Code Review Strategy
Team 14 – Devlopp

Check List

- Is the code it easy to understand?
- Does the code do what it's supposed to do?
- Is there dead-code or debug statements?
- Are exceptions and errors handled appropriately?
- Is the documentation good enough?
- Are the naming conventions appropriate?
- Is there a lot of bad copy and pasted code?
- Are constants used when possible?
- Does the code correspond to our original design?
- Does the code follow SOLID principles?
- Consistent use of whitespace?

Code Review Summary

Christian

Code Review Summary of:
src\main\java\com\devlopp\teq\database\DatabaseInserter.java

There is good commenting and easy to understand code throughout. The code is clear and uses white space effectively. There are no dead-code or debug statements and naming conventions are easy to understand and follows Java convention. This code follows our initial design intention. DatabaseInserter could better follow SOLID, by reducing the responsibility of the class and abstracting more of the methods. Also, it could be refactored and redundancy in the methods could be reduced. The exceptions could be handled better rather than just printed. Docstring is good where implemented; however, some methods are missing their docstring, like insertCourseEnroll and insertCourseEnrollDetails. Magic numbers are used throughout in preparedStatements, try using constants instead so the code can be more maintainable.

Zifan

Code Review Summary of:
src\main\java\com\devlopp\teq\excel\ExcelReader.java

The code is well formatted, meeting indentation and whitespace standards and free from parse errors. The code did what is supposed to do: read the excel file and parse the value. It includes helper functions of getMandatoryColumns, createAllowedValues and getAllHeaders. These helper functions are necessarily included in this class. It is nicely commented and can easily understand.
However, the issue with this part of code is the error checking function is too big, it slightly violated the SOLID design principle. Regarding to design principles it should break down this function's responsibility

or have the error checking be another class. It should be architecturally clearer and more maintainable. I think overall this part of code is good because they are correct and effective solutions for the project requirements at hand.

Jeffrey

Code Review Summary of:
src\main\java\com\devlopp\teq\course\Course.java
src\main\java\com\devlopp\teq\course\CourseBuilder.java
src\main\java\com\devlopp\teq\course\ICourseBuilder.java

This code is used in storing data parsed from the LT Course Setup template. The builder classes, from where the concrete object is created, are obvious implementations of the builder design pattern, thus following some concepts from SOLID. Since the concrete Course object has a lot of attributes, a builder design pattern is very much preferred. The code is also well formatted, and each method and attribute are appropriately named according to common Java coding style conventions.

Although the methods are simple getters and setters, there maybe context needed as to why and how the user would use these methods. There is a lack of documentation nor any helpful commenting in the code. Some methods, like setStartDate and setEndDate in ICourseBuilder, actually require the argument to be in a specific format.

Kelvin

Code Review Summary of:
src\main\java\com\devlopp\teq\databasehelper\DatabaseInsertHelper.java
src\main\java\com\devlopp\teq\databasehelper\DatabaseSelectHelper.java

The code functions as expected, does the required tasks, and follows concepts from SOLID. For instance, the methods in DatabaseInsertHelper only inserts to the database, and methods from DatabaseSelectHelper only selects from it. This adheres to the “Single responsibility principle” as a class only performs and is responsible for a single function (i.e. select, insert or update). Furthermore, the naming convention is appropriate for the methods and easily understandable. From the name “getCourseExit”, users can understand that the method gets a CourseExit object from the database returns it.

Some small issues with the code would be the style and commenting. Although the actual code is consistent, easy to understand, and well formatted, some methods may not be documented. DatabaseInsertHelper in databasehelper does not have a docstring, so it can be confusing for new users or TAs to understand. Another issue would be that the classes are very large. DatabaseSelectHelper contains almost 600 lines so it is possible that we have missed one of the forty tables in the database. It may be possible to refactor the code into smaller pieces and have abstract classes for the inserters, updaters or selectors. Most of the methods are copy and pasted and only have minor differences between each other.

Code Review Video

Link: https://www.youtube.com/watch?v=Iw1jfMi9qr8