EECS 3311 Lab 6 Report

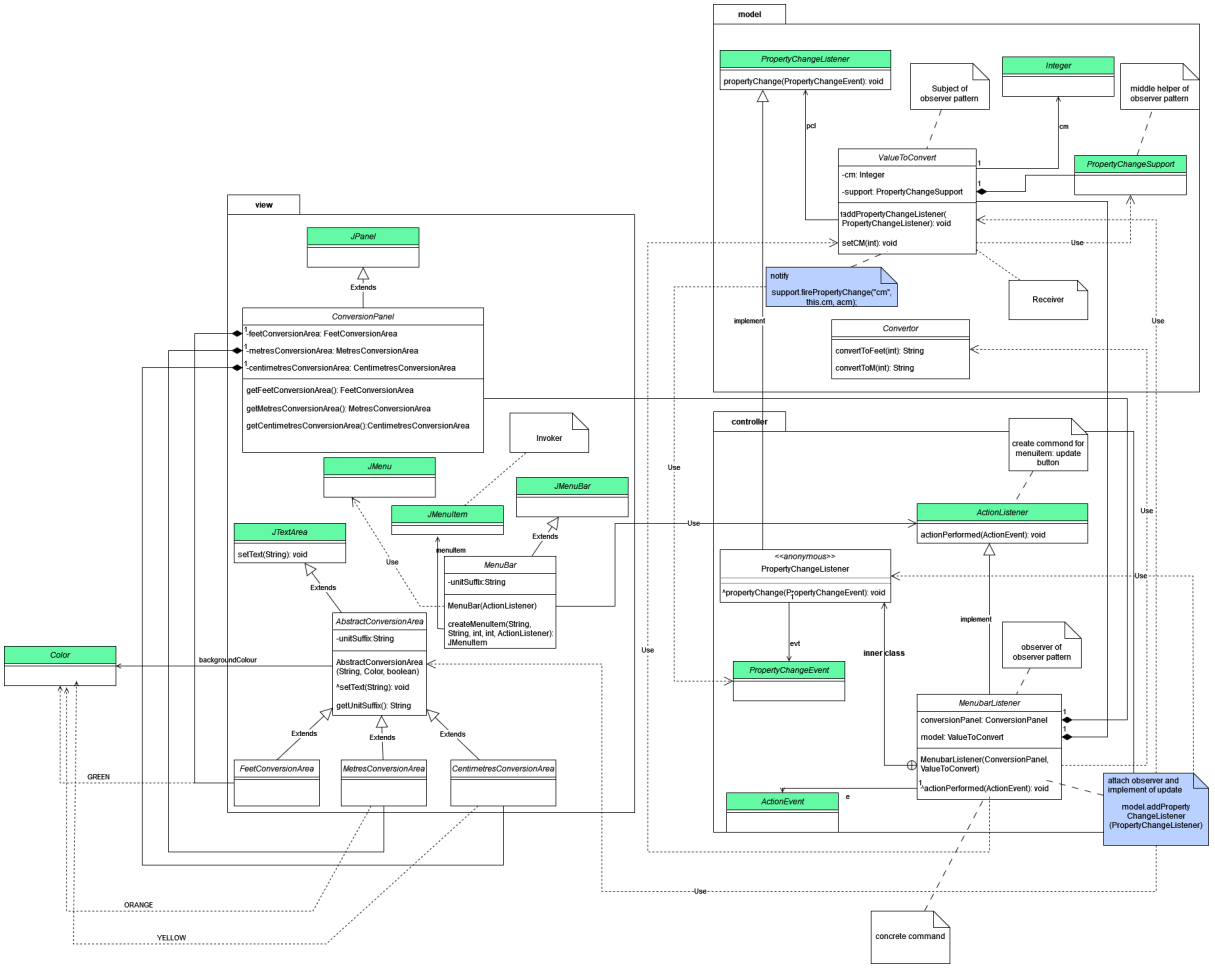
Andrew Hocking, 215752835
Adrian Koduah, 216793887
Yun Lin, 214563449
Saad Shahid, 216178477

Part 1: Introduction

This software project is an application that behaves like a unit converter. It converts a given value of one unit to other units, specifically it converts centimeters to meters and feet. The goal of this project is to design and implement the Converter application using the principles and patterns of Object Orientation. The primary challenge for this project was to fully implement the application using the Model/View/Controller design pattern from scratch, where no partially-implemented code was given. Object Oriented Design Principles used in this project are *Inheritance, Encapsulation,* and *Abstraction.* Inheritance is the process where a class can inherit methods and attributes of another class through the relationship of parent class and child class. Encapsulation is the process of privatizing certain methods/ attributes that are specific to a class and don't pertain to any external classes, preventing direct access to them from the external classes. Abstraction is the overarching process of hiding internal information from other components that are not important to them, only showing what is important to them. This report will be structured in the following order: 1.Introduction, 2. Design of the Solution, 3. Implementation of the Solution, and 4. Conclusion.

Part 2: Design of the Solution

Class diagram:



Design Patterns Used

Command pattern
Invoker class: MenuBar class a container with menuItem.
Receiver class: ValueToConvert class.
Command class: ActionListener Interface -> it declares an interface that handles the execution of an operation through the actionPerformed method.
Concrete Command class: MenubarListener class-> it implements actionPerformed by invoking the setCM method on Receiver.

Observer pattern
Concrete subject class: ValueToConvert class -> the method addPropertyChangeListener() to attach observer objects.
Concrete observer class: MenubarListener class -> an anonymous inner class implementation of PropertyChangeListener interface implements propertyChange() method as update method of observer class.

Helper class: PropertyChangeSupport instance -> it help send notification to observers when an specify attribute of subject class changed.
The implementation of notifying emthod of subject class is in ValueToConvert class named setCM(). This method get help from helper class by calling firePropertyChange method from instance of PropertyChangeSupport.
At first, while initializing step of controller class: MenubarListener, this step will calling subject instance' attach method (addPropertyChangeListener) attach its' instance as a listener to listener maps of PropertyChangeSupport's.
Later when controller class: MenubarListener calling setCM method, the PropertyChangeSupport will notify the observer objects by searching though its' listener map and then pass an PropertyChangeEvent to observer object, after that, observer object change the view containers base one passed event.

Design Principles Used

Inheritance

Inheritance simplifies the coding process by allowing classes to inherit and access attributes and methods that were already created by parent-classes. That's why inheritance is apparent throughout the project. For example, the MenuBar class extends the JMenuBar class, inheriting attributes from the preexisting JMenuBar class. The ConversionPanel class extends the JPanel class, inheriting attributes from Jpanel. CentimetresConversionArea, FeetConversionArea and MetreConversionArea classes are all subclasses of the superclass AbstractConversionArea abstract class.

Abstraction

The AbstractConversionArea abstract class defines the structure of a ConversionArea object, which is a View component that visually constructs a square on the panel. The specific details for how the square is visually constructed is contained within the AbstractConversionArea class, and it is not necessary information for its subclasses (CentimetresConversionArea, FeetConversionArea and MetreConversionArea). The subclasses only require basic information such as the unit the square represents, the colour of the square, and whether or not the user can edit it, all of which are all passed as parameters to the superclass' constructor method and they are UnitSuffix, BackgroundColour and isEdible respectively.

Part 3: Implementation of the Solution

The tools used to complete the coding portion of this software project were Eclipse build id: 20210910-1417.
With jre: org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v20210721-1149.
Detail implementation explanation see javadoc: at github /doc
Video demo: https://youtu.be/CMPQEQwKick

Part 4: Conclusion

Conclusion in note form (temporary)

Benefits

- Implementing the software project in components such as Model View and Controller from scratch solidified our understanding of the design process. A hands-on approach for the implementation was a great way for learning it

Challenges

- Unlike the previous software projects, this is the first one where we had to start from scratch. It can quickly become challenging if replanning is not done and UML diagrams are not made before the implementation process of the project

Advantages of working as a group

- Implementation of the project was simplified with multiple people working on it.

Disadvantage

- In theory, communication via online is readily accessible and simple, but on the contrary can be challenging when everyone has different schedules and availability, because of that it was a challenge to set hard deadlines, instead the project was passively worked on during the time given

Three ways to ease the completion for the software project

- Communication with members as soon as possible.
- Understanding the requirements of the project
- Delegate work to members and sticking to the part assigned to you, ensuring that it is complete on time
- To keep everything organized and work faster, adopte the SCRUM model for simplifying the process, and sticking to it.

Tasks Completed			
Andrew Hocking	**Adrian Koduah**	**Yun Lin**	**Saad Shahid**
Implemented app view		Implemented app controller	Report: Introduction
Assisted development of app model		Lead development of app model	Report: Design Principles
Javadoc		Report: Design Pattern	Report : Conclusion notes