Code Review Strategy:

Team members will focus on the following categories when reviewing code:

- Code Readability (styling, meaningful variables)
- Quality of Documentation (Javadoc, internal comments)
- Quality of Design (adherence to SOLID and other best practices)
- Test Coverage

Each package in the codebase will be reviewed by different members of the team:

Package	**Assigned Team Members**
Action	Andy, Sin
Command	Samuel
Database	Andy, Aradhya
DatabaseAPI	James, Samuel
Exceptions	Aradhya, Sin
GUI	Andy, James, Samuel
Models	Aradhya
Tests	All

Code Review Summary:

Andy:

- There are still some “dead” imports
- Inconsistent styling in the use of spaces for tabs, 4 spaces for some and 2 for others

Aradhya:

- The classes have be well structured as per responsibilities but there are cases where inheritance could be used to reduce class size
- The package are well structure so as to minimize dependencies
- There are cases with code repetition which could be reduced by using helper functions

James:

- The use of exceptions could be more consistent, in particular with the interactions between the database and databaseAPI packages.
- Documentation of independent classes is good, but there could be an improvement on how classes from different packages interact with each other.

- Inconsistent use of this.x for private variables of classes where x is the variable, particularly in GUI package.
- Should find a way to put databaseTester with the rest of the jUnit tests.

Samuel:

- Actions and Commands need further decoupling; perhaps use dependency injection
- API interfaces for interacting with the database are too general; can separate APIs into classes for interacting with each object model type, inheriting from a generic interface
- Usages of general Object types and instanceof in commands should be changed to be more statically typed
- In general, more interfaces should be used to program to an abstraction rather than an implementation
- More focus on test coverage is needed

Sin:

- Inconsistent styles, some are using 4 spaces and some are using 2 spaces for tabs
- Errors are not being handled properly
- There are unused variables and imports
- Follows the SOLID principles for the most part
- No dead code or unnecessary debug statements
- Documentation is good overall, methods and variables have meaningful name

Code Review Debriefing Meeting:
https://www.youtube.com/watch?v=XDAmmxR0koY

Code Review - CHECKLIST:

Readability and Maintainability:
Can I understand the code reading it?

Is the documentation good enough?

How is the coding style?

Do we follow appropriate naming conventions?

Do we have bad copy and paste of code?

Do we use constants whenever it is possible?

Code design
Does the code corresponds to our original design?

Does the code follows the SOLID principles?

Can we better structure the code using design patterns?

Functionality

Can I understand what the code does?

Does the code do what it is supposed to do?

Is there some dead code or unnecessary debug statements?

Do we handle the exceptions and errors correctly?

Testing

Can I understand what the tests do?

Do I understand the tests results?

Do we have a good test coverage?