Object Oriented Programming
Programming report
assignment X

Lee Yi Ju Elvis Deng
S3587797 S2772817

June 18, 2018

1 Problem description

In the final assignment a graph editor is required to be generated with a GUI(graphical user interface). Specifically, With the program the users are capable of utilizing functionality like creating and deleting new nodes(vertices), adding new edges and so on to draw an ideal graph on the interface.

This report is written for the users to better clarify how the whole program was constructed. And the whole report is constructed in the following manner: in the next section the problem analysis where the major difficulties encountered during the built-up process would be found. Besides the corresponding solving procedures would be introduced and explained. This follows with the program design, in which how the conceptual ideas were transformed to the demo is to be presented, in particular the knowledge to be employed: including the Model-View-Controller(MVC) structure , Java IO and Java Swing and so on. Afterwards, the evaluation of the whole project would be given to indicate if the final demo corresponds to the primarily ideas. This report concludes with the the discussion part in which the efficiency of the program, the comparisons with the alternative methods would be given.

2 Problem analysis

Before the methodologies to be achieved with coding, it is indispensable to primarily have ideas in mind regarding the architecture of the program. Considering that the program involves the users' aspect and they are not likely to access the data/methods to construct the application(model), instead they simply achieve the goal with the usage of application, in other words they would simply apply what the program presents interface(View). For those reasons we applied the MVC structure into the whole program. Specifically, the MVC architecture consists of three parts: model, view and controller. Taking a basic calculator as an example, if the appliers intend to calculate the multiplication of two values, they would first insert two values then press the buttons represents for this feature. The controllers would then reconise which action the user want to perform and they would send this message to the model to react the action by modifying data, in this case determining the value after multiplication. The view would then get the value instead of the data inside the model at the end and paint the result to the user.

Since we are going to create a graph editor, the model then is represented by an undirected, simple, unweighted graph of a set of vertices that are connected by edges. In that sense there should exist methods to construct corresponding graphs and edges. As for the vertex, we are going make it as rectangle which the 2D position as well as the height width and name should be initialized and specified. The edges, on the other hands, should be created with the notification that which two vertecies it is connected to, and a general model where both created edges and corresponding vertices should contain. Moreover, for data-reuse it is better to have saving and loading feature regarding the data of those objects.

As for the view, which presents the UI through the Panel and frame to the userïijŇthere should be buttons including adding and deleting nodes and adding vertices for the user, and for each of the button contains controllers behind who would react the decision of the user to the model. Then the model would push the data to the view to paint the ideal shapes. What other extended functionality would be adding copy paste shortcuts and redo-undo choice for the user to be more convenient to construct the ideal shape. With those throughs we are going to show the program methods in detail.

3 Program design

For user Interface :
Adding edge : first click two vertex and click add edge button on toolbar
Adding new Vertex : first click add node button on the toolbar and on the button would pop up a textField and insert the name of the vertex and click Enter on the keyboard.
Delete Vertex : click one vertex that want to delet and click delet node.
save file : Click File and save

For program :
In our program, we have three different packages: View, Controller, and Model. We tried to use MVC as a program pattern when we developed this program. Inside the View, there are three classes: GraphFrame.java, GraphPane.java, and VertexView.java.
For GraphFrame.java which is extended from JFrame and use it as the container for Panel which is the place we put all the components (Vertexs, Edges) on. Inside the JFrame there is a function called $createMenus()$. The program uses this function to initialize the menu on the user interface. In the menu, there is a button called file and inside of it there is a menuItem which is called save and allows users to save the current state of the program in a file called data.txt. When users click Save button it will call $save()$ function in GraphModel.
For GraphPanel.java which is extended from JPanel and implements Observer for observing the model class. In this class, the program uses $paintComponent()$ function to paint when users change the state of the components or initialization. When user changed the component position or add some vertex or edges the update() class will be triggered by $notifyObservers()$ function in GraphModel.java and repaint all the components again. There is also a function called $createToolBar()$. It is responsible for creating all the buttons which can allow users to manipulate the component. For VertexView.java which is the view of Vertex that would be shown on the panel. VertexView.java is a class extended from JLabel.java. It is a component we can use it to detect the click and drag action by the mouse. In this way, the Panel component doesn't have to keep listening one the mouse action and this can prevent developers from writing logic code to distinguish if the mouse is on a vertex component or not. In the constructor of VertexView.java, the program add DragLabelAction which is a controller for changing the position data in GraphModel.java. In Contoller package, there are several classes and they are used for calling GraphModel to manipulate the data inside of it.
Inside of Model package, There are three different classes: GraphVertex.java, GraphEdge.java, and GrpahModel.java. For GraphEdge.java and GraphVertex.java it is the data type for the corresponding view. GraphModel.java contains the instances of GraphVertex and GraphEdge. Besides, GraphModel.java is the class extending Observable and there is a $notifyObsers()$ function to trigger view to repaint the components when users manipulate them.
Inside the package, There is also a file called Main.java which is the starting point of this program.

4 Evaluation of the program

As we mentioned in the front , we use MVC as a design pattern for developing. But there is something that we didn't do well : since we use JLabel as VertexView instead of following the instruction in the reader file, there are some pros and cons. For the pros, the mouse action would only be detected inside the vertex view. However, since JLabel is a component in swing library, it can only be added to JPanel by calling add() function instead of paint on the panel and this can let the division of View and Model in MVC be more difficult. Third, we didn't finish the functionality of undo and redo.

5 Conclusions

To conclude, With combination of the Java swing, Java IO and MVC structure, we provided a program relates to graph editor in which the user can decide to draw or delete the an undirected, simple and unweighted graph. Even though the success of making such an editor, there still have more possibilities to extend the features to make user obtain better experience with the usage of UI such as shortcuts including copy paste vertecies and graph or redu and undo methods.