BÀI TẬP LỚN

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: Electrical circuit simulator

Giáo viên hướng dẫn: Nguyễn Thị Thu Trang

Mã lớp: 151965 – Nhóm 15

Sinh viên thực hiện:

STT	Họ và tên	MSSV	Email
1	Thân Cát Ngọc Lan	20225646	lan.tcn225646@sis.hust.edu.vn
2	Bùi Thành Long	20225874	long.bt225874@sis.hust.edu.vn
3	Mai Huy Long	20225735	long.mh225735@sis.hust.edu.vn
4	Phạm Xuân Long	20225648	long.px225645@sis.hust.edu.vn
5	Phan Hoàng Long	20225738	long.ph225738@sis.hust.edu.vn

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

STT	Họ và tên	MSSV	Vai trò	Công việc
1	Thân Cát Ngọc Lan	20225646	Thành viên	Tìm hiểu bài toán, Dựng biểu đồ lớp, lên ý tưởng logic hệ thống
2	Bùi Thành Long	20225874	Trưởng nhóm	Tìm hiểu bài toán, Xây dựng logic hệ thống, viết báo cáo, làm slide
3	Mai Huy Long	20225735	Thành viên	Tìm hiểu bài toán, Xây dựng GUI, logic UI , viết báo cáo, kiểm thử
4	Phạm Xuân Long	20225648	Thành viên	Tìm hiểu bài toán, Viết báo cáo
5	Phan Hoàng Long	20225738	Thành viên	Tìm hiểu bài toán, Lên ý tưởng & xây dựng biểu đồ use case, làm slide thuyết trình

Lời mở đầu

I. Lý do chọn đề tài

Mạch RLC là một trong những mạch điện cơ bản, được ứng dụng rộng rãi trong các hệ thống điện tử như radio, lọc tín hiệu, và điều chỉnh tần số. Chúng em chọn đề tài này nhằm làm trực quan hơn thông qua việc tương tác với các thông số và theo dõi kết quả thay đổi.

II. Mục tiêu đề tài

Hiểu khái niệm phân tích và giải quyết bài toán lập trình hướng đối tượng: Xác định các đối tượng lớp, mỗi quan hệ giữa các lớp trong một hệ thống thông tin.

Mục tiêu đề tài mục đích nhằm nghiên cứu môi trường phát triển, ngôn ngữ java xây dựng những ứng dụng cụ thể.

III. Công nghệ sử dụng

Trong xu thế phát triển công nghệ thông tin như vũ bảo hiện nay, đặc biệt là trong ngành công nghệ phần mềm ngày càng đòi hỏi trình độ cao trong kĩ thuật lập trình. Chính vì vậy mà phương pháp lập trình hướng thủ tục cổ điển trước đây không đáp ứng được nhu cầu đặt ra của thời đại một phương pháp lập trình mới được xây dựng theo nguyên lý Alan-Kay đã được ra đời nhằm đáp ứng những nhu cầu cấp thiết đó : “Phương pháp Lập Trình Hướng đối Tượng”.

Đồ án này được thiết kế theo phương pháp LTHDT bằng ngôn ngữ Java do SunMicroSystem đưa ra vào năm 1991 .Chính vì vậy mà nó giải quyết được những vướng mắc gặp phải khi thiết kế theo phương pháp lập trình thủ tục thuần túy

- Mã chương trình rõ ràng, dễ đọc, dễ hiểu và cô đọng
- Chương trình được tổ chức thành những Class lắp ghép lại với nhau thành một khối thống nhất
- Mỗi Class gồm có nhiều Method đảm nhận các vai trò khác nhau trong chương trình
- Chương trình có tính mềm dẻo cao
- Có khả năng tái sử dụng tài nguyên

IV. Phạm vi nghiên cứu

Nghiên cứu môi trường phát triển, ngôn ngữ java trong lập trình hướng đối tượng. Tìm hiểu cách xây dựng các phương thức, thuộc tỉnh đối tượng trong java.

Nội dung báo cáo

I. *Cơ sở lý thuyết*

1. Giới thiệu

Java là một ngôn ngữ lập trình được Sun Microsystems giới thiệu vào tháng 6 năm 1995. Từ đó, nó đã trở thành một công cụ lập trình của các lập trình viên chuyên nghiệp. Java được xây dựng trên nền tảng của C và C++, do vậy nó sử dụng các cú pháp của C và các đặc trưng hướng đối tượng của C++.

Vào năm 1991, một nhóm các kỹ sư của Sun Microsystems có ý định thiết kế một ngôn ngữ lập trình để điều khiển các thiết bị điện từ như tivi, máy giặt, lò nướng.... Mặc dù C và C++ có khả năng làm việc này nhưng trình biên dịch lại phụ thuộc vào từng loại CPU.

Trình biên dịch thường phải tốn nhiều thời gian để xây dựng nên rất đắt, vì vậy để mỗi loại CPU có một trình biên dịch riêng là rất tốn kém. Do đó nhu cầu thực tế đòi hỏi một ngôn ngữ chạy nhanh, gọn, hiệu quả và độc lập thiết bị tức là có thể chạy trên nhiều loại CPU khác nhau, dưới các môi trường khác nhau. “Oak” đã ra đời và vào năm 1995 được đổi tên thành Java. Mặc dù mục tiêu ban đầu không phải cho Internet nhưng do đặc trưng không phụ thuộc thiết bị nên Java đã trở thành ngôn ngữ lập trình cho Internet.

2. Hướng đối tượng

Java là ngôn ngữ lập trình thuần hướng đối tượng, mọi chương trình viết trên Java đều phải được xây dựng trên các đối tượng. Nếu trong C/C++ ta có thể tạo ra các hàm (chương trình con không gắn với đối tượng nào) thì trong Java ta chỉ có thể tạo ra các phương thức (chương trình con gắn liền với một lớp cụ thể). Trong Java không cho phép các đối tượng có tính năng đa kế thừa mà được thay thế bằng các giao diện (interface)

II. *Đặc tả bài toán*

1. Miêu tả yêu cầu bài toán.

- Ứng dụng yêu cầu mô phỏng mạch RLC, giúp người dùng thiết kế và kiểm tra các mạch có chứa thành phần điện trở (R), cuộn dây (L), và tụ điện (C). Người dùng có thể xem và kiểm tra đáp ứng tần số, phản hồi mạch, và các thông số quan trọng của mạch.

2. Mục tiêu

- Người dùng có thể thiết kế mạch logic bằng cách kéo thả các thành phần.
- Tính toán giá trị điện thế, cường độ của mạch và của từng thiết bị của mạch.

Yêu cầu chi tiết

i. Giao diện người dùng

1. Bảng thiết kế mạch

- Các thành phần R, L, C nút nguồn, và dây kết nối giữa các thành phần.

2. Các thành phần

- **Điện trở (R)**: Cho phép người dùng nhập giá trị.
- **Cuộn dây (L)**: Cho phép người dùng nhập giá trị độ tự cảm (L).
- **Tụ điện (C)**: Cho phép người dùng nhập giá trị dung kháng (C).
- **Nút nguồn và tải**: Tạo nguồn cung cấp điện và tải kết nối đầu vào/đầu ra.

ii. Tính năng chính

- **Thiết kế mạch**: Người dùng kéo thả các thành phần và kết nối chúng theo logic mong muốn.
- **Kiểm tra logic**: Ứng dụng tính toán đầu ra dựa trên đầu vào đã được đặt trong mạch.

iii. Yêu cầu kĩ thuật

1. Giao diện người dùng

- Giao diện desktop.
- Tối ưu hóa tốc độ xử lý, tương thích với thiết bị.

2. Xử lý login và đầu ra

- Thuật toán tính toán các thông số RLC như tần số cộng hưởng, Q-factor, đáp ứng tần số.

III. Biểu đồ use case

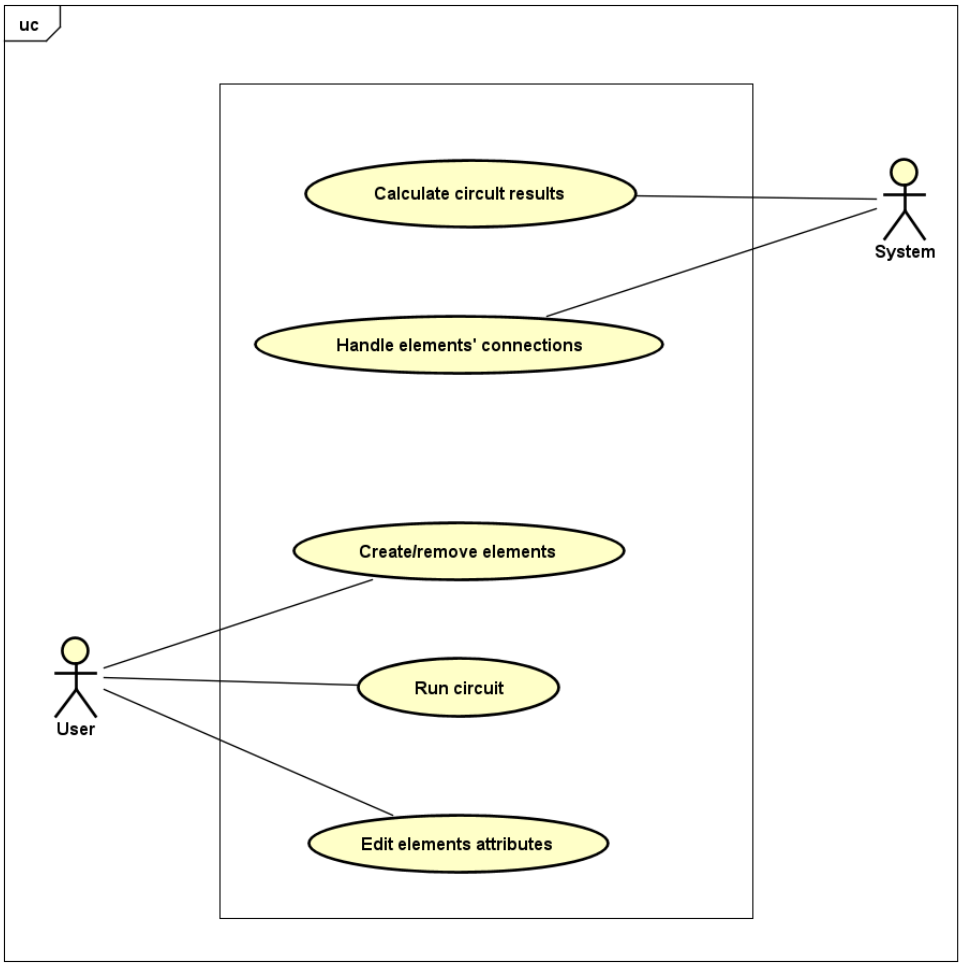
1. Biểu đồ use case tổng quan

- Tác nhân: User, System

- *Các use case với tác nhân User:*
  + Create/Remove elements
  + Run circuit
  + Edit elements attributes

- *Các use case với tác nhân System:*
  + Calculate circuit results
  + Handle elements' connections

- *Hình ảnh cho usecase*



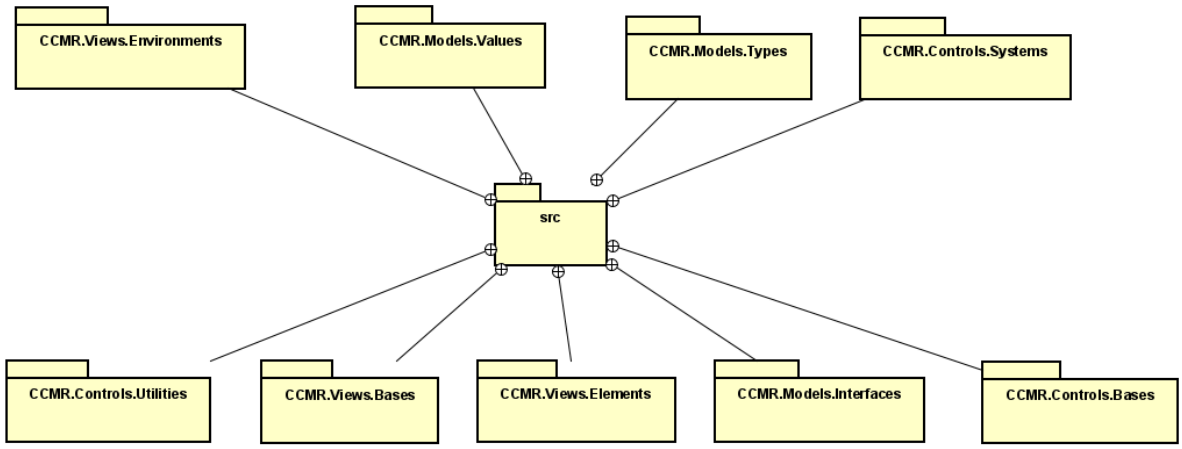
2. Đặc tả use case

Điều kiện trước	Người dùng khởi động simulator	
Tác nhân	Người dùng chương trình	
Luồng thực thi chính		
No.	**Thực hiện**	**Hành động**
1	User	Tạo mới thiết bị
2	System	Hiển thị thiết bị lên bảng mạch
3	User	Điền các thông số của thiết bị
4	User	Kết nối các thiết bị
5	User	Chạy mạch

6	System	Xử lý logic hiển thị thông tin bảng mạch

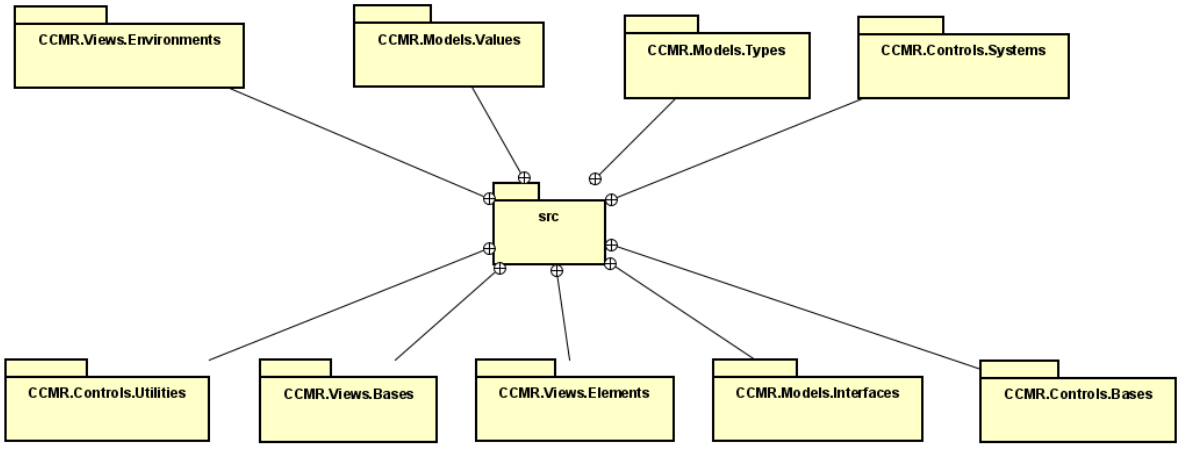
IV. Class diagram

1. *Biểu đồ lớp tổng quan:*



PHÂN TÍCH THIẾT KẾ BÀI TOÁN

1. Biểu đồ lớp tổng quan:

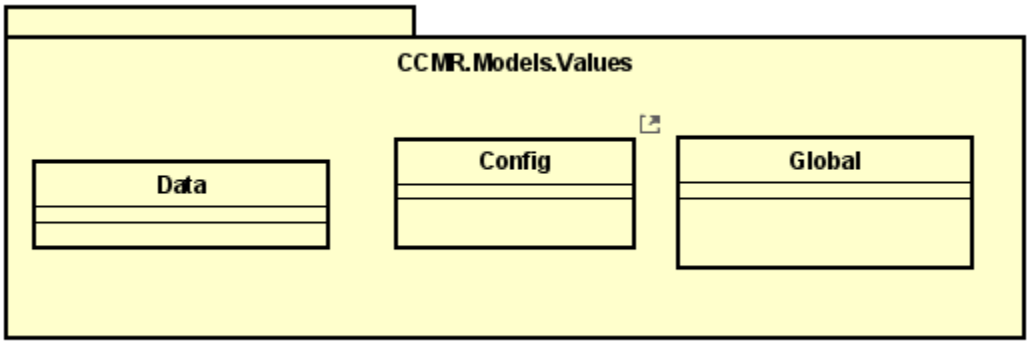


- Gói CCMR.Views.Environments: Chứa các đối tượng UI liên quan tới không gian và các thiết bị mạch.

- Gói CCMR.Models.Interfaces: Gồm các mã về interface.
- Gói CCMR.Controls.Utilities: Gồm các hàm tiện ích.
- Gói CCMR.Models.Types: Chứa các kiểu dữ liệu/đối tượng.
- Gói CCMR.Controls.Elements: Chứa các thành phần mạch.
- Gói CCMR.Controls.Systems: Chứa các đối tượng điều khiển/quản lí.
- Gói CCMR.Models.Values: Chứa các giá trị/tham chiếu toàn cục .trong hệ thống.
- Gói CCMR.Views.Elements: chứa UI của các thiết bị mạch.
- Gói CCMR.Views.Bases: Chứa các object trừu tượng cho các thành phần UI khác.

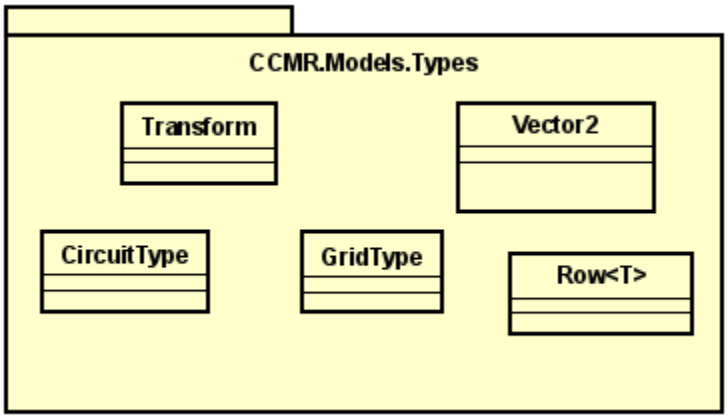
2. *Biểu đồ cho các gói*

a. *Gói CCMR.Models.Values*



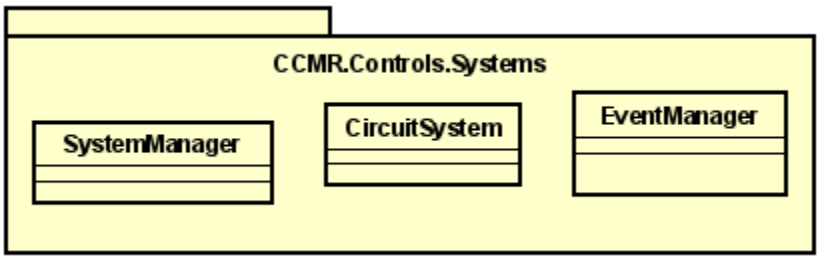
- Class Config: lớp static chứa các giá trị mặc định của hệ thống.
- Class Data: lớp static chứa các giá trị lưu động của hệ thống.
- Class Global: lớp static chứa tham chiếu tới singleton của các đối tượng duy nhất trong hệ thống.

b. *Gói CCMR.Models.Types*



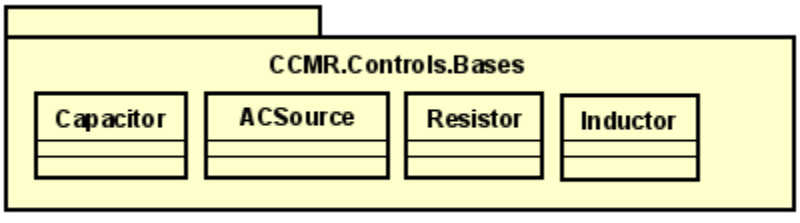
- Enum CircuitType: là các kiểu loại của thiết bị mạch.
- Enum GridType: là các kiểu dạng lưới của GridView (UI).
- Row: lớp generic đại diện cho một danh sách các đối tượng với các chức năng cơ bản như Add(), Remove(), Count().
- Transform: là lớp đại diện cho vị trí và góc xoay của UI của các thiết bị.
- Vector2: là lớp đại diện cho kiểu dữ liệu vector có 2 giá trị double X và Y.

c. Gói CCMR.Controls.Systems



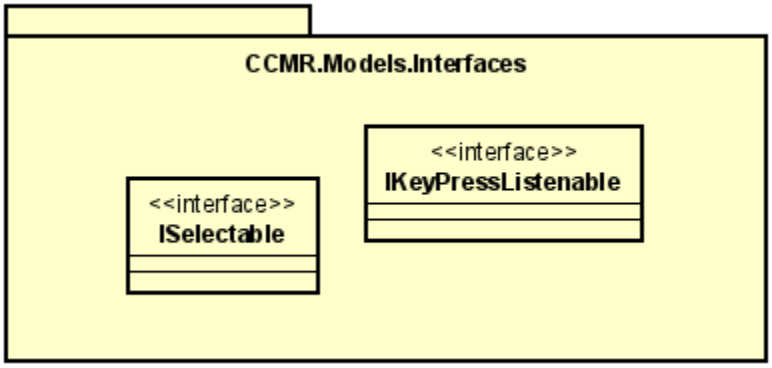
- Class CircuitSystem: là lớp đại diện cho hệ thống mạch, có khả năng thêm các thiết bị cũng như tính toán các giá trị cần thiết.
- Class BaseCircuitElement: là lớp cơ sở cho các phần tử mạch, có công dụng xác định các hành vi của phần tử mạch.
- Class EventManager: là lớp trung tâm điều phối sự kiện nhấn phím và cung cấp cơ chế gọi tất cả các đối tượng lắng nghe sự kiện.

d. Gói CCMR.Controls.Bases



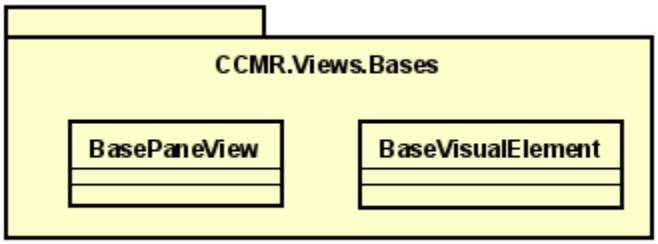
- Class BaseCircuitElement: là lớp trừu tượng định nghĩa cho các thiết bị mạch trong hệ thống.
- Class CircuitConnection: là đối tượng đại diện cho các đầu kết nối giữ thiết bị này với thiết bị khác.

e. Gói CCMR.Models.Interfaces



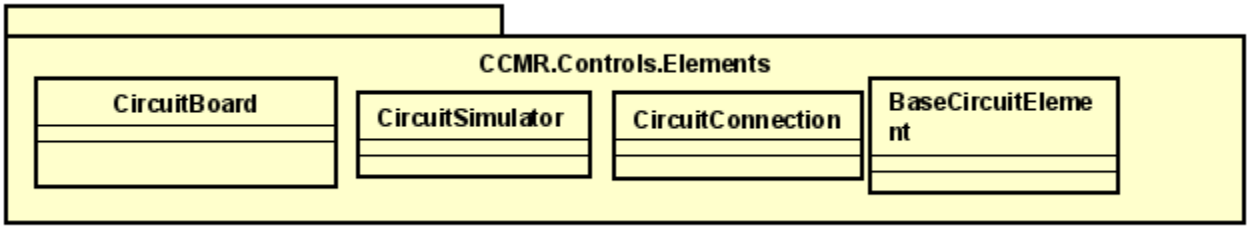
- Interface IKeyPressListenable: cung cấp khả năng lắng nghe sự kiện khi có phím keyboard được nhấn.
- Interface ISelectable: cung cấp khả năng chọn/bỏ chọn đối với các đối tượng UI.

f. Gói CCMR.Views.Bases



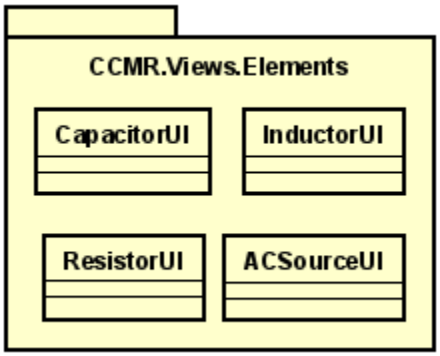
- Class BasePaneView: là lớp trừu tượng dùng để tạo dựng lên các UI Pane phức tạp hơn (GridView) với các phương thức để tạo UI và tương tác với môi trường.
- Class BaseVisualElement: là lớp trừu tượng dùng để tại dựng lên các UI Element cho các thiết bị mạch với các phương thức tạo dựng UI, cập nhật vị trí và tương tác với môi trường.

g. CCMR.Controls.Elements



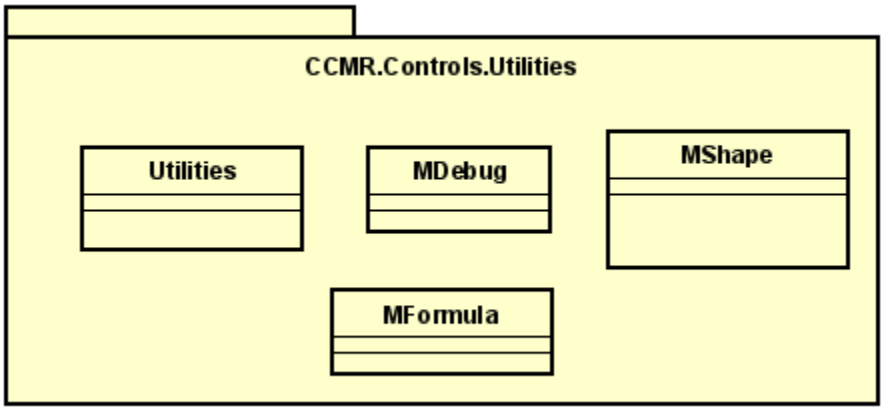
- Class ACSource: mô phỏng một thành phần nguồn điện xoay chiều (Alternating Current Source) trong mạch điện. Có chức năng mô hình hóa 1 AC trong mạch, tính toán các giá trị điện áp tần số, trở kháng.
- Class Capacitor: mô phỏng một tụ điện trong mạch điện, được thiết kế để mô hình hóa hành vi của tụ điện và quản lý các thuộc tính liên quan.
- Class Inductor: mô phỏng một cuộn cảm trong mạch điện, thể hiện các thuộc tính và hành vi đặc trưng của cuộn cảm.
- Class Resistor mô phỏng một điện trở trong mạch điện, thể hiện các thuộc tính và hành vi đặc trưng của điện trở.

h. Gói CCMR.Views.Elements



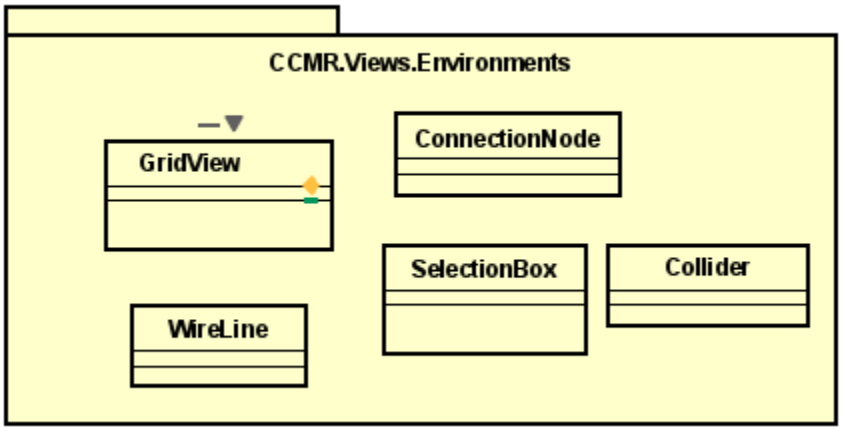
- 4 class ACSource, Capacitor, Inductor, Resistor: Kế thừa từ lớp BaseCircuitElement, đại diện cho các thiết bị mạch trong hệ thống, chứa các giá trị riêng và tính toán, trả về các giá trị tương ứng.

i. Gói CCMR.Controls.Utilities



- Class MDebug: Chứa các hàm tiện ích dùng để in log ra console của IDE.
- Class MFormula: Chứa các hàm tiện ích phục vụ cho việc tính toán trong hệ thống.
- Class MShape: chứa các hàm tiện ích tác động tới các đối tượng Shape trong UI.

j. Gói CCMR.Views.Environments



- Class Collider: kế thừa từ Rectangle, đại diện cho viền ngoài của UI, có khả năng phát hiện va chạm với các Collider khác khi được kéo thả chồng chéo lên nhau.
- Class ConnectionNode: kế thừa từ Circle, đại diện cho các đầu kết nối của UI, có khả năng kết nối với các ConnectionNode khác khi được chọn.
- Class GridView: Kế thừa từ BasePaneView, là UI của background lưới trong hệ thống, có khả năng kéo thả, phóng to thu nhỏ theo input từ chuột.
- Class SelectionBox: là UI cho hộp chọn, sử dụng khi kéo thả trên màn hình để chọn nhiều thiết bị cùng lúc.
- Class WireLine: Kế thừa từ Polyline, là UI của các đường dây dẫn nối giữa các thiết bị mạch.

Các design pattern được sử dụng

Singleton

- Mô tả: Sử dụng tính chất của 'static' để đảm bảo trong hệ thống chỉ có duy nhất một instance của đối tượng.
- Ưu điểm:
    + Đảm bảo chỉ có duy nhất một instance của đối tượng trong toàn bộ hệ thống.
    + Làm giảm thiểu lượng tham chiếu chéo trên các đối tượng, đảm bảo tính 'clean code' của hệ thống.
- Nhược điểm:
    + Khó kiểm tra và bảo trì do singleton làm giảm khả năng unit test của hệ thống.
    + Vi phạm nguyên tắc 'Single Responsibility' của SOLID.
- Ứng dụng: Pattern này được sử dụng ở CCMR.Models.Values.Global để lưu trữ tham chiếu tới các đối tượng độc nhất trong hệ thống.

Observer

- Mô tả: Sử dụng một đối tượng (Publisher) làm trung gian để thông báo một sự kiện tới các đối tượng theo dõi (Listener).
- Điểm mạnh:
    + Giảm thiểu sự phụ thuộc giữa các đối tượng mà vẫn đảm bảo được các đối tượng đó có thể kích hoạt hành động khi có một sự kiện xảy ra.
    + Dễ dàng quản lý các đối tượng nào được cho phép nhận sự kiện hay không do ốc thể dễ dàng thêm hoặc bớt chúng vào danh sách listener.
    + Khả năng mở rộng hệ thống tốt do không tạo sự gò bó và độc lập với các phần khác của hệ thống.
- Điểm yếu
    + Khó kiểm soát thứ tự thông báo sự kiện tới các đối tượng lắng nghe.

+ Không thể hủy/đảo ngược thông báo.
+ Tiêu tốn bộ nhớ nếu không được quản lý tốt.

- Ứng dụng: Pattern này được sử dụng ở CCMR.Models.Interfaces.IKeyPressListenable (interface cho listener) và CCMR.Controls.Systems.EventManager (đối tượng thông báo).