EECS 3311 Section B Third Software Project

The ConverterProject

Xinger Yao 215106578

Chang Liu 216882813

Jiakang Chen 216952012

Yuyu Ren 216547911

TA's Name: Naeiji Alirez

Table of Contents

Introduction

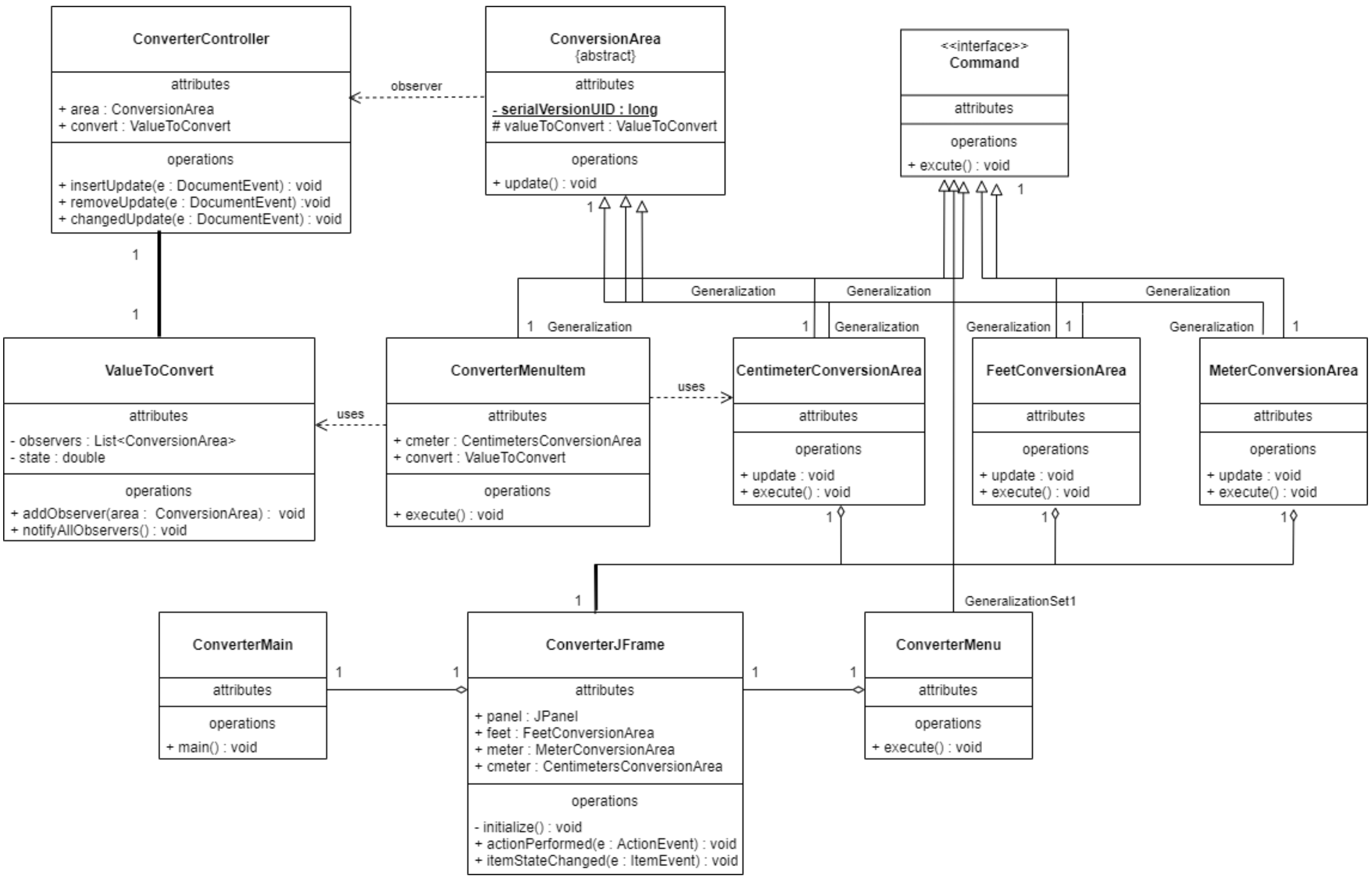
This software project is a Converter. The goal is to display a user interface and convert the user’s input centimetres value in feet and meters successfully. The application displays an interface with a JMenuBar and a JPanel. The JPanel contains three views: a yellow view where the user can input a value in centimetres, a green view where the input value is displayed in feet, and an orange view where the input value is displayed in metres. The JMenuBar comprises a JMenu named “Update model” and a JMenuItem called "Save input centimetres" is included in the JMenu. In the yellow view, the user can input a value that they want to convert in centimetres, the user next clicks "Update model" then selects "Save input centimetres" to request the value be saved. The input value is then retrieved from the yellow view and stored in the Model by the controller. The Model is observed by the green and orange views, these two views are automatically notified when the Controller stores the input value in the Model. Finally, the input value converted in feet is displayed in the green view, while the input value converted in meters is displayed in the orange view.

There are many challenges associated with this software project. For example, the biggest challenge we have faced is how to apply the Observer design pattern and Command design pattern in this project as well as how to apply Model View Controller (MVC) in this project. We will explain all of them in detail as well as how to overcome them in the following paragraphs.

In this software project, we will be using many concepts from OOD, OOD principles and design patterns to carry out the project. Such as Object-oriented analysis and design workflow, Encapsulation, Abstraction, Inheritance, Polymorphism, Observer design pattern and Command design pattern. All of them will be explained in detail in the following paragraphs.

This report includes four main parts including introduction, design, implementation and conclusion, we will structure it strictly followed by the project requirement, all the mentioned details from the document will be explained. The report starts with a brief introduction, after that, we will explain how we have used design patterns and design principles in detail and followed by a design solution with the UML class diagram. Third, we will be presenting the details of the implementation, a short video presentation will be provided. At last, we will give an overall conclusion to this project, explaining what went well, what went wrong and what we have learned from this project as well as the advantages and drawbacks of completing the lab in a group. At last, a table will be provided that shows the various duties allocated to each group member, as well as the percentage of the work done by each group member Indicate whether or not each group member participated.

Design



In this project, our group implemented two design patterns which are observer behavioral pattern and Command behavioral pattern. The reason we choose to use Command pattern is because it could let the parameterize objects by a command to execute. In our case, class ConverterMenuItem inherits the behaviour of JMenuItem. By doing this, when the user commands the user interface with a mouse click event, the ConverterMenuItem will automatically ask the execute method to execute the input. Since the input will be an integer value that matches the value from class CentimersConversionArea. This will lead to the next behavioral pattern we designed which is the observer pattern. After the object centimetre changes its state, we want all its dependents such as feet and meter are notified and updated automatically. Therefore, the observer pattern is the best to implement. ConversionArea is an abstract class also the concrete observer in our case. It has three subclasses which are CentimeterConversionArea, FeetConversionArea and MeterConversionArea class. The modification in the CentimeterConversionArea class's value will affect the other two classes. The execute method in the command pattern will modify the state attribute in class ValueToConvert. Wait until the update is finished, it will invoke each data that has been observed and according to the conversion algorithm in the update method, the new value will be converted and executed.

According to the UML class diagram that we created, we implement Inheritance, Encapsulation, Polymorphism and Abstraction in our project. We design an abstract class which is ConversionArea. It contains one abstract method update. Since it is an abstract method, it does not have a body. Now, we can introduce the inheritance principle. CentimeterConversionArea, FeetConversionArea and MeterConversionArea class are the subclasses of the abstract class ConversionArea. In our code, these three subclasses inherit the method update from its superclass and each of these update methods has a unique conversion algorithm such as converting the input centimetre value to meter or to feet. Until now I believe you could already find out the method to update this single action in our project is performed in different ways. Therefore, principle polymorphism is also applied in our project. The last principle is encapsulation. We implemented this principle in the ValueToConvert class. This class has its own mutator and accessor. The two attribute observers and state are all set to private in order to protect it from accidental access by other classes.

Implementation

*We use “prec” to denote the precondition and “poc” to denote the postcondition.

Class ConverterMain

This class is the main class of the project. It is used to create the application of converting input in centimeters to value in meters and value in feet. The main method *run()* is used to create the JFrame we implemented and set it to visible. It would give an exception if there were something wrong.

Interface Command

This interface is implemented for the command pattern. The method *execute()* would be overridden by the three subclasses of ConversionArea to create three different views.

Class ConverterController

This class is implemented as the controller of MVC. It can use the three methods to send information about updates by views to the model.

Class ConversionArea

Invariant: The serialVersionUID is always 1L. The numbers of rows and columns are 10 and 20.

This class is used to create views for input and display convert values. Its method *update()* would be inherited by the three subclasses to convert the value.

Class ConverterMenu

This class is used to create a JMenuBar named “Update model”.

Class ConverterMenuItem

This class is used to create a JMenuItem named “Save input centimeters”.

Class ValueToConvert

This class is implemented as the subject in the observer pattern. A list of ConversionArea objects is created as the list of observers. The method *getState()* can be used to get the value we input as a double number. And we can use the method *setState()*(prec: the state is not null and the length of numbers should be larger than 0; poc: the state should have been set to the double type of the parameter state, and the method *notifyAllObservers()*(poc: all observers should have been updated) should have been called) to use the method *notifyAllObservers()* to notify all the observers when the state is changed and update these observers. The method *addObserver()*(prec: the area should not be in the list; poc: the area should be in the list and can be used by other methods and classes) is used to add a new ConversionArea object as observer to the list.

Class CentimetersConversionArea

Invariant: The serialVersionUID is always 1L. The numbers of rows and columns are 10 and 20. The color of the background is always yellow.

As the subclass of **ConversionArea**, this class is used to create a view for users to enter a value in centimeters.

Class ConverterJframe

Invariant: The JMenuBar named “Update model” with JMenuItem named “Save input centimeters” and the JPanel that comprises three views are always in the JFrame.

This class is used to create a JFrame and add the JMenuBar and the Jpanel that comprises views to it. When a new object of ConverterJframe is created, it will use the method *initialize()*(poc: same as the invariant) to initialize the content of the frame.

Class FeetConversionArea

Invariant: The serialVersionUID is always 1L. The numbers of rows and columns are 10 and 20. The background color is always green. The value can not be edited.

As the subclass of **ConversionArea**, this class is used to create a view displaying the input value in feet. It uses the overridden method *update()*(prec: the value should be the input in centimeters; poc: the value should have been converted to the corresponding one in feet) to convert the input value (in centimeters) to value in feet.

Class MeterConversionArea

Invariant: The serialVersionUID is always 1L. The numbers of rows and columns are 10 and 20. The background color is always orange. The value can not be edited.

As the subclass of **ConversionArea**, this class is used to create a view displaying the input value in meters. It uses the overridden method *update()*(prec: the value should be the input in centimeters; poc: the value should have been converted to the corresponding one in meters) to convert the input value (in centimeters) to value in meters.

The code is written with Eclipse 2021-09, the version of JDK is JDK-13.0.1.

All other requirements and details can be found in our code.

Conclusion

Overall, we did a good job in this lab. Initially, the MVC(model-view-controller) pattern is used to build the main system of this software as the requirement asked. Classes are assigned to the three packages: mode, view and controller. They are implemented separately, but the controller is associated with the model and view. Contributed by it, the process of subsystem design is eased and with high efficiency.

Moreover, the OOD principles are used a lot in this project with a significantly positive impact. For instance, the principles of inheritance and polymorphism are strictly followed in the design of class ConversionArea and its subclasses: MeterConversionArea, FeetConversionArea and CentimetersConversionArea. Since the generalization relationships did not only exist among the 4 classes mentioned above (most classes in this project are implemented as a subclass), the design can be hard and the code can be complex. But the principles used avoid lots of design debt and messy code, improving the quality and efficiency of the project.

As we mentioned above, our group did an excellent job, not only just satisfying the basic requirements, we try to make this to be a perfect software. This project went pretty well overall, nothing really went wrong, although the coding part is very heavy, the clear instructions and well teamwork let us finish it in a short time. However, we still faced some challenges. The biggest challenge we have faced is how to apply the Observer design pattern and Command design pattern in this project. Initially, we have no experience with the Observer design pattern and Command design pattern, we spent a lot of time reviewing the lectures and all the suggested videos, we also researched many related examples to help us understand the concepts clearly. Finally, we figured out how they work and how to apply these design patterns in our project and found out that actually made our design solution much easier. We practiced and learnt a lot by overcoming this challenge.

From this project, we learned a lot. What I need to mention first is that we learned how to give a good overview of the project. An overview is critical to the software design, especially in this lab. Since this project is designed with the MVC pattern, it's necessary to have an idea about the subsystem design for the three parts: model, view and controller. And it's also expected to think about the association among these parts before the implementation. That's why an overview is important.

Another thing we learned from this project is the use of the Command Pattern and Observer Pattern, which are used for the whole process of implementation. To be honest, they are not easy to be mastered and the process is challenging. But it is certainly worthy. They help to avoid messy code structure, defects and bugs.

Working as a group, we assigned different parts of work to each other. The burden for each member is significantly eased. And we can choose the work we're skilled in to improve the quality and efficiency of the project. Moreover, group work means that we will have a lot of discussions. During the discussions, we can solve each other's problems for the course material and programming skills. In other words, we can improve our ability for programming during group work.

But the disadvantages for group work also exist. It will bring negative impacts to the project design ability. Group work means that a member would have a little chance to take part in every stage of the project. And if we always choose the work we're good at, we would lose the chance to improve our ability for other work.

Xinger Yao 25%	Works on the designing and coding part. Works as the team leader clarifies the team's objectives, makes sure every member understands their role and assigns tasks to members so they can help the team achieve the goals. Gather ideas and lead group meetings.
Chang Liu 25%	Works on the report part. Contribute ideas and suggestions for resolving problems within the group. Monitor interactions and progress, set deadlines to keep members on task, ensure everyone is on track and the project continues to meet its objectives. Check and modify the overall quality of the project.
Jiakang Chen 25%	Works on the report part. Contribute ideas and suggestions for resolving problems within the group. Encouraging communication between group members, ensures that all group members have a chance to participate and learn. Recording team meetings and maintaining documentation of group activities.
Yuyu Ren 25%	Works on the report. Contribute ideas and suggestions for resolving problems within the group. Communicating the project's goals throughout the entire project, providing feedback and concerns on progress as well as identifying and eliminating them. Help the team overcome challenges.