Software Design: Lab 6 - Unit Converter

Group Members:

Omar Saeed: 215911555

Arian Mohamad Hosaini: 214969638

Mustafa Ahmad (Section B)

Feng Lian: 217673815

TA: Naeiji Alireza

Part 1: Introduction:

- **Project goals:**
  This project requires us to create an application that uses users input, taken in centimeters, to convert it into feet and meters. The JFrame consists of 3 squares, in different colors, to be displayed on the screen. The square at bottom takes the input value from the user to be converted and displayed on the other two squares above. A JMenuBar is given to update the results after the user is done inputting their value.

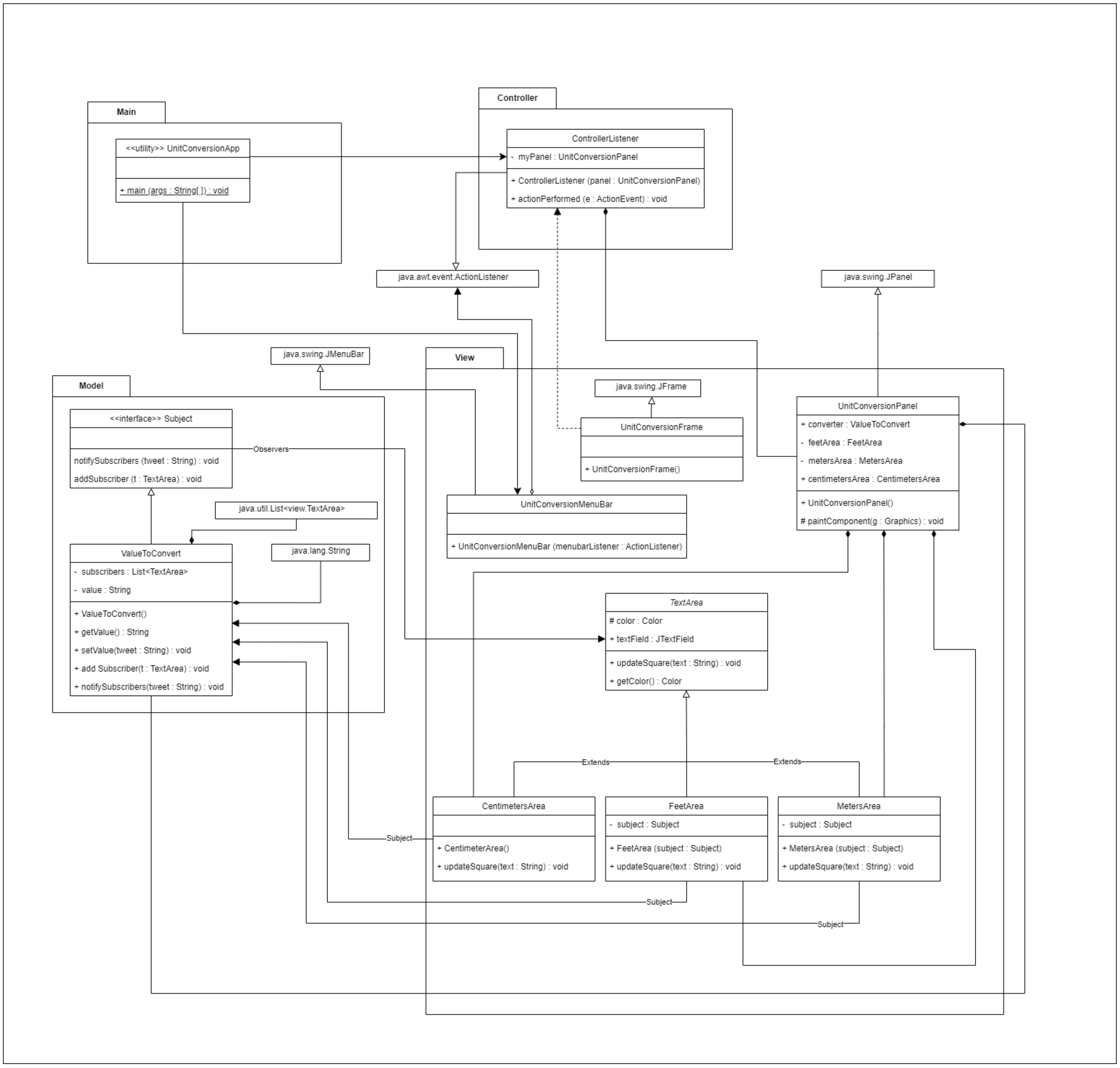
- **Challenges faced:**
  The biggest challenge faced while completing this project was implementing the code as MVC. Keeping classes in different packages and figuring out which class goes in which package was one of the biggest issues to overcome. There were also some challenges in designing the UML with the two design patterns in mind, as it was required to change the diagram multiple times.

- **OOD and Design patterns used:**
  We used MVC (model-view-controller) for the architectural pattern. We also used observer and command patterns for our UML diagram and implementation. We also used the four Object-Oriented Principles such as Abstraction, Encapsulation, Inheritance, and Polymorphism.

- **Report Structure:**
  This report will feature 4 parts, as required. Part 1 is an overall introduction, part 2 will be an overview of the design, part 3 will cover the implementation details of the project and part 4 will conclude the report.

Part 2: Design of the Solution:

- UML Diagram:


Main
<<utility>> UnitConversionApp
+ main (args : String[ ]) : void
Controller
ControllerListener
- myPanel : UnitConversionPanel
+ ControllerListener (panel : UnitConversionPanel)
+ actionPerformed (e : ActionEvent) : void
java.awt.event.ActionListener
java.swing.JPanel
java.swing.JMenuBar
View
java.swing.JFrame
Model
<<interface>> Subject
notifySubscribers (tweet : String) : void
addSubscriber (t : TextArea) : void
Observers
UnitConversionFrame
+ UnitConversionFrame()
UnitConversionPanel
+ converter : ValueToConvert
- feetArea : FeetArea
- metersArea : MetersArea
+ centimetersArea : CentimetersArea
+ UnitConversionPanel()
# paintComponent(g : Graphics) : void
java.util.List<view.TextArea>
UnitConversionMenuBar
+ UnitConversionMenuBar (menubarListener : ActionListener)
ValueToConvert
- subscribers : List<TextArea>
- value : String
+ ValueToConvert()
+ getValue() : String
+ setValue(tweet : String) : void
+ add Subscriber(t : TextArea) : void
+ notifySubscribers(tweet : String) : void
java.lang.String
TextArea
# color : Color
+ textField : JTextField
+ updateSquare(text : String) : void
+ getColor() : Color
Extends
Extends
CentimetersArea
+ CentimeterArea()
+ updateSquare(text : String) : void
FeetArea
- subject : Subject
+ FeetArea (subject : Subject)
+ updateSquare(text : String) : void
MetersArea
- subject : Subject
+ MetersArea (subject : Subject)
+ updateSquare(text : String) : void
Subject
Subject
Subject

- **Design Pattern usage:**
  - ❖ Observer Pattern
    - ➢ Subject: Subject interface that knows its observers and proposes operations to attack and detach them.
    - ➢ Concrete Subject: ValueToConvert class modifies a list of subscribers when changes have been made to the CentimetersArea class.
    - ➢ Observer: TextArea abstract class that updates the data in its textfield and has three child classes: CentimeterArea, FeetArea, and MeterArea.
    - ➢ Concrete Observer: FeetArea and MetersArea classes that handle the data in their JPanel square and each of these subclasses instantiate a new area and convert the value from cm to either feet or meter depending on class, and update the new value in the textfield.
  - ❖ Command Pattern
    - ➢ Client: UnitConversionFrame class extends to JFrame and holds the UnitConversionPanel that handles the text areas.
    - ➢ Invoker: UnitConversionMenuBar class contains a JMenuItem and a constructor to create the MenuBar and menu item to it.
    - ➢ Command: ActionListener Interface
    - ➢ Concrete Command: ControllerListener class implements ActionListener Interface that receives controller events. When this controller event occurs, that object's appropriate method is invoked.
    - ➢ Receiver: Not implemented due to only requiring one function.

- **OO principles in UML diagram:**
  - ❖ Abstraction: The attributes of some classes are declared as private. For instance, the ValueToConvert has 2 private attributes, the getter() and setter() method have also been created publicly so that they can be accessed.

  - ❖ Encapsulation:The ValueToConvert class encapsulates the values in centimeter area, the ControllerListener class can set the new state of ValueToConvert through the public setValue() method while keeping the logic inside.

  - ❖ Inheritance:Inheritance: TextArea class is the parent class of CentimetersArea, FeetArea, and MetersArea class. The ValueToConvert implements the Subject Interface. The ControllerListener implements ActionListener.

  - ❖ Polymorphism: FeetArea and MetersArea class are both inherited from TextArea class. They will perform different actions and convert the value accordingly when the updateSquare() method is called.

Part 3: Implementation of the Solution:

- Documentation: JAutoDoc, JavaDoc (refer to the doc folder on Github)

- Tools/Libraries used:

  We used Eclipse as an IDE and Java version "14.0.1" for the implementation of this project.

Part 4: Conclusion:

- **What went well in the software project?**

  For the most part, completion of this project went smoothly. We were able to create a draft UML diagram that assisted us in the implementation. Although the UML was adjusted multiple times, we did manage to complete the code properly due to it. It also helped in the completion of the report.

- **What went wrong in the software project?**

  Time management could have been better. Most of the group members had a lot of work due from other courses which led to delayed completion of the UML diagram and the report. However, we were able to get it all completed on time.

- **What have you learned from this software project?**

  After completing this project we have better familiarity with writing reports, documenting code, using Java graphics and the MVC model. It also helped us get more experienced in working in groups and dividing tasks when required. Design patterns was another topic that we feel have improved upon.

- **What are the drawbacks and advantages of completing the lab in a group?**
  - **Advantages:**
    - It becomes very easy to take on the whole project when it's divided into 4 parts. No one is overburdened with work.
    - Some parts of the lab are more appealing to some members. For example, coding and report/UML were done by different members of the group because of their preferences. This leads to everyone doing what they like doing.
    - Time and workload is significantly reduced as compared to completing the project individually.

- **Disadvantages:**
  - No guarantee of work being done on time.
  - Poor communication can lead to further problems.
  - Group members may not be on the same page at all times.

- **Work each member did:**

Name.	*Assigned task.*	*Collaborative?*
Omar Saeed	Completing the report	Yes
Mustafa	UML, Report, Implementation	Yes
Arian Mohamad Hosaini	Implementation, UML	Yes
Feng Lian	Implementation, Report	Yes