AGENDA

At first, we had no idea which drawing software we should use. We searched all the interne, and then found an Eclipse plug-in named UMLet. It’s been very useful and has many options for drawing various classes, object diagrams.

We created the class diagram. In the beginning, the class named FileWriter was connected to the class named Gameplay and changed its name to FileWriterAndReader. We tied it to the Game class instead. We expect more efficiency between Game and GamePlay by this way. Player was tied to GamePlay. However, we thought that it wouldn’t have made sense in our class diagram. We rather kept it alone.

We created the interaction diagram. We were worried about where Exit should be gone. We thought that it should be gone to Menu class, as it was introduced in our earlier UML class narratives accordingly. Earlier, we have had hard times drawing the diagram of PlayGame. We have questions that need to be answered about that by TA. For Gameplay, we weren’t sure which class whom and when. It would have been called when swapping occurs or when the player was joined the game at the beginning. We are still working on it to improve our diagram. For the isSwappable method, we couldn’t understand if it’s false. Because, we know that swap is called when isSwappable is true. But, when it’s false, we created a way that we should first call unselectlokums then selectlokum again. By that way, we have lots of methods piled in our two main classes; Board and Gameplay. We have hard times connecting and calling the specific methods under crucial circumstances as explained earlier. We also found out that destroy method must be in Gameplay class, because it’s involved in many other methods that stay in Gameplay class. We also make sure that if isSwappable is true, we get to check the row and the column, later on check the method isColorMatched under the condition of three of more same lokums in a row. We are still having problems with creating our GUI plan for the future application. Thus, we have to consider creating our diagrams by thinking of our future brainstorms now. We are basically inspired by our future explanations.

Finally, we decided that our main class diagram has difficulties for adding and upgrading new features and extensions, but our alternative class diagram provides conveniences for adding and upgrading new features and extensions. Hereby, we selected alternative class diagram for our design and added it some adapters which made our design more optimized.