CSCI 4448
Team Members: Serena Evans-Lutterodt, Garrett Hempy, Fallyn Logan
Date: 12/4/2020

Project 6: Funi

Project Video Link (also in ReadMe on Github): https://youtu.be/jFdkYWE7nr0

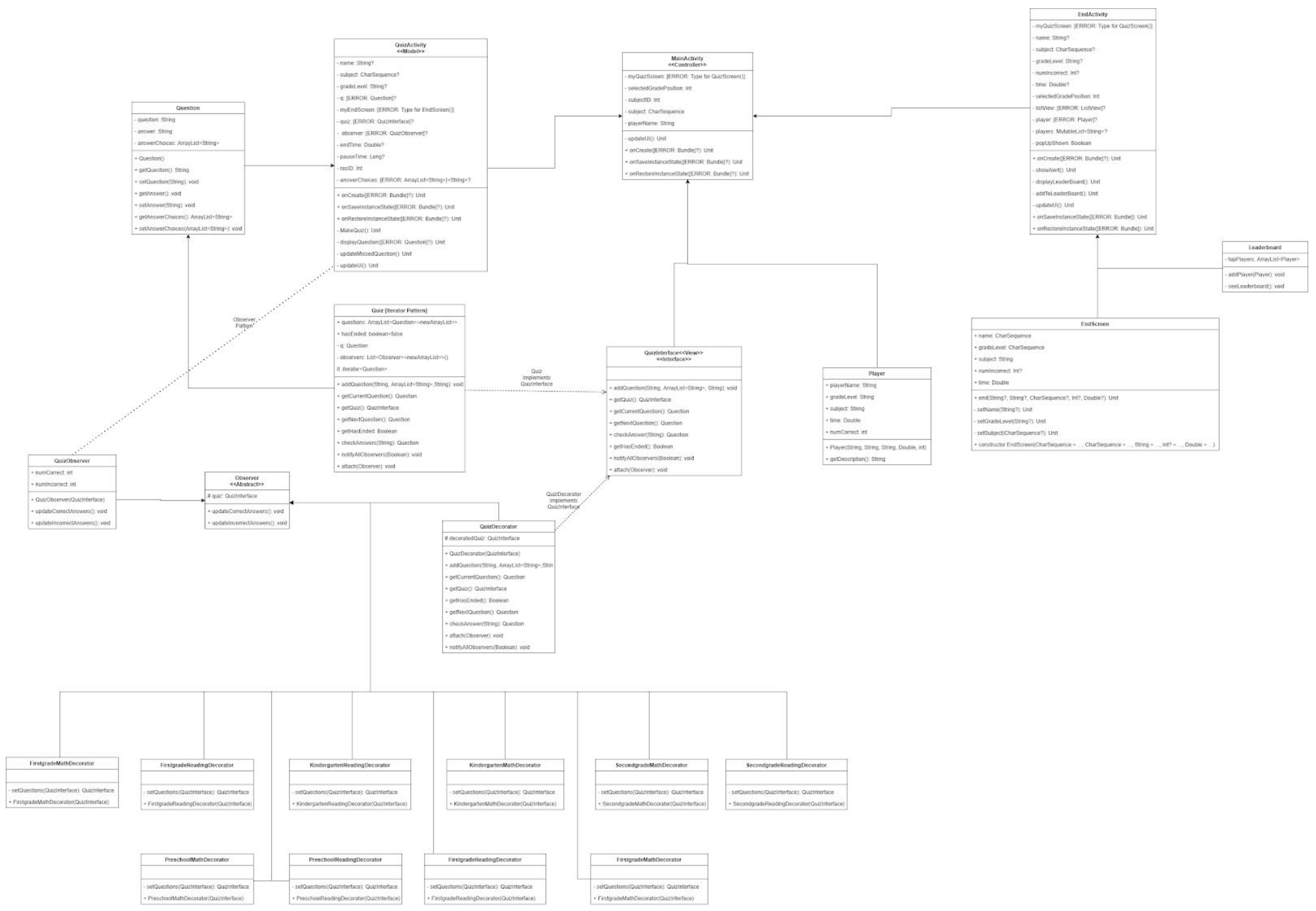
Final State Statement:

The Funi application is an educational app for grades preschool through third grade, to encourage fun ways to learn basic spelling and math. The system asks the user for their name, grade level, and subject they would like to attempt, either reading/grammar or math. Once the user submits their credentials, the quiz displays on the screen based on the chosen grade level and subject. The user gets 10-12 questions and the system displays one question at a time until the user either gets three questions wrong or the subject quiz runs out of questions. If the user never answers three questions incorrectly, and the questions run out for that grade level, their name and time spent on the quiz can be added to the leaderboard. Lastly, after the user finishes a level, they can be displayed on the leaderboard and are asked if they would like to Try Again, Switch Subject, or to Start Over.

Since Project 4, we also implemented five different design patterns. The Observer pattern is used to keep track of the questions that were either answered correctly or incorrectly. The Decorator was used to set the different questions for each grade for each subject. The MVC pattern was used to implement the different activities/ screen views. Two smaller patterns were also implemented in smaller parts of the program: Iterator was also implemented to iterate through the Questions array as well as Builder pattern for altering dialog.

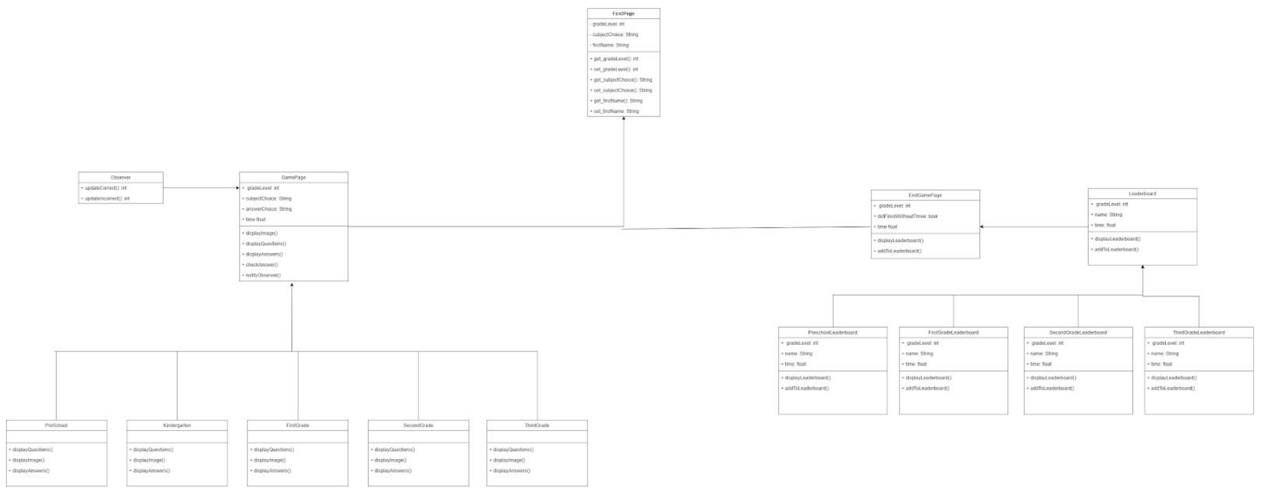
After working through Project 5, the two features that were not implemented in the final state of the project were the Strategy pattern for the quiz implementation and the style of the quizzes. After our check in with the TA, we realized that the strategy pattern for implementing all the quizzes was not the best nor efficient implementation choice: It caused too much extra work that was not needed. The transition from Strategy pattern to Decorator was a very easy and quick change for us that indeed cleaned up our code. We also altered the style of our quizzes as we shifted our focus more towards the reading section to a matching-spelling section for Reading where our initial “sentences” question prompt was changed to just having images representing the “sentences” plan from Project 4. The quantity of the questions also decreased because 30 questions per quiz per topic was not feasible in the time frame we had because we had to create our own images in Draw.io on top of coming up with creative yet different variations for each word per question for every quiz. Lastly, we decided to get rid of the leaderboard object, and combine it with our end activity. Overall, we stuck to our initial plan from the beginning and made slight changes throughout the process to better our code, inspired by feedback and arising errors that we worked through; many of which were caused by Android Studio itself. .

Final Class Diagram:

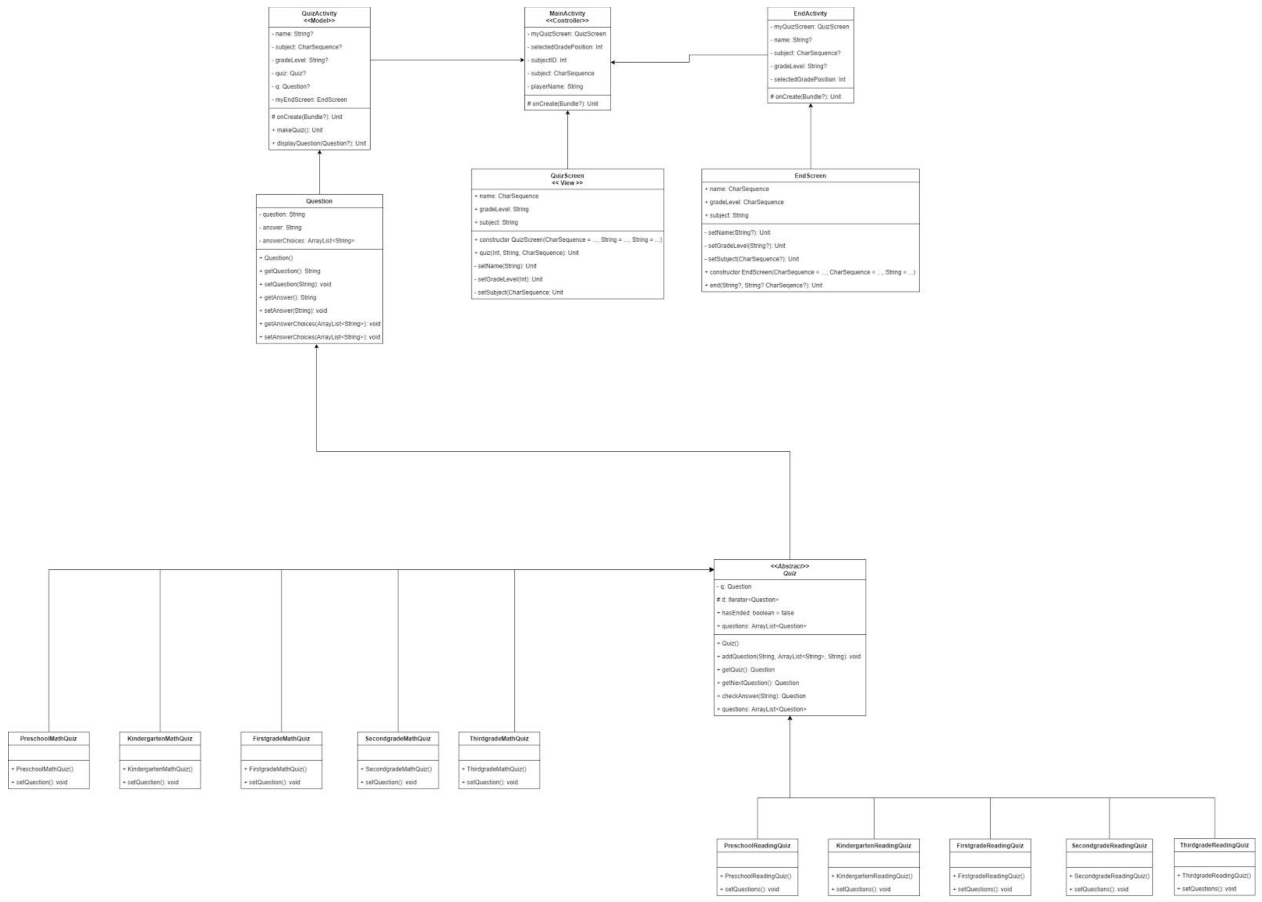

QuizActivity <<Model>>
MainActivity <<Controller>>
EndActivity
Question
Leaderboard
Quiz (Iterator Pattern)
Observer Pattern
Quiz implements QuizInterface
QuizInterface<<View>> <<Interface>>
Player
EndScreen
QuizObserver
Observer <<Abstract>>
QuizDecorator implements QuizInterface
QuizDecorator
FirstgradeMathDecorator
FirstgradeReadingDecorator
KindergartenReadingDecorator
KindergartenMathDecorator
SecondgradeMathDecorator
SecondgradeReadingDecorator
PreschoolMathDecorator
PreschoolReadingDecorator
FirstgradeReadingDecorator
FirstgradeMathDecorator

Comparison Diagrams:

Project 4 UML Diagram:



Project 5 UML Diagram:



Comparison Diagram Statement:

In Project 4, we defined our overall project objective, project requirements, and project planning tools for planning the architecture and features for the project. The objective of the project was to create an educational Android application for the elementary years, grades preschool to third grade, to have a creative opportunity to learn basic math and reading skills according to the “player” or student’s respective grade level. At this point in the process, there was minimal code implemented, but the UI mockups, code planning diagram such as the architecture diagram, activity, use-case, and class uml diagrams were completed.

By Project 5, there was great progress regarding actual code implementation as the MVC had been implemented for all three screens, which includes the start, quiz, and end screens; Iterator was implemented to iterate through the questions data structure for each quiz; and finally, created all the witten questions with their respective answers for each reading and math quiz. The initial implementation of the quizzes was using the strategy pattern, which was a lot of extra work, was changed to decorator where each quiz will be created based on the user input from the start screen that is within the quiz screen file.

For project 6, the Funi application had been created with the slight changes we had to make along the way. The last two design patterns, observer and builder, were successfully added. The ‘Try Again, Switch Subject, and Start Over’ features were debugged and a few of the quiz questions with their corresponding answers had to be changed, as finding the question-prompting icons resulted in unclear answers. Although some words like ‘balance’ or ‘laughing’ would be great words to know, they were hard to find pictures that properly depicted the words the student has to learn.

Third-Party code vs. Original code Statement:
The use of 3rd party code was implemented in this project for the chronometer widget on the quiz screen
Tutorials used:

- https://abhiandroid.com/ui/chronometer
- https://stackoverflow.com/questions/526524/android-get-time-of-chronometer-widget

Along with the listview widget for the end screen leaderboard

- https://www.youtube.com/watch?v=5PJ6GZHs1bQ

Also, the AlertDialog was implemented with the help of this tutorial

- https://www.tutorialspoint.com/android/android_alert_dialoges.htm

The rest of the code included in the project is original code learned from either *CSCI 4448* or *ATLS 4120*

OOAD process for Semester Project Statement:
Patterns used:

- Observer (keeps track of questions missed/answered correctly/score)
- MVC (for different activities)
- Decorator (setting the different questions for each grade/subject)

(these two are smaller in implementation than the rest ie like just one or two lines of code instead of entire classes)

- Iterator (iterating through Questions array)
- Builder (for alert dialog)