EECS 3311 Lab: Software Project 6

Wenjing Qu - 214568612

Kurt Magsipoc-Wilson - 216047953

Oreoluwa Ogunlude - 216594285

Corneille Bobda - 217548595

Part I: Introduction

The software project requires us to create an application that allows converting a value in centimeters to feet and meters. One of the challenges I notice is applying both the command and observer pattern when implementing. The concepts we will be using are the Observer pattern, the command pattern, the factory pattern, encapsulation, polymorphism, and inheritance. Our report will be structured as the implementation is being done, gradually.

Part II: Design of the Solution

The 2 design patterns used in our implementation are the observer pattern and the command pattern

- Observer pattern: this pattern involves the Observer interface, the Subject interface, the ValueToConvert class, and the ObserverConversionArea class. The Observer interface observes the Subject interface for when the centimeter value is saved. The ValueToConvert class updates the feet and meter values then notifies the ObserverConversionArea class.
- Command pattern: this pattern involves the ConvertCommand interface, the FeetConvertCommand class, the MeterConvertCommand class, and the ObserverConversionArea class. The ObserverConversionArea class asks the ConvertCommand interface to convert the centimeter value to feet and meter. That request is then executed by both the MeterConvertCommand and the FeetConvertCommand classes.
- Factory pattern: the ConversionAreaFactory class is used to create JTextArea instances.

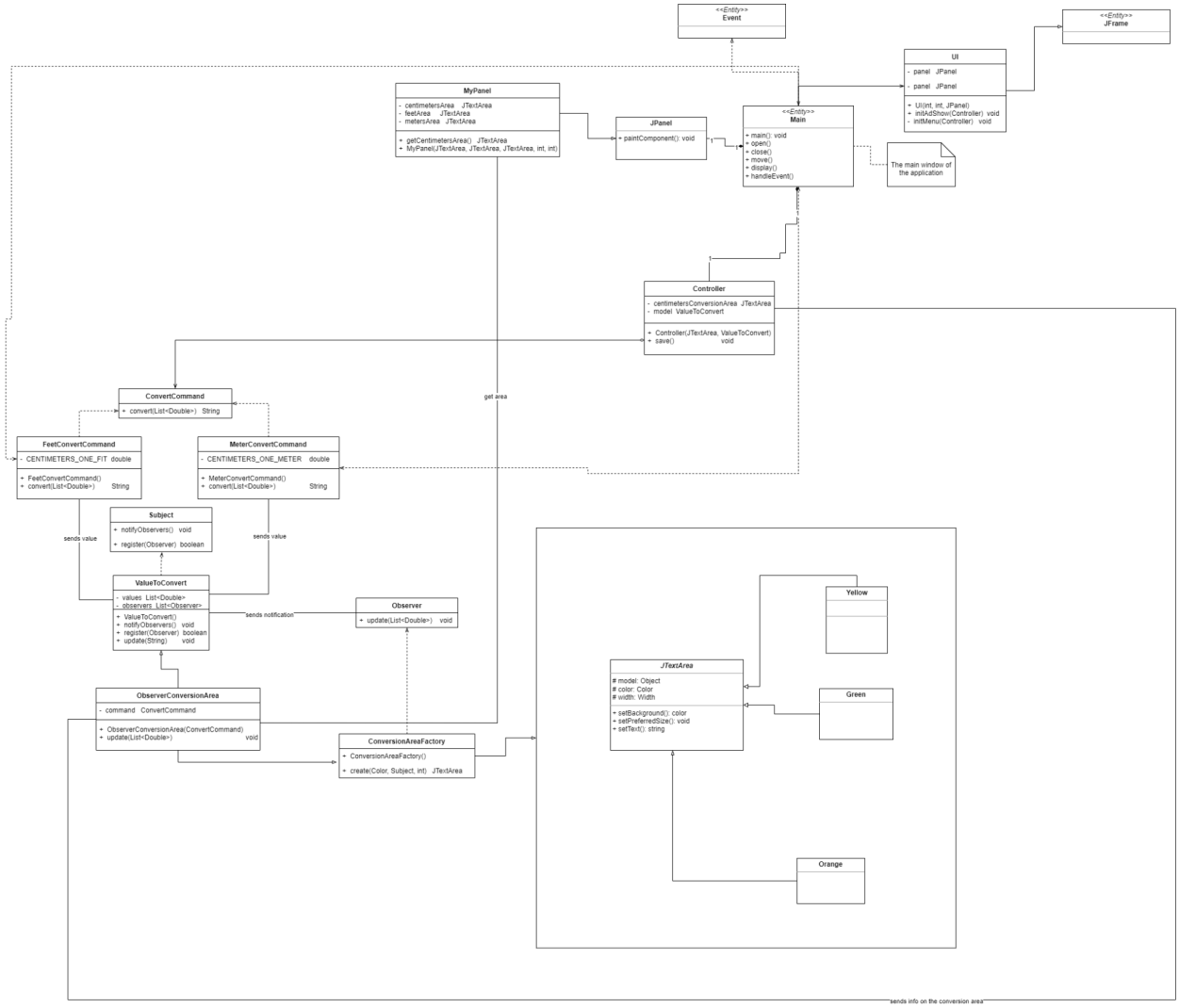
The Object-Oriented design principles used are the following:

- Inheritance: we make use of inheritance through the ObserverConversionArea class, the MyPanel class, and the UI class. The ObserverConversionArea class extends JTextArea, the MyPanel class extends JPanel, and the UI class extends JFrame
- Encapsulation: we hide unnecessary information between classes by using private variables and private methods. The Controller class, the FeetConvertCommand class, the MeterConvertCommand class, the ValueToConvert class, the MyPanel class, the ObserverConversionArea class, and the UI class all contain private variables. In the case of private methods, the UI class is the only class with one.
- Polymorphism: the ObserverConversionArea class applies polymorphism by overriding the update() method from its parent class. Both the FeetConvertCommand

and the MeterConvertCommand classes override the convert() method from ConvertCommand

Part III: Implementation of the Solution

UML Class Diagram of the Converter App



Tools and Libraries Used

- Git - Used for source code management by tracking changes in the source code and collaboration among group members.
- Discord - An instant messaging platform that was used to communicate with group members, assign tasks, and manage deadlines.
- GitHub - A version control system that uses Git, which allows for team members to collaborate together and make changes to the code.

- Eclipse - The IDE that was used for development.
- Terminal - A command line system that was used to input commands to run the software and utilize Git.
- JDK - The Java Development Kit was used to develop the application using its API.
- Swing - A toolkit in Java that was used to create the application's GUI.
- Diagrams.net - The website used in creating the UML Class diagram.

Part IV: Conclusion

The communication between group members and engagement play a huge part in accomplishing this project. On the flip side, we had some difficulties when creating the UML class diagram. This project reinforced our knowledge of the Model View Controller pattern. One of the advantages of completing a lab in-group is that the workload can be divided but the drawback is present if there is no cooperation or good communication.

Name	Work assigned	Completed	% of work
Corneille Bobda	Introduction, conclusion, and part 2 of the report	Yes	25
Kurt Magsipoc-Wilson	Part 3 and presentation of project	Yes	25
Wenjing Qu	Design and implementation	Yes	25
Oreoluwa Ogunlude	UML Diagram	Yes	25