**REPORT**

**1.** **Assignment of Members:**

20194436 - Nguyen Duy Hung:

* + ElectricalApplication.java
  + guiController.java (90 %)
  + guiElectrical.fxml

20194448 - Nguyen Hoang Nhat Quang:

* Component, Resistor, Capacitor, Inductor.java
* Source.java
* Calculator, Circuit, ParallelCitcuit.java
* guiController.java (10 %)
* Slide presentation

20194449 - Le Hai Son:

* Serial Circuit.java
* UML diagram
* Report

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

**2.1.** **Project overview:**

The aim of our project is to build an application to create a simple electrical circuit simulator and let users build an electrical circuit and calculate U, I of the circuit.

Lastly, the visualization shows users details about the circuit.

**2.2.** **Design requirements:**

We have Create new circuit button, 2 button types of Circuit for users to choose, 3 buttons types of Component to add, Remove component button, Submit button, Exist button:

+ User chooses type for circuit.

+ After that, we can choose one component to add each time and type the details of its in the text box.

+ Remove component button helps user to remove component.

+ User can only submit and see the visualization when they enter all the details about the circuit they are creating.

+ After submitting, the application shows the image of the circuit with details about the circuit.

+ User can reset all information of their last circuit by using Create new circuit button.

+ User can close the application by Exit button.

**2.3. Use case diagram explanation:**

We have five use cases:

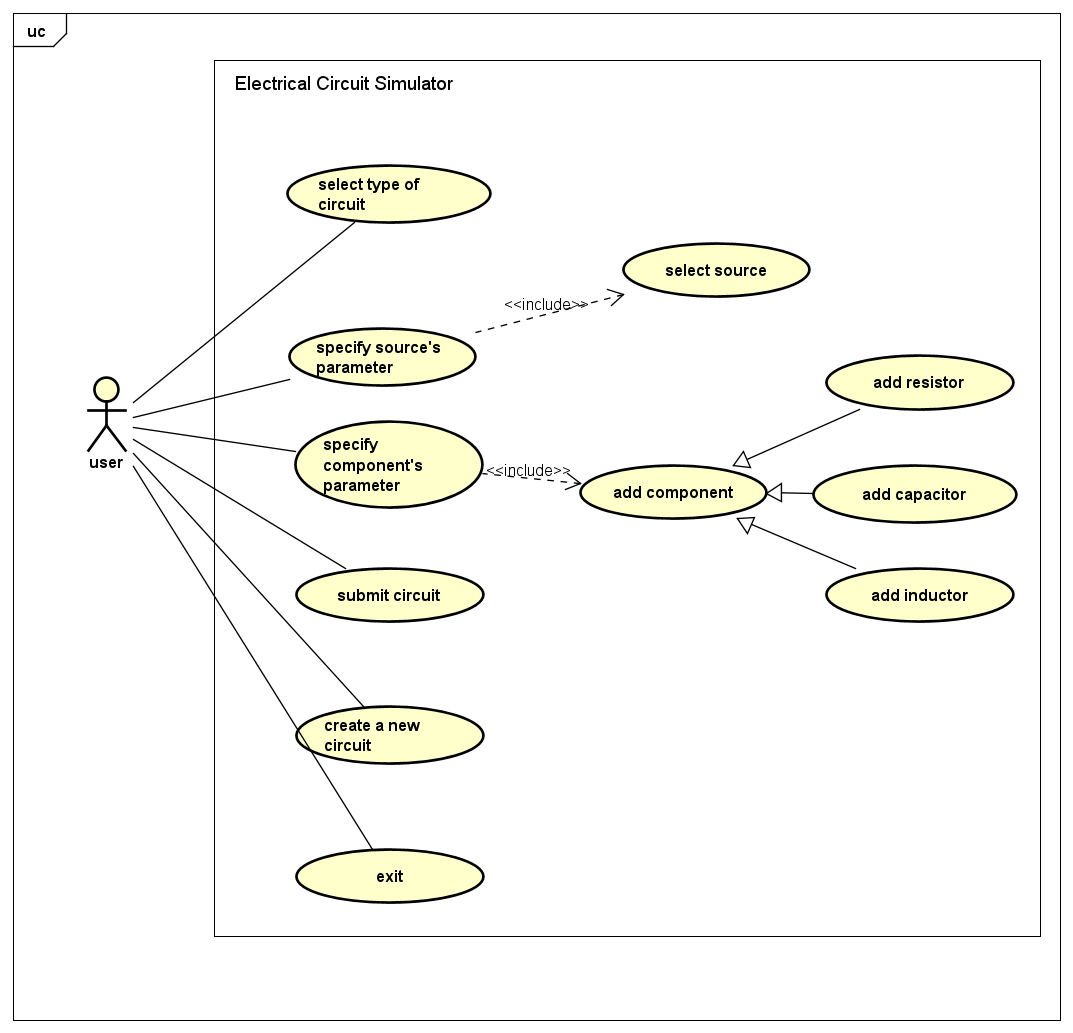


figure 1. Use Case diagram.

To be more specific:

* The application will take details from the user based on GUI: which type of circuit user wants to create (serial/parallel), details about source(AC/DC, Frequency), which component, number of components user wants( max. 5).
* If there is any problem with the input or the user doesn’t supply enough information about the circuit, the application will notice the user.
* After entering all the details, they can submit it. The application will show the visualization of the circuit, moreover, there is a table containing all details that the application has calculated.
* Wait for the user’s command whether they want to create a new circuit or exit the application.

**3. OOP Design idea explanation:**

**3.1. General design:**

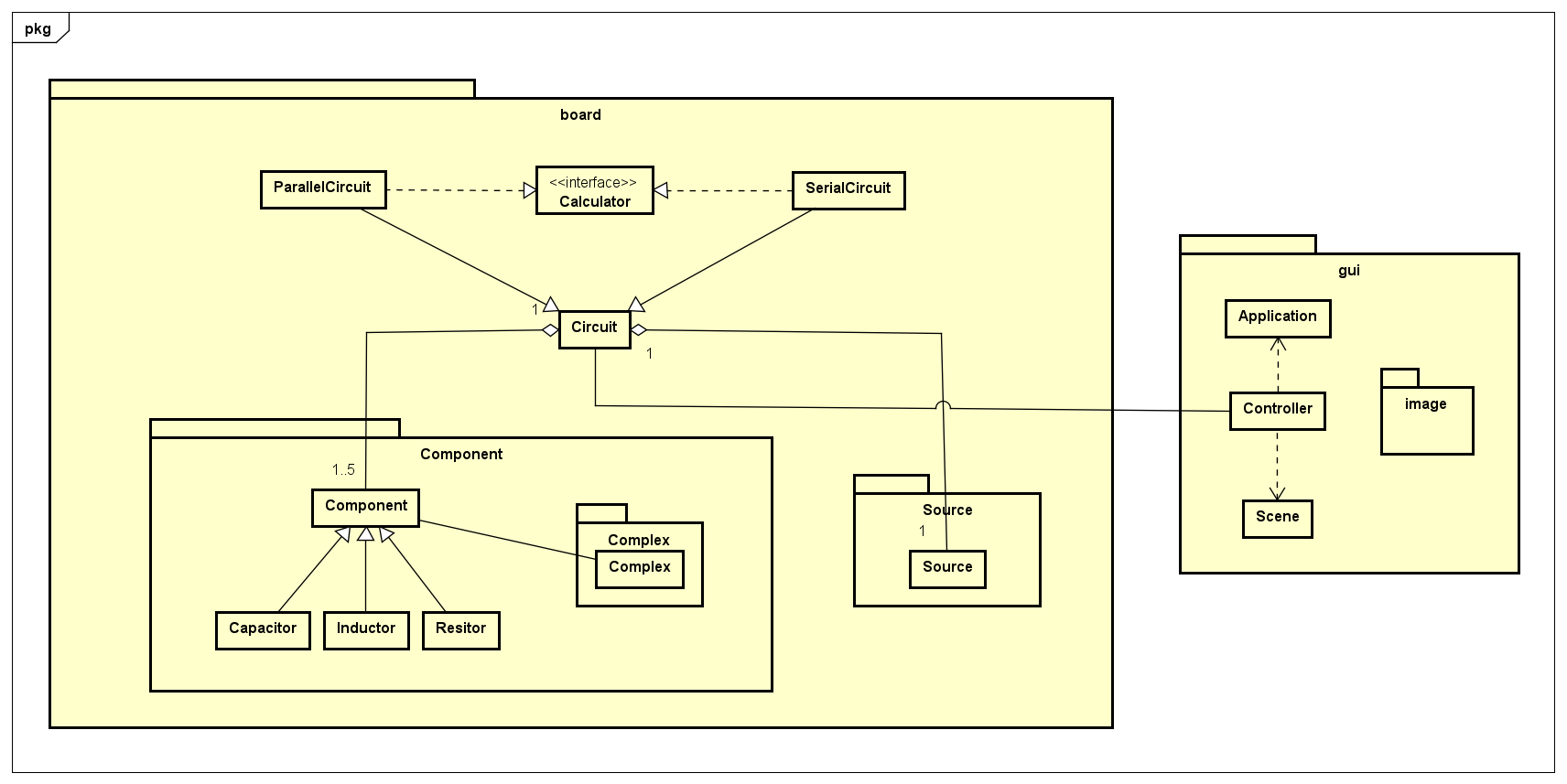


figure 2. General class diagram.

Packages:

* Package ‘board’ contains package ‘component’, ‘source’, ‘complex’ to process information throughout the project.
* Package ‘gui’ (include package ‘image’) contains screen controllers.

**3.2. Package details:**

**3.2.1. Package ‘source’:**

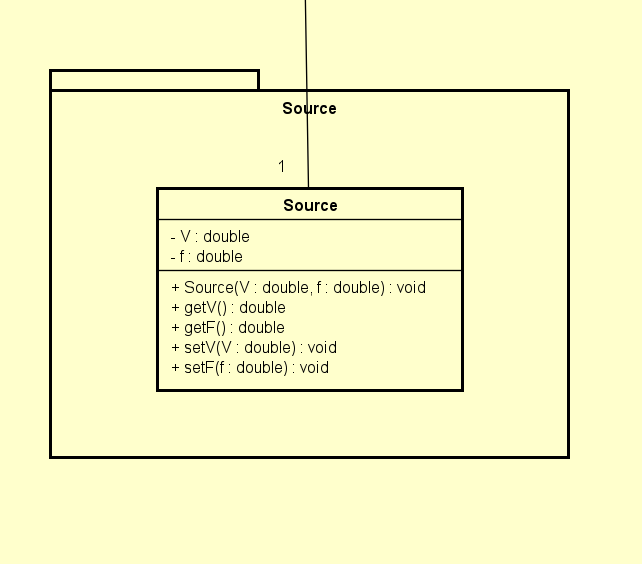


figure 3. package ‘source’.

The class ‘Source’ contains 2 attributes and 5 methods:

* V: voltage of circuit.
* f: frequency
* Source(V: double, f: double): constructor.
* getV(): return input voltage.
* getF(): return input frequency.
* setV(): set initial voltage.
* setF(): set initial frequency.

**3.2.3. Package ‘complex’:**

* We use source code from [*princeton.edu https://introcs.cs.princeton.edu/java/32class/Complex.java.html*](https://introcs.cs.princeton.edu/java/32class/Complex.java.html)*)*
* Have attributes and methods to process complex numbers, which are used to describe values of components.

**3.2.4. Package ‘Component’:**

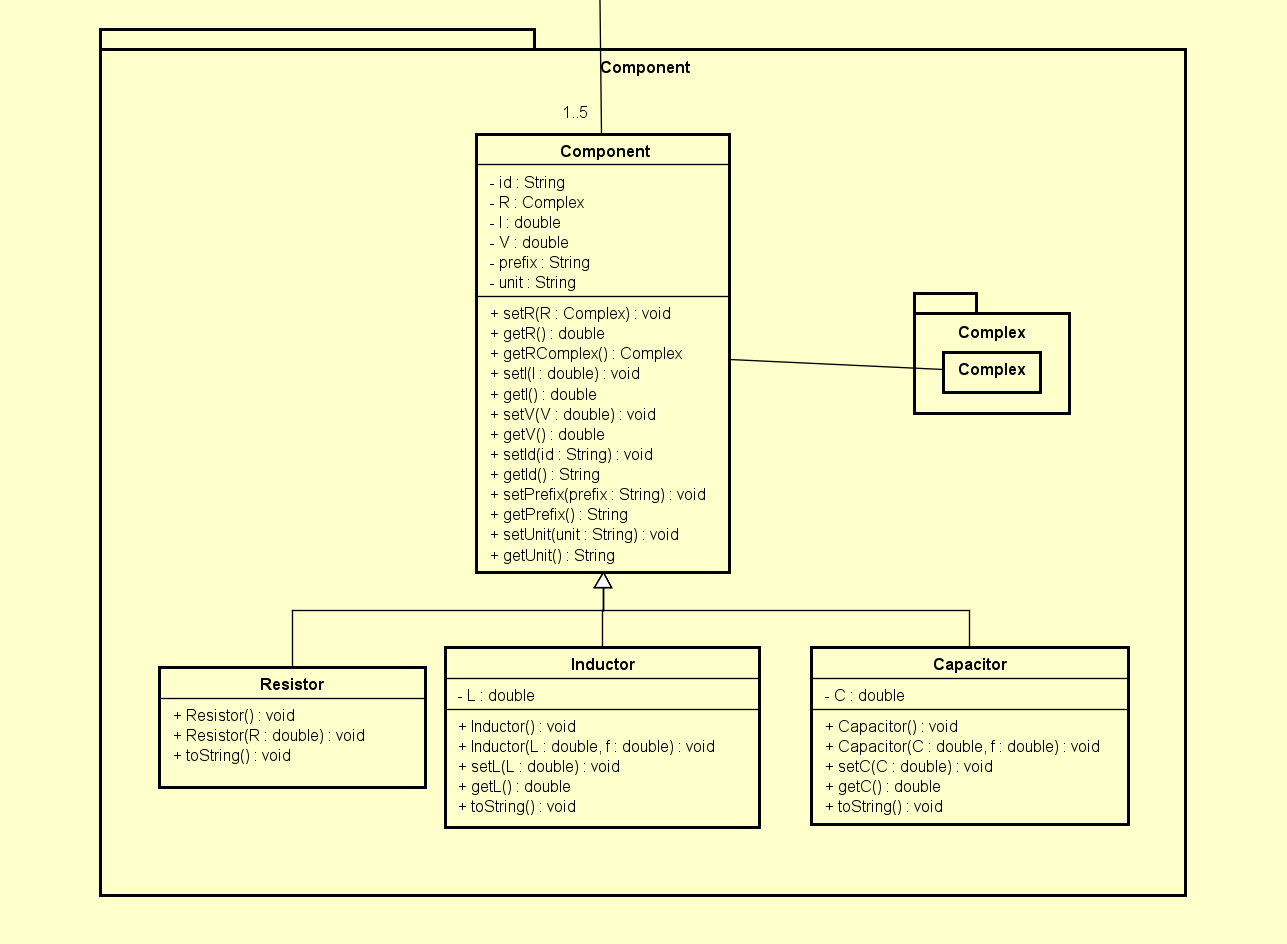


figure 4. package ‘Component’.

* Contains abstract class Component and class Resistor, Inductor, Capacitor inherit from Component class.
* In class component:
* 5 attributes: id, R: impedance, I: current, V: voltage, prefix, unit: sign of component.
* There are setter/getter methods for those attributes.
* In 3 class Resistor, Inductor, Capacitor:
* Every class has methods to interpret the prefix, value of each component.

**3.2.5. Package ‘board’:**

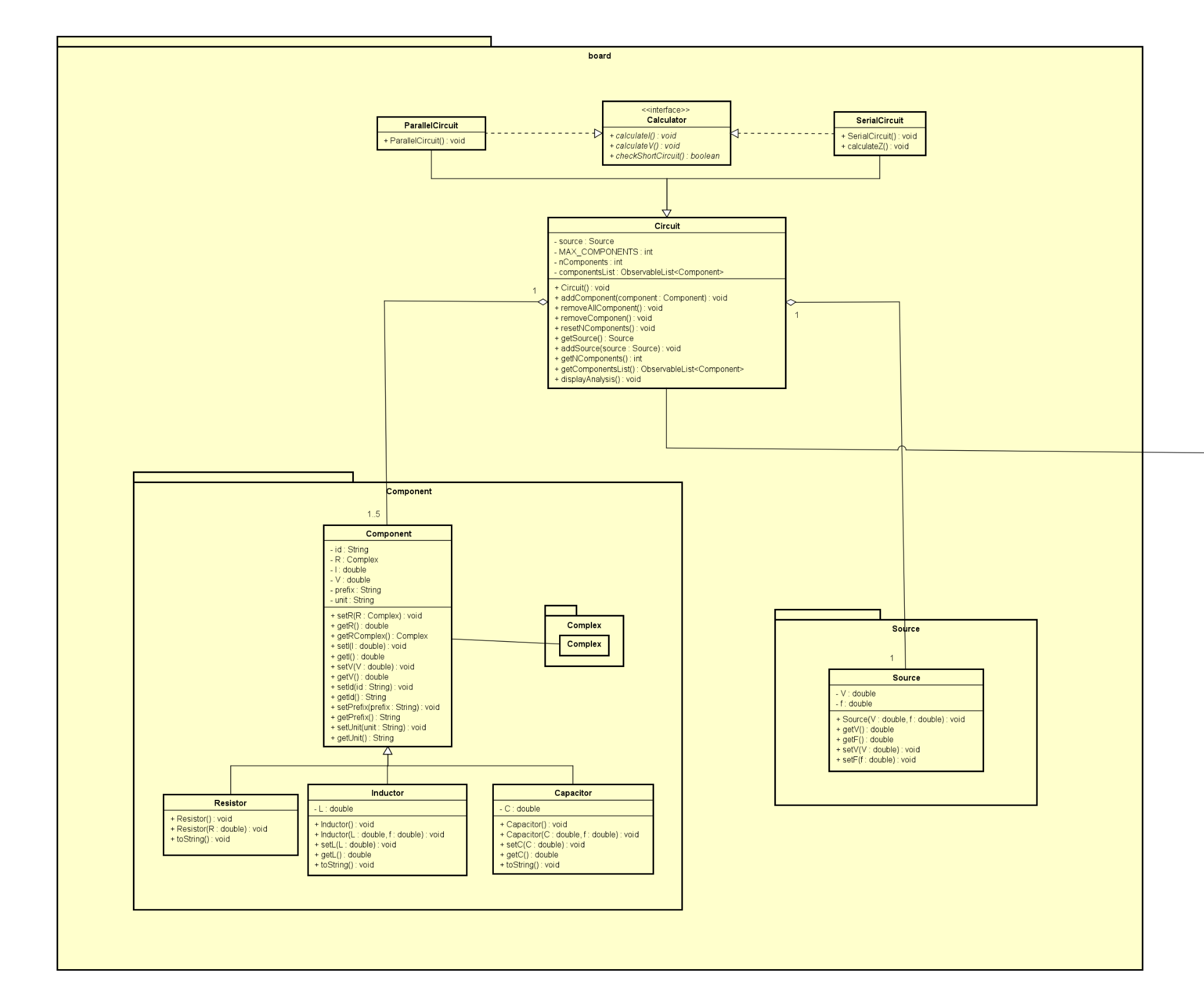
****

figure 5. package ‘board’.

* Contains package ‘Component’ and ‘Source’.
* Class Circuit:
* Attributes:
  + source: contains detail of circuit (voltage, frequency).
  + MAX\_COMPONENTS: restrict number of components added
  + nComponents: current number of components added.
  + componentsList: contains all components added.
* Methods:
  + Circuit(): constructor.
  + addComponent(),removeComponent(), resetComponent(): fix the number of components added to component list.
  + getSource(),addSource(): gain, fix information about voltage, frequency of the current circuit.
  + getNComponents(): return number of components.
  + getComponentsList(): return a list containing all added components.
  + displayAnalysis(): show details of all calculations of the circuit.
* We also have class ParallelCircuit and SerialCircuit implements interface Calculator which calculate voltage, current of the circuit and check if the circuit is short:
* calculateV()
* calculateI()
* checkShortCircuit()

**3.2.6. Package ‘gui’:**

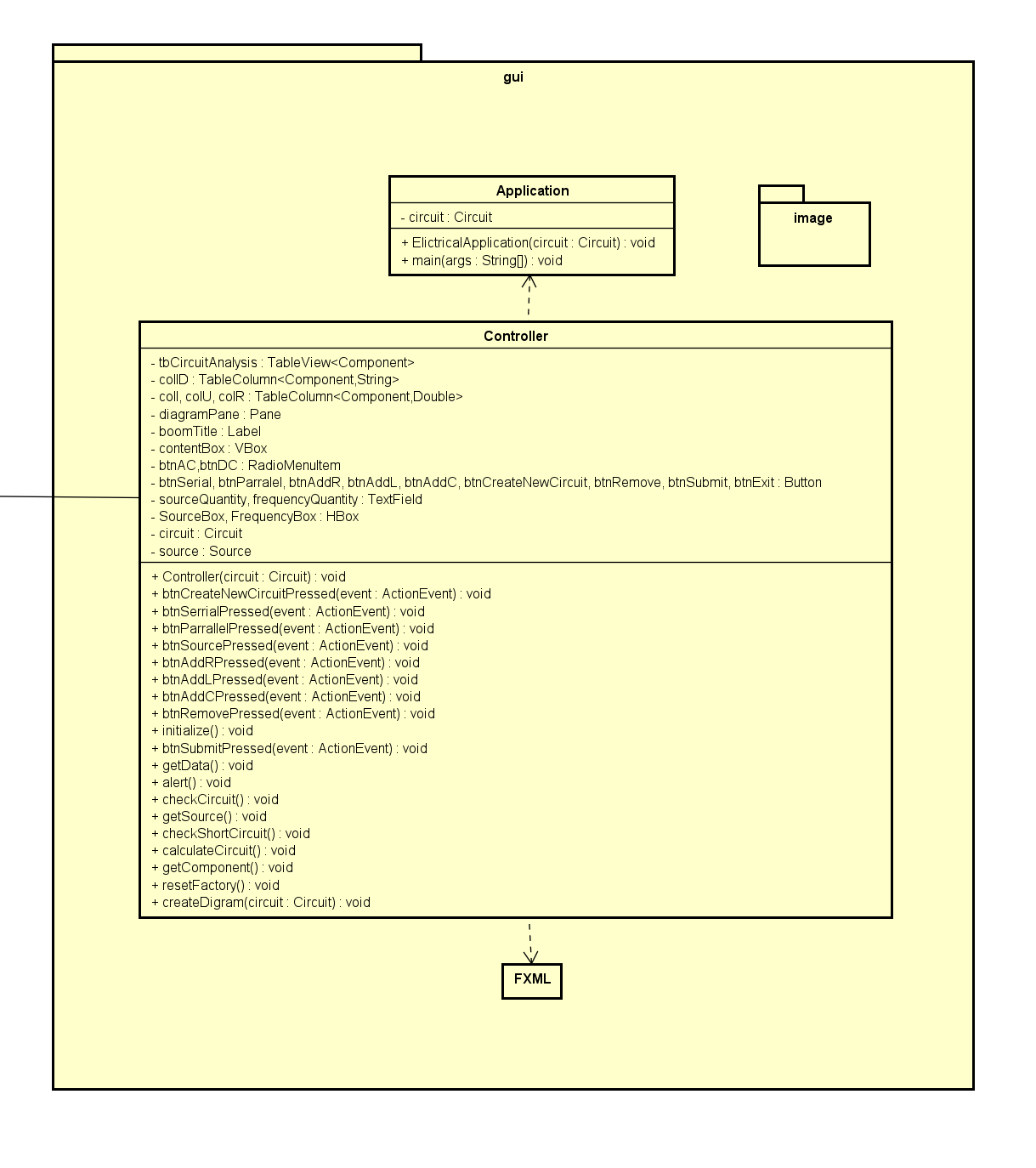
****

figure 6. package ‘gui’.

* Class Controller:
* Methods: Besides fundamental methods, there are some more important methods that help the application run smoothly:
  + alert(): notice when there is not enough information about the circuit and the user must enter more details so as to run the application.
  + getData(): get analysis information of the circuit to display.
  + createDiagram(): visualize the circuit.
  + resetFactory(): used to clear last information to create a new circuit.