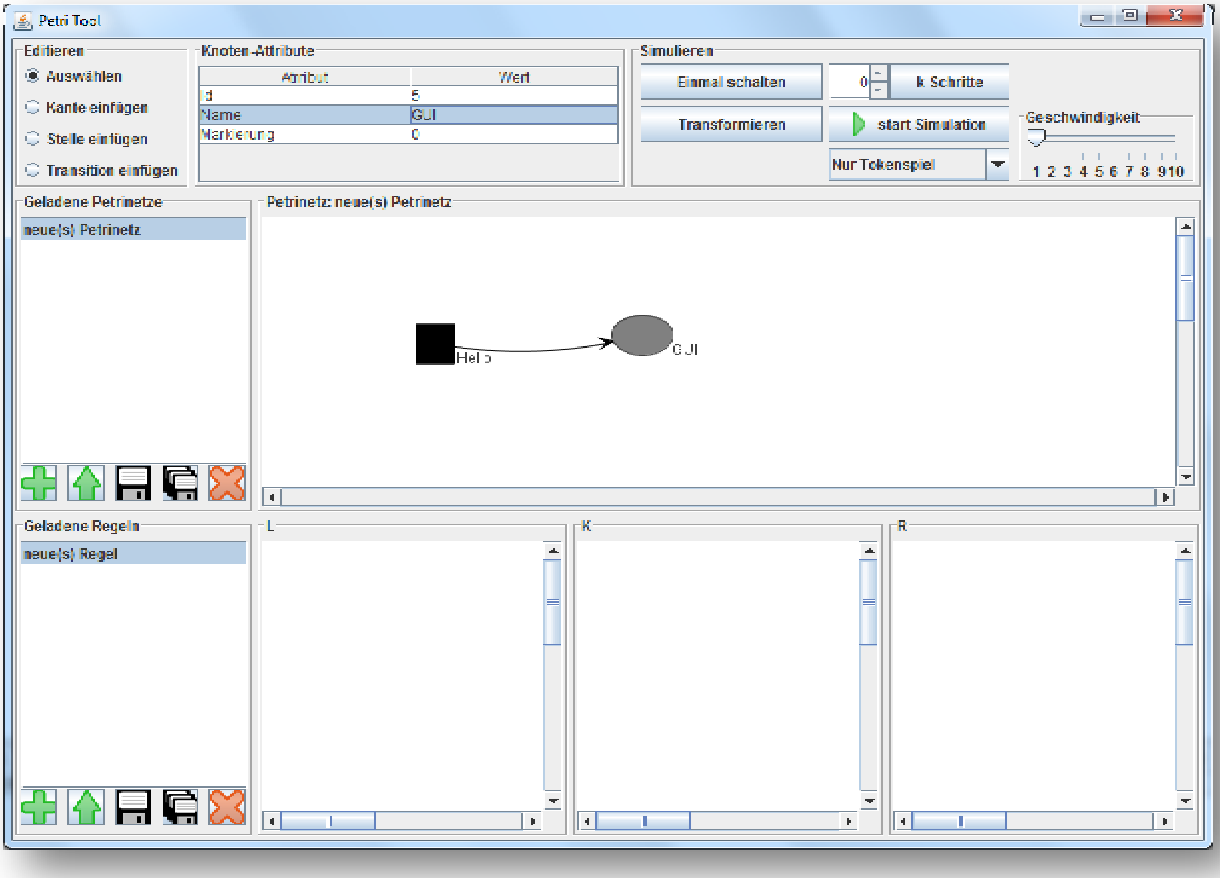
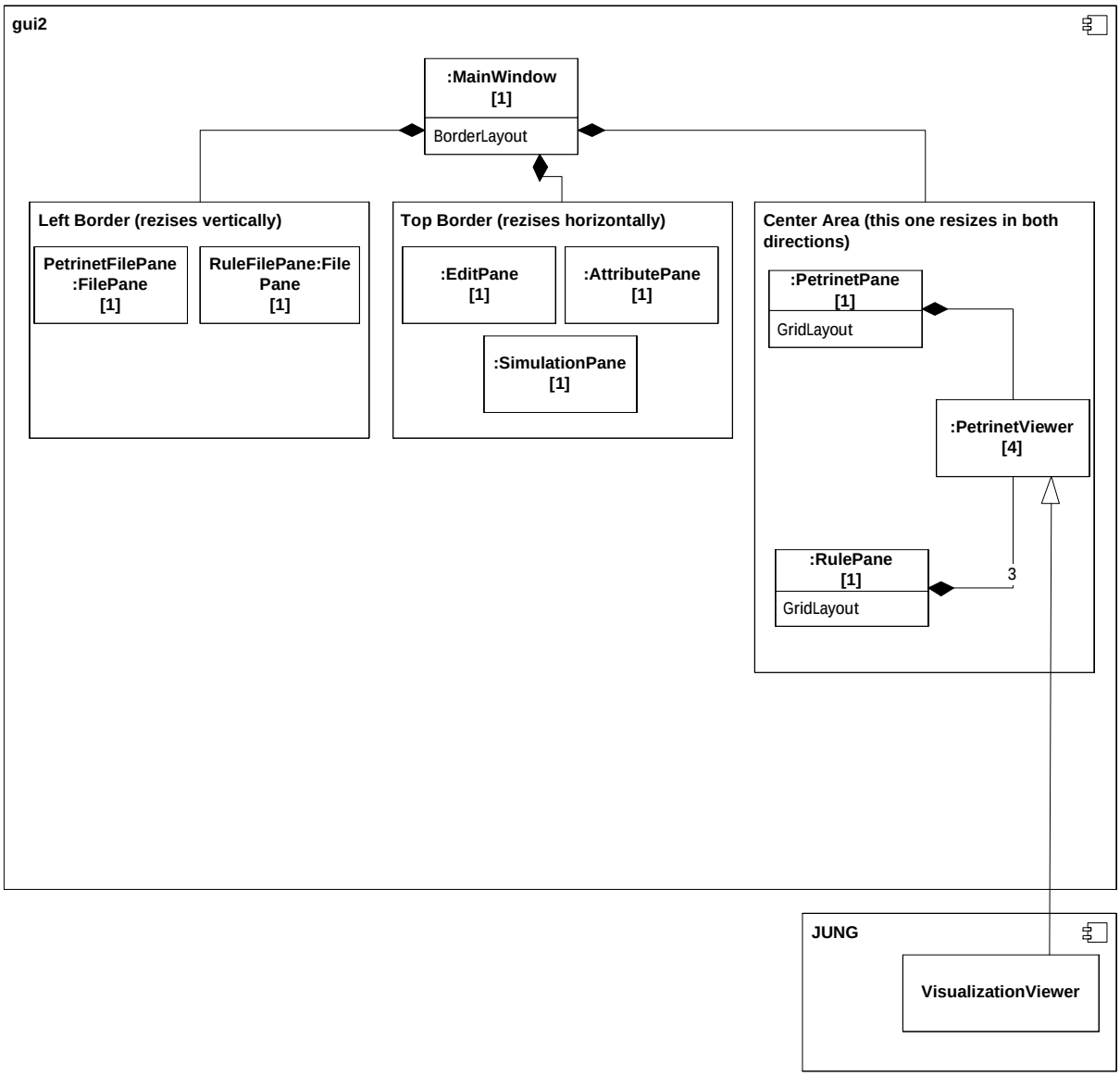
Welcome to GUI

So you've decided to work on the GUI, eh? Atta boy! (Or girl...)

To give you some little introductory help, we created this document for you. So let's get a brief overview:



This is the main window of the GUI. Here you can create, safe and delete petrinets and rules. You can also edit them and start a simulation. You have several areas, which can be used for the different tasks.



So the main window you see when the GUI starts. It holds the three basic areas within its border layout so they adjust themselves automatically when the window is resized.

So for the top area we have 3 tasks: Editing PN, viewing and editing attributes of graph elements, simulating. For each of those panes/tasks we have a class with a single instance.

> EditPane contains 4 radio buttons within one button group (in a group there can only be one button selected). Within the EditPane class there are static private classes – the listeners to the buttons. A Listener is needed for each GUI element so something happens when the user interacts with the element. The Listeners are private because they should not be visible to any other class outside the pain. Other classes outside the pane can just call the getter `getCurrentMode`() to see what radio button is selected.
>
> All the other Panes work in a similar way: They have one or two instances with many private helper classes (mostly listeners) and have getters for information that is needed in other panes.

AttributePane has a Table (each Table-Type needs its own Model) for viewing and editing node attributes. The Model is changed depending on what element is selected. The Listener sets respective attributes from table to actual node.

SimulationPane offers all buttons that are needed for simulations. The slider has an abstract speed value 1 (slowest) to 10 (fastest). The Spinner has a special Model that e.g. accepts only natural numbers.

The left area supplies just 2 tasks:

Saving/loading petrinets and saving/loading rules. As those two tasks are very similar most of the code needed is very similar. We refactored those two Panes into one class with only 2 instances. Some gui texts need to know whether it is a PN or a rule pane, therefore an instance of FilePane.java has the String "type" and its plural form "types".

Also the Listeners differ between the instances so there are many redundant private Listeners.

The PopUp Windows for saving/loading are out-of-the-box elements of swing (JFileChooser). The same goes for all other PopUps (JOptionPane). Also we map between the gui name (list entry) of an PN/rule and the file it is associated with and also between list entry and ID because the ID would be lost after creating a net but the interface to the engine requires an ID.

Now the centers area. Here we have two panes in a grid layout:

The PetrinetPane is just a frame for a PetrinetViewer. It also displays the name of the petrinet on its border.

The RulePane is just a frame for 3 PetrinetViever in a grid layout.

So all the magic happens in the PetrinetViewer. It is a perfectly happy bundle for anyone who wants to display a petrinet (rules are just 3 petrinets). It inherits from JUNG's VisualizationViewer that inherits from JPanel. So a PetrinetViewer can be handled as a special JPanel (can be added to any swing container etc). It handles all the mouse actions (Listeners). E.g.: If the viewer is part of a rule, a change on a node means something different, as other nodes needs to be changed as well. A Listener that is part of the PetrinetViewer of the PetrinetPane (not part of a rule) will call methods of IPetrinetManipulation. But a Listener that is part of the RulePane will call methods of IRuleManipulation.

So someone that uses the PetrinetViewer (e.g. AttributePane) does not need to know whether the petrinet is part of a rule or not. He just says "change the attribute of that node" and the PetrinetViewer knows what to do (see paragraph above).

In a sense the PetrinetViewer is also a connector to the engine as it calls almost any method of the engine.

It also encapsulates the whole JUNG framework from the rest of the GUI.

Now we wish you the best ~~luck~~ fun coding swing.