Object-Oriented Programming
Programming Report
Graph Editor

Denis Garabajiu *Constantin Cainarean*
s4142551 *S4142152*

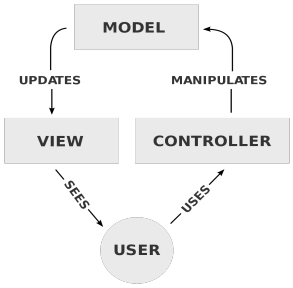
June 26, 2020

1 Introduction

As a task our group had to create a Graph Editor in Java. Mathematically speaking, a graph is a pair G = (V, E), where V is a set whose elements are called vertices (singular: vertex) or nodes, and E is a set of two-sets (sets with two distinct elements) of vertices, whose elements are called edges. Our program creates these graphs, with some functionality such as : drawing a graph, creating a new empty graph, saving a loading the graphs into a file, adding / removing nodes, adding / removing edges, renaming the nodes, selecting and moving the nodes, and of course undo / redo functional. The program allows the user to construct a graph, using the graphical interface. Also the user can save his progress, and load it to work later on the graph. The program is very simple to use, intuitive and efficient for simple graphs.

2 Program design

The program is designed using the Model-View-Controller software design pattern. The program is divided into 3 essential parts. The Model is the central component of the software. This part is accountable for all the data, logic and rules of the program. The View is the part that is seen by the user. It consists of a graphical user interface (GUI) . The pattern which controls these 2, is Controller. This part manages all user input (rather it is on keyboard or mouse) and makes sure that the model treats the changes accordingly to the input, and the view is updated with the following changes. Together, these parts form a functional program used by the user.



Model : In the model section of the program we have 4 Classes :

1. Node
2. Edge
3. GraphModel
4. SaveAndLoad

The names of the classes explain pretty much what do these do. The *Node* class is used for creation of the graph nodes. It contains the information about position , dimensions , name of the node (so that a user distincts a node from another) , and a unique id of the node. All these parameters are used for creation of the nodes, since we store

the nodes as an ArrayList, unique id of the nodes is basically the index of the array, which is allowing us to make further implementations easier and more efficient. The *Edge* class is used for creation of the edges that connect the nodes. This class contains the information of the first and the second node, so that we know the 2 nodes between which is the edge. *GraphModel* class is the most essential class of the program. All the operations occur here. This class contains all data our program needs in order to construct and edit the graph. It holds the collection of nodes and edges in the graph, but it also holds information like what objects are selected, what buttons are pressed, the methods to operate with the input, as well as the UndoManager to allow the user to undo or redo the last changes. We chose to keep the program variables, methods and the graph itself in the same class so that everything we need from the model is accessible in one object.

View : The view package as we mentioned above is what the user sees (Graphical User Interface). Our view section consists of 2 Classes :

1. GraphFrame
2. GraphPanel

The *GraphFrame* class is a class that extends JFrame, it construct the window in which the editor is represented, this window can be resized, it also holds the menu that is filled with buttons to be able to interact with the program. The *GraphPanel*, in Java, the Panel is like the canvas for a painter, it is the place where all drawing, repainting, updating occurs. This class implements Observer, meaning it can detect the changes that are made in Observable classes, to make sure that the changes made will be immediately represented and the user will be able to see them. Everything from the graph structure (nodes, edges, name of the edges) is being represented using this class. We wanted a text scanner on the panel, where the user can type a text that will be used to rename a node. We chose JTextField implementation for this action, because it is more aesthetically pleasant, and also we left this Textfield on the main panel and frame, so the user is able to change the names of the nodes fast and easy.
Controller : This section contains 3 packages : Edit, Actions and Buttons. In the edit package we have the classes necessary for the undo / redo operations.These classes all extend AbstractUndoableEdit, which makes it possible for a UndoManager to handle them, and perform undo and redo actions when the user will request this. In the action package we have the classes that perform the actions conditioned by the buttons, and in button package we have the buttons needed for the graph editing. Also we have 3 Classes :

1. ButtonBar
2. KeyboardController
3. SellectionController

The Button Bar consists of a few simple menus, one for file operations, one for editing operations and the buttons needed for graph editing. Next, we have a few ”Listener” classes: KeyboardController and SellectionController. The first handles all the actions performed by the user on the keyboard, when a key is pressed a specific action is done. This class is also used to move the nodes with the help of the keyboard arrows. SelectionController handles all mouse events. When the mouse is clicked, dragged, released or moved, this controller handles the event and performs the corresponding action. This class is used for node moving with the help of the mouse.

3 Evaluation of the program

We tested in many different ways our program, including with IntelliJ Debugger and all possible combinations of factors that are likely to cause a failure. The final version is as we intended it to be at the beginning. However, we have a known bugs in the program, which might have a minor impact on user experience, when we try to resize the window, after the JTextField appears (when we rename the node) it behaves strangely, occupying the whole screen.

4 Extension of the program

4.1 Keyboard Movement

The nodes can be moved by the arrow keys on the keyboard, this is efficient for precisely positioning a node on the panel.

4.2 Dark Mode

A pleasant dark-themed simple graph editor.

5 Process evaluation

The process starting from the first line to the end of the code and the report was easy at some points which were similar, maybe a bit more complex that the last assignments, and pretty challenging in other, such as UndoManager. This process of course wasn't done without mistakes, a mistake frequently done by us was not creating "new" instances, and wondering why a specific thing doesn't work. From this assignment we consolidated the previous knowledge on GUI, as well as with the Model - View - Controller design structure, also we have learned how to implement undo and redo functions with UndoManager, we discovered how to use JTextField and enjoyed constructing our own Graph Editor.

6 Conclusions

To sum it up, we can afirm that the program satisfies all the requirements, moreover it has some extra functions implemented. The program works properly, satisfying all the user needs from a simple graph editor. The code is consistent and maintainable. As a future extensions for the program we thought about implementing trees, and also the algorithms that can be used on them, as well as algorithms on the graphs, some functionallity to check if the graph satisfy specific properties (undirected , complete, has Euler path etc.).