Code Review

Video of Debrief

https://drive.google.com/open?id=1ZNFiO9Rac4kwg-YsM5I2MCBnOl9jqKH2

Code Review Strategy

Each teammate will look at another person's code and will review the following:

- Readability and Maintainability
  - Is the reviewer able to read through the code and understand it?
  - Are there chunks of code that can be condensed or split up.
  - Are the Javadocs present and descriptive to an acceptable degree?
- Design
  - Is the structuring of classes coherent?
  - Do the methods, variables, and internal classes have their privacy modifiers set appropriately?
- Functionality
  - Does the code do what it proposes to do?
  - Are exceptions and errors caught and managed appropriately?
- Testing
  - Are there tests for the reviewed code?

Review of Matthew

Reviewed by David Wan

Code Reviewed

- AssignmentDao.java
- QuestionDao.java

Readability and Maintainability

- variable names make sense
- consistent naming conventions (camel casing)
- Reading through the code, I can understand what is going on, but could use a little more commenting in the DAO classes
- JavaDocs are present and informative
- Constants aren't used, but there wasn't any need for them in the DAO classes

Design

- Didn't really have original design, but design is coherent.
- Classes have their methods/variables private/public appropriately

Functionality

- Code does do what it claims to do
- No debug statements leftover
- Exceptions during database connection are handled using a general try-catch.

Code Review of Shi Feng

Reviewed by Matthew Lau

Code Reviewed

1. StudentDao.saveMark
2. StudentAction.validateAnswer
3. AssignmentUI

Readability and Maintainability

- Many issues that could be solved via IDE auto reformat code including the following
  - Wrong lexicographical order for imports [3]
  - Incorrect white space such as two spaces or no spaces in some areas [1] [3]
- Using the '.*' form of import should be avoided [3]
- AssignmentUI should be AssignmentUi as names must contain no more than 2 consecutive captial letters [3]
- Hard coded strings [3]
- Missing Javadoc which is especially useful for DAO functions [1] [3]
- Parameter name of QA should be just qa [2]
- Fire sentence of Javadoc is missing an ending period [2]
- Javadoc line continuation of return is not indented [2]
- ValidateAnswer Javadoc does not seem to match which the function does [2]

Design

- Variables are not specified as public, private, protected etc [3]
- DAO function should not be static [1]
  - It is not thread-safe, could lead to concurrency issues and affects testability negatively
  - See https://stackoverflow.com/questions/2523804/using-static-methods-or-none-static-methods-in-dao-class
- DAO function is void when it should return something that at the very least can indicate if the DB was updated correctly or not [1]
- All events appear to be handled within a single function [3]
  - When actionPerformed checks which button is clicked, it should then call a different function
  - This would make it easier to edit specific functionality of specific buttons

Functionality

- Some import statements are unused [3]
- Everything functions as it is supposed to do

Testing

- No test coverage

Code review of David

Reviewed by Shifeng

Code Reviewed

mainly reviewed GUI with java swing

Readability of code:

- Generally good, understandable by just looking at it.
- Would recommend to separate consecutive code with a comment that describes each subsection
- Consistency on variable naming: most of variables are named properly with respect to their uses, but with a few exceptions, where those are named as button1, button2. etc.

Maintainability of code:

- Main class should have a single class(to separate it from the MainMenu class)
- Recommended to move all the constant out into a single class.
- As the we have multiple GUI panel(for examples, listing of class and list of student etc.), would recommend to move them into individual class, for future maintainability (i.e move them out of MainMenu class, where they currently reside)
- Don’t call the DAO classes directly, call the API class instead, which in turn invoke DAO classes. As it would bypass the security mechanism implemented in the API class,

Code Review of Junzhi Chen (Jackson)

Reviewed by Ryan Young

Code reviewed

1. AssignmentDao.setVisibilty
2. AssignmentAction.changeAssignmentVisibility
3. AssignmentAction.checkAssignments
4. Main.case “getAssignments”

Readability and Maintainability

- Lack of Javadoc, makes understanding AssignmentAction functions slightly harder. [1][2][3][4]
- Variable names could be more meaningful
- Due to lack of Javadocs and comments, it was a little harder to check for required input and output of functions
- Was still fairly understandable and could be understand by a quick read through

Design

- Design is coherent despite lack of original design
- Classes have appropriate methods, but variables are not private

Functionality

- Code functions as required
- No excess or unused statements
- Exceptions are handled by basic try catch, but only returns flag

Code Review of Ryan Young

Reviewed by Junzhi Chen (Jackson)

It is functional and the general format of the code is good. However, the code does not have good Javadocs and the naming of variables is poor.

T3:3

1. When adding an assignment, there is no limit for input. It is recommended that he should avoid Null inputs.
2. Variable JTextField should be private.
3. Recommend checking whether courseCode exits or not.

T3:4

1. Recommend checking the input.
2. Recommend checking whether courseCode exits or not.

T4:1

1. Some generic classes do not declare types.e.g.(“ArrayList<String> list“ is better than “ArrayList list”)

T4:2

1. Recommend adding the attention that there is no result for searching.

Overall. It’s good and works.