HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

*MINI PROJECT REPORT*

**Electrical Circuit Simulator**

Object-Oriented Language and Theory

May 26th, 2021

Team 1:

Phan Nguyên Anh - 20194727

Ngô Quốc Thắng – 20194839

Trần Đức Thắng – 20194842

Lecturer:

Nguyen Thi Thu Trang Ph.D.

1. **Assignment of members**

**Members Assignment**

|  |  |  |
| --- | --- | --- |
| **Full name** | **Student ID** | **Assignments** |
| Phan Nguyên Anh | 20194727 | * Use case Diagram * Source code of backend Package * Source code of complex Package * Package Class Diagram for above packages |
| Ngô Quốc Thắng | 20194839 | * General Class Diagram * Use case Diagram * Source code of guiWindow.input Package * Source code of guiWindow.output Package * Source code of circuit Package * Package Class Diagram for above packages |
| Trần Đức Thắng | 20194842 | * Source code of components Package * Source code of guiWindow.drawcircuit Package * Source code of guiWindow.output Package * Source code of circuit Package * Package Class Diagram for above packages |

***For all remaining documents, we do together.***

Now we will describe in detail what did each of members do.

1. Phan Anh

Phan Anh

1. Quoc Thang

In my assignment, all I had done is frontend of the mini-project.

I will divide it into 4 parts to explain:

1. Duc Thang

Duc Thang

1. **Mini-project description**
2. Requirement:

* Frontend
* Create friendly GUI for user to input parameters
  + - Demonstrate two types of circuits: parallel circuit and serial circuit.
  + User can pick a type of circuit by choosing one or two tabs in navigation bar.
* Can build the electrical circuit corresponding to user’s inputs
* Display results table to analyze
* Backend
* Accurately calculate results
* Handle special exception cases

1. Use case diagram and explanation:



Modify Circuit Type: User will choose one of two circuit types (parallel or series)

Modify Source of the circuit: User choose AC or DC Source with its value (Voltage and Frequency for AC or Voltage for DC).

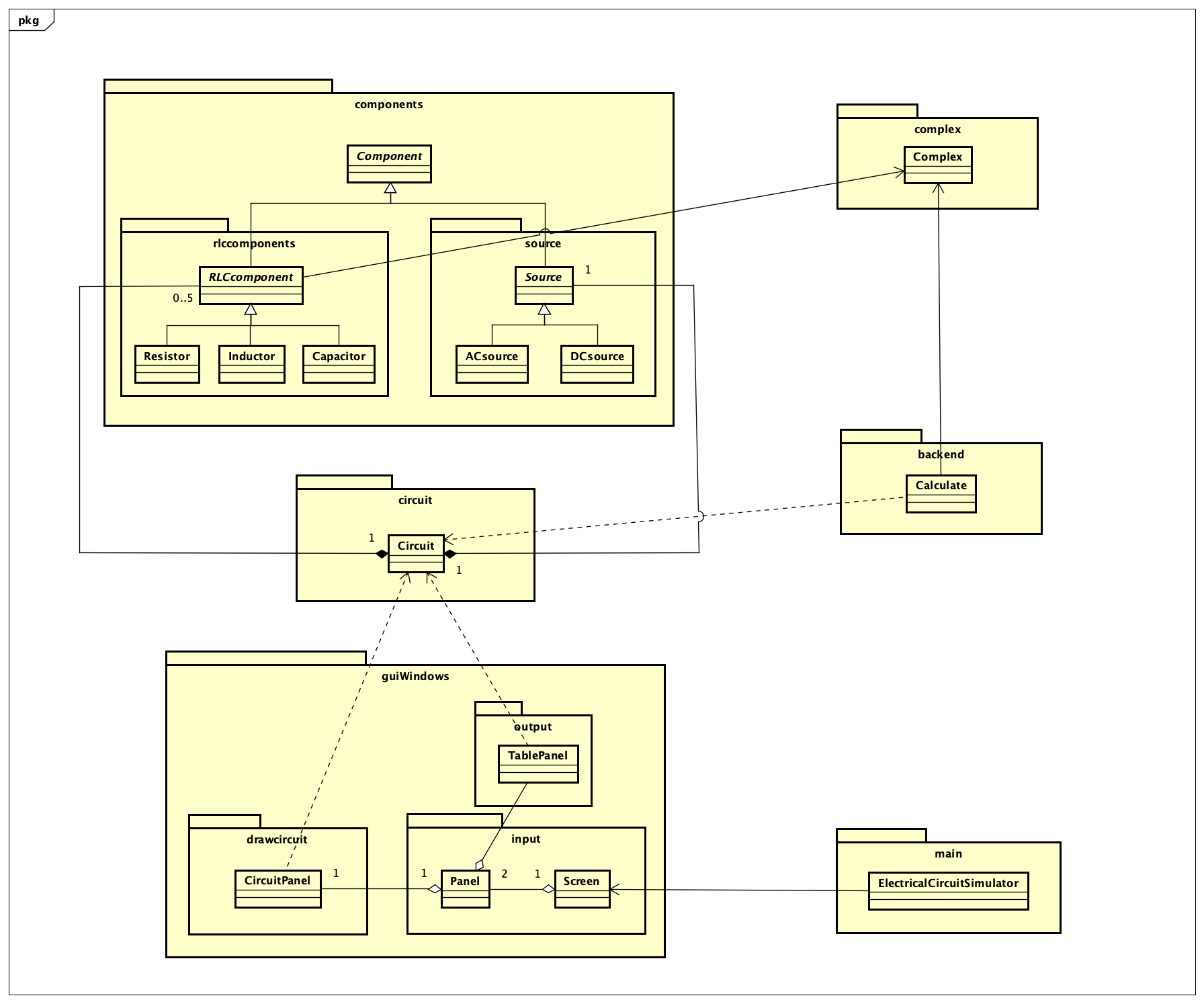
Add Component to the circuit: User can add up to maximum of 5 components

(Resistor, Inductor or Capacitor) with its value.

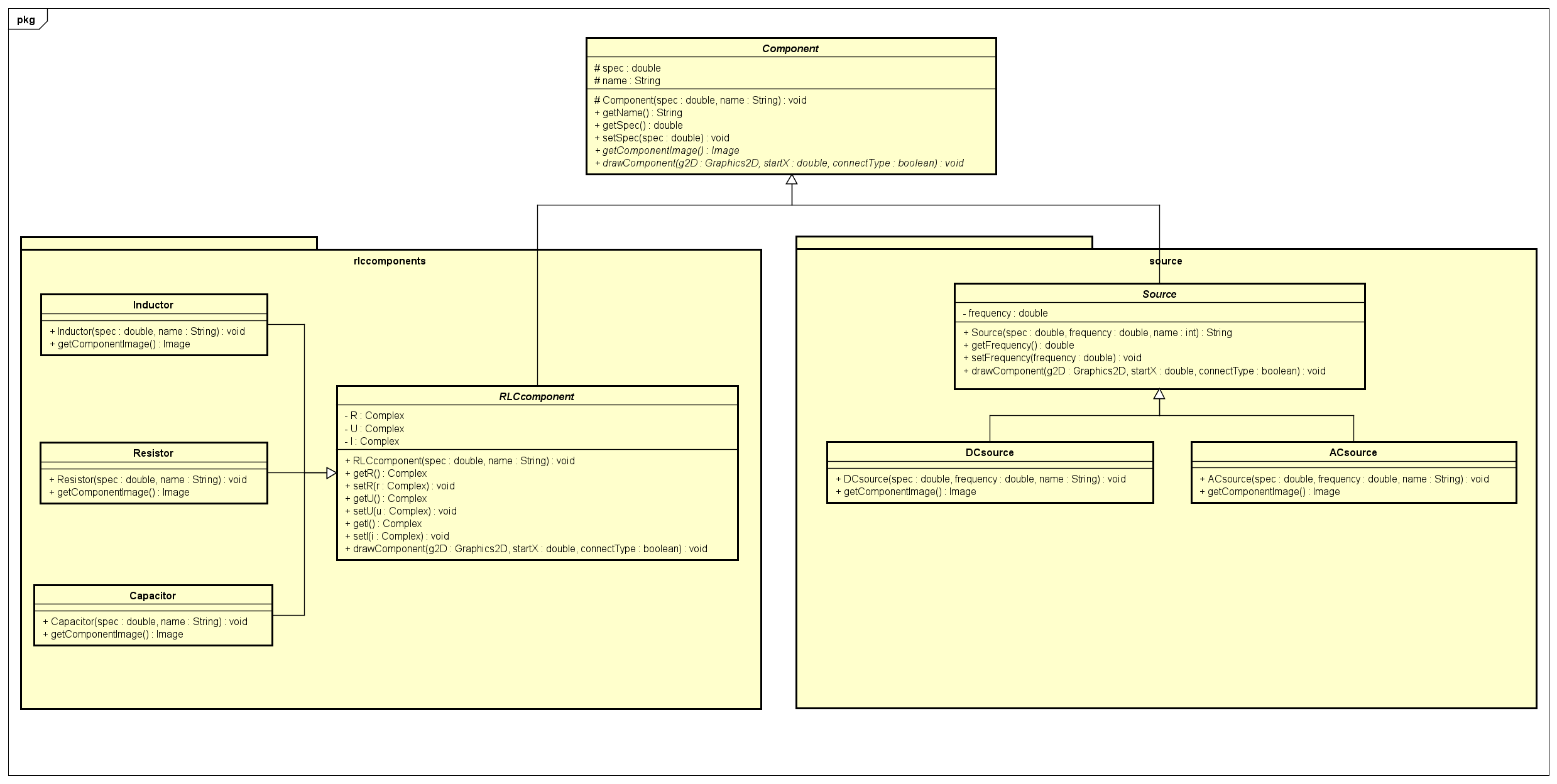
Remove Component from the circuit: User can remove the last component of the current circuit or remove all of them.

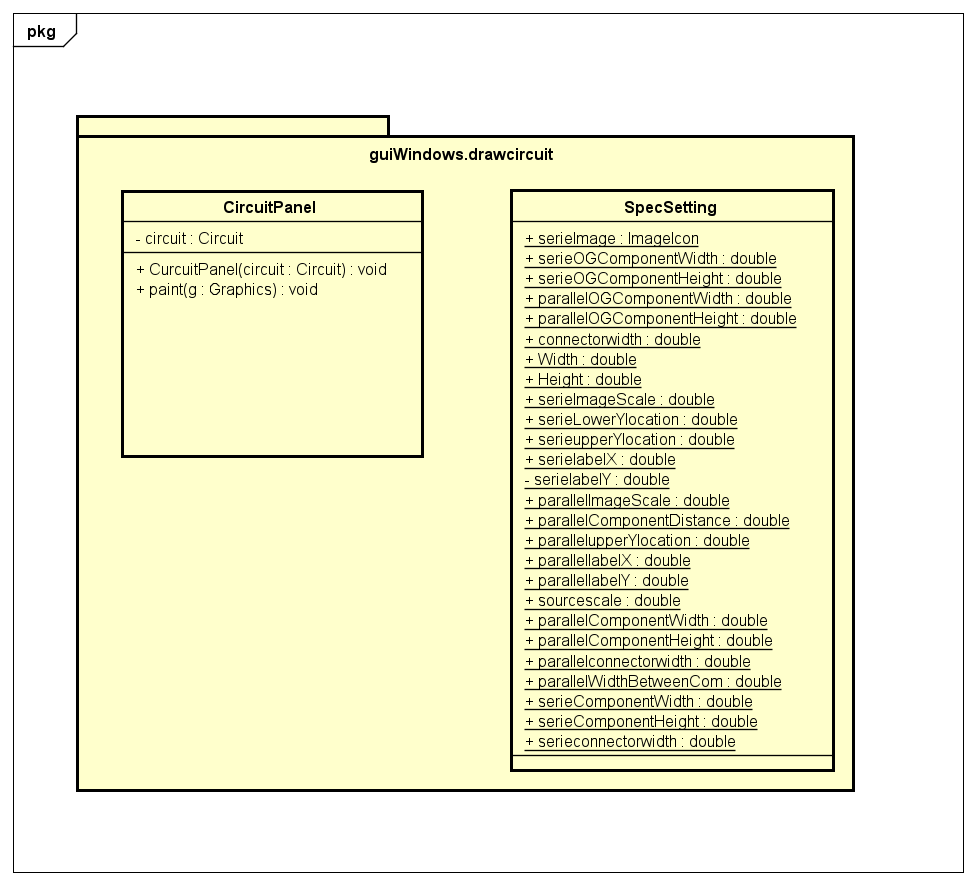
View circuit analysis: User can view the circuit figure corresponding to its components. After pressing submit button, the user can view the results table to analyze.

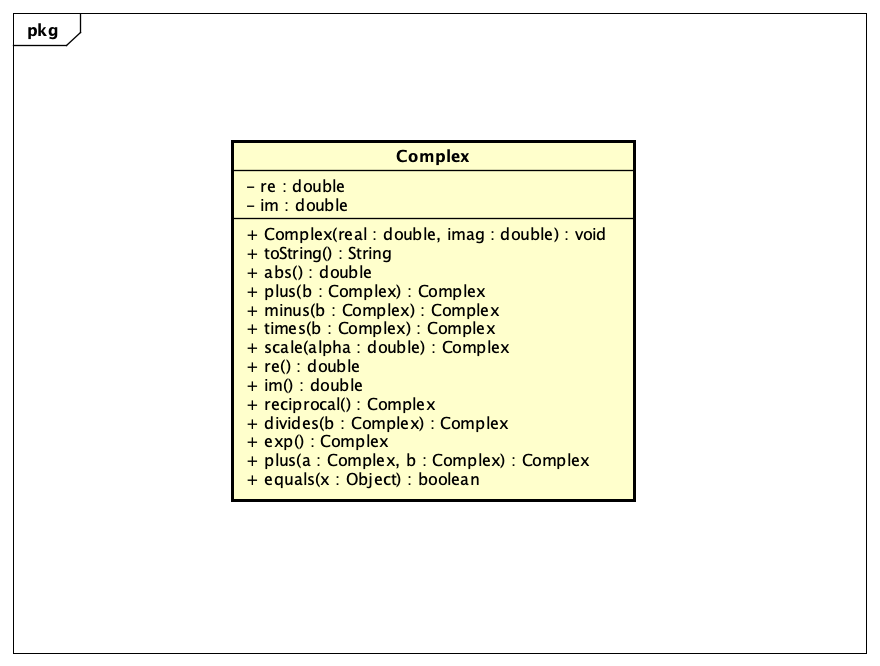
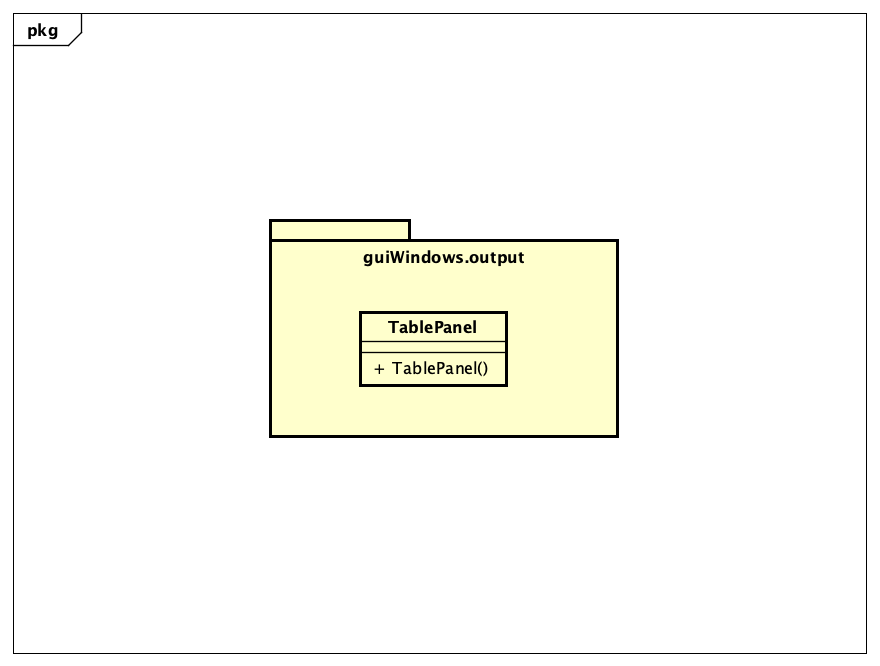
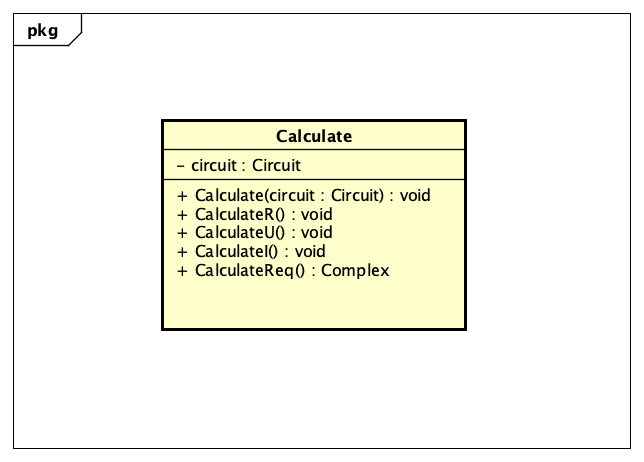
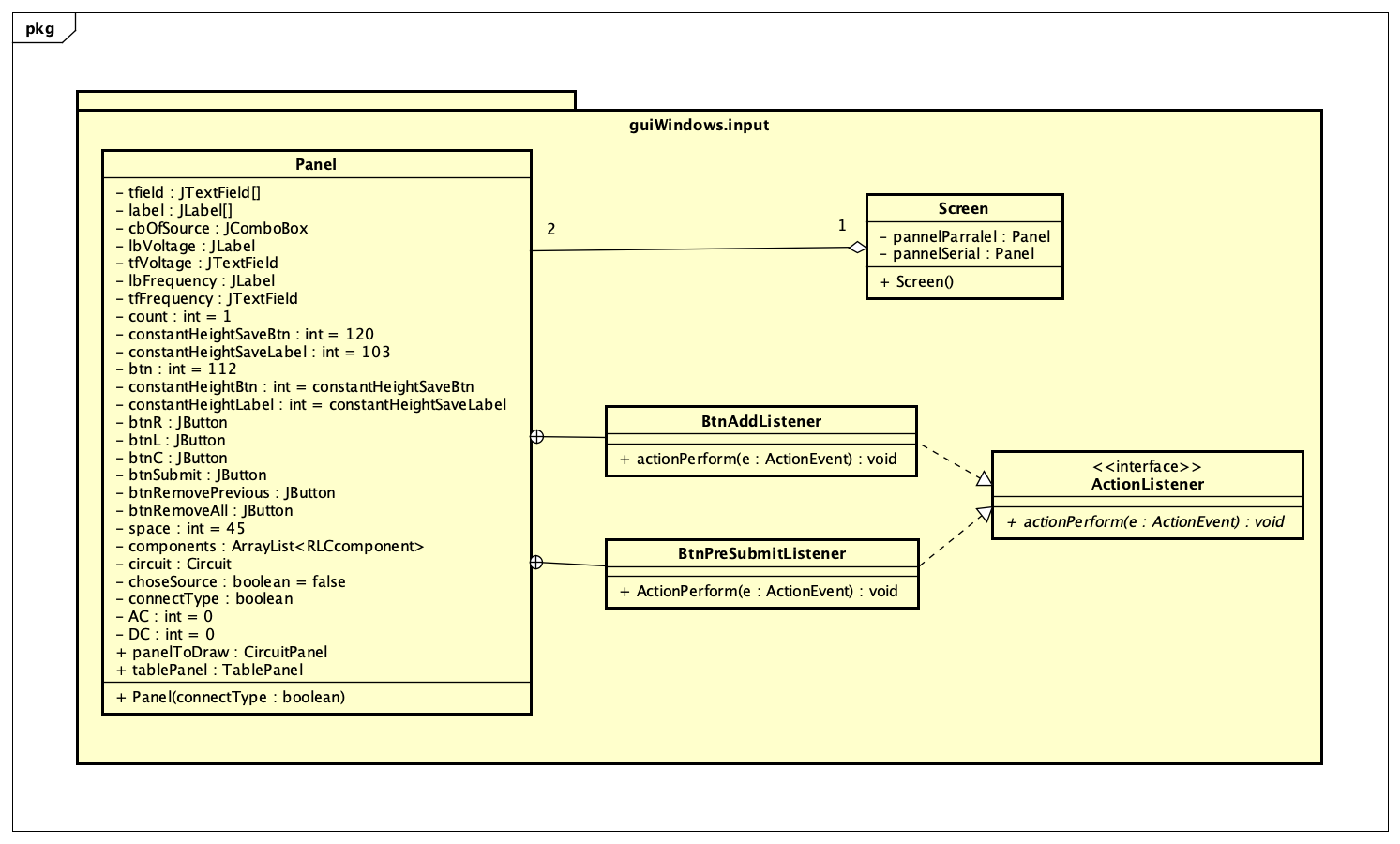
1. **Design**
2. General Class Diagram

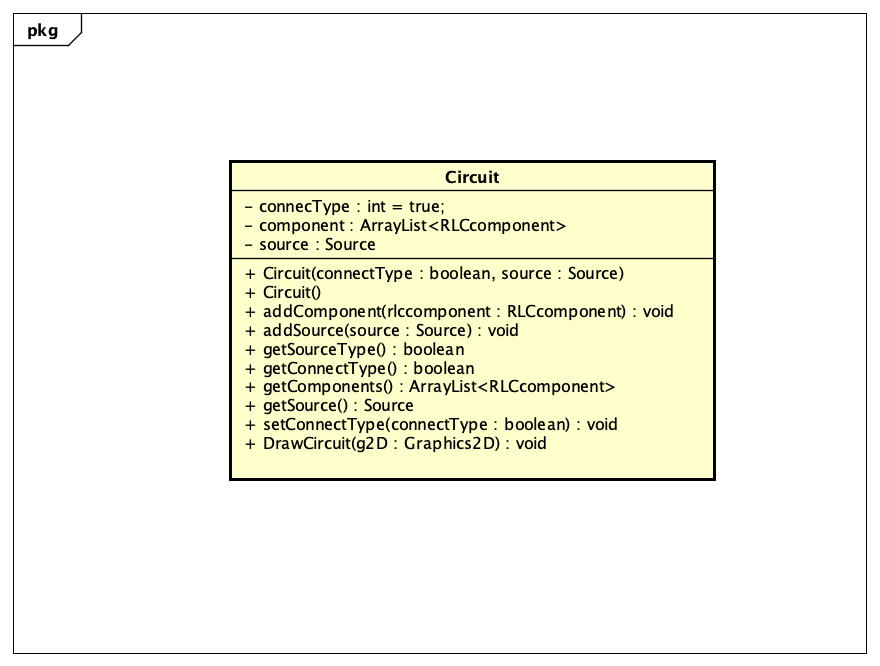


1. Package Class Diagram









1. Explanation of the design

General Class Diagram Explanation:

We want to divide the application into 4 main modules.

Input module: Package guiWindows.input

Purpose: Allow user to input parameters

Input: Circuit parameters

Output: An instance of circuit object

Draw Circuit module: Package guiWindows.drawcircuit

Purpose: Display the circuit figure

Input: An instance of circuit object

Output: A panel includes the input circuit figure

Calculate module: Package backend and complex

Purpose: Calculate accurately the results

Input: An instance of circuit object

Output: Calculate the required values in complex form

Output module: Package guiWindows.ouput

Purpose: Display the analysis table

Input: An instance of circuit object after being calculated

Output: The analysis table

Package Class Diagram Explaination:

Because the circuit has its components (source and RLC components) so we separate it into 2 **packages** (circuit and components) for easier management.

Approaching components package, we use **inheritance** in OOP to utilize source code and be more logical. Since all components have 2 identical attributes which are name and value and same methods also. Although components class has 2 major subclass which are source and RLC component, we do not need instances of these 3 classes (included components class itself). So, we implemented them as **abstract** class.

When applying 4 modules in the mini-project, we often use **Polymorphism** in OOP to upcasting and down-casting objects (specifically when using components object) to process data.

Frontend implementation (some important methods):

Draw circuit method: implemented in circuit class and component class. When we call the method, it will call draw method of each components.

Main panel (in input package): divide screen into 3 main areas (input area, table analysis area and circuit figure area).

Backend implementation (important method):

Calculate method: