Peer Review for Grupp 23 - GrOOP

Do the design and implementation follow design principles?

The project uses a consistent coding style except for the naming of some functions and classes. The project follows Single Responsibility Principle which makes the code both reusable and easy to maintain. The project uses Adapter pattern for their recyclerviews so that the items in the recyclerview can be used.

Is the code documented?

The packages “Model” and “memory” are well documented according to Javadoc standards but the remaining packages mostly lack documentation of any kind.

Are proper names used?

Throughout the project the naming is relatively good and mostly describes the purpose of the entity (variable/method/class). There is however some inconsistency when it comes to capitalization and abbreviation. Furthermore there is some inconsistency in the order of the words when naming classes, for example the classes “Start/EndFragment”, “MemoryFragmentStart/End”.

FirstFragment and SecondFragment could be named differently. Without much insight “first” and “second” doesn't say much, it doesn't describe what it is. Also first might not be first always, maybe it becomes the third or fourth fragment in the future.

Is the design modular? Are there any unnecessary dependencies?

It could be argued that design is too modular in some places where multiple classes work towards the same purpose. These classes could be merged together and still follow the Single Responsibility Principle.

Is the code well tested?

The models are well tested with relevant tests, making sure that the app is running as expected. However, there are no tests for the UI.

Are there any security problems, are there any performance issues?

Methods and variables are isolated and only public when necessary. FlashCard works well except that the “go back”-button sometimes takes you to the end fragment. The answer on the back of the card changes before the card rotates which makes cheating possible.

Is the code easy to understand? Does it have an MVC structure, and is the model isolated from the other parts?

Model seems to be isolated from other parts. However, there seems to be several files in the model package that don't necessarily belong there or over complicate it.

- Card, Deck and Difficulty should definitely be in the Model package.
- Flashcard, FlashcardProgress should maybe be combined and be inside the FlashcardViewModel or in their own package?

- GameLogic is more of a controller than a model.
- MemoryPairUpLogic Interface seems unnecessary and unsure how it would be used as you would most likely use GameLogic for abstraction. Maybe a better idea is to make another abstract class that extends GameLogic such as MemoryPairUpLogic which holds the common code and then Memory and PairUp would extend that abstract class instead.
- TimeKeeper feels more like a util class or a controller than a model.

Can the design or code be improved? Are there better solutions?

Id instance variables for Card and Deck seem unnecessary. A cleaner solution might be to override the HashCode method in Card and Deck and then use Object.equals method to check if two cards/decks are equal.