Design Rationale

Our team decided to follow an MVC architectural pattern in assignment 2. Thus, we were able to extend our application to incorporate the extra functionality required for assignment 3 with minimal changes to classes and packages from assignment 2.

This is owing to the fact that our packages and classes were highly modularized to adhere to the SoC principle.

The package structure of the project has been refactored to incorporate the Common-Reuse Principle and The Common-Closure Principle. This is achieved by grouping together classes that were likely to change together for instance all classes related to the cholesterol monitor were put in to one monitor package and inside this package the classes were sub packaged to group classes that function together.

No packages have cyclic dependencies thus the package architecture follows the Acyclic dependencies principle.

We refactored the PatientService interface into an abstract class and we also refactored the PatientServiceImpl class and broke up it into multiple classes. Thus we created a seperate class for each functionality like fetching patient details, cholesterol levels, blood pressure records and smoker records. All these classes were extended from PatientService abstract class. This is to satisfy the Single Responsibility principle as now each class will be responsiblefor one singular task. And thus everytime new functionality needs to be added we can extend the PatientService abstract class. This also satisfies the Open-Close principle as now everytime we need to functionality we will not be modifying existing classes but instead extend from the PatientService class.

The cholesterol-monitor.html template file was refactored to remove all repeating components and attributes.