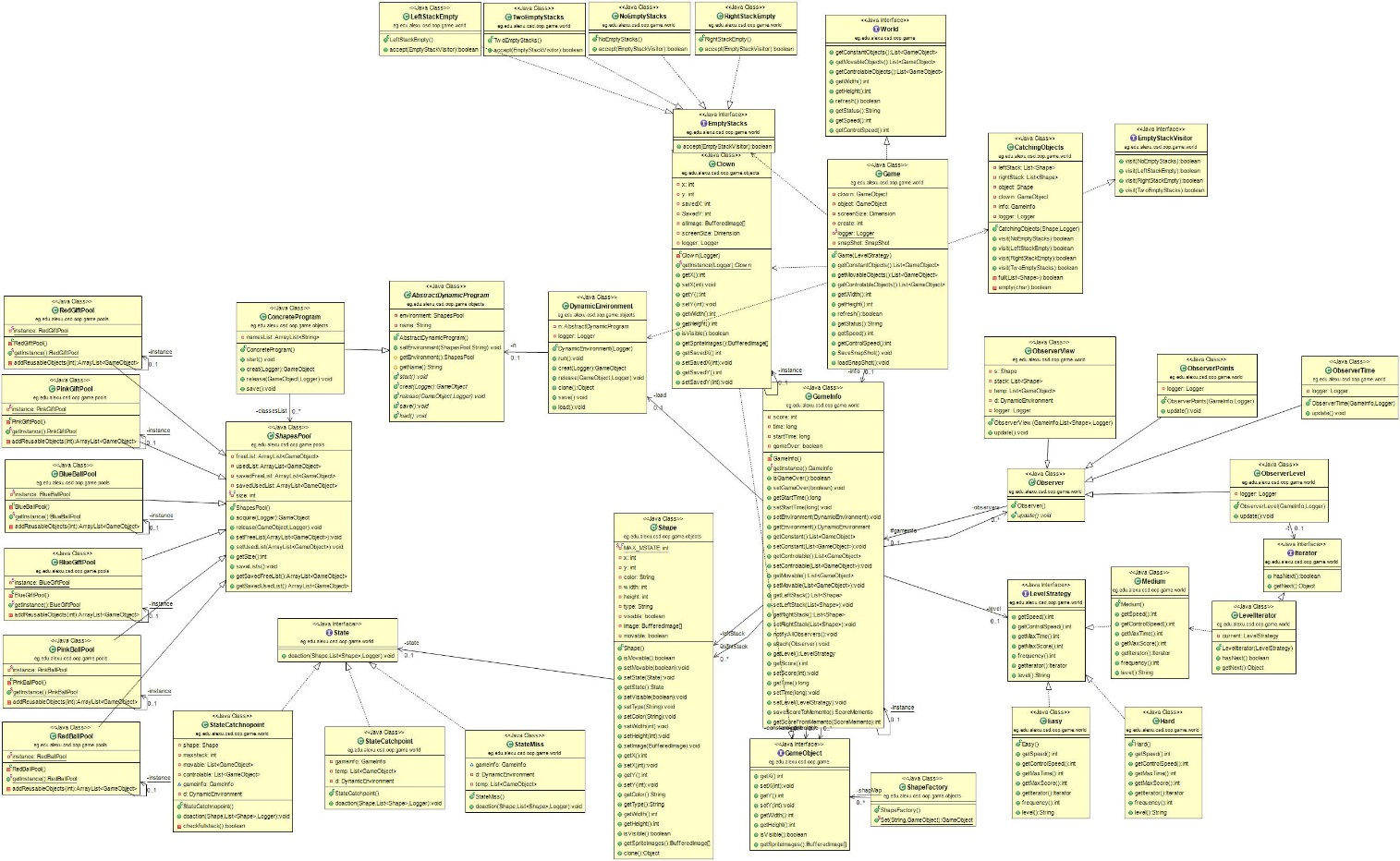
2 - Clown class is a singleton class implementing the GameObject interface and representing the user controllable clown object. Instance of Clown is firstly created in the Game class.

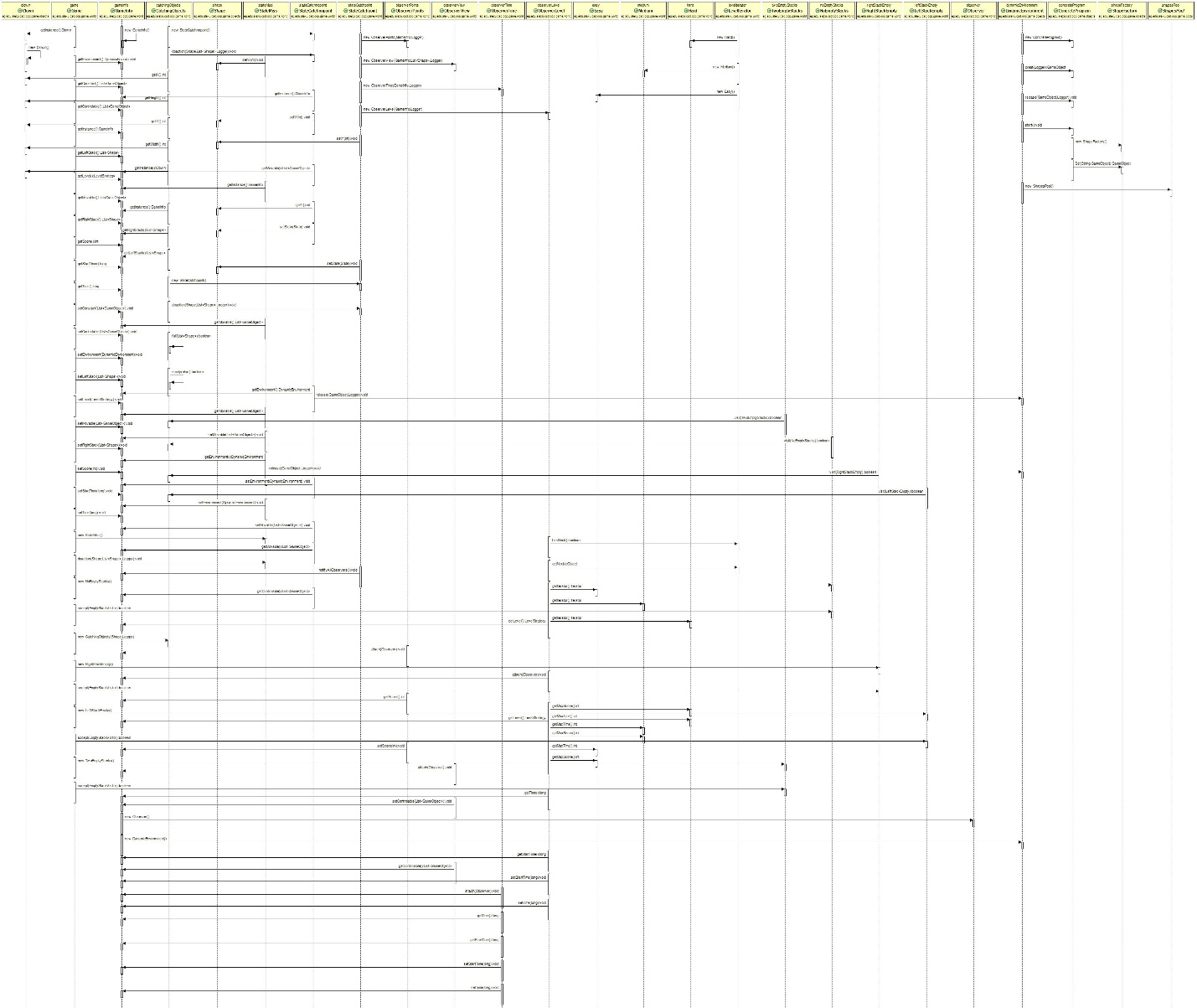
3 - Abstract class Shape implements the GameObject interface and various objects in the game extend the abstract Shape.

4 - Abstract class ShapesPool is inherited by 6 types of shapes classes. They are blue ball pool class, pink ball pool class, red ball pool class, blue gift pool class, pink gift pool class and red gift pool classs.

5 - LevelStrategy interface is implemented by each class representing a level difficulty supported in the game.

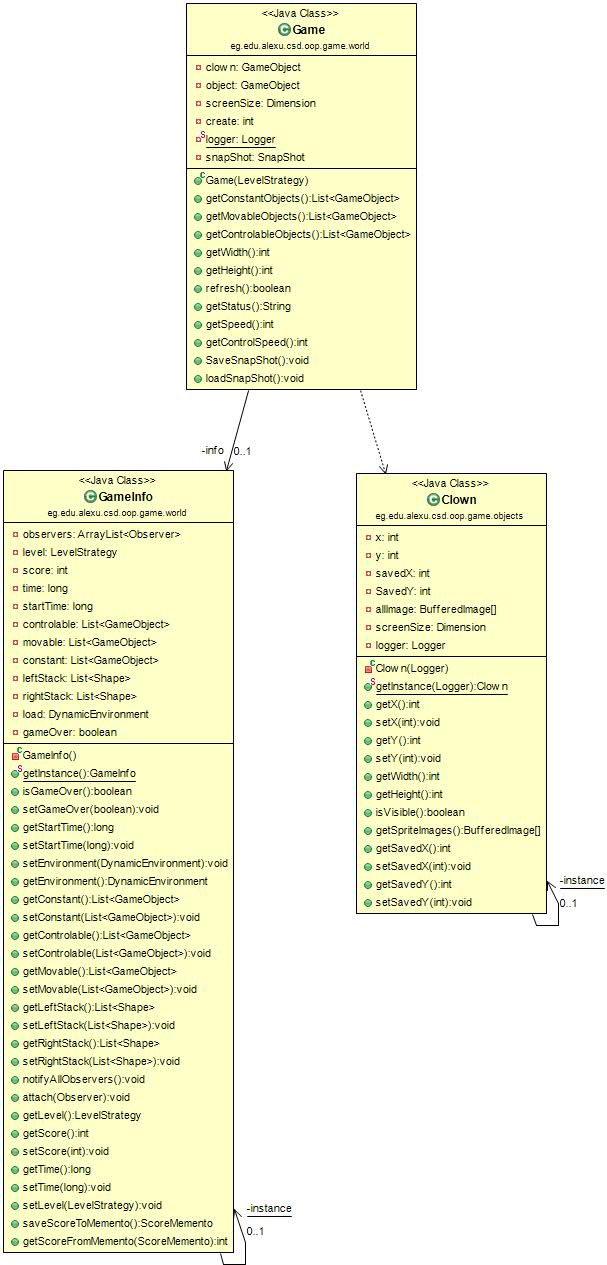
6 - Classes that represent different states of falling objects at which their behavior varies implements the State interface. Observer classes which extend the abstract observer are all notified and updated on catching an object that scores a point.

* **Class Diagram:**
* **Sequence Diagram:**

****

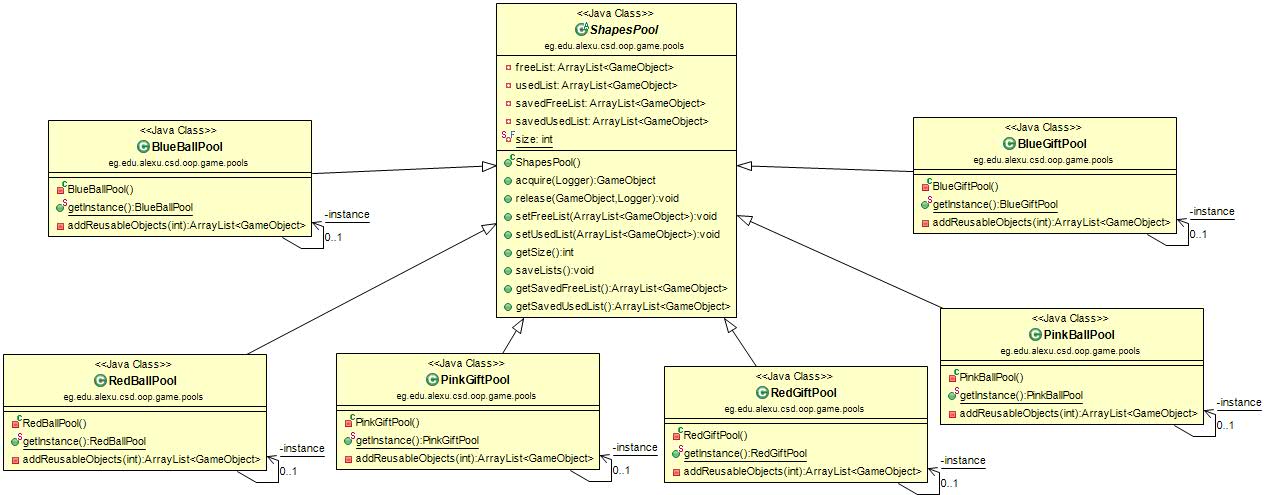
* **Design Patterns:**
* Singleton

We use this pattern to create one object of the clown of the game and to create one game information “gameinfo” for each game.where we all information in class of gameinfo and this class create once only from game and put all information in it which we can get them again in any class.



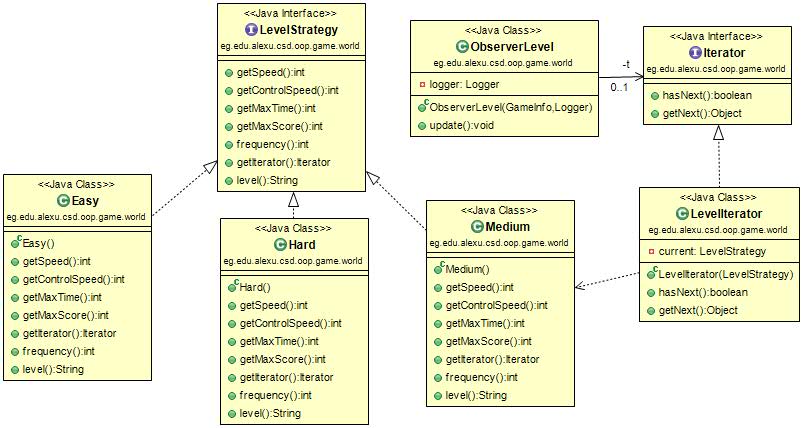
* Pool

Pool design pattern is used with the falling shapes. We create abstract class called "shapes pool" with 6 types of shape pool that which extend this class. Each type creates 20 object of its own shape.



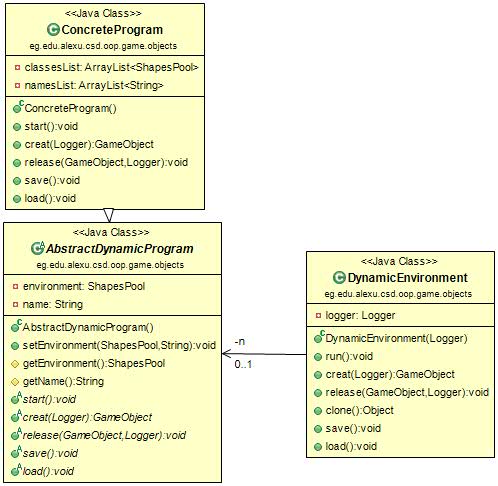
* Iterator

Iterator pattern is used to access sequence of different levels supported in the game. Class LevelIterator implements Iterator interface and takes parameter of LevelStrategy interface in its constructor. Every concrete class implementing LevelStrategy has a function getIterator to return a level iterator on it.



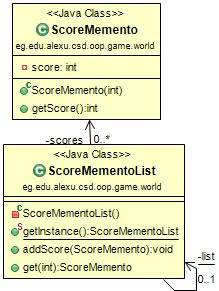
* Dynamic Linkage

Dynamic linkage design pattern is used to load the 6 types of shapes pool from a jar file before the game starts.



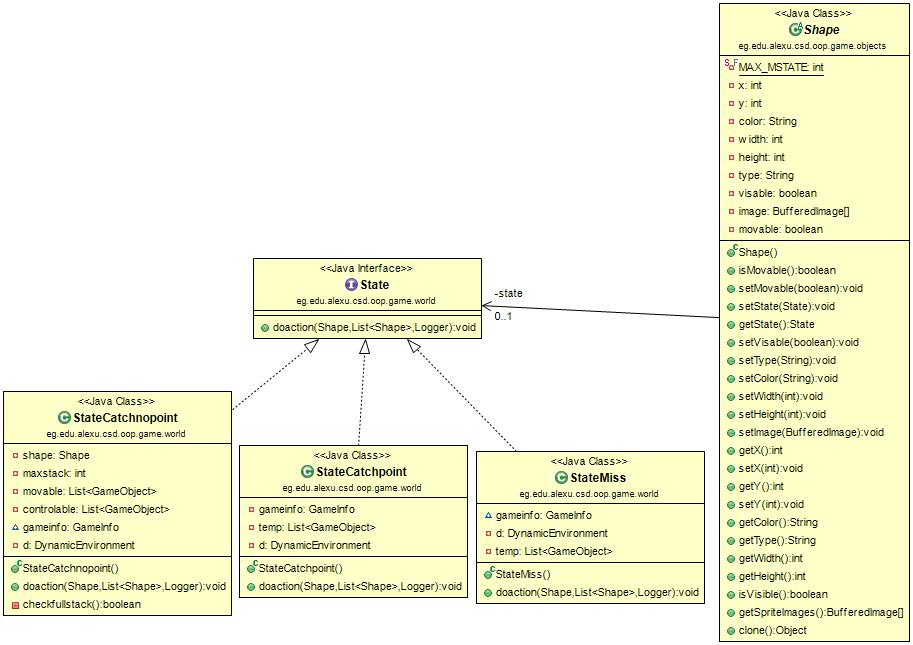
* Snapshot

Snapshot is used to save the game at some point and then return to that point again before game over. Class Snapshot holds all the details of the saved point. Class game calls the Snapshot either to save or load it.



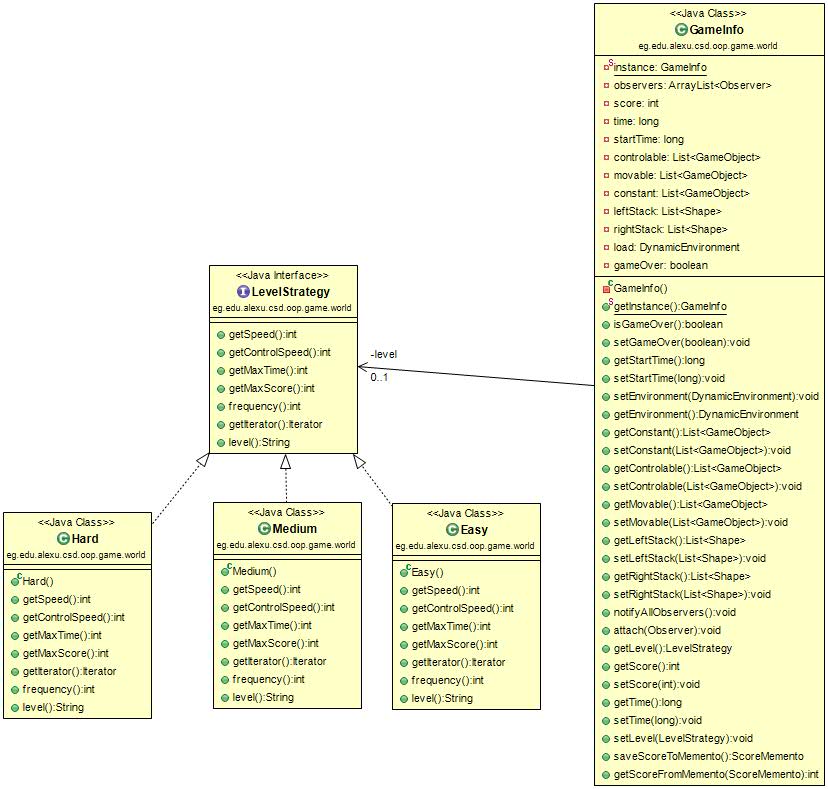
* State

State pattern is used to determine the state of each shape on falling. Three concrete classes: SateMiss, Statecatchnopoint and state catchpoint implements the state interface. Class shape carries the state and its behavior varies as the state varies.



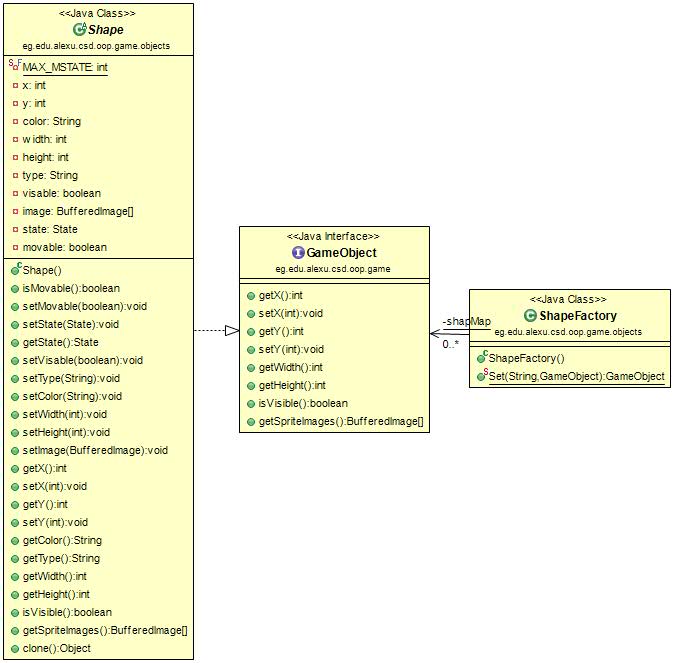
* Strategy

Strategy pattern is used to supprt different level strategies in the game. Three classes: Easy, Medium and Hard implements LevelStrategy interface to represent the behavior of different game levels. LevelStrategy is used in gameInfo to differ as level differs, it is firstly set in the constructor of Game class then iterated depending on the game sequence.



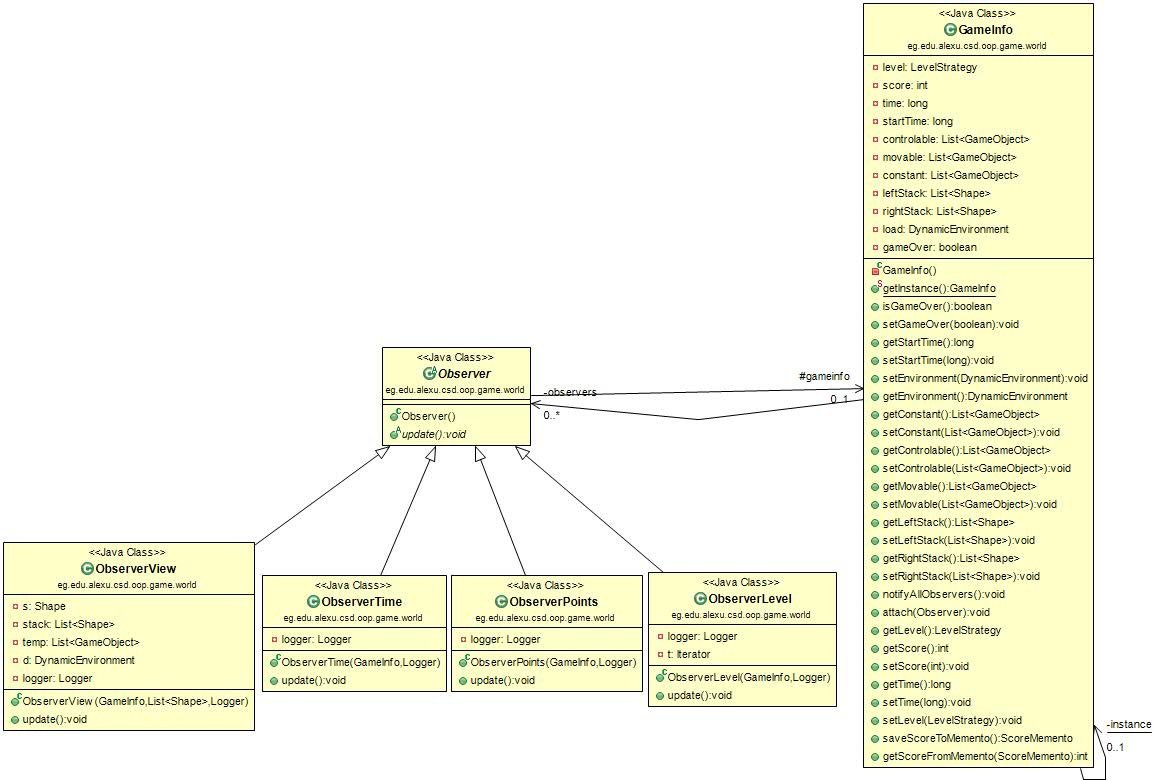
* Flyweight

Flyweight design pattern is used with the shapes pool before shapes appear in the view to assign a random number of the position for the falling shapes.



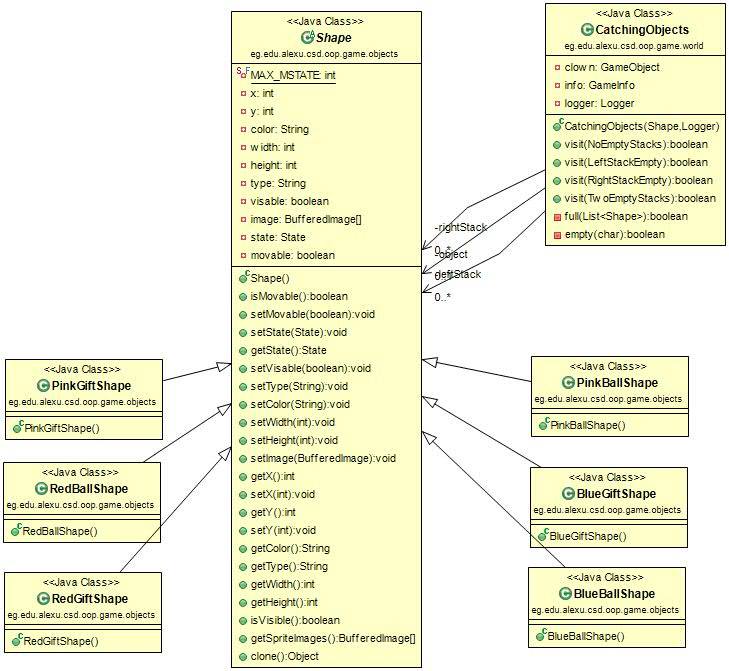
* Observer

Observer pattern is used to modify the game on getting a point. Four concrete classes: ObserverLevel, ObserverView, ObserverTime and ObserverClass extent the abstract class Observer. Game has a list of observers to be notified and updated on getting a point.



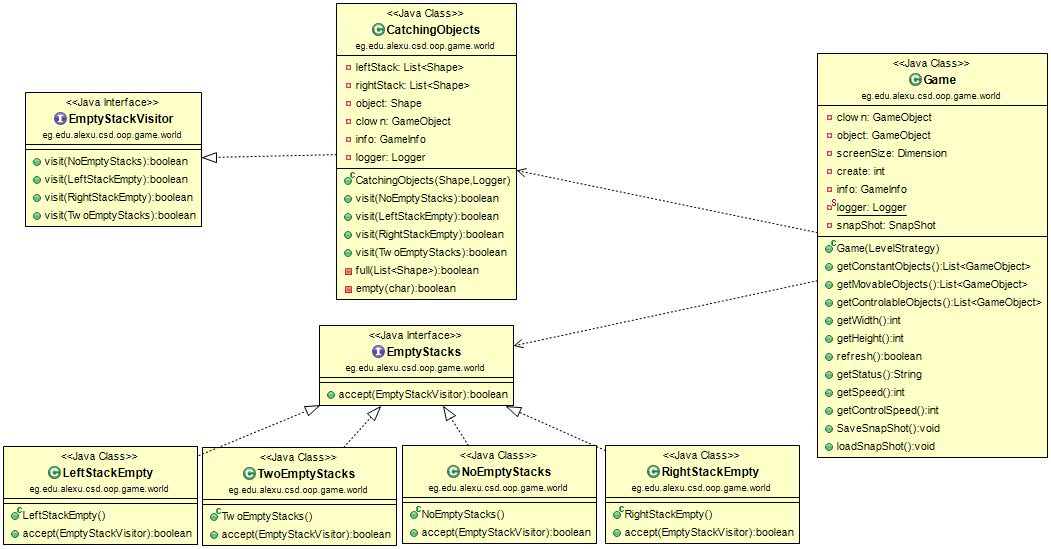
* Prototype

Abstract class Shape implements the cloneable interface to get a duplicate of a shape on being translocated from the movable list to the controllable list after being caught. Duplicated shape is created in Catchnopoint class.



* Visitor

Visitor pattern is used to handle and execute different algorithms of objects to be caught in one of the stacks held by the clown. Four classes: NoEmptyStacks, LeftStackEmpty, RightStackEmpty and TwoEmptyStacks implements EmptyStacks interface which accepts EmptyStackVisitor interface. Class CatchingObjects implements the EmptyStackVisitor interface to implement various algorithms of the four cases.



* **Design Decisions:**

1. Any class to be loaded as a falling object must extend abstract Shape.
2. Snapshot is only loaded before game over.
3. Before saving or loading a snapshot, game must be paused.
4. Time increases when player collects 3 shapes with the same color or when the game transfers to the next level.
5. The game ends when time out or the stack is full.
6. There are 3 levels of difficulty (easy – medium – hard); when player gets the required score of the current level, the game transfers to the next level.
7. The type and position of the falling shape is chosen randomly.
8. The number of the falling shapes increases in each level.
9. There is a single clown in the game for all levels.