Game Report

Co\_workers:

Abdelrhman Yasser ID : 39

Ahmed Ashraf ID : 6

Mohammed Deifallah ID : 58

**Game**

* Introduction:

That application introduces a two-player game using the keyboard. Every player tries as much as he can to catch the falling different-shaped plates in such a way to win. The player gets a score if he could collect three consecutive plates from the same color.

* Overall design:

The code of that game is coded using design patterns to apply the principles of OOP. At the beginning, all shapes are loaded using a class loader component. Then, a factory is used to produce those plates, and there's also a pool to store an enough quantity of plates in order to reuse the unused falling plates. Also, the GUI is implemented using the MVC design pattern. The game introduces different states as difficulty levels. All components that are needed to be created only once in the game are implemented with the Singleton design pattern as the components of saving, loading, logging….etc.

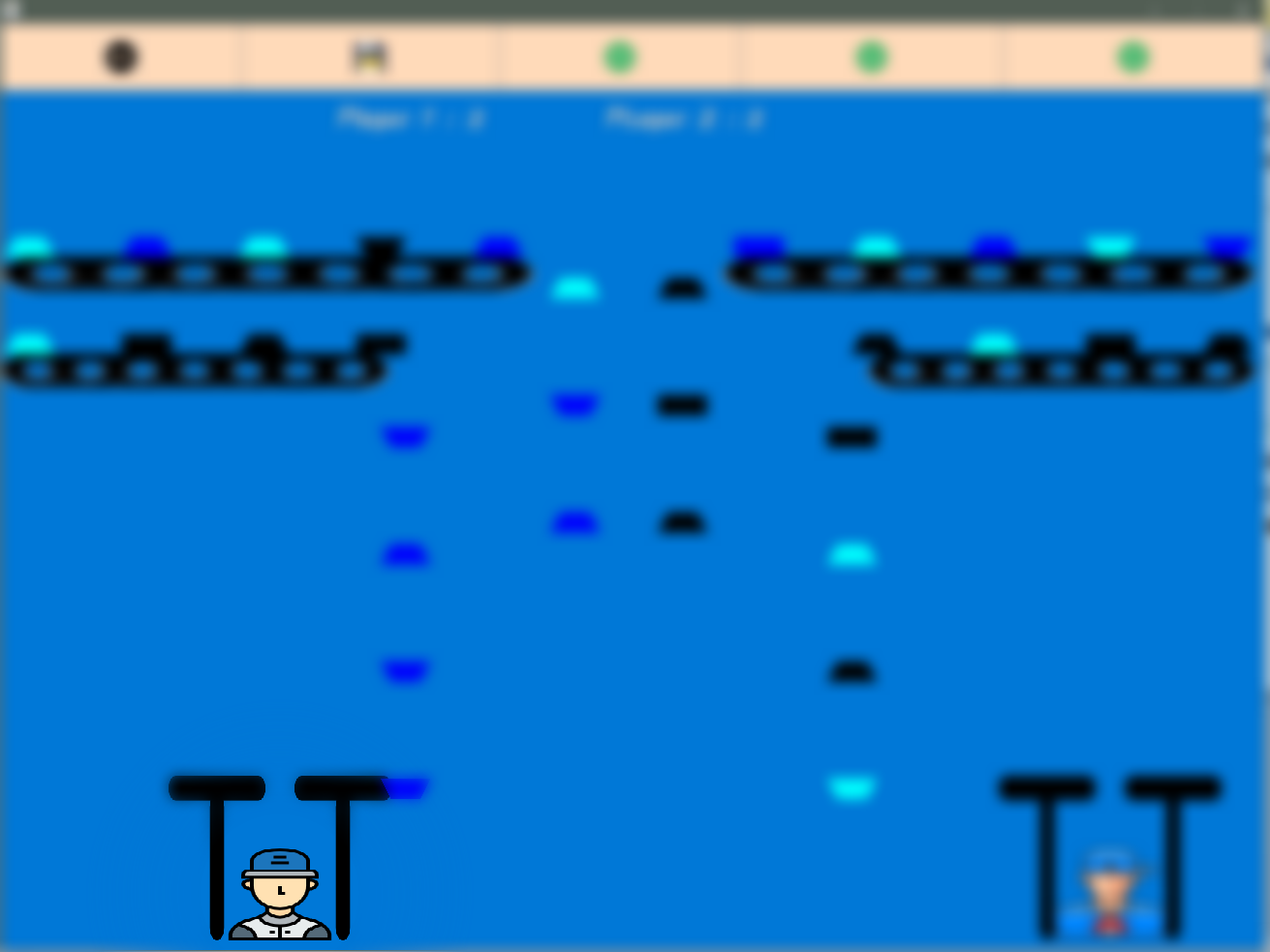
* Design patterns:

1. Singleton: That pattern is used as shown above as a creational design pattern to create the components that are needed to be created only once through all the game. So, it was convenient to use it when implementing saving, loading, class loading, logging ….etc.
2. Factory: As a creational design pattern, it was suitable to use it for creating the plates with their different shapes and color.
3. Dynamic linkage: That pattern was used in that case when the game loads all the plates' classes at the beginning.
4. State: There are different states in that game. That game provides 3 different levels, such as: easy, medium and hard.
5. Observer: As there are threads in the game, so that pattern was used thoroughly to keep an eye on all changes.
6. MVC: That pattern was applied with the GUI, as it separates model, view and controller.
7. Object Pool: In order to reuse the unused plates which fall without being collected by any player, that pattern was used to store those plates. As a result there's no waste of plates, and the factory produces the plates when needed.

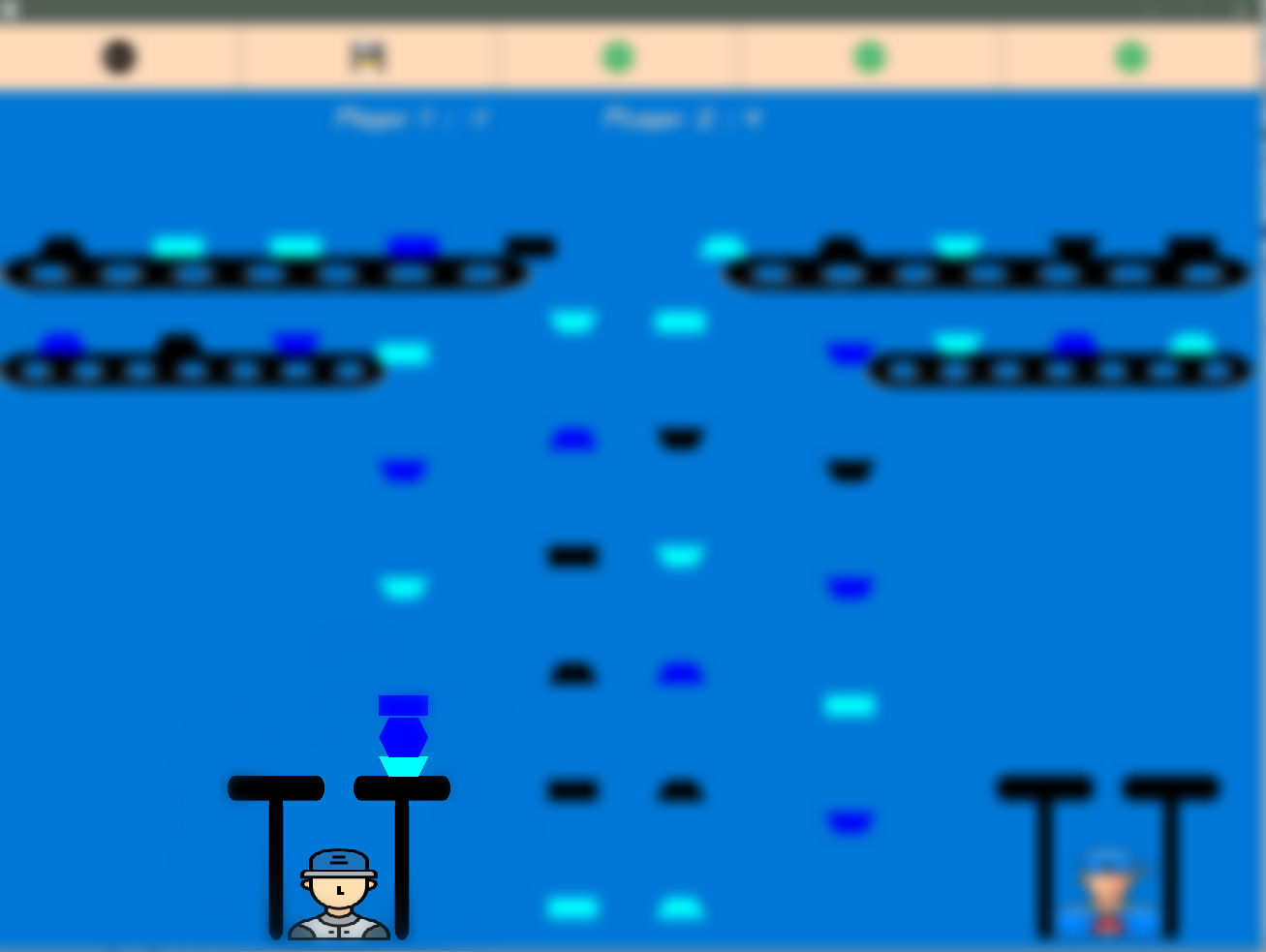
* User guide:

The GUI starts with a menu of options to select, whether to start a new game or to load a saved game, or even to choose a specified level. Then it introduces the players to the game itself.

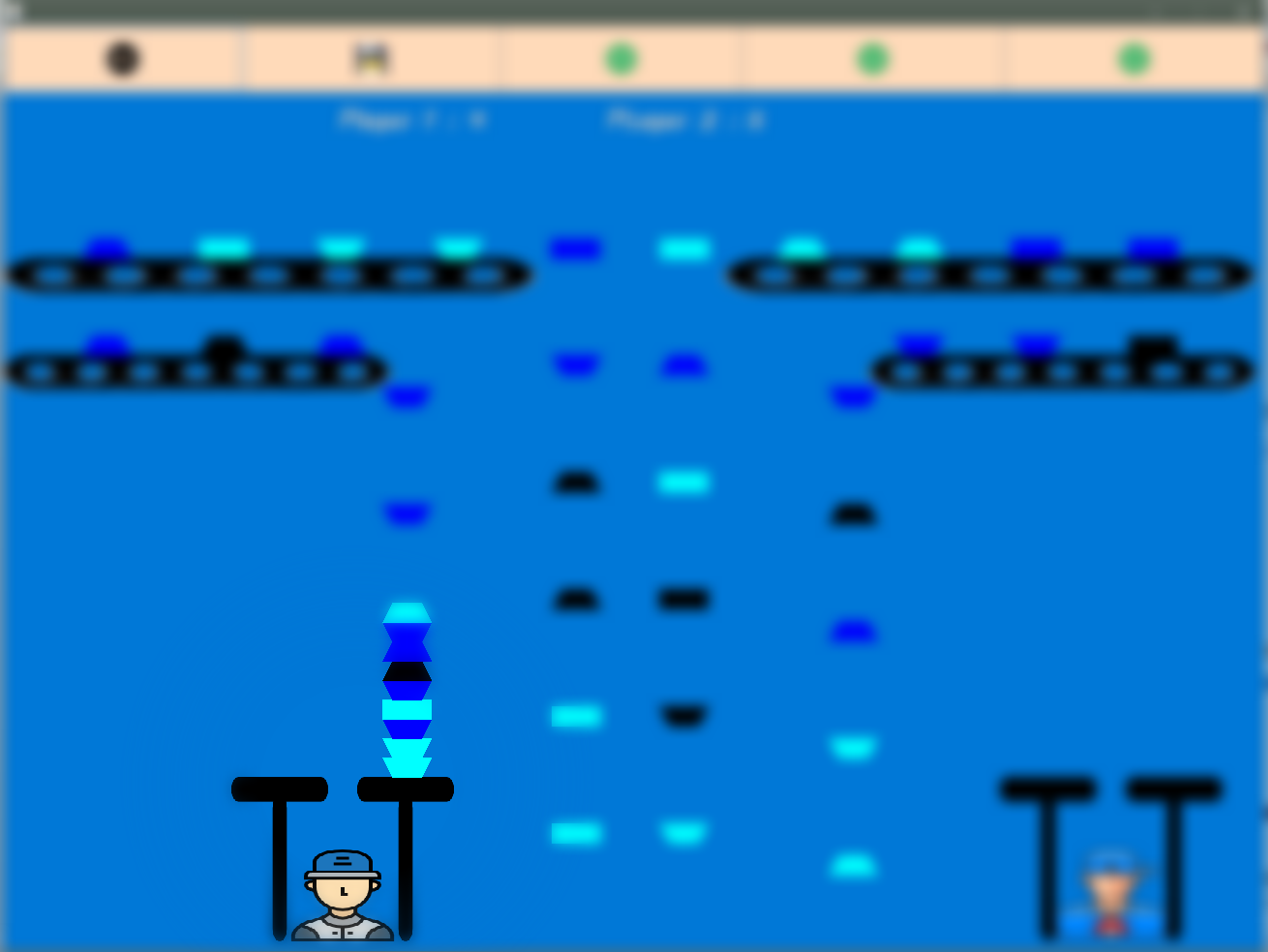
* In order to move the players, player1 uses the left and right arrows, and player2 uses 'A' & 'D' buttons.



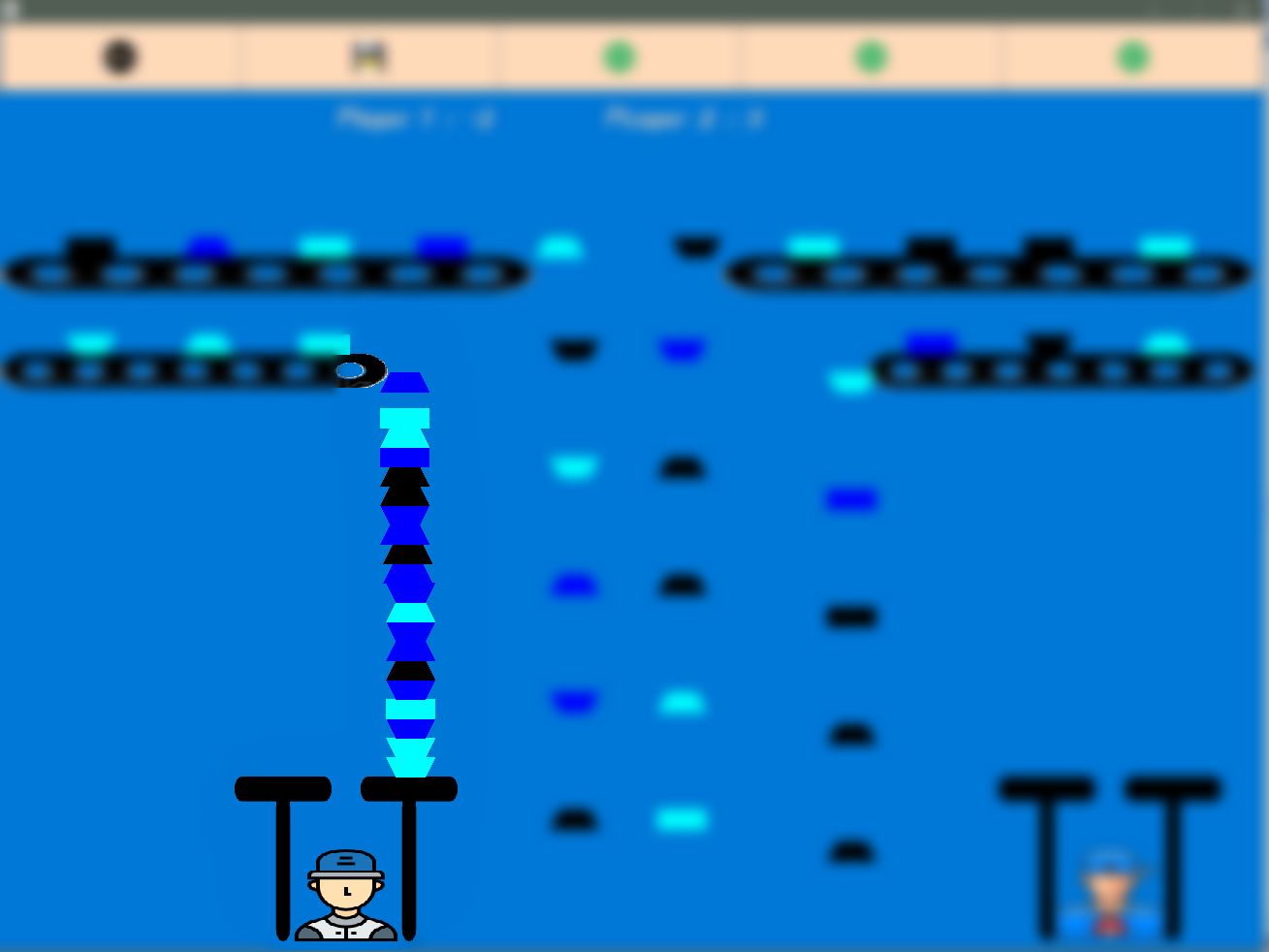
* When the player managed to collect three consecutive plates from the same color (even if different shapes), he gets a point, and those three plates vanished from his stack.



* There's a synchronized score list to show the scores of the two players.



* When some player collect to many plates without any winning condition, such that those plates crashes one of the shelves, that user loses a point and half of that collection.



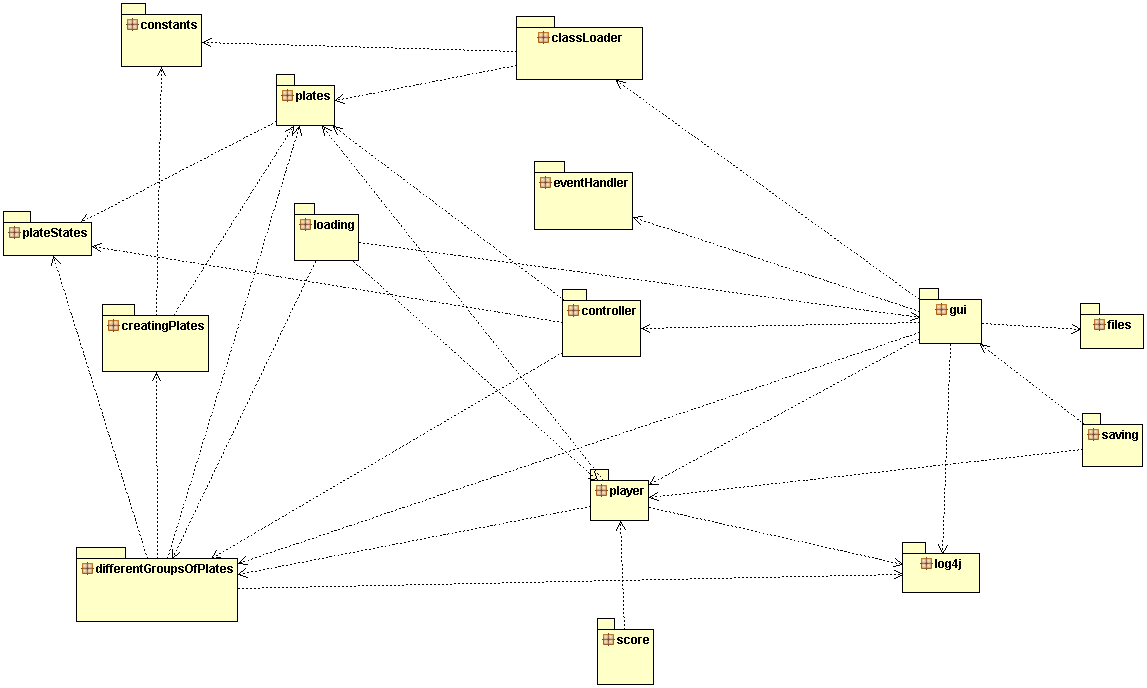
* There's a fantastic list of buttons at the top of the frame. That list introduces the user to many nice features in the game.



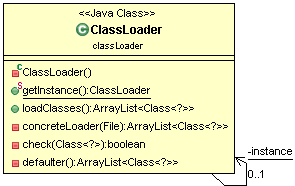
* That list has two buttons next to others having to opposite-direction arrows. Those two buttons can be used to change the difficulty level of the game, even if the game is currently running.
* That list provides a button to save the game, and load it again if needed.
* There's also a back button to return to that start menu.

UML’S :

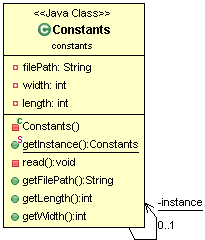
All packages :



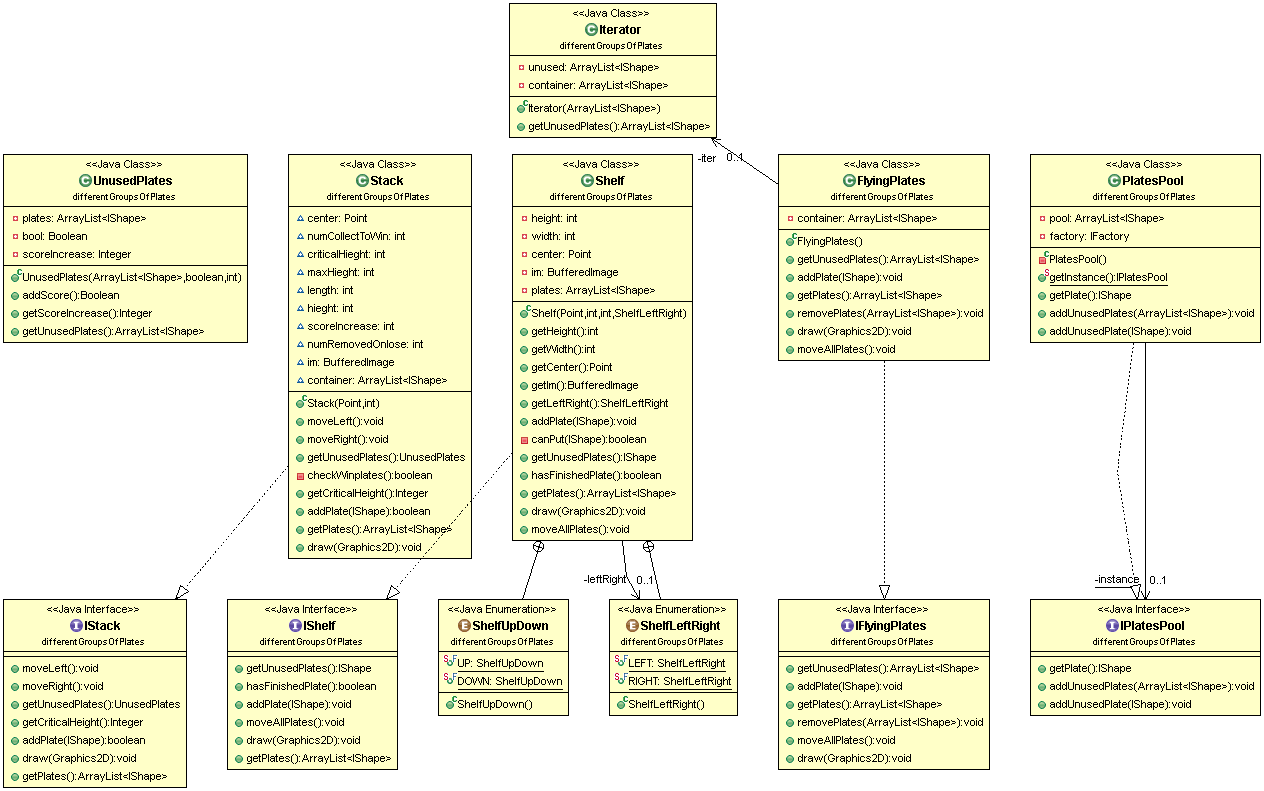
Class Loading :



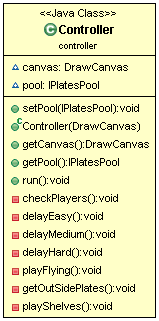
Constants :



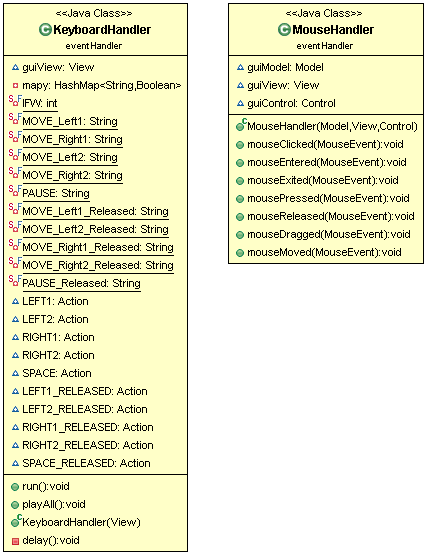
Containers :



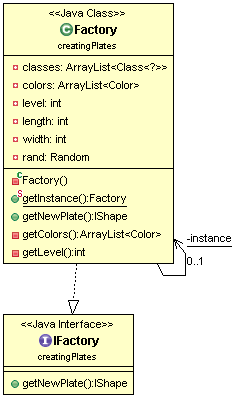
Controller:



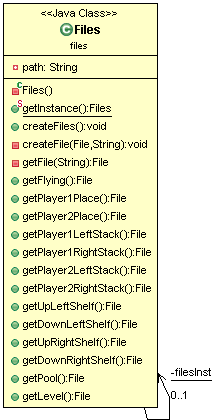
Event Handler :



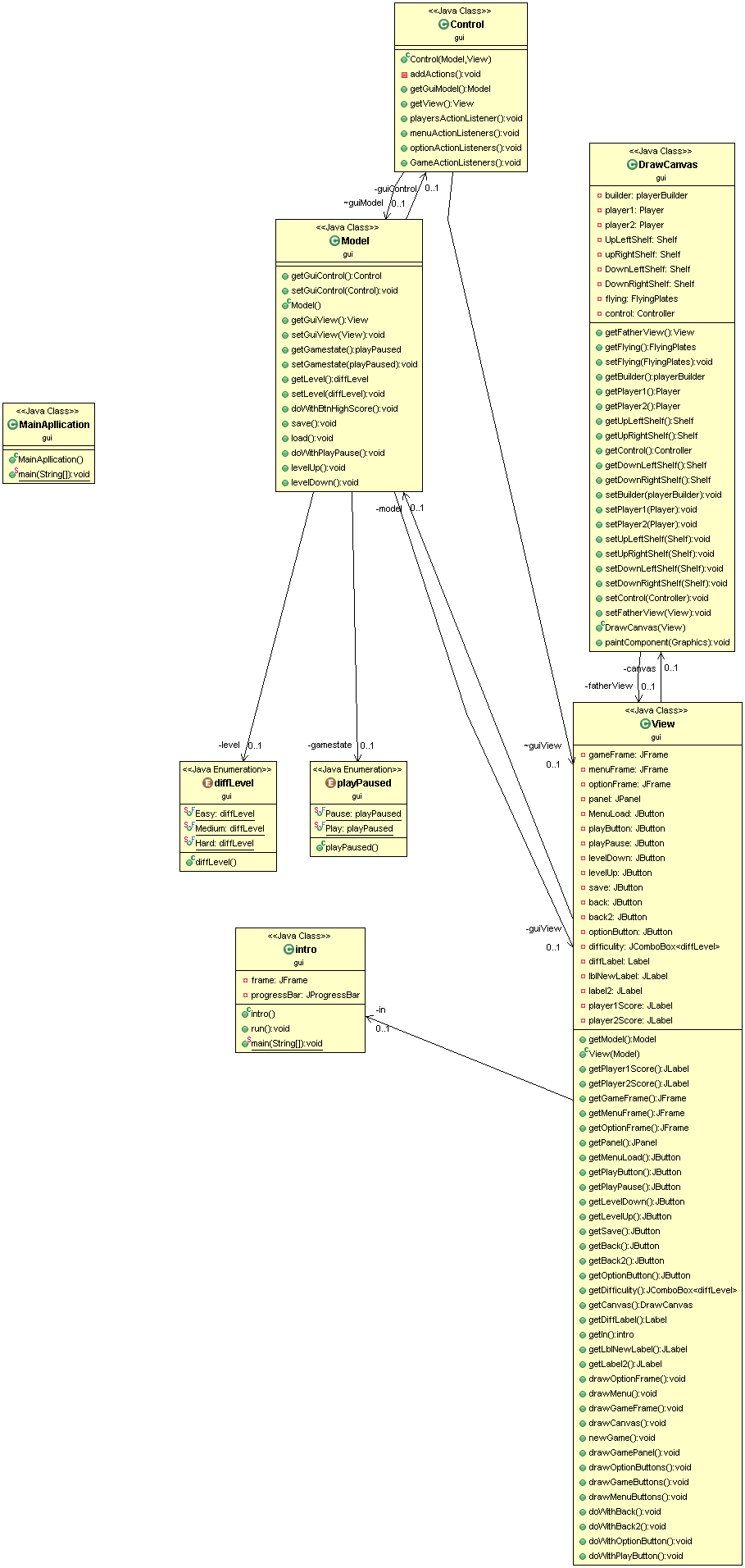
Factory :



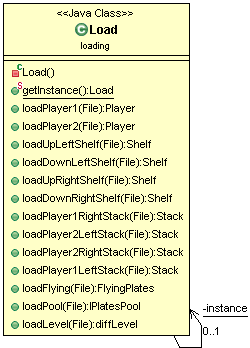
Files:



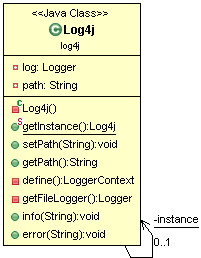
Gui:



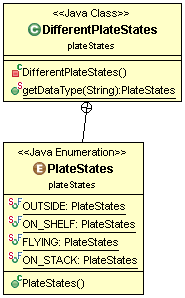
Loading:



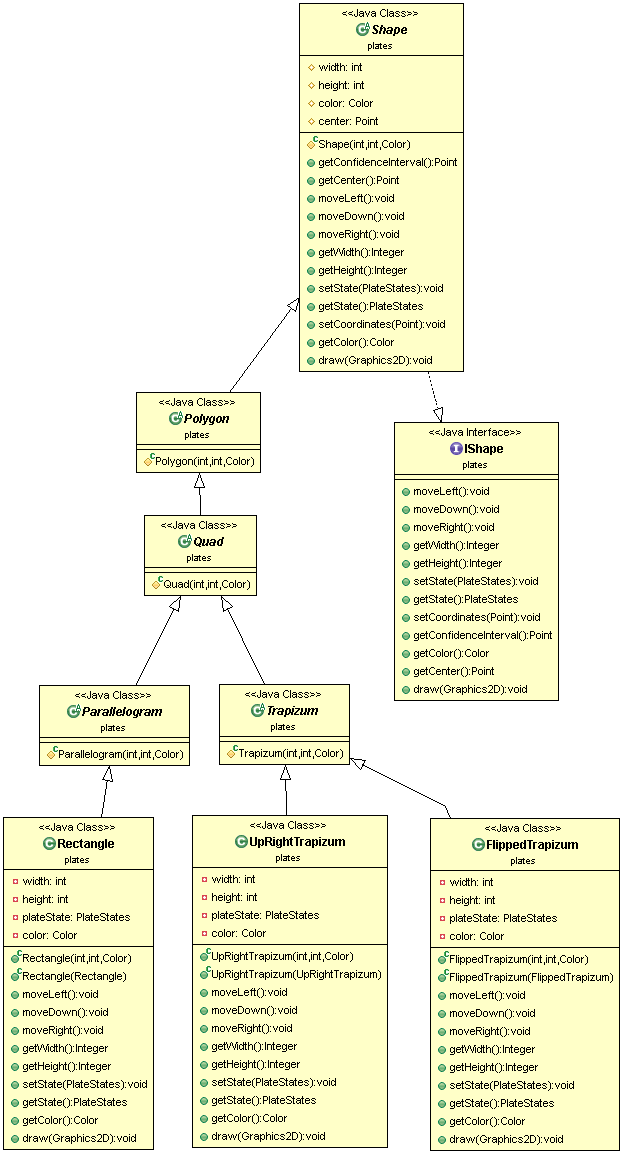
Log :



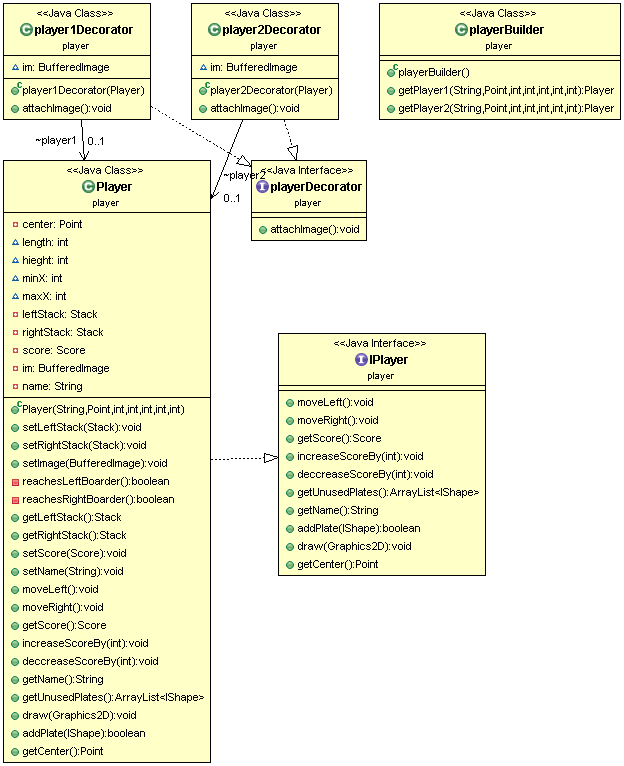
Plates states:



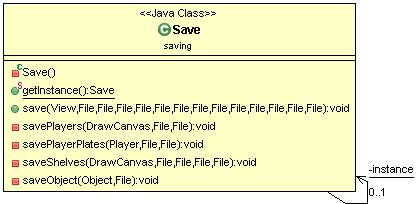
Platestypes:



Player :



Save:



Score:

