Alternatives and Discussion

Our final class diagram took form by passing three main processes. At the first processes, we created very basic class diagram as shown below. This class diagram was created based on system sequence diagram, and domain model.

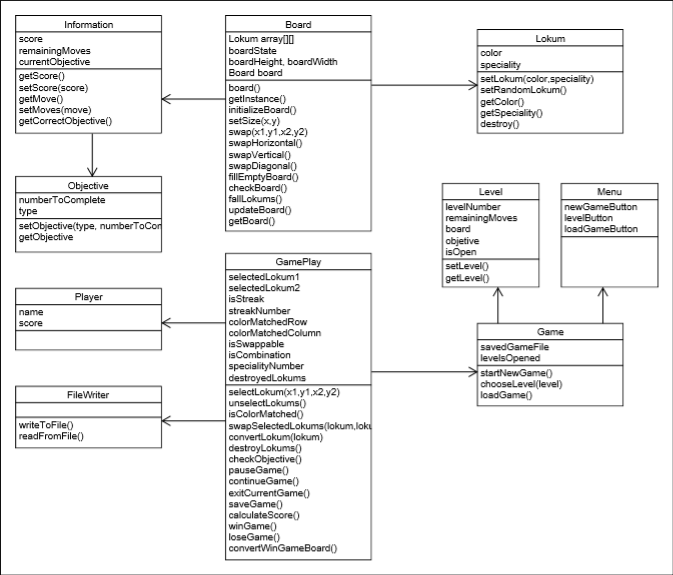


Figure 1: First Class Diagram

At the second process, we enhanced the design and created an alternative class diagram. Enhanced version of first class diagram is shown below.

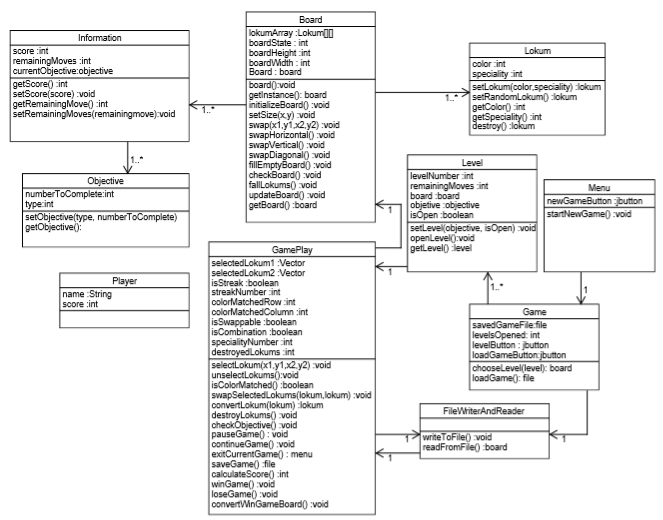


Figure 2: Enhanced Version of the First Class Diagram

At this design, we have a lokum class that enables every instance to store its specialty which is normal, stripped, bomb or chocobomb, and its color. We have board class that has an array of lokums and related methods. We have level class that has its own unique board and objective variables. We have objective class that holds the information to complete the level. We have gameplay class that plays a role as a controller. We have filewriterandreader class that provides a game to be saved or loaded from file.

For an alternative design, we created separate classes for all kind of lokums. At the project meeting, we decided it is better to have separate classes for all kind of lokums for extensions like adding new type of lokum. Therefore, we decided to use our alternative class diagram as the main class diagram, and make this design even easier to add extensions.

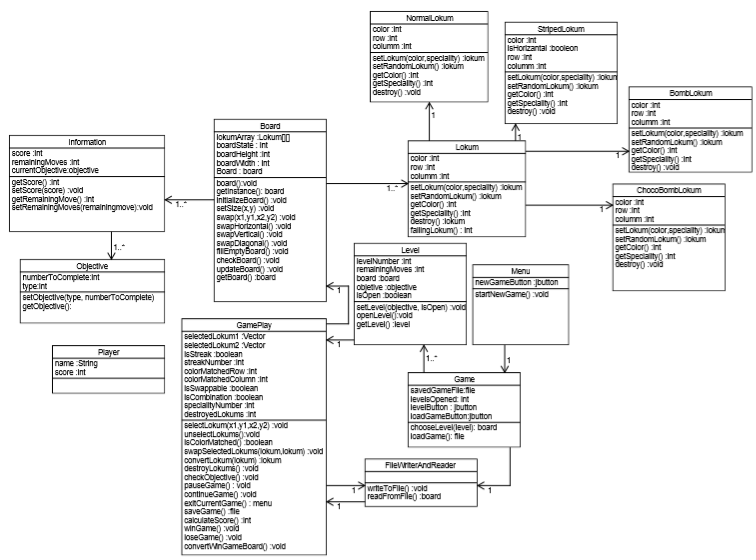


Figure 3: Alternative Class Diagram

At the third and last process, we decided to use our alternative class design as the main class design. At this design, every kind of lokum has its own class, and we have adapters to optimize the design. With these adapters, we aimed to simplify design and provided to ease adding new extensions. In this way, we achieved our final class diagram which can be seen below.

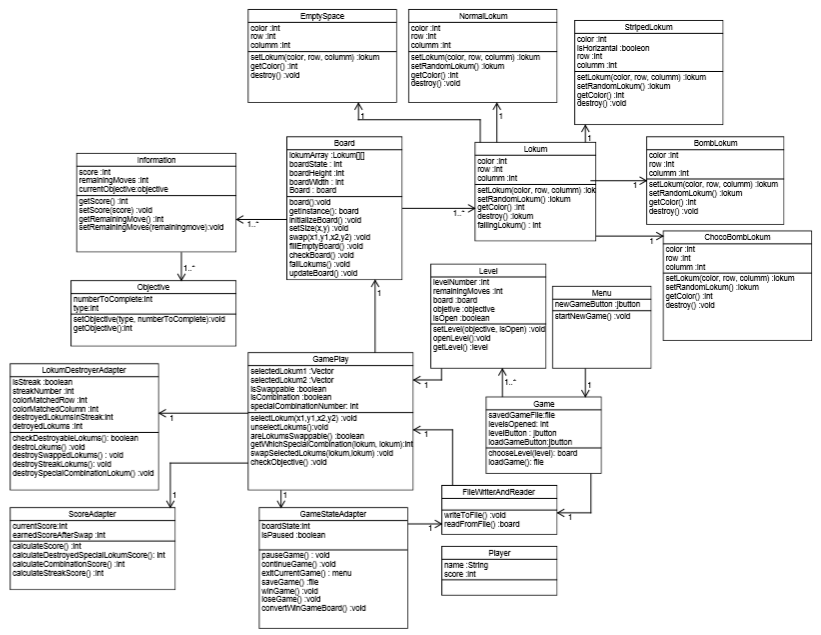


Figure 4: Our Final Class Diagram