DESIGN RATIONALE

Packages
The design of packages complies with the **Common Reuse Principle (CRP)** and **Stable Dependency Principle (SDP)**. In particular, the package *model* is stable as it doesn't have any outgoing dependency and hence package *studentview, tutorview and mainview* depend on it (SDP). StudentView and its subclasses are enclosed in one separate package as they are reused together, as well as TutorView and its subclasses. The classes in one package are tightly related and it is impossible to reuse one without also reuse the others. (CRP)

Abstract class View (extended by all -View classes in the system)
The design of abstract class View follows **Factory Method pattern**, whereby the factory method is the *placeComponents()* method. This gives the system the flexibility to extend to new View classes without modifying its codebase.
The use of View classes also enabled several instances of **Dependency Inversion**, e.g. *addSwitchPanelListener* method and *Display* class. These methods and classes operate on the abstraction level of View class rather than a concrete class which is more bound to change.

StudentView and Tutor View
The design Student View and TutorView subclasses were motivated by **Open-Closed Principle (OCP)**. These views work as the template for their subclasses Views which tend to serve a more narrow-scoped functionality. Thanks to these classes, functionality addition can easily be made by extending from these parent views rather than modifying the existing classes in the system.

StudentView subclasses and TutorView subclasses
Since the business logic for a student and tutor is largely separated, combining the functionalities to fully support both student and tutor needs may cause the issue of class bloating and went against **Single responsibility principle (SRP)**. The separation makes the system more expandable should other roles be included alongside student and tutor. It is also more adaptable to functional change, as the code relating to each functionality is divided into manageable sized classes.

ListPanel extending JPanel
ListPanel was created out of the need for fast-visualizing Java's List data type. It keeps true to **Dependency Inversion Principle (DIP)** and polymorphism by working on the abstraction level of JComponent class. Hence, it can be used for JComponent's subclasses such as JPanel, JButton, etc.

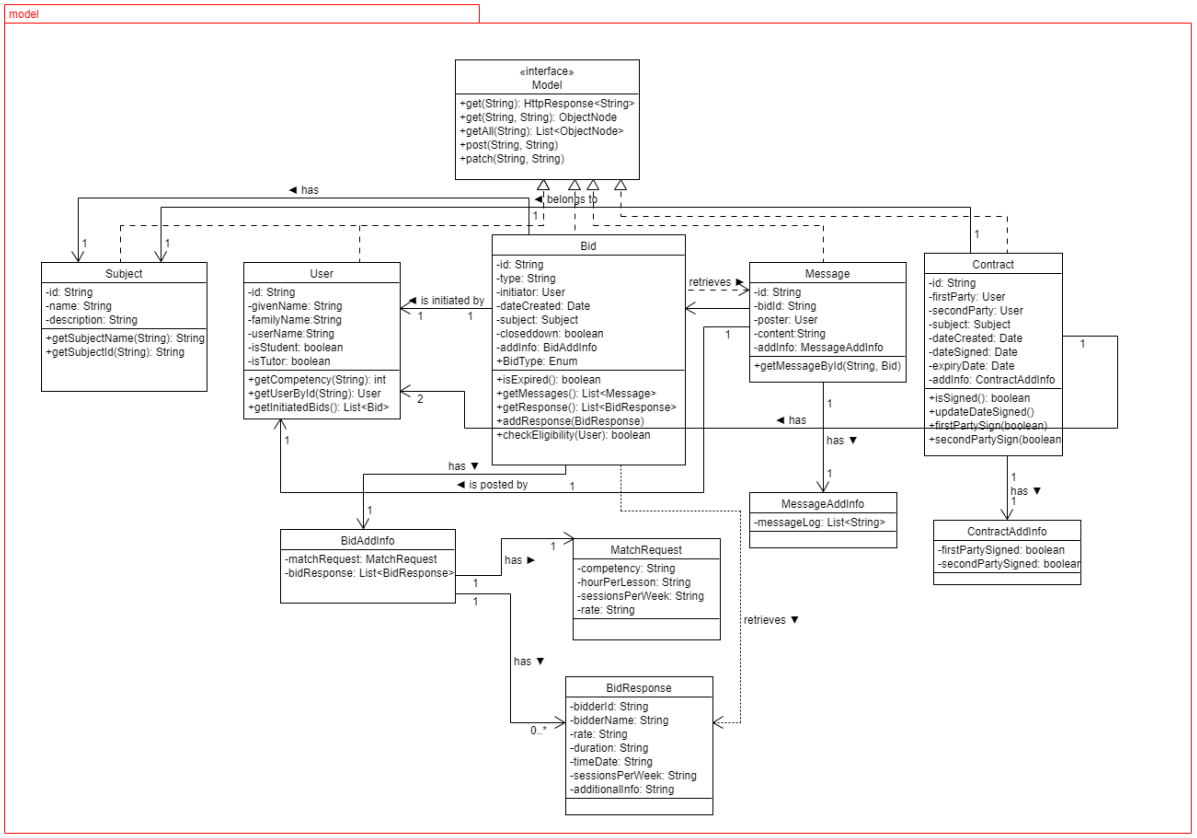
MouseClickListener interface inheriting from MouseListener interface
Since the majority of operations only have the need to observe mouse click events, a MouseListener interface with 5 unimplemented methods of different MouseEvents is an overkill and increases code repetition. The use of MouseClickListener which gives default value to unnecessary methods, was to follow the principle **DRY (Don't Repeat Yourself)** and was an attempt to cut down the complexity of the codebase.

Model interface

Model interface follows **Facade Pattern**,which provides an easy-to-use interface to the HTTP operations. It provides a multitude of default classes that handles HTTP requests and can easily be reused by classes implementing it. It greatly reduces the amount of repeated code and keeps the Model classes nice and tidy. Hence, it also abides by **Single Responsibility Principle (SRP)** by helping to keep the HTTP responsibility separated from business logic.

CLASS DIAGRAM

1. Model package



2. Mainview, studentview and tutorview packages with inter-package dependency