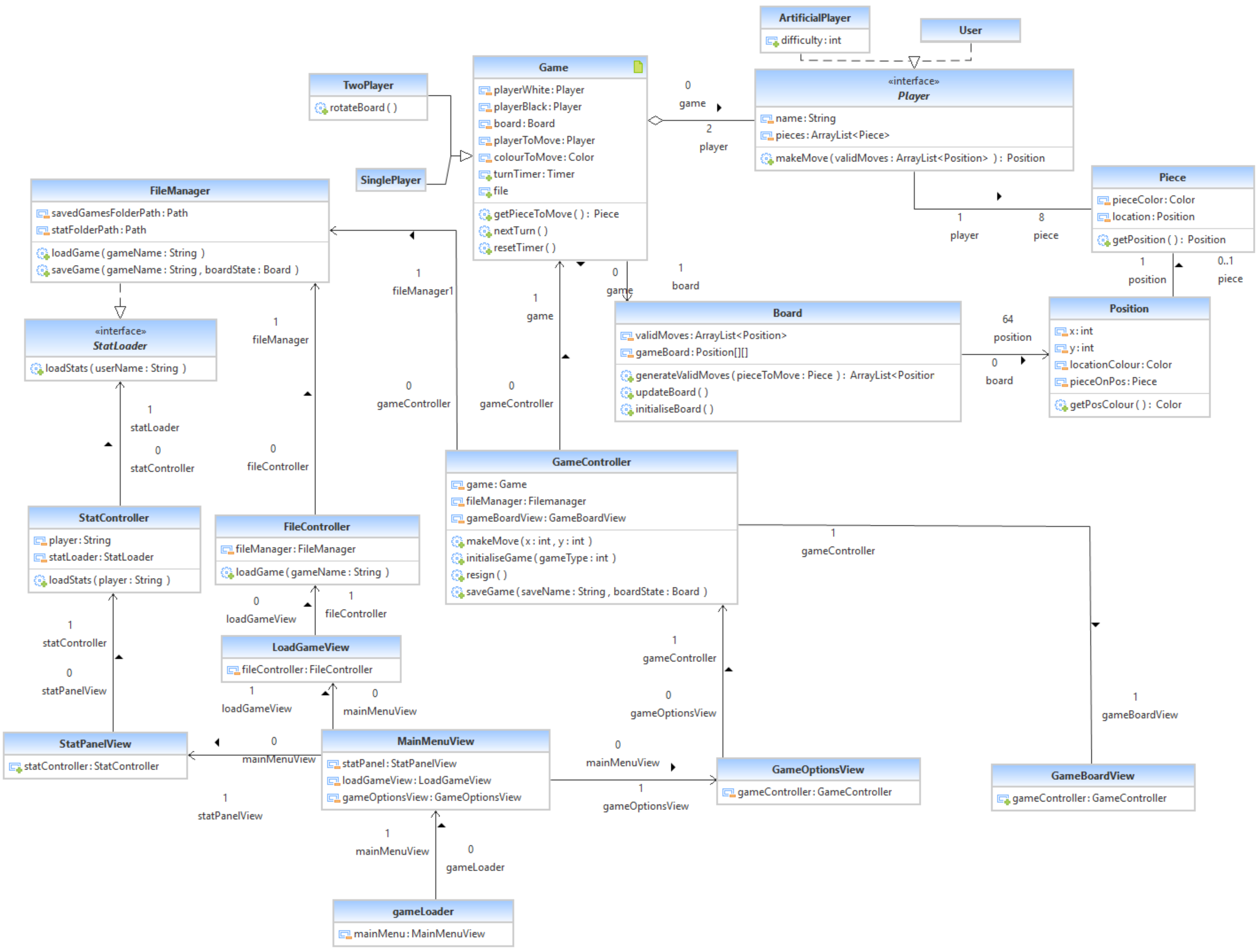

Our main idea behind this class structure was to create an initial model-view-controller design. We chose MVC because it allows easier modification than alternative designs.

First the model we came up with included these main classes: Game, Board, Piece, Position, ArtificialPlayer, User, SinglePlayer, TwoPlayer. And the Interface: Player

We later realised we needed StatLoader and FileManager for more specific tasks to do with load/save game and loadStats.

We decided to make SinglePlayer and TwoPlayer extend game because we found there was a lot of shared functionality. The different game modes meant that they will each have a different implementation of what game holds (e.g. TwoPlayer will take advantage of the Timer during the playing of a game while SinglePlayer might just use it for stats)

AI and User classes implement Player as these two will have to implement the same methods (primarily makeMove()) where the User class with prompt the human player for input and the AI class will determine the best move to make.

We decided it was best for the modularity to have multiple views and controllers. This meant that only the class with the desired action is triggered. The multiple Views also makes it easier to split the code and not have all the JPanel’s functionality in one class.