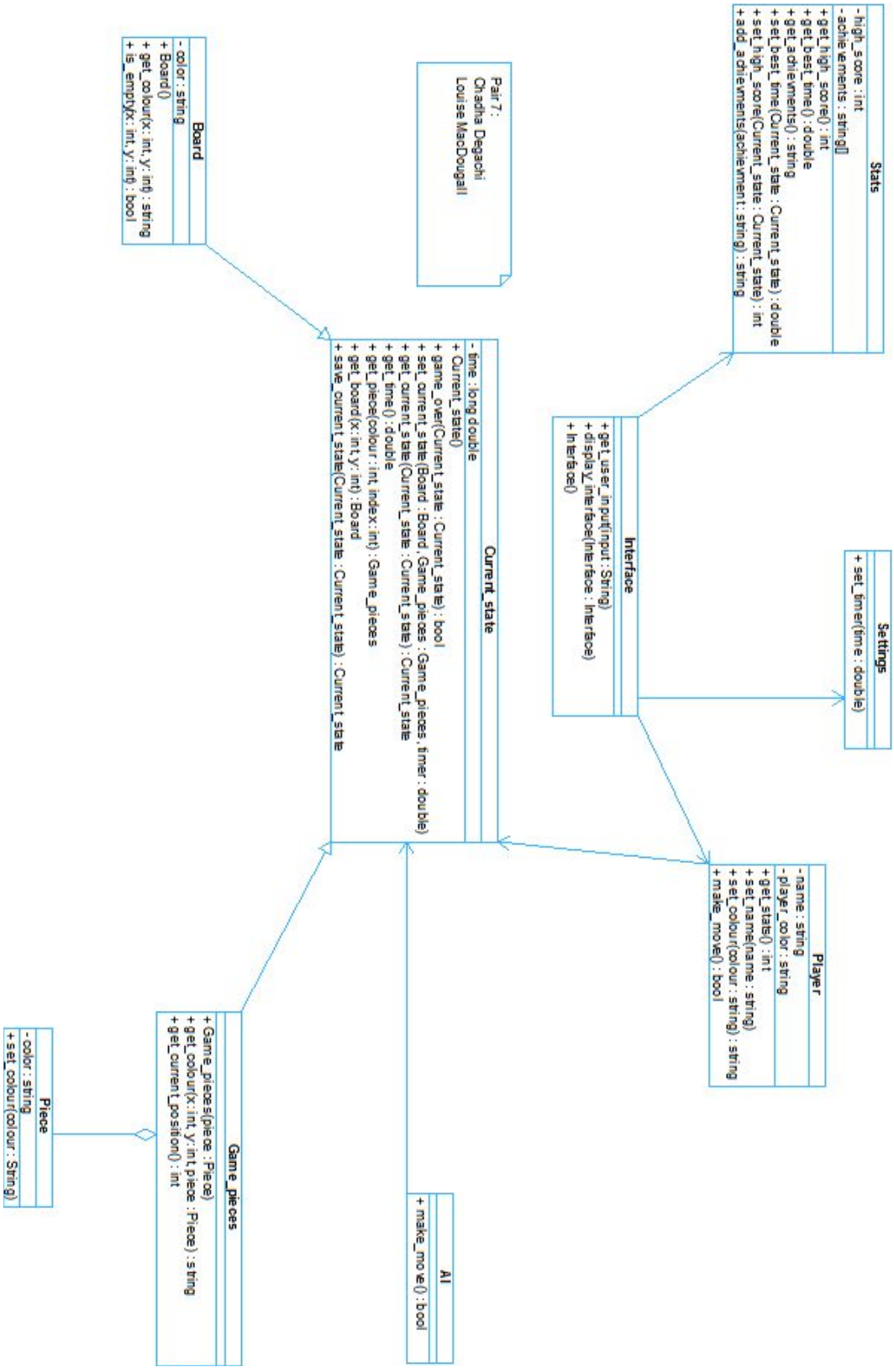
Class Diagram: Kamisado



We chose this structure as we believe that it best represents how the game user would interact with the game, while also keeping the interactions between different components within the system as simple as possible. By inspecting the design requirements - specifically the Use Case diagram - we identified the following sub-systems within the overall architecture:

- Current State
- Interface
- Player
- Stats

Where, for example, the Stats sub-system will implement the View High Scores use case, the Current State sub-system will implement the Save and Resign use case, and so on.

With this design, we present fairly isolated components which allow the system flexibility and scalability, and further present a system which is easy to maintain and extend. While initially challenging, we were eventually capable of reworking our diagram so that low level classes such as board and pieces would never need to interact with the more high level classes, such as interface - hence, achieving the desired loose coupling.

The design created is, in itself, fairly self-explanatory in terms of showcasing the various interactions between the different components of the system, hence making it simple to implement.