**Programming Report**

Martin Stoev

Group Green3.13

**Overall Design**

**Target Audience**

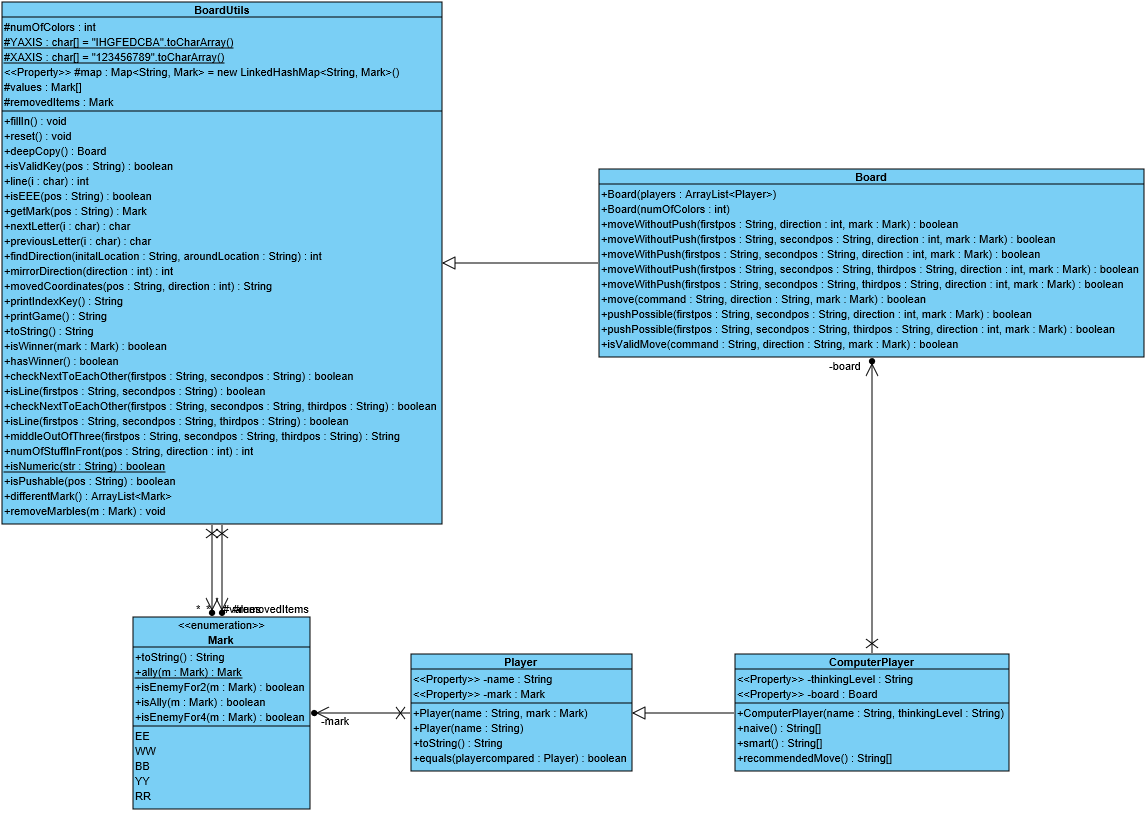
This document is meant for future software developers to understand all the functionalities/features I have implemented in my program. To allow developers develop my application further in the future, I have kept in mind to make the game methods understandable and also constructed them in a way such that it can be expanded more.

**Discussion of Overall Design**

The class diagram for the project was made long before I started the actual coding and was changed many times but always remained to be a very useful means to develop my game.

. The original idea was the board class to be the main and only source of movement control but after the class grew over 600 lines, I decided to split it in two, having my moves in the Board and all other methods that are required to be in another class called BoardUtils. The Marbles were in an enum called Mark, the Player class which is the representation of the client who uses the application and the ComputerPlayer who takes the role of a bot.

The protocols Group Green 3 made was by far the most important element of the project as it allows communication between other clients and servers to be fluid with no miscommunication.



**Server**

While the package with the board takes care of the logical part of the game, the client and server packages take care of the networking part.

When the server is started a port number should be given in order for it to run. If a used port is given, an error message will be displayed and then the person running the server will be asked for another port.

The server supports multiple clients and multiple games at the same time due to its multithreaded nature.

The server has a ServerTUI class that ensures the server owner that all commands that are used are shown.

The Server follows the protocol strictly and sends the right messages based on the client’s input. This was tested during the bot competition.

**Client**

The ClientTUI provides a readable interface which begins by asking for the Server IP address and the port number and if given a invalid IP and port asks again.

The Client support multiple users and also provides a bot option which has two levels of intelligence which can be chosen by the client.

The client also provides a hint option which can be accessed by a specific command and will give a recommended move.

After the game is finished, the party leader has the option to start a new game.

If a person disconnects or his client crashes, the other players are informed and the protocol is executed. The disconnected player is presented with the option to reconnect to the server.

The moment the Server crashes the player closes smoothly.

The client does everything according to protocol.

**Global Requirements**

The Client and Server have their game states synchronized.

All user input is validated and handled accordingly. If there is an error in connecting, the ClientTUI repeatedly asks for input until a correct one is used.

The application follows the typical MVC pattern by having a ClientHandler, Board, Room and Server as my Model, ServerTUI and ClientTUI as both View and Controller

All errors and exceptions are handled properly in a way that signals the client what is happening and does not break the client or server.

**Exceptions**

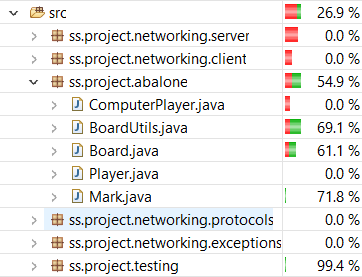
I defined the certain exceptions (ExitProgram, PortNotAvailableException, ProtocolException, ServerUnavailableException). However, I still had to use predefined java exceptions such as IOException.

**Testing**

Multiple tests were made to check if the board display the right marbles depending on the player count, if moves are possible and if pushing marbles is possible.

**Unit testing**

In my Junit I test the movement of marbles for 2 and 3 people game and a 4 people ally move. I also check if, when made, the board is filled with the right marbles for a 2 people game. During testing I saw the when an ally move is made on two different enemy marbles which are from different color the enemy colors switch which led to an uneven amount of marbles on the board. After I saw this issue, I fixed it and tried several different moves. For better or worse, I did not discover more errors to fix. The coverage of the test for the entire project is approximately 27% with BoardUtils and Board being the largest percentage coverage (69.1% and 61.1% respectively) as they are the most complex classes used to test as part of my Junit tests.



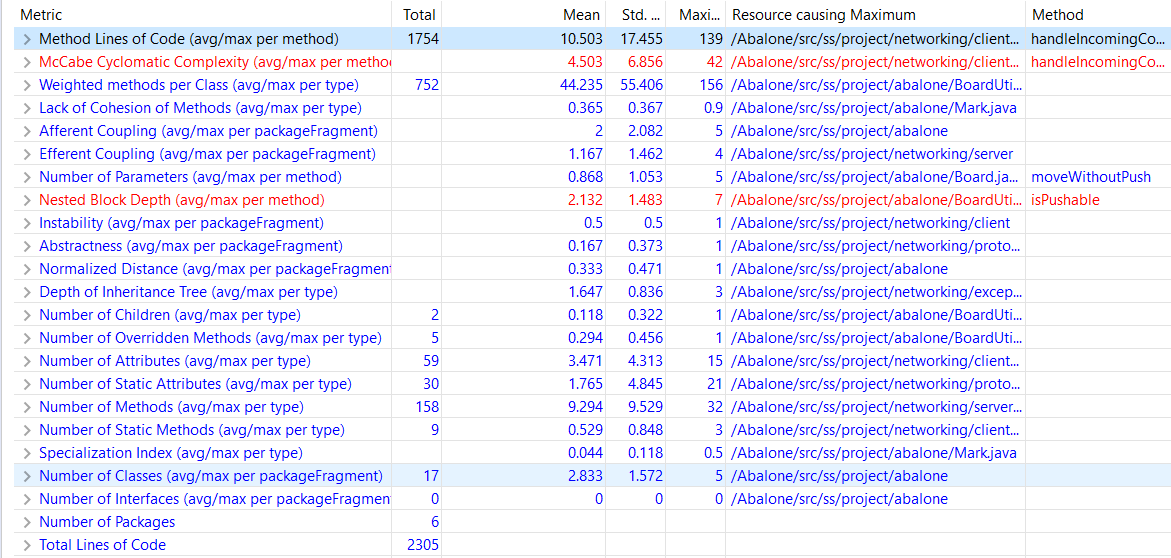
**System testing**

System testing was by far the most frequent way I have done my testing while developing the program. Especially when implementing functional requirements on the server and client side, I had to do a lot of system testing.

|  |
| --- |
| System testing |
| Functional Requirement 1- Validating connection with server and client |
| Expected result: User is asked to enter input again until server/client validates it |
|  |

|  |
| --- |
| System testing |
| Functional Requirement 2- Ensuring a client disconnecting does not disrupt server or other clients |
| Expected result: The client disconnect should not affect a game in all types of game modes and the server must acknowledge this by removing all instances of the client from the room and server. |
|  |

**Metrics**



**Weighted Methods per Class (WMC):** My BoardUtils class had the highest WMC (156) which is very high. This makes it difficult for future developers to understand my code.

**Cyclomatic Complexity (CC):** 4503 was the mean value of the CC in project. For future project I should ensure that the complexity of the project is lower

**Lack of Cohesion of Methods (LCOM):** The average LCOM is found to be 0.365 which means the project is cohesive and well balanced.

**Reflection on Planning**

During the Design Project we did not have the coordination and understandability of the big picture of the tasks that were supposed to be performed by us which led to doing everything in the last minute and being sloppy. That led to me deciding to start the project during the Christmas break and finishing the board and moves classes and planning everything in advance.

Something that made me deviate from the planning was the fact that Networking took me too much to understand and the separation of my partner led to progress becoming slower. After a significant amount of time, I understood how Networking worked and implemented it in the short time I had before the bot competition. The fact that I was alone allowed me to work more on the bonuses. I did the challenges, texting in the room, leaderboard (I could not implement the time of wins or statistics, I did it because the game did not feel complete without it) and attempted to make a GUI but run into the problem of multithreading in javafx.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Task |  |  |  |  |  |  |
|  | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday | Sunday |
| Week 7 | - | - | Protocols | Protocols | Creating class diagram | Design | Split up tasks |
| Week 8 | Implementation of Game | Testing Game | Testing Game | Documenting Code of Game | Implementation of Protocol | - | Testing protocol |
| Week 9 | Implementation of Game | Documentation | Implementation of Game | Testing Game(Beta testing) | Final changes and documentation of code | Final changes | Report |
| Week 10 | Repairs | Repairs | Submission | - | - | - | - |

As countermeasures I worked a lot everyday and asked friend for help the moment and interesting errors show up (Index out of bound exception for a scanner).

For the next project I will make sure that I understand the material well enough in order to not lose time. This can be preventing be studying more and asking friends for help but earlier than I did this time.

Do’s and Don’ts:

Do: finish the logical part of the project during the vacation and share work properly

Don’t: waste time to understand something complex just ask a friend and ask other people for their progress and approaches in order to get more fresh ideas and improve your project.