The implemented version of the Game Server has the same “basic” structure as the planned class diagram. What’s important to note that, based on the first feedback, server class diagram was not complete, for example it misses most of the business rules altogether and uses custom data classes instead of the network provided ones. With that said, implemented version does not have separate Player class, but instead makes use of provided PlayerState, same goes for Coordinates class. Game class has same structure but instead of separating players and maps into different variables implementation uses Lists. Separate classes were also made to check business rules of players HalfMaps and overall game rules which are missing in the diagram. The other main thing that could not be found in initial planning but was implemented is exceptions package and all the created exception that go with it.

In the context of adding players and their half maps to the game class, given that only 2 players or maps could be in the game, we have chosen using lists of those players and maps and neglected separate variables for them to achieve scalability, simple iteration, and uncluttering of class parameters, at the cost of use simplicity because of code overcomplication and unreadability.

In the context of making many separate exceptions, given that server will probably receive game clients with many issues, we have chosen to create separatee exceptions for each rule and check, and neglected using a single one to achieve correct exception handling, at the cost of time consumption of creating new exception for each rule and check because of code readability and understandability.

In the context of not using Player class as was planned in class diagram, given already provided network classes, we have chosen to use PlayerState class provided by the course and neglected creating a separate internal class to achieve course requirements and simplicity, at the cost of not implementing server as per planning because of time consumption to implement additional converters for a single class.

In the context of having 2 full maps stored inside 1 game, given different map view for each player, we have chosen to have 2 different views of the same full map for each player and neglected having only one stored map inside game class to achieve simplicity of sending correct map to different players, at the cost of using list for full map storage and double map changes tracking because of complexities changing single map for each players game state request.