



SOICT

ELECTRICAL CIRCUIT SIMULATION

MINI-PROJECT PROPOSAL

CLASS CODE: 141177

OUR GROUP INCLUDES FOUR MEMBERS:

Vũ Lâm Anh	20214876	anh.lv214876@sis.hust.edu.vn
Trương Gia Bách	20210087	bach.tg210087@sis.hust.edu.vn
Nguyễn Thanh Bình	20210106	binh.nt210106@sis.hust.edu.vn
Lê Ngọc Bình	20214878	binh.ln214878@sis.hust.edu.vn

SUPERVISOR: PhD Nguyễn Thị Thu Trang

CONTENTS

Table of Contents	1
1 Assignment of members	1
1.1 List of Tasks	1
2 Mini-project Description	3
2.1 Mini-project overview	3
2.2 Mini-project requirement	3
2.3 Use Case Diagram	4
2.3.1 Set up Electrical Circuit Type	4
2.3.2 Adding Circuit Components	4
2.3.3 Review on Circuit Display and Circuit Analysis	5
3 Design	6
3.1 General Class Diagram	6
3.2 Package Details	7
3.2.1 model package :	7
3.2.2 GUI package	10
3.2.3 Execute	17
4 References	19

Chapter 1

ASSIGNMENT OF MEMBERS

1.1 LIST OF TASKS

In this part, we will inform the work each group member has done by listing classes and methods where he mainly involved and then enumerate all materials which we based on in this project. We break our project into four separate parts for the sake of assigning tasks for each member.

Member ID	Full Name	Student ID
1	Vu Lam Anh	20214876
2	Truong Gia Bach	20210087
3	Le Ngoc Binh	20214878
4	Nguyen Thanh Binh	20210106

Task	Responsible Member ID	Percentage Contribution
General Class Diagram	Nguyen Thanh Binh	40%
	Truong Gia Bach	60%
Detail Class Diagram	Vu Lam Anh	20%
	Le Ngoc Binh	70%
	Nguyen Thanh Binh	10%
Use-case Diagram	Le Ngoc Binh	
Model Package	Vu Lam Anh	5%
	Truong Gia Bach	20%
	Nguyen Thanh Binh	75%
View Package	Vu Lam Anh	25%
	Truong Gia Bach	35%
	Le Ngoc Binh	20%
	Nguyen Thanh Binh	20%
Report	Nguyen Thanh Binh	40%
	Le Ngoc Binh	60%
Slides	Vu Lam Anh	

Details for classes

- **ElectricalElement package**
 - Capacitor.java: 2, 4
 - ElectricalElement.java: 2, 4
 - Inductor.java: 2, 4
 - Resistor.java: 2, 4
 - TestElement.java: 1
- **VoltageSource package**
 - AC.java: 2, 4
 - DC.java: 2, 4
 - VoltageSource.java: 2, 4
- **Circuit package**
 - Circuit.java: 4
 - ParallelCircuit.java: 4
 - SerialCircuit.java: 4
 - TestCircuit.java: 1
- **CircuitDrawing package**
 - ACGUI.java: 3
 - CapacitorGUI.java: 4
 - DCGUI.java: 3
 - ElementGUI.java: 4
 - InductorGUI.java: 4
 - MainGUI.java: 3
 - ResistorGUI.java: 4
 - TestDrawing.java: 3
- **screen package**
 - AddAC.java: 2
 - AddCapacitor.java: 2
 - AddComponent.java: 2
 - AddDC.java: 2
 - AddInductor.java: 2
 - AddResistor.java: 2
 - BuildScreen.java: 1, 2
 - DisplayScreen.java: 1

Chapter 2

MINI-PROJECT DESCRIPTION

2.1 MINI-PROJECT OVERVIEW

In this project, our group needs to create an Electrical Circuit Simulator that let users build an electrical circuit and perform circuit analysis on it.

Throughout the collaborator, we control our progress by using Version Control, specially on Github.

To build this Simulator successfully, we also need to construct an Use Case Diagram and Class Diagrams in the development process.

For GUI, we decided to use Swing as our UI toolkit because Swing provides a rich set of GUI components such as buttons, labels, text fields, tables, and more. It also includes layout managers for arranging components within containers, event handling mechanisms for capturing user interactions, and various utility classes for handling graphics, fonts, and colors.

2.2 MINI-PROJECT REQUIREMENT

As mentioned above, we have to develop a Simulator to display a custom circuit and the circuit analysis which strictly follow the Modern Physic Rules.

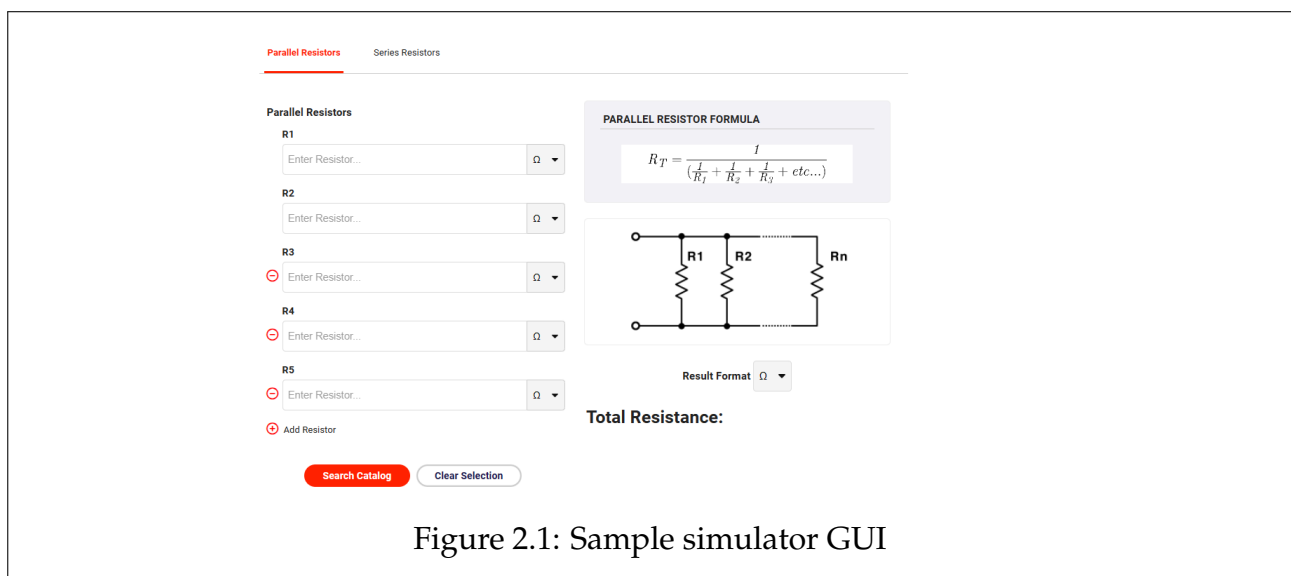


Figure 2.1: Sample simulator GUI

About GUI, Figure 2.1 illustrates a demo Interface which include two main parts: Choos-

2.3. USE CASE DIAGRAM

ing two types of circuits: Parallel circuit and serial circuit by choosing first or second tab on the navigation bar; Adding Circuit Components with values and unit then submit.

In our simulator, Users can add difference kinds of components: resistors, capacitors, inductors. After pressing the "Submit" button, a Circuit Display and Circuit Analysis will appear on the next frame.

Short Circuit Detection is also implemented to warn the Users whenever they are constructing an Short Circuit in order to make appropriate changes on Circuit Structure.

2.3 USE CASE DIAGRAM

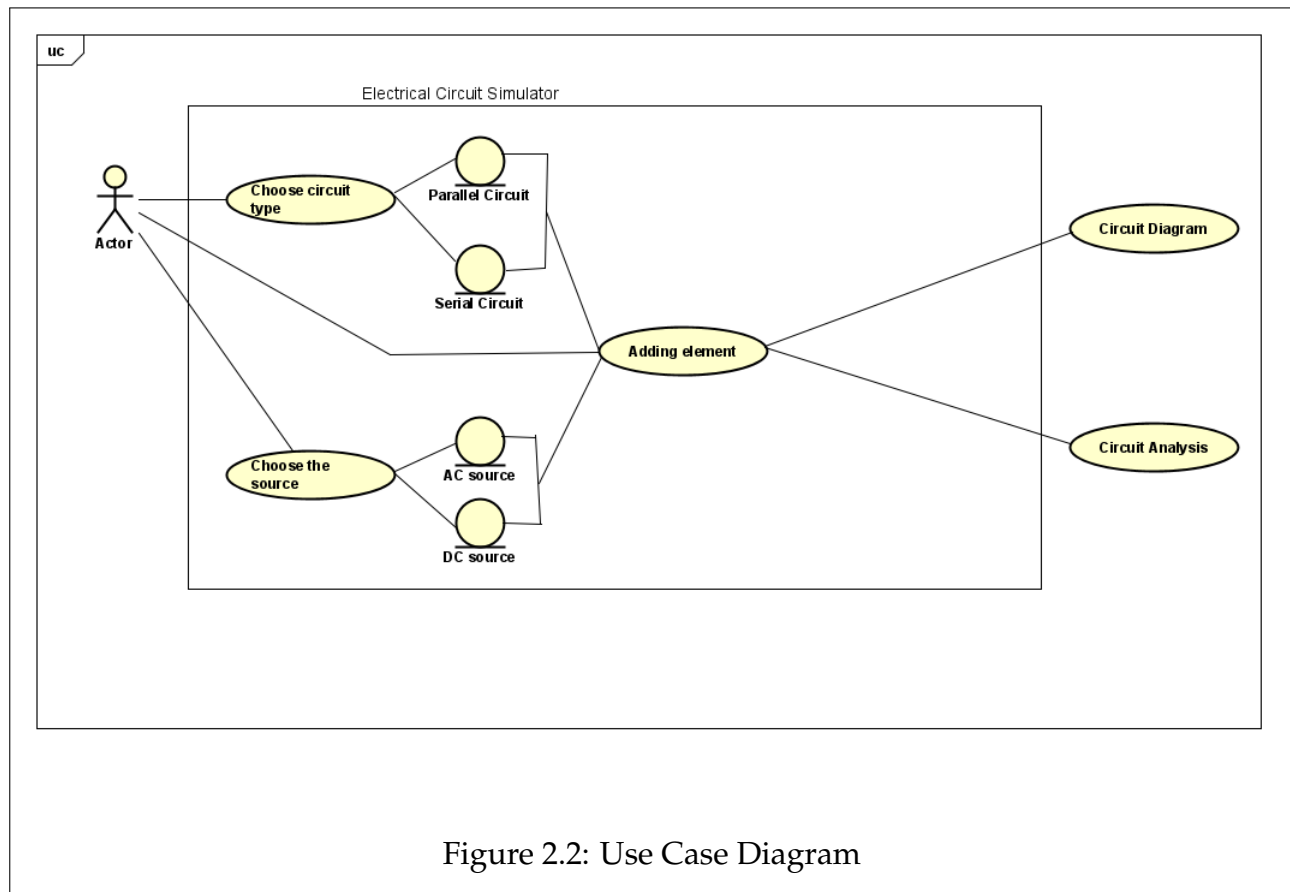


Figure 2.2: Use Case Diagram

2.3.1 Set up Electrical Circuit Type

- At first, Users have to choose which type of Circuit they want to construct: Parallel Circuit or Serial Circuit.
- While setting up the Circuit, Users also have to choose which type of Voltage Source: AC Source or DC Source. Users will add the value of Voltage Source following with the unit.
- In Figure 2.3, We choose Parallel Circuit from navigate tab and choosing AC Source with Voltage of 200V and frequency of 50Hz

2.3.2 Adding Circuit Components

- Users will choose what components to be added into the Circuit with maximum number of components is 5.

- By adding new components with values, the Simulator will inform the Users if Short Circuit is detected.

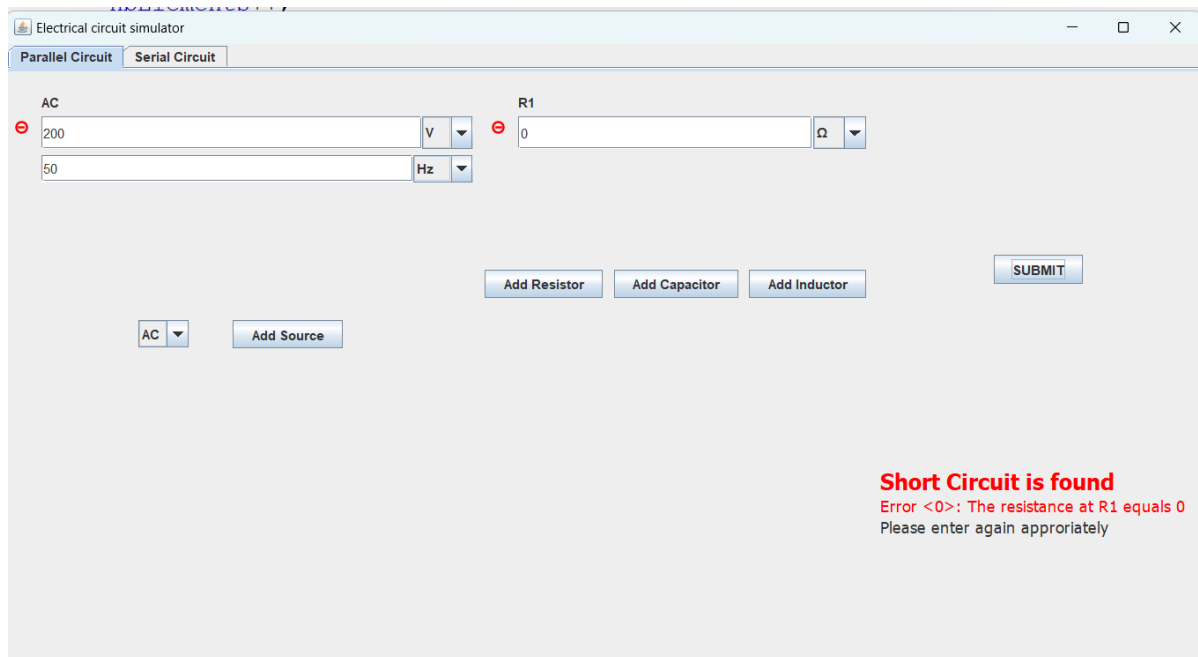


Figure 2.3: Short Circuit Detection

2.3.3 Review on Circuit Display and Circuit Analysis

- After pressing the "SUBMIT" button, a new frame will be displayed with the image of constructed circuit with a table of circuit analysis.
- From the result, Users can easily determine what to substitute and go back to the first frame to re-construct the circuit

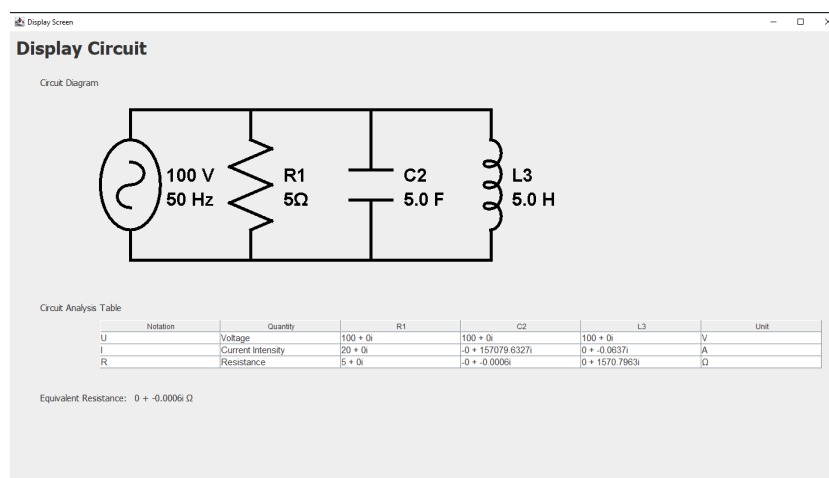


Figure 2.4: Example Circuit

Chapter 3

DESIGN

3.1 GENERAL CLASS DIAGRAM

Our General Class Diagram consists of 3 packages: Execute, GUI and model.

In **model** package, we divided three main sub-packages which are **Circuit**, **ElectricalElement** and **VoltageSource**.

The same in **GUI** package, we split it into two main sub-packages which are **screen** and **CircuitDrawing**

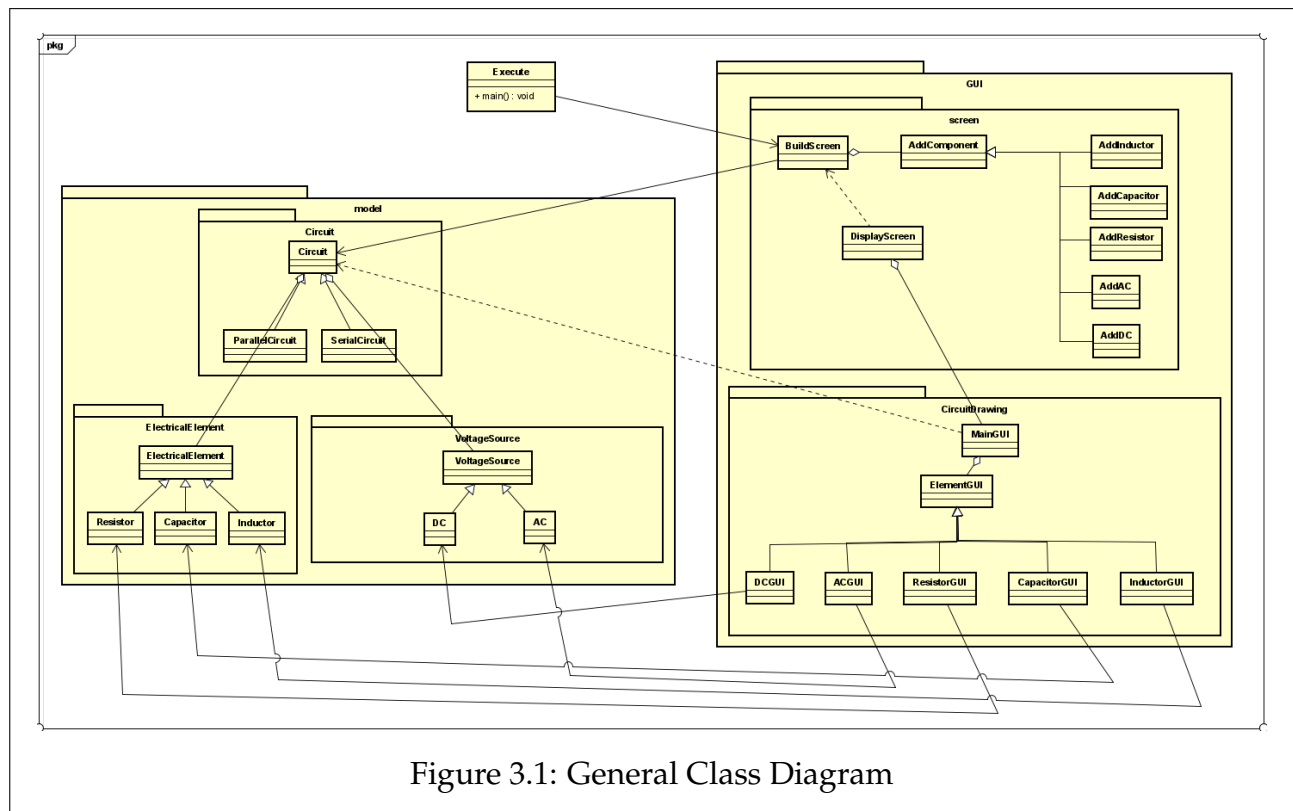


Figure 3.1: General Class Diagram

3.2 PACKAGE DETAILS

3.2.1 model package :

Here we observe a specific view of model package which was developed based on the needs of elements in an Electrical Circuit.

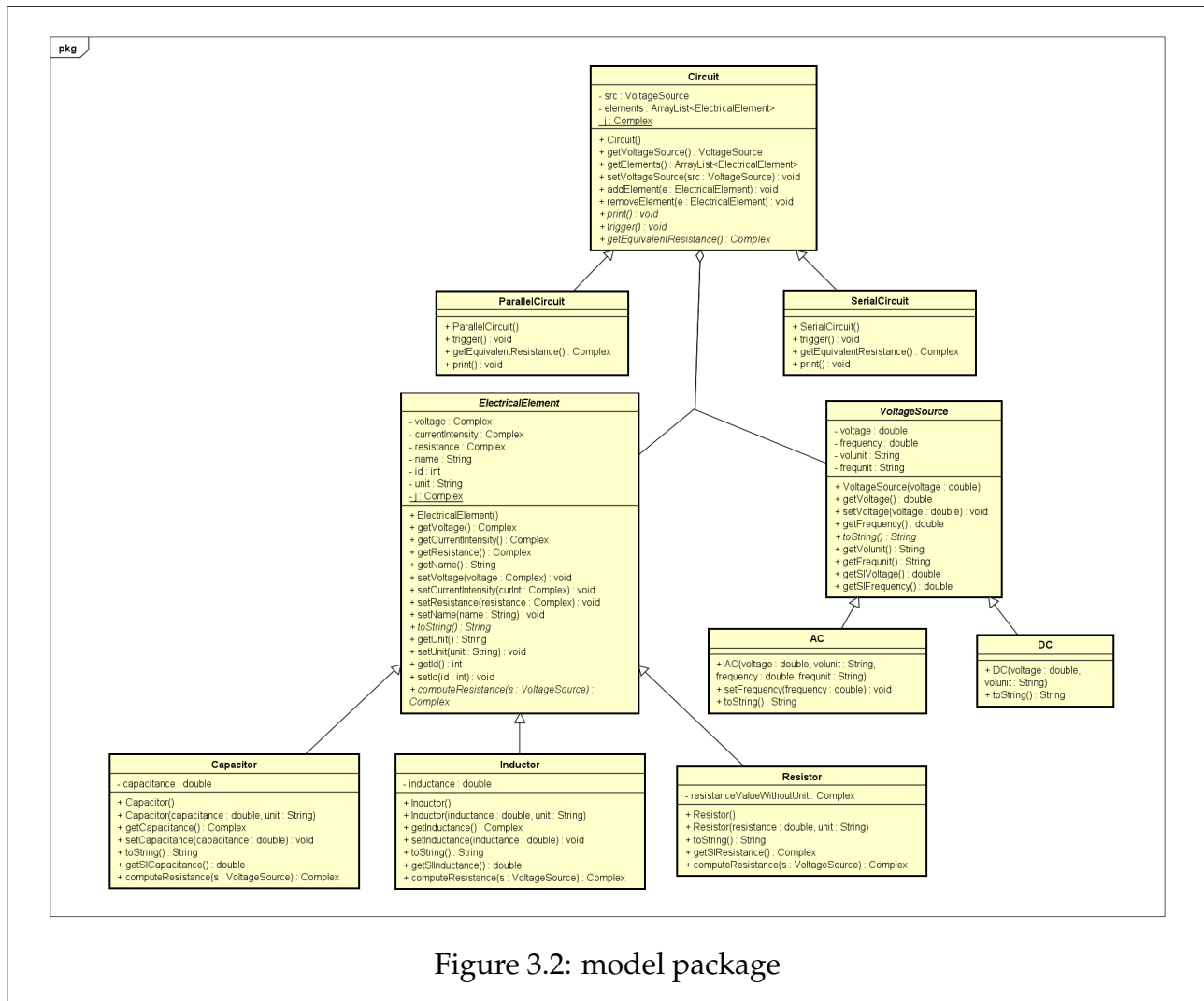


Figure 3.2: model package

Circuit: This class is used to create an Electrical Circuit as the Users constructed.

– Its Attributes are:

- * *src* : this attribute is an instance of **VoltageSource** class.
- * *elements* : this attribute is a list of added **ElectricalElement** class as the Users chose to add into the Circuit.
- * *j* : this is attribute represent a complex number $0 + 1i$.

– Its Methods are:

- * *Circuit()* : standard constructor for the class.
- * *getVoltageSource()* : return the value of **VoltageSource** class.
- * *getElements()* : return the list of added **ElectricalElement** instances by the Users.
- * *setVoltageSource(src : VoltageSource)* : set the value of **VoltageSource** class.

- * *addElement(e : **ElectricalElement**)* : add an **ElectricalElement** instance to the Circuit.
- * *removeElement(e : **ElectricalElement**)* : remove an **ElectricalElement** instance from the Circuit.
- *trigger()* : this method create a specific behavior which is pre-programmed in **ParallelCircuit** class and **SerialCircuit** class.
- *getEquivalentResistance()*: return the value of equivalent resistance of the Circuit.

ParallelCircuit : this class define one out of two type of Circuit.

- Its Attributes are:
 - * this class inherited from **Circuit** so it have all the Attributes and Methods of **Circuit** class.
- Its Methods are:
 - * *ParallelCircuit()* : standard constructor.
 - * *trigger()* : this method set the *voltage* attribute of **ElectricalElement** equals to the *voltage* of **VoltageSource** class and compute the resistance and intensity of all **ElectricalElement** in the Circuit in Complex form.
 - * *getEquivalentResistance()* : return the equivalent resistance of the Circuit.

SerialCircuit : this class define the last type of Circuit.

- Its Attributes are:
 - * this class inherited from **Circuit** so it have all the Attributes and Methods of **Circuit** class.
- Its Methods are:
 - * *SerialCircuit()* : standard constructor.
 - * *trigger()* : this method set the *voltage* attribute of **ElectricalElement** equals to the *voltage* of **VoltageSource** class and compute the resistance and intensity of all **ElectricalElement** in the Circuit in Complex form.
 - * *getEquivalentResistance()* : return the equivalent resistance of the Circuit.

VoltageSource : this class define the source of the Circuit that chose by Users.

- Its Attributes are:
 - * *voltage* : the value chose by Users.
 - * *frequency* : the value chose by Users.
 - * *volunit* : the type of units for voltage chose by Users.
 - * *frequnit* : the type of units for frequency chose by Users.
- Its Methods are:
 - * *VoltageSource(voltage : double)* : standard constructor.
 - * *getVoltage()* : return the *voltage* attribute value.
 - * *setVoltage(voltage : double)* : set the value for *voltage* attribute.
 - * *getFrequency()* : return the *frequency* attribute value.
 - * *toString()* : print out the chosen result values by the Users.
 - * *getVolunit()* : return the *volunit* attribute value.
 - * *getFrequnit()* : return the *frequnit* attribute value.
 - * *getSIVoltage()* : return the actual voltage value corresponds to *volunit* attribute value.

- * *getSIFrequency()* : return the actual frequency value corresponds to *frequnit* attribute value.

AC : this class define one out of two types of **VoltageSource** that chose by Users.

- Its Attributes are:
 - * This class inherited from the **VoltageSource** class so it consists all the Attributes and Methods of **VoltageSource**.
- Its Methods are:
 - * *AC(voltage : double, volunit: String, frequency : double, frequnit : String)* : standard constructor.
 - * *setFrequency(frequency : double)* : set the *frequency* values chose by Users.

DC : this class define the last types of **VoltageSource** that chose by Users.

- Its Attributes are:
 - * This class inherited from the **VoltageSource** class so it consists all the Attributes and Methods of **VoltageSource**.
- Its Methods are:
 - * *DC(voltage : double, volunit : String)* : standard constructor.

ElectricalElement : this class is programmed to denote real-life electrical elements which is **Capacitor**, **Inductor**, **Resistor**.

- Its Attributes are:
 - * *voltage* : define the voltage values of the element.
 - * *currentIntensity* : define the intensity values of the element.
 - * *resistance* : define the resistance values of the element.
 - * *name* : define the alias of the element.
 - * *id* : define the index of the element.
 - * *unit* : define the unit of the element.
 - * *j*: this is attribute represent a complex number $0 + 1i$.
- Its Methods are:
 - * *ElectricalElement()* : standard constructor.
 - * *getVoltage()* : return the *voltage* values.
 - * *getCurrentIntensity()* : return the *currentIntensity* values.
 - * *getResistance()* : return the *resistance* values.
 - * *getName()* : return the *name* values.
 - * *setVoltage(voltage : Complex)* : set the *voltage* values.
 - * *setCurrentIntensity(curInt : Complex)* : set the *currentIntensity* values.
 - * *setResistance(resistance : Complex)* : set the *resistance* values.
 - * *setName(name : string)* : set the *name* values.
 - * *toString()* : print out the chosen result values by the Users.
 - * *getUnit()* : return *unit* attribute value.
 - * *setUnit(String unit)* : set *unit* attribute values.
 - * *getId()* : return *id* attribute value.
 - * *setId()* : set *id* attribute values.
 - * *computeResistance(s: VoltageSource)* : abstract class compute resistance based on types of electrical element

Capacitor : this class define a capacitor in the Circuit

- Its Attributes are:
 - * This class inherited from the **ElectricalElement** class so it consists all the Attributes and Methods of **ElectricalElement**.
 - * *capacitance* : define the capacitance value.
- Its Methods are:
 - * *Capacitor()* : standard constructor.
 - * *Capacitor(capacitance : double)* : constructor by a capacitance value.
 - * *getCapacitance()* : return the value of *capacitance*.
 - * *setCapacitance(capacitance : Complex)* : set the value of *capacitance*.
 - * *toString()* : print out the chosen result values by the Users.
 - * *getSICapacitance()* : return the actual capacitance value corresponds to *unit* attribute value.
 - * *computeResistance(s: VoltageSource)* : override method in **ElectricalElement** class, compute resistance with capacitor

Inductor : this class define an inductor in the Circuit

- Its Attributes are:
 - * This class inherited from the **ElectricalElement** class so it consists all the Attributes and Methods of **ElectricalElement**.
 - * *inductance* : define the inductance value.
- Its Methods are:
 - * *Inductor()* : standard constructor.
 - * *Inductor(inductance : double)* : constructor by an inductance value.
 - * *getInductance()* : return the value of *inductance*.
 - * *setInductance(inductance : Complex)* : set the value of *inductance*.
 - * *toString()* : print out the chosen result values by the Users.
 - * *getSIInductance()* : return the actual inductance value corresponds to *unit* attribute value.
 - * *computeResistance(s: VoltageSource)* : override method in **ElectricalElement** class, compute resistance with inductor

Resistor : this class define an resistor in the Circuit

- Its Attributes are:
 - * This class inherited from the **ElectricalElement** class so it consists all the Attributes and Methods of **ElectricalElement**.
- Its Methods are:
 - * *Resistor()* : standard constructor.
 - * *Resistor(resistance : double)* : constructor by a resistance value.
 - * *toString()* : print out the chosen result values by the Users.
 - * *computeResistance(s: VoltageSource)* : override method in **ElectricalElement** class, only return the *resistance* attribute value.

3.2.2 GUI package

In GUI package, we divided the diagram into two main diagrams: **screen** diagram and **CircuitDrawing** diagram.

BuildScreen : this method define a GUI display where the Users can manipulate the **Simulator** directly to customize their own **Electrical Circuit**.

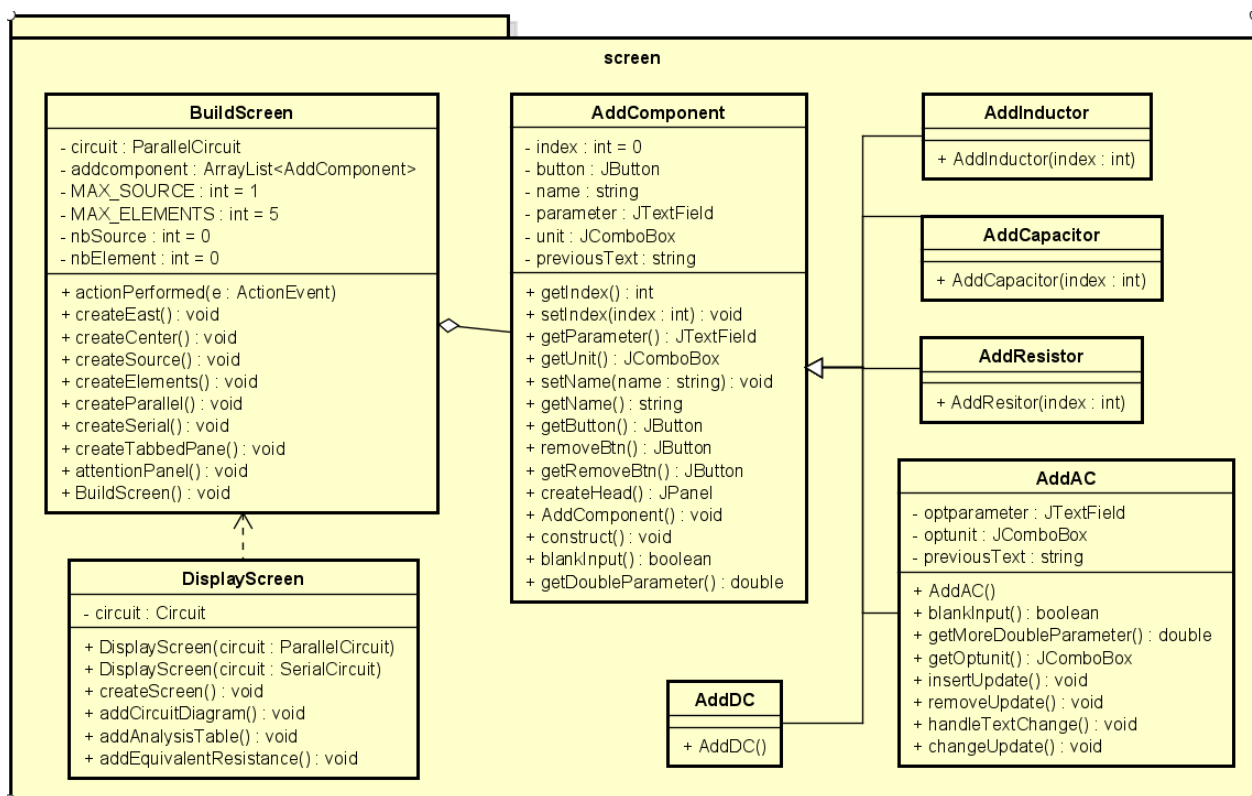


Figure 3.3: screen package

– Its Attributes are:

- * *circuit* : Represents the **Circuit** being built. It can be either a **ParallelCircuit** or a **SerialCircuit** object.
- * *addcomponents* : ArrayList containing the components like **VoltageSource** and **ElectricalCElement** being added to the **Circuit**.
- * *nbElements* : Current number of **ElectricalElement** added to the **Circuit**.
- * *nbSource* : Current number of **VoltageSource** added to the **Circuit**.
- * *MAX_SOURCE* : Maximum number of **VoltageSource** allowed.
- * *MAX_ELEMENTS* : Maximum number of **ElectricalElement** allowed.

– Its Methods are:

- * *createEast()* : Creates and returns the JPanel for the east side of the **Build-Screen** GUI, which includes the "SUBMIT" button.
- * *createCenter()* : Creates and returns the JPanel for the center of the **Build-Screen** GUI, which includes the source and elements panels.
- * *createSource()* : Creates and returns the JPanel for the source panel, which allows the User to add **VoltageSource** to the **Circuit**.
- * *createElements()* : Creates and returns the JPanel for the elements panel, which allows the User to add **ElectricalElements** to the **Circuit**.
- * *createParallel()* : Creates and returns the JPanel for the **ParallelCircuit** mode.
- * *createSerial()* : Creates and returns the JPanel for the **SerialCircuit** mode.
- * *createTabbedPane()* : Creates and returns the JTabbedPane that contains the **ParallelCircuit** and **SerialCircuit** modes.
- * *attentionPanel()* : Creates and returns a JPanel that displays an attention message if a **Short Circuit** or other errors are found in the **Circuit**.

- * *BuildScreen()* : Constructor of the **BuildScreen** class. Sets up the GUI components and initializes the **Circuit** and component lists.

AddComponent : this method helps the Users to manipulate the **Simulator** directly to customize their own **Electrical Circuit**.

– Its Attributes are:

- * *index* : An integer variable that represents the index of the component.
- * *button* : A **JButton** object that represents the button associated with the component.
- * *name* : A string variable that stores the name of the component.
- * *parameter* : A **TextField** object that represents the parameter input field.
- * *unit* : A **ComboBox** object that stores the units for the component.
- * *previousText* : A string variable that holds the previous text value of the parameter input field.

– Its Methods are:

- * *getIndex()* : Returns the index of the component.
- * *setIndex(index : int)* : Sets the index of the component.
- * *getParameter()* : Returns the **TextField** object representing the parameter input field.
- * *getUnit()* : Returns the **ComboBox** object representing the unit selection.
- * *setName(String name)* : Sets the name of the component.
- * *getName()* : Returns the name of the component.
- * *getButton()* : Returns the **JButton** object associated with the component.
- * *removeBtn()* : Private method that creates and returns a **JButton** with a custom "remove" icon.
- * *getRemoveBtn()* : Returns the remove button associated with the component.
- * *createHead()* : Creates and returns a **JPanel** representing the header section of the component.
- * *AddComponent()* : Constructor for the **AddComponent** class. Sets the component as not visible.
- * *construct()* : Constructs and initializes the component by setting its layout, adding labels, input fields, and listeners.
- * *blankInput()* : Checks if the parameter input field is empty and returns a boolean value indicating the result.
- * *getDoubleParameter()* : Retrieves the double value from the parameter input field.
- * Note: The **RemoveIcon** class is an inner class that implements the **Icon** interface and is responsible for painting a custom "remove" icon for the remove button.

AddInductor : this method helps the Users to manipulate the **Simulator** directly to add **Inductor** object as they constructed.

– Its Attributes are:

- * this class inherited the **AddComponent** so it consists all Attributes and Methods of **AddComponent** class.

– Its Methods are:

- * *addInductor(index : int)* : Constructor with *index* values.

AddCapacitor : this method helps the Users to manipulate the **Simulator** directly to add **Capacitor** object as they constructed.

- Its Attributes are:
 - * this class inherited the **AddComponent** so it consists all Attributes and Methods of **AddComponent** class.
- Its Methods are:
 - * *addCapacitor(index : int)* : Constructor with *index* values.

AddResistor : this method helps the Users to manipulate the **Simulator** directly to add **Resistor** object as they constructed.

- Its Attributes are:
 - * this class inherited the **AddComponent** so it consists all Attributes and Methods of **AddComponent** class.
- Its Methods are:
 - * *addResistor(index : int)* : Constructor with *index* values.

AddAC : this method helps the Users to manipulate the **Simulator** directly to add **AC** object as they constructed.

- Its Attributes are:
 - * this class inherited the **AddComponent** so it consists all Attributes and Methods of **AddComponent** class.
 - * *optparameter* : A **JTextField** object that represents the optional parameter input field.
 - * *optunit* : A **JComboBox** object that stores the units for the optional parameter.
 - * *previousText* : A string variable that holds the previous text value of the optional parameter input field.
- Its Methods are:
 - * *AddAC()* : Constructor for the **AddAC** class. Sets the name of the component as "AC" and initializes the unit selection for the main parameter and optional parameter.
 - * *blankInput()* : Overrides the *blankInput()* method from the parent class. Checks if both the main parameter and optional parameter input fields are empty and returns a boolean value indicating the result.
 - * *getMoreDoubleParameter()* : Retrieves the double value from the optional parameter input field.
 - * *getOptunit()* : Returns the **JComboBox** object representing the unit selection for the optional parameter.
 - * *insertUpdate()* : method is triggered when text is inserted into the *optparameter* text field. It calls the *handleTextChange()* method.
 - * *removeUpdate()* : method is triggered when text is removed from the *optparameter* text field. It updates the *previousText* variable with the current text value of the *optparameter* field.
 - * *handleTextChange()* : method is defined within the *insertUpdate()* method. It is responsible for handling the text change event. In this case, it attempts to parse the text as a double value. If an exception occurs, indicating that the

text is not a valid double value, it sets the text of the *optparameter* field back to the previous value stored in the *previousText* variable.

AddDC : this method helps the Users to manipulate the **Simulator** directly to add **DC** object as they constructed.

- Its Attributes are:
 - * this class inherited the **AddComponent** so it consists all Attributes and Methods of **AddComponent** class.
- Its Methods are:
 - * *addDC()* : standard constructor.

DisplayScreen : this method define a GUI display where the Users can manipulate the **Simulator** directly to customize their own **Electrical Circuit**.

- Its Attributes are:
 - * *circuit* : Represents the **Circuit** object to be displayed.
- Its Methods are:
 - * *DisplayScreen(circuit : ParallelCircuit)* : Constructor that creates a **DisplayScreen** object for a **ParallelCircuit**.
 - * *DisplayScreen(circuit : SerialCircuit)* : Constructor that creates a **DisplayScreen** object for a **SerialCircuit**.
 - * *createScreen()* : Creates the display screen by setting the size, title, layout, and other properties of the frame.
 - * *addCircuitDiagram()* : Adds the circuit diagram to the display screen.
 - * *addAnalysisTable()* : Adds the circuit analysis table to the display screen.
 - * *addEquivalentResistance()* : Adds the equivalent resistance label to the display screen.

CircuitDrawing Diagram is a GUI package which show the image of inserting components into the User's **Electrical Circuit**.

MainGUI : this method define the main GUI display in the **Electrical Circuit Simulator**.

- Its Attributes are:
 - * *circuit* : Represents the **Circuit** object to be displayed.
- Its Methods are:
 - * *MainGUI()* : The constructor of the **MainGUI** class. It takes a **Circuit** object as a parameter and initializes the *circuit* attribute.
 - * *paintComponent(g : Graphics)* : An overridden method from the **JPanel** class that is responsible for painting the components on the panel. It takes a **Graphics** object as a parameter and is called automatically when the panel needs to be painted. Inside this method, it iterates over the elements in the **Circuit** and calls the appropriate drawing method (draw) based on the type of element (**Resistor**, **Capacitor**, **Inductor**). It also draws the **VoltageSource**(AC or DC) based on its type.

ElementGUI : this method define a GUI display for all the **ElectricalElement** objects and **VoltageSource** object in the **Electrical Circuit**.

- Its Attributes are:
 - * *circuit* : Represents the **Circuit** object to be displayed.
 - * *seed* : An integer representing the seed value for the **ElementGUI**. It is based on the size of the elements in the **Circuit**.
- Its Methods are:
 - * *ElementGUI()* : The constructor of the **ElementGUI** class. It takes a **Circuit** object as a parameter and initializes the *circuit* and *seed* attributes.
 - * *getCircuit()* : A method that returns the **Circuit** object associated with the **ElementGUI**.
 - * *getSeed()* : A method that returns the *seed* value of the **ElementGUI**.
 - * *draw(g : Graphics)* : An abstract method that defines the drawing functionality for the **ElementGUI**. It takes a **Graphics** object as a parameter and is implemented in the subclasses to draw specific elements on the graphics context.

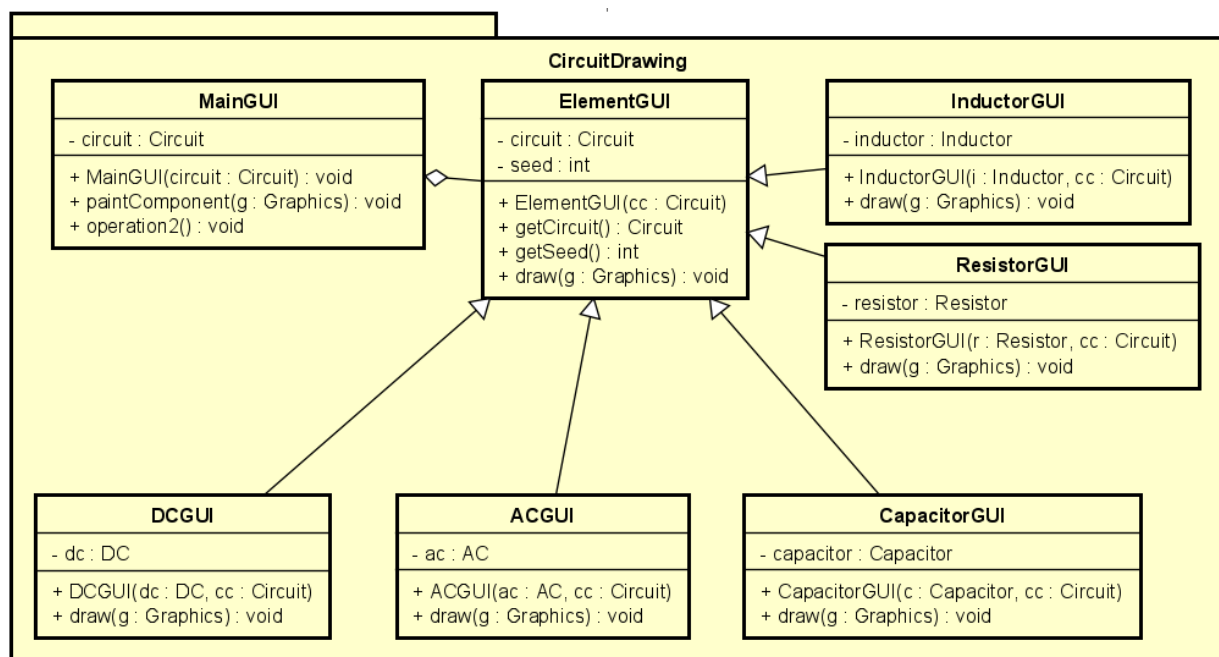


Figure 3.4: CircuitDrawing package

InductorGUI : this method define a GUI display for the **Inductor** object in the **Electrical Circuit**.

- Its Attributes are:

- * This class inherited **ElementGUI** so it consists all the Attributes and Methods of **ElementGUI** class.
- * *inductor* : Represents the **Inductor** object to be displayed.
- Its Methods are:
 - * *InductorGUI(i : Inductor,cc: Circuit)* : The constructor of the **InductorGUI** class. It takes an **Inductor** object and a **Circuit** object as parameters and initializes the *inductor* attribute and the inherited attributes from the **ElementGUI** class.
 - * *draw(g : Graphics)* : An abstract method that defines the drawing functionality for the **ElementGUI**. It takes a **Graphics** object as a parameter and is implemented in the subclasses to draw specific elements on the graphics context.

ResistorGUI : this method define a GUI display for the **Resistor** object in the **Electrical Circuit**.

- Its Attributes are:
 - * This class inherited **ElementGUI** so it consists all the Attributes and Methods of **ElementGUI** class.
 - * *resistor* : Represents the **Resistor** object to be displayed.
- Its Methods are:
 - * *ResistorGUI(r : Resistor,cc: Circuit)* : The constructor of the **ResistorGUI** class. It takes an **Resistor** object and a **Circuit** object as parameters and initializes the *resistor* attribute and the inherited attributes from the **ElementGUI** class.
 - * *draw(g : Graphics)* : An abstract method that defines the drawing functionality for the **ElementGUI**. It takes a **Graphics** object as a parameter and is implemented in the subclasses to draw specific elements on the graphics context.

CapacitorGUI : this method define a GUI display for the **Capacitor** object in the **Electrical Circuit**.

- Its Attributes are:
 - * This class inherited **ElementGUI** so it consists all the Attributes and Methods of **ElementGUI** class.
 - * *capacitor* : Represents the **Capacitor** object to be displayed.
- Its Methods are:
 - * *CapacitorGUI(c : Capacitor,cc: Circuit)* : The constructor of the **Capacitor** class. It takes an **Capacitor** object and a **Circuit** object as parameters and initializes the *resistor* attribute and the inherited attributes from the **ElementGUI** class.
 - * *draw(g : Graphics)* : An abstract method that defines the drawing functionality for the **ElementGUI**. It takes a **Graphics** object as a parameter and is implemented in the subclasses to draw specific elements on the graphics context.

ACGUI : this method define a GUI display for the **AC** object in the **Electrical Circuit**.

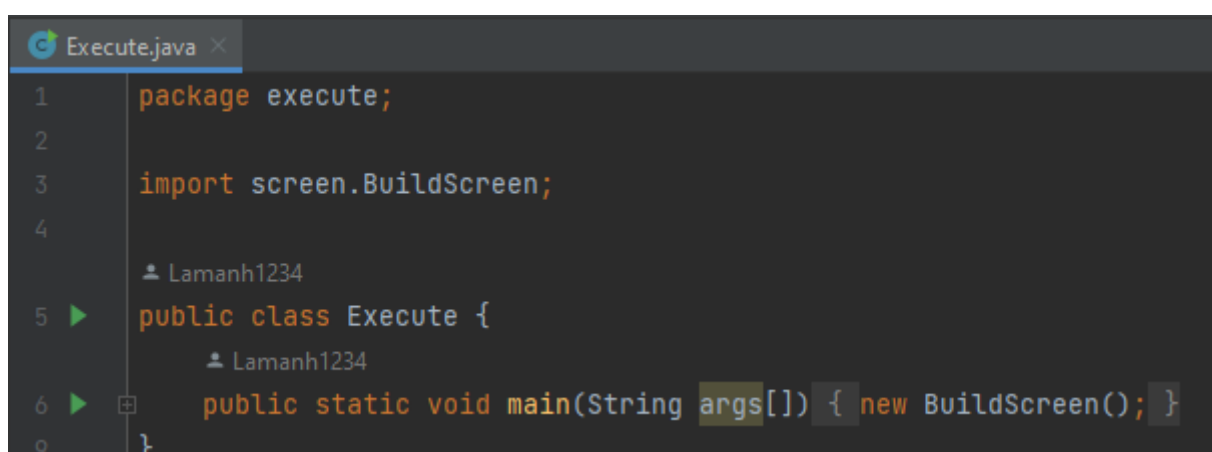
- Its Attributes are:

- * This class inherited **ElementGUI** so it consists all the Attributes and Methods of **ElementGUI** class.
- * *ac* : Represents the **AC** object to be displayed.
- Its Methods are:
 - * *ACGUI(ac : AC,cc: Circuit)* : The constructor of the **ACGUI** class. It takes an **AC** object and a **Circuit** object as parameters and initializes the *ac* attribute and the inherited attributes from the **ElementGUI** class.
 - * *draw(g : Graphics)* : An abstract method that defines the drawing functionality for the **ElementGUI**. It takes a **Graphics** object as a parameter and is implemented in the subclasses to draw specific elements on the graphics context.

DCGUI : this method define a GUI display for the **DC** object in the **Electrical Circuit**.

- Its Attributes are:
 - * This class inherited **ElementGUI** so it consists all the Attributes and Methods of **ElementGUI** class.
 - * *dc* : Represents the **DC** object to be displayed.
- Its Methods are:
 - * *DCGUI(dc : DC,cc: Circuit)* : The constructor of the **DC** class. It takes an **DC** object and a **Circuit** object as parameters and initializes the *dc* attribute and the inherited attributes from the **ElementGUI** class.
 - * *draw(g : Graphics)* : An abstract method that defines the drawing functionality for the **ElementGUI**. It takes a **Graphics** object as a parameter and is implemented in the subclasses to draw specific elements on the graphics context.

3.2.3 Execute



```

1 package execute;
2
3 import screen.BuildScreen;
4
5 public class Execute {
6     public static void main(String args[]) { new BuildScreen(); }
7 }

```

Figure 3.5: model package

On **Execute** Class, we define a **main()** class where we create new **BuildScreen** and run as starting our **Electrical Circuit Simulator**.

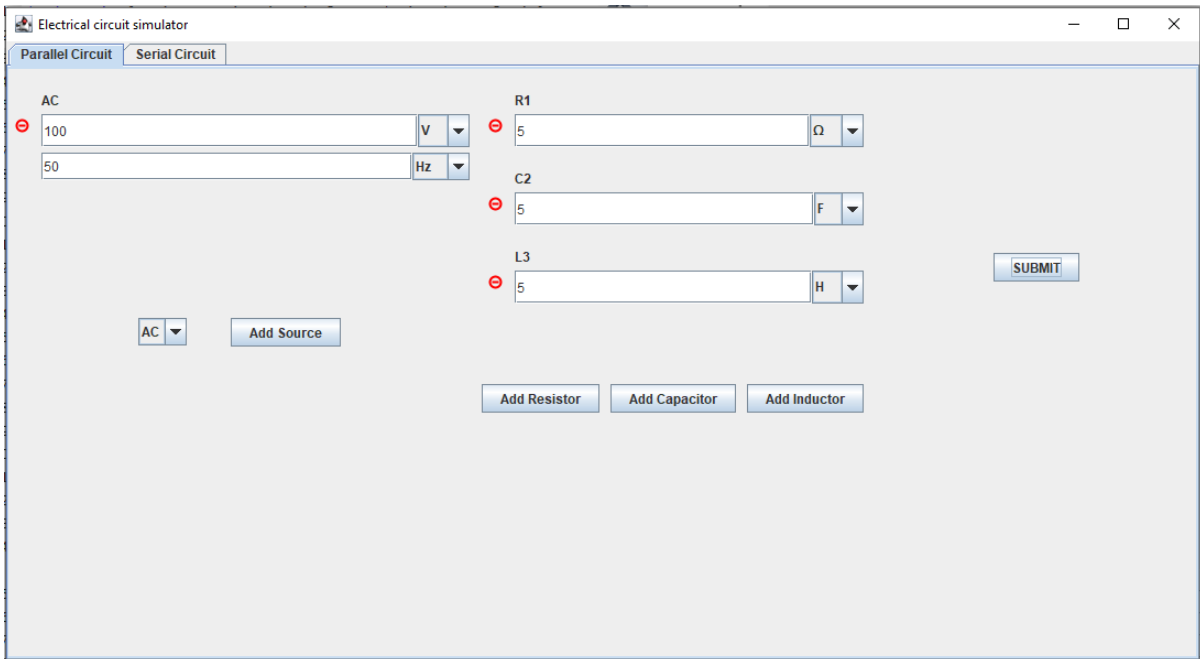


Figure 3.6: model package

Chapter 4

REFERENCES

- We applied this Documentation as our guide for Graphics drawing:
 - <https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html>
- Our GUI is referred from the link allocated in the Mini-project Topics:
 - <https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-parallel-and-series-resistor>
- Our Mini-project requires downloading The Apache Commons Mathematics Library which is through the link below:
 - <https://archive.apache.org/dist/commons/math/binaries/commons-math3-3.6.1-bin.zip>