|  |  |
| --- | --- |
| Phase 1 Design – MODULAR STUDY APPLICATION | ABSTRACT  A Project Design for the development of a Modular Study Application for students. The design outlines the interface and functionality, application structure, build instructions and UML Diagram for the application.  Kory Bennett / Brock Allton / Dylan Sawyer / Justin Casey / Dan Page / Cale Ward  UMUC 495 6380 10 Feb 2020 |

TABLE OF CONTENTS

|  |  |
| --- | --- |
| Revision History | 2 |
| 1. Key Functionality | 3 |
| 2. Application Structure | 3 |
| 3. Build Instructions | 4 |
| 4. UML Diagram | 5 |
| 5. Milestones | 6 |
| 6. Progress | 6 |

Revision History

|  |  |  |
| --- | --- | --- |
| Date | Description | Author |
| 10 Feb 2020 | Initial Draft | K. Bennett |
| 14 Feb 2020 | Review | B. Allton |
| 15 Feb 2020 | Milestones / Progress | D. Sawyer |
|  |  |  |
| 15 Feb 2020 | Final Review / Formatting | K. Bennett |
|  |  |  |
|  |  |  |

Phase 1 Design

1. **Key Functionality**

Team Modular Study created a Graphic User Interface (GUI) software that allows a user to access a database with questions and be tested with multiple choice options to answer. A score is provided to tell the user the success of the test. The application works by allowing the user to select a course topic and importing those questions from a database, displaying them for the user to be tested on in a random order. Functionality includes:

* A drop-down pane that holds the subjects that a User can be tested on
* An area for the User to input how many questions they want to be tested on
* Multiple choice questions in random order with four options the User can select from
* The results of the test shown
* An option for the User at the end of the test to exit or restart the application

1. **Application Structure**

**Main.java**

* Started by the user
* Initializes and handles GUI.java
* Queries Maria DB for above information
* Parses database results for optimal viewing in GUI
* Receives requests from GUI.java
* Sends results to GUI.java
* Serves as primary interface to database
* Responsible for randomizing questions
* Tracks questions completed out of total
* Tracks number of correct answers out of total

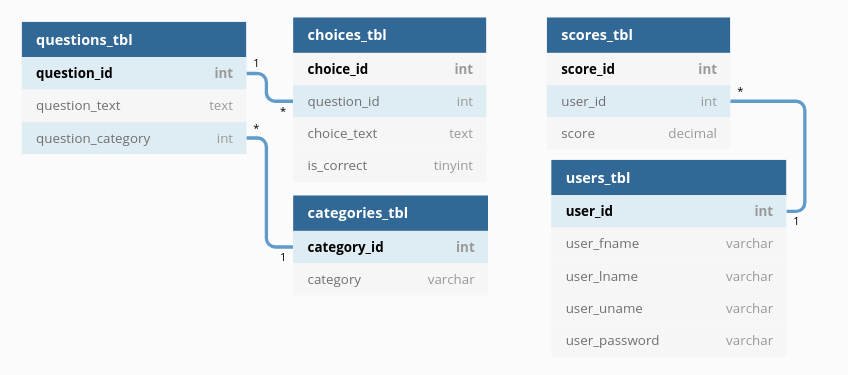
**GUI.java**

* Contains JFrame user interface
* Accepts user input via button and text
* Outputs questions, answers and results to user
* Sends requests to Main.java
  + Available Databases
  + Number of Questions within each database
  + Individual Questions
* Displays resulting scores to user

**SQL**

* Stores all subject and question information
* Only interfaced by MiddleMan.java
* Hosted remotely to prevent setup complicated database set up

Entity Relationship Diagram



1. **Build Instructions**

**cd into the ‘Code’ directory**

* 1. **Compile your Java code, generating class files**

*javac -cp “.:./mysql-connector-java-8.0.19.jar” <ALL\_JAVA\_FILES>*

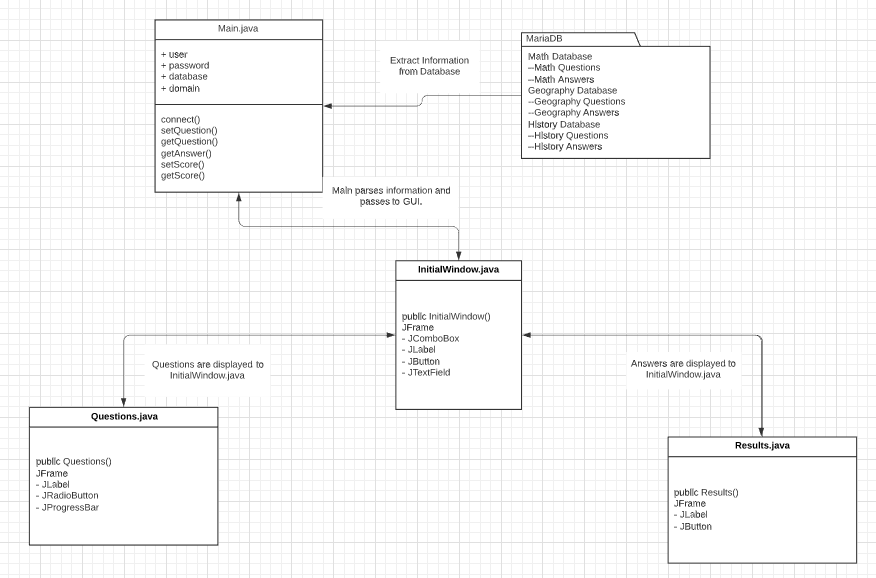
* 1. **Create a JAR file called study**

*jar cmf manifest study <ALL\_CLASS\_FILES>*

* 1. **Execute the JAR file**

*java -jar study*

1. **UML Diagrams**

****

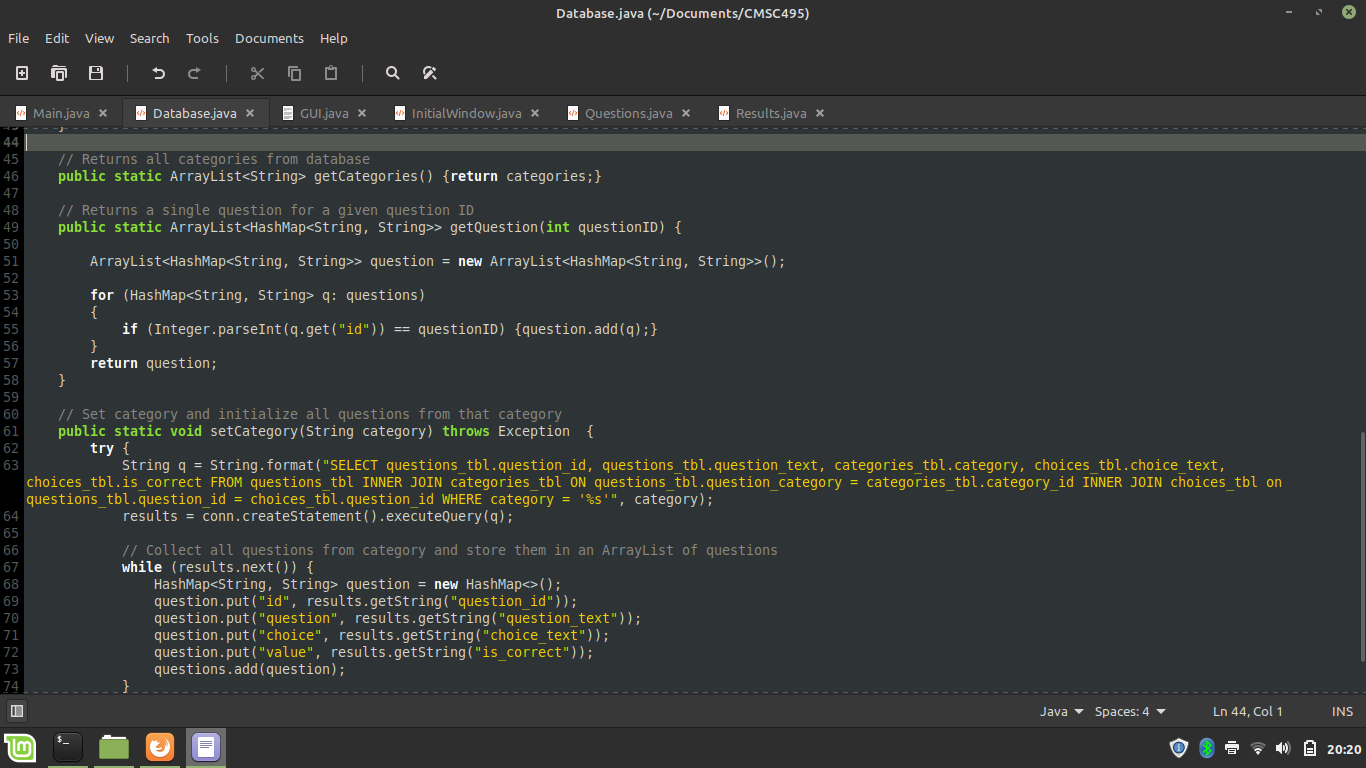
1. **Milestones**

For the Phase 1 Design week the milestones were to have the GUI of the program function developed to the point of being able to launch the program and select the subject to test on, choose the number of questions to be tested on, and progress through the question database with a score result displayed once all questions have been answered and submitted.

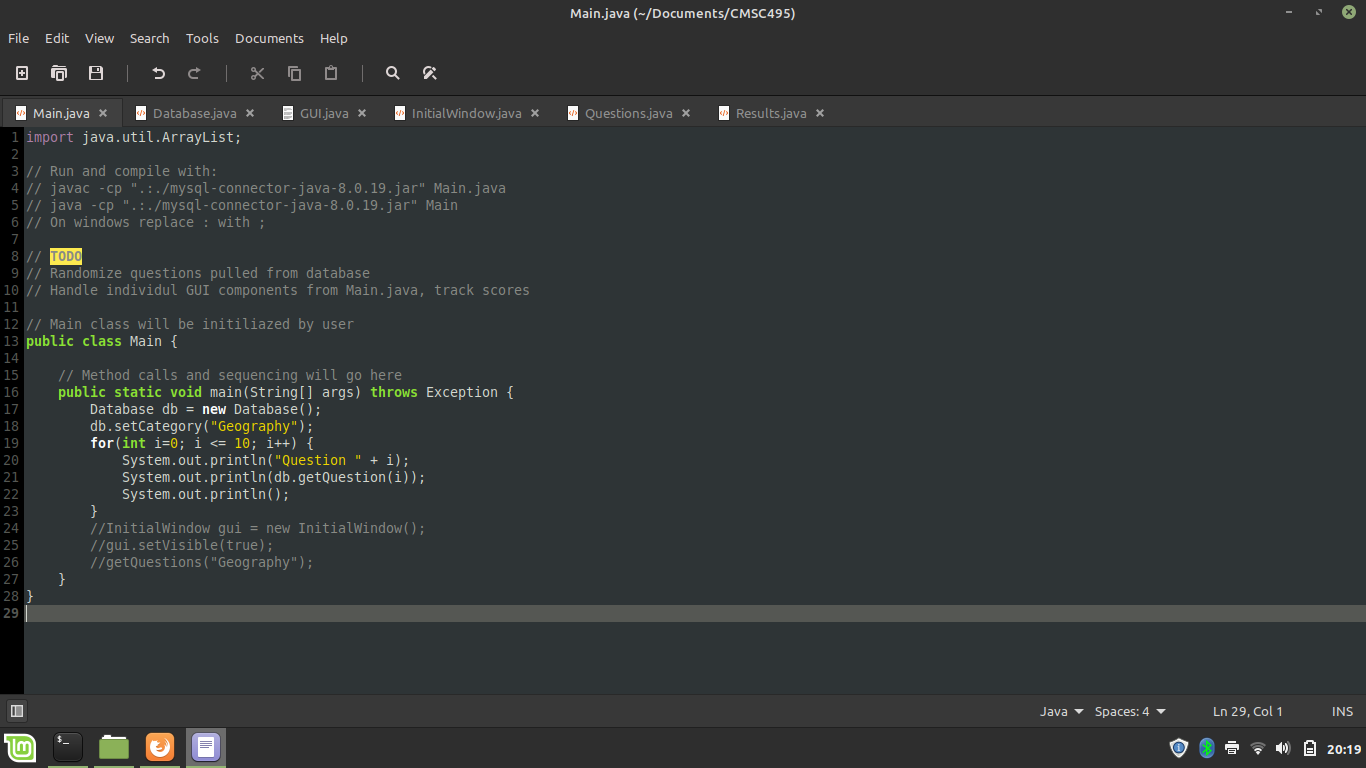
This team has been performing on task and on schedule since the first week of class. The coding of the project is progressing at an acceptable rate. A few re-evaluations have been noted with exceptional turnaround each time. The overall team dynamic is good with good communications throughout the week vie Hangouts chat, and a standing video teleconference (VTC) every Monday and Saturday. During the VTC, goals for the week are discussed with expectations set for the end of the week with any last-minute changes being discussed on Saturdays.

1. **Progress**

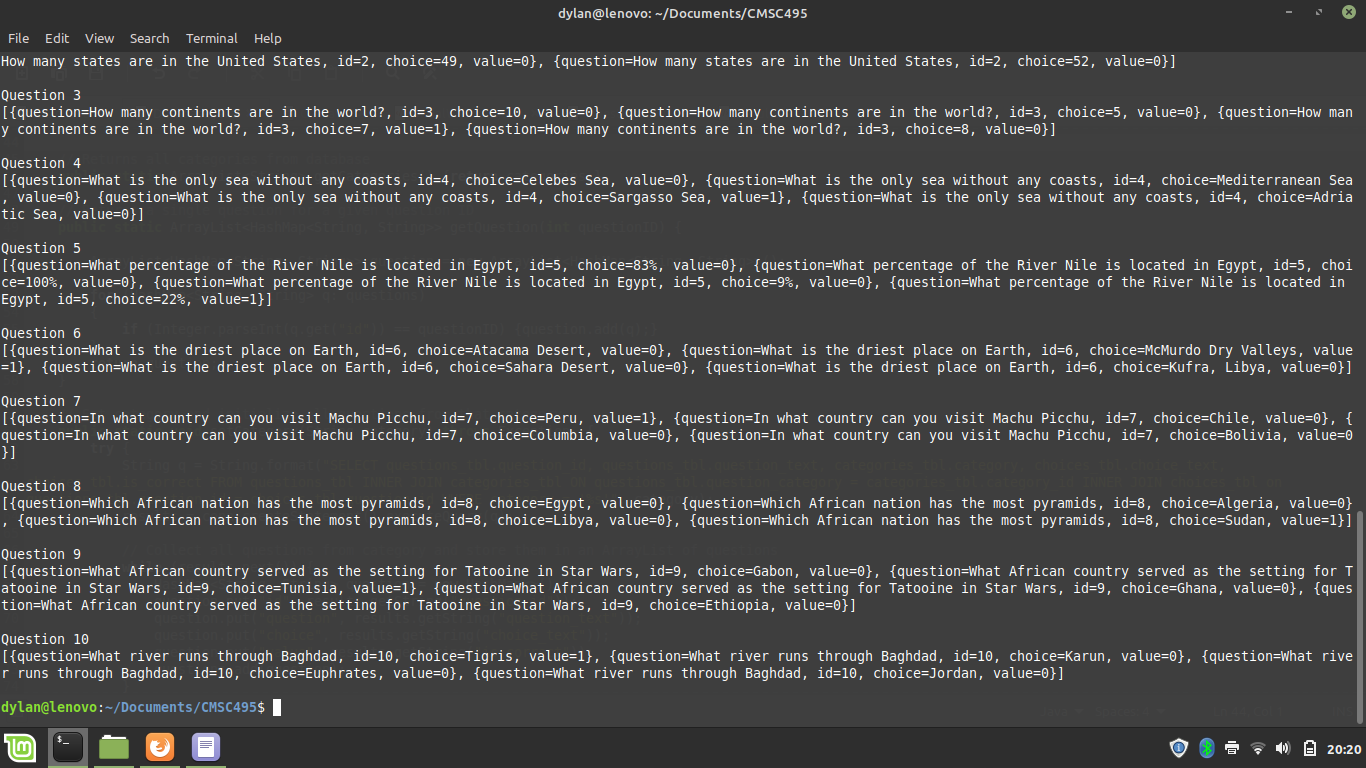
Some decisions that were made early in the project were to develop a username/password login, however, this process was quickly deemed to be unnecessary and removed from the program. Another decision was to add the Database.java to remove some of the logical load on the Main and GUI components. The database class will be responsible for connecting and querying the SQL database directly, where the Main and GUI components simply request subjects and questions from Database.java in a user-friendly format. Screenshots of operable code below for Database communication: (Screen Shots below)



In the above screenshot of the Database.java source code, it shows an example of a query to the SQL database that selects all questions for a given category (requested by the user) and parses them into ArrayLists containing HashMaps. This format was chosen for simplicity and readability within the source code.



The above screenshot details an excerpt from the Main class that operates the Database.java. In this case, selecting the Geography database and requesting the first 10 questions in numerical order. Later this numeric choice will be randomized.



The above screenshot showcases the output of the Database.java requests in a programmatic context. It may not be very reader-friendly for a human, but works very well when matching four choices to a given question (Choice A, B, C, D) and determining which choice is correct for user scores.