

Design Rationale

The Oktanauts - Meiya Lian and Nicholas Chmielewski

<u>Design Principle:</u>

The Open-Closed Principle:

Our current design has been purposely left open for extension in terms of future functionality. For example, measurement functionality hasn't been defined in terms of cholesterol levels specifically. This allows for future dynamic control of the desired measurements. By using this principle, the design can be easily extended without having to modify the underlying data structures to accommodate the changes.

The Acyclic Dependency Principle:

DIP and ISP are used in the system referring to this principle. Interfaces such as GetPractitionerCallback and GetMeasurementCallbak are implemented to invert the dependency between the Controller package and the Model package. In this way, any changes to the controller classes will not affect the model package, as the model classes interact with the controller classes via interfaces.

<u>Design Pattern:</u>

The Model-View-Controller Architectural Pattern:

The system is designed with the MVC model through the use of fxml and java.

The "Model" consists of application-specific domain objects, the "View" consists of FXML, and the "Controller" is Java code that defines the GUI's behavior for interacting with the user. (Implementing JavaFX Best Practices | JavaFX 2 Tutorials and Documentation, 2020) By using MVC, multiple views can be implemented and the UI components can be changed easily without changing the business logic inside the model, which follows OCP.

The active MVC model is chosen over the passive model, as the active model is more suitable to use under the context that the measurements (model) need to be refreshed every N seconds. The two interfaces GetPractitionerCallback and GetMeasurementCallbak allow the model to report to the controller when they change their state, which the controller will then notify the view to refresh and display the updated data to the user. However, on the other side, this design introduces extra levels of dependency as one of the motivations for using MVC was to make the Model independent from the others. To minimize these dependencies, interfaces are implemented in between the components.

The Iterator Pattern:

The Iterator pattern is used for an aggregate class **PatientList** in the system. The iterator functions have not been used in any functionality in the current system. However, we decide to keep this pattern by following OCP. With this pattern, future functionalities such as filtering patients with age/gender/other attributes can be implemented easily.

Reference:

Docs.oracle.com. 2020. Implementing Javafx Best Practices | Javafx 2 Tutorials And Documentation. [online] Available at: https://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm [Accessed 19 May 2020].