**EECS 3311 LAB 6**

**Report**

Akif Prasla, Hasnain Saifee, Saniz Momin

Dec 6, 2021

**Contents**

Part 1: Introduction:

1) The software project is about an application that allows the user to input a number and produce an output in feet and centimeters. The yellow rectangle is the input box, and the output boxes are in green(feet) and orange(centimeters). The goal is to input any real number and output that number in feet, and centimeters in their respective boxes.

2) Some of the challenges include:

· Not able to fit the rectangles into one window

· Overlapping the rectangles

· Not outputting the right conversions for the input number

· Not able to position the menu bar in the top left corner

· Action listener on the “Update Model” button was producing bugs

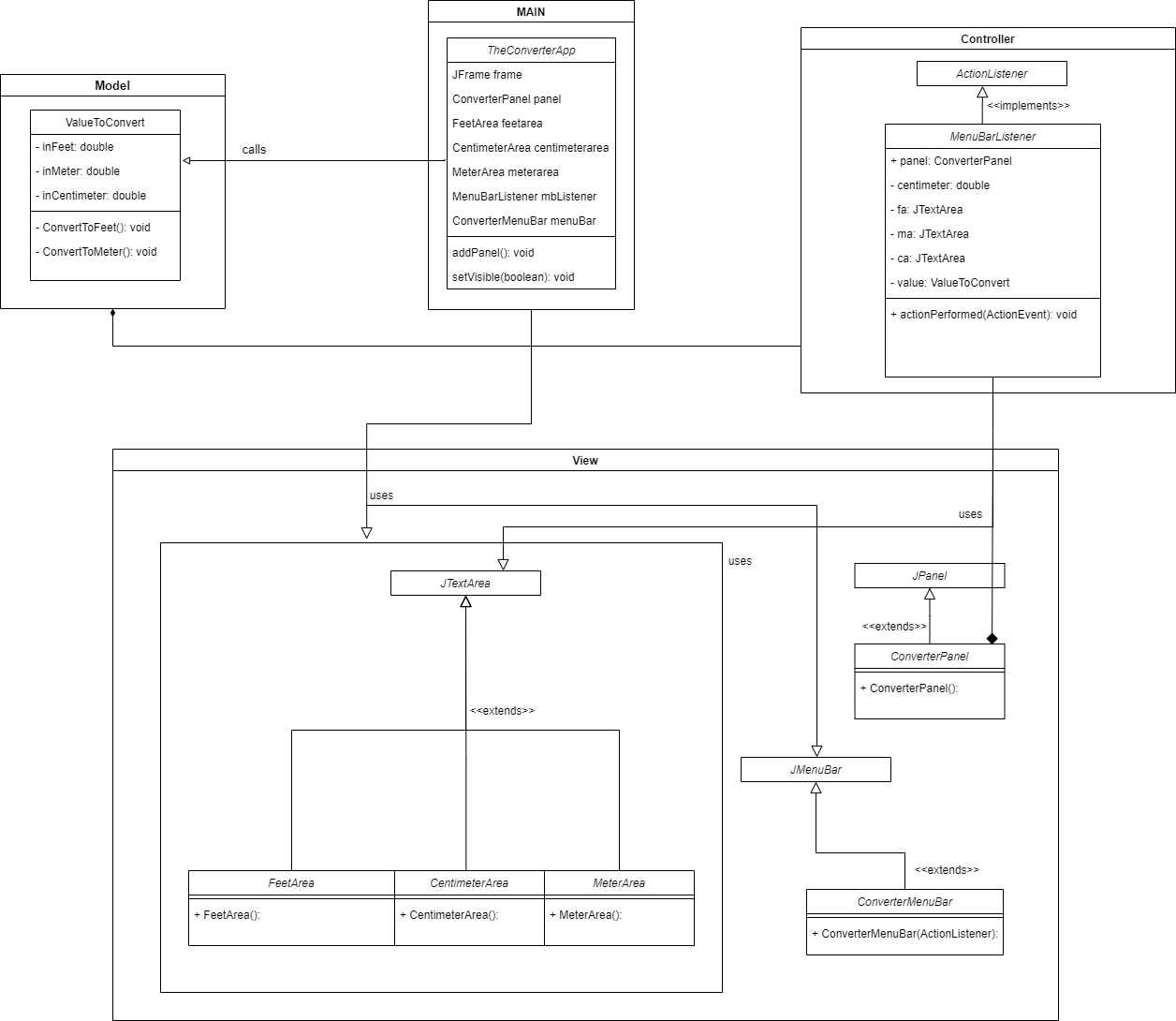
3) - In this project, all the OOD principles should be used. Abstraction will be used to hide unnecessary details from the user. Methods such as *ConvertToFeet()*, *ConvertToCentimeters()* are easily identifiable and readable by users. Each class will be functioning differently, and the methods and states will be encapsulated in those classes.

- Moreover, the design patterns will help us construct the foundation of the program that is to be created. It will give us the blueprint, and help us identify loopholes that need to be filled in.

4) The report will be structured as follows: Introduction, OOD principles explained, design patterns, UML Class diagrams, Implementation of the code & conclusion.

Part 2: Design:

**UML:**



**Design Patterns:**

* We used the Observer and Command design pattern as specified. **The Observer** pattern is used when there is a one-to-many relationship between objects such as if one object is modified, its dependent objects are to be notified automatically. **The observer pattern** uses three actor classes. Subject(MenuBarListener), Observer(JTextArea) and Client. A subject is an object having methods to attach and detach observers to a client object. We have created an abstract class *Observer(JTextArea)* and a concrete class *Subject(MenuBarListener)* that extends class *Observer*. We can see clearly that when the button ‘save input centimeters’ is clicked, the rest of the two boxes automatically update their values showcasing the Observer pattern.
* **Command pattern** is a data driven design pattern and falls under the behavioral pattern category. A request is wrapped under an object as a command and passed to the invoker object. The JMenuBar option ‘Update menu’ acts as a ActionListnerCommand. The client is being shown in the TheConvertorApp class which has the main function. The ConvertorMenuBar invokes the function taking place and calls the necessary functions. The receiver is the MenuBarListener which receives the call to make the necessary changes to the view and implements the changes before updating the correct values which are shown on the screen using the view files.

**Implementation of the solution**

In this software project there are software design patterns and OOP Principles are used to implement the classes and for the efficient implementation of the code. Basically the project is using the MVC design pattern which is the Model, View, Controller. View is concerned with the Look and UI of this project whereas Model’s role is to make sure there is proper implementation of any calculation made by the user. On the other hand, the controller has the job of passing data to and from view to model and vice versa.

**Classes:**

**Model**:

This contains the class named **ValueToConvert** which essentially takes care of how the conversion will take place and holds variables related to three different values which are centimeters, feet and meters. This class will also take care of the conversion taking place once required and has respective methods for it.

**View**:

View is concerned with the design of this project, how it will appear to the user and what visual functionalities are added. For example the color of the three text area which is implemented in **CentimeterConversionArea** , **FeetConversionArea** , **MeterConversionArea** having the Jmenubar with required JmenuItem and also which shortcut key the user can use to get instant output on the text area for example in this project we use Alt F to display the calculated value in feet text area and Meter text area.

**Controller**:

The controller holds the important class named **MenuBarListener**. This class has the ability to control the entire project and also run the app as required by the user. This class will accept the inputs as ActionListner and obey them according to the requirements. This enables the user to change the values of the neighboring boxes when clicked by the user.

**Main**:

This is the Main class where objects are created for example **JFrame**, **convertPanel** and also passing of different object instances as parameters in other objects also takes place. Without the main class there is no existence of any object or the UI that appears.

**OOP Principles:**

**Abstraction**: In this software project, some unnecessary details are hidden from other classes. Some instances where abstraction is used can be seen in all the variables that are private and hidden from the classes which use instances of the class. The ValueToConvert class holds certain details such as values for feet and meter which isn’t required by the controller class MenuBarListener which holds the instance of the class. This is an example of abstraction.

**Encapsulation**: In this software project, you will see the concept of encapsulation used where some logical implementation required for some classes to give required functionality is set to private so that other classes cannot access it. We can see this in the ValueToConvert class where the methods used for conversion are kept private as the implementation is not required to be public and is only accessible to the local class for converting the values.

**Inheritance**: The concept of Inheritance is one of the most required principle used in this software project for better performance reasons. Always keeping the **(DRY)** principle in mind which is **Don’t Repeat Yourself**. Inheritance is used widely. For example each class **ConverterPanel** extends **JPanel**, the three textarea class

Which is **CentimeterConversionArea**, **FeetConversionArea**, **MeterConversionArea** extends **JTextArea** because of inheritance we are able to use methods that are implemented in the parent class without writing it again.

**4 Conclusion**

This software project helped us to work as a group and also get an insight of how to collaborate while working on the same project. We learned how to work in team and coordinate with each other. This project also helps us understand the design principles and use them in an adequate way according to the project need. One big thing we learned from the previous mistake was to take a brief look at how to implement different aspects of the project in the MVC manner. This helped us stay organize and keep progressing towards the final goal. The things that I learned through the journey of creating this software project is initial planning, if it’s a project which requires lot of demand to be fulfilled. I used divide and rule principle where I breakdown the demands of this project into subpart and tried to find what should be the role of that subpart in the project how many classes will be required and is there a way to implement code reusability by using OOP Principles.Disadvantages can be difficult to find a common time for collaborating, too many ideas to deal with, etc. Few recommendations for easy completion of project for upcoming students are that they should ask for help understanding the flow of code in the project, attend lab session and ask for help from TA, make sure that they understand the basic concepts of JAVA and also the software concepts taught in the class.

Yes, all of the group members were collaborative and we worked as a team to finish the project. Following is the division for the work:

**Setup**: Saniz

Setup the project on github to make sure the team could collaborate together.

**Uml**: Akif, Hasnain

Created the uml.

**Report**: Saniz, Akif, Hasnain

Created the report.

**Tools** used were Eclipse, Draw.io, Github.