HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

LOGIC EXPRESSION NORMALIZER

FINAL PROJECT REPORT

Object Oriented Languge and Theory - Lab

lecturer:

Trần Thế Hùng

group 31:

Lê Thanh Tùng 20226070

Nguyễn Tài Huy 2018 4372

June 23rd, 2024

Table of Contents

[**A.** **Description** 4](#_Toc73810071)

[I. Mini-project requirements 4](#_Toc73810072)

[1. Overview: 4](#_Toc73810073)

[2. Specifications: 4](#_Toc73810074)

[II. Use case diagram 4](#_Toc73810075)

[**B.** **Design idea** 6](#_Toc73810076)

[I. The Model – View – Controller design patterns 6](#_Toc73810077)

[II. General class diagram 8](#_Toc73810078)

[III. Class diagrams and explanation 9](#_Toc73810079)

[1. Model package: 9](#_Toc73810080)

[2. View package: 11](#_Toc73810081)

[3. Controller package: 12](#_Toc73810082)

[4. OOP-Principle: 13](#_Toc73810083)

[5. WorkFlow: 15](#_Toc73810084)

**MEMBERS ASSIGNMENT**

|  |  |  |
| --- | --- | --- |
| **Full name** | **Student ID** | **Assignments** |
| Lê Thanh Tùng  Nguyễn Tài Huy | 20226070  20184372 | * Designed the Model. * Developed the input and output screens. * Created the tables. * Implemented all the classes to build a complete model. * Wrote all the methods in the Controller to streamline the project workflow. * Documents and Slide |

# **Description**

## Mini-project requirements

### Overview:

1. Tạo một ứng dụng để chuẩn hóa biểu thức logic bằng cách sử dụng phương pháp Quine-McCluskey. Biểu thức cuối cùng phải có số lượng các hạng tử ít nhất. Số lượng các mệnh đề càng ít thì độ tin cậy càng cao và chi phí sản xuất càng thấp. Ứng dụng này chấp nhận các đầu vào có ba hoặc bốn biến

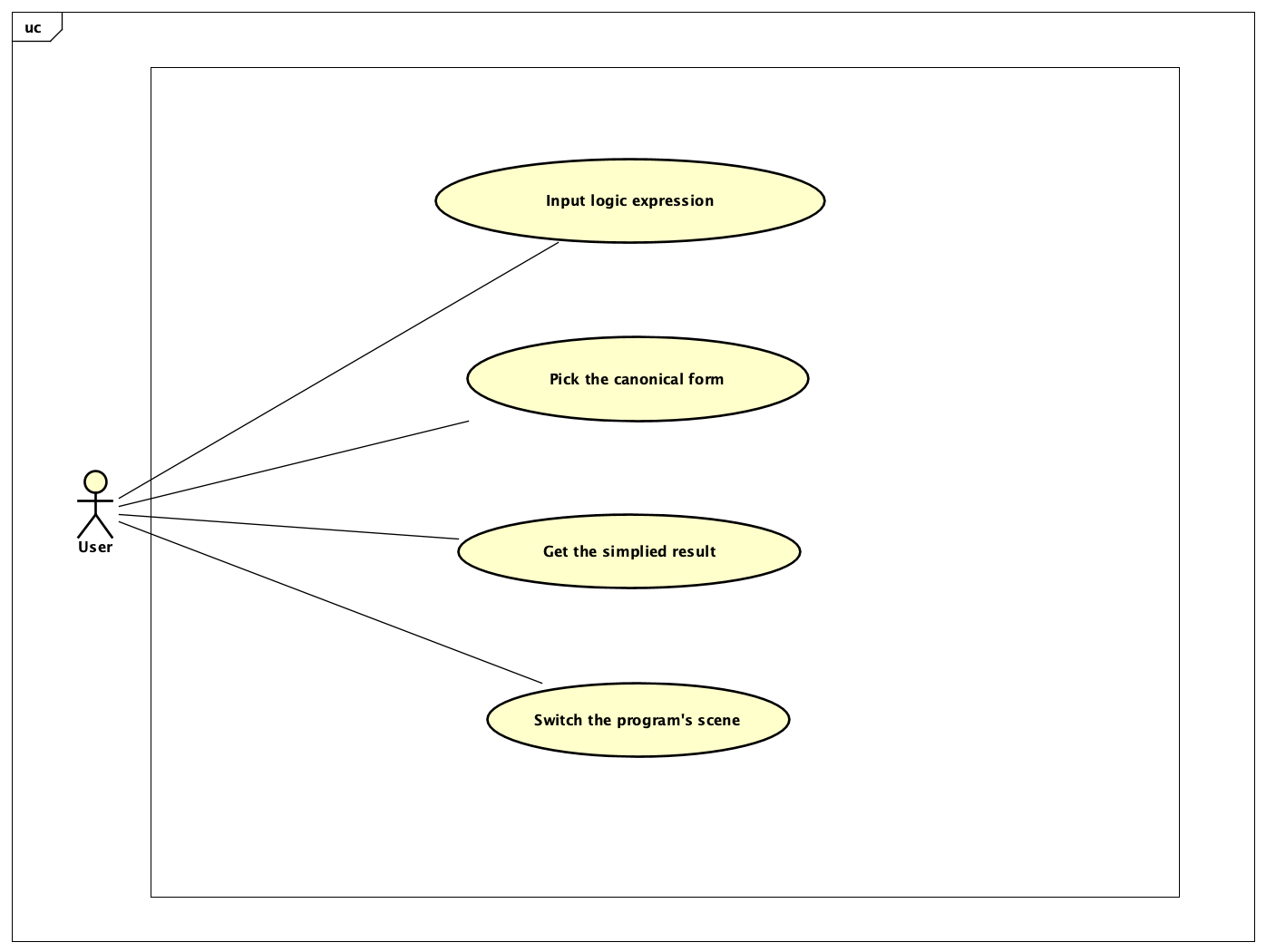
### Specifications:

Ứng dụng này nhận một biểu thức boolean từ người dùng thông qua một giao diện mô phỏng truth table. Giao diện người dùng bao gồm:

* Main menu(Giao diện chính): Người dùng có thể lựa chọn một trong hai trường hợp là biểu thức 3 biến hoặc 4 biến từ thanh điều hướng.
* Input interface(Giao diện nhập dữ liệu): Người dùng có thể nhập một biểu thức logic thông qua truth table. Giá trị không liên quan sẽ không được chấp nhận. Họ cũng có thể chọn một dạng chuẩn tắc cho biểu thức đơn giản hóa - hoặc là SOP (Tổng của các tích) hoặc là POS (Tích của các tổng). Sau khi người dùng đã hoàn thành việc chọn các giá trị cho truth table, họ có thể nhấn nút "Submit" để xem kết quả đầu ra.
* Output interface(Giao diện đầu ra): Ứng dụng sẽ hiển thị cho người dùng bảng trung gian (bao gồm các cột trung gian), bảng các hàm biểu diễn chính (PI table), bảng tạo phương trình (bao gồm các EPI và phương trình đặc tính của chúng), và biểu thức đơn giản cuối cùng.

## Use case diagram

* Nhận kết quả đã đơn giản hóa sau khi người dùng gửi đầu vào đã chọn, và kết quả được hiển thị ngay sau đó.
* Duyệt các dạng chuẩn bao gồm các dạng kết quả chuẩn được người dùng chọn (dưới dạng SOP hoặc POS).
* Nhập biểu thức logic bao gồm một chuỗi các số 0 hoặc 1 thông qua truth table trong đó 0 có nghĩa là không được chọn và 1 có nghĩa là được chọn
* Khởi chạy chương trình khi người dùng nhấn chọn các thanh 3 biến hoặc 4 biến



*Figure 1. Logic expression normalizer use case diagram*

# **Design idea**

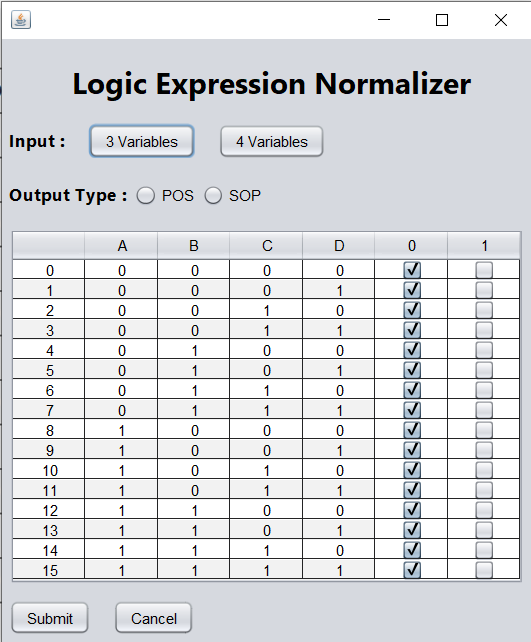
## The Model – View – Controller design patterns

Trong dự án nhỏ này, bọn em sẽ thiết kế chương trình của mình dựa trên Mô hình MVC (Model – View – Controller). Bằng cách này, mã nguồn được chia thành các phần khác nhau với mục đích riêng của chúng và khi chúng ta xem xét mã nguồn sẽ dễ dàng hơn

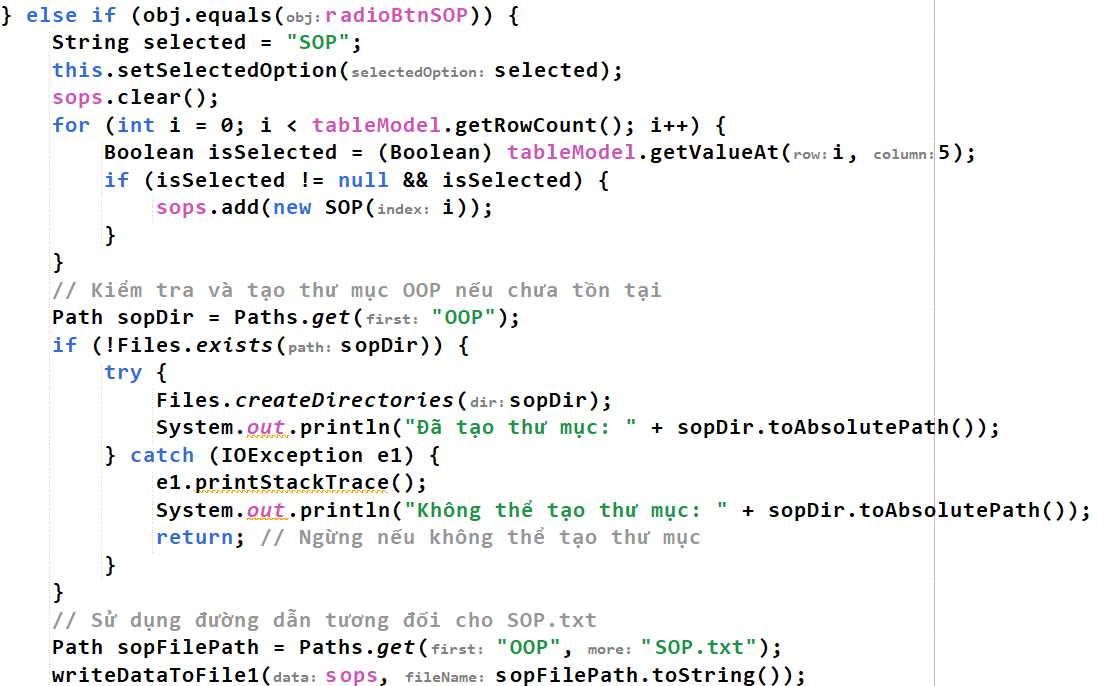
**The Model Package**: Phần này chịu trách nhiệm lưu trữ dữ liệu và tất cả các thành phần được sử dụng trong thuật toán Quine-McCluskey..

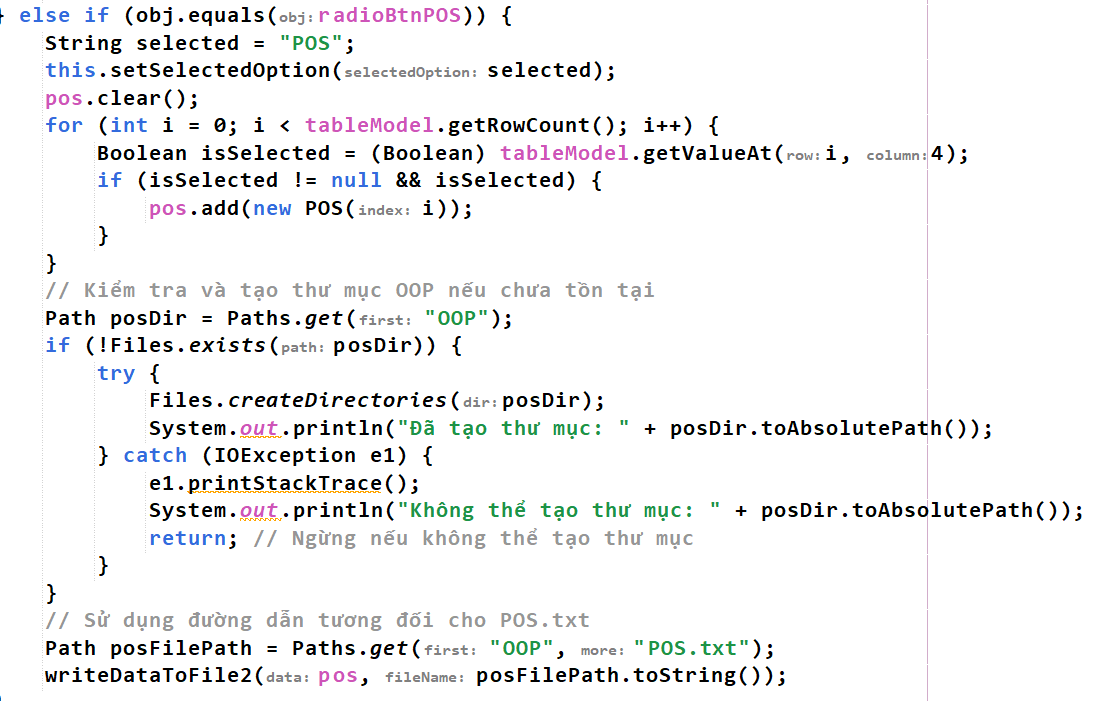
**The View Package**: Chương trình sử dụng Java Swing để hiển thị dữ liệu được người dùng nhập vào và đưa ra kết quả khi người dùng nhấn nút "submit". Cụ thể, phần này được thiết kế để trực quan hóa dữ liệu được quản lý bởi gói Model.

**The Controller Package**: Phần này sẽ liên kết 2 phần Model và View, nó xử lý luồng dữ liệu từ đầu vào của người dùng trong Model và cập nhật hiển thị qua View khi có thay đổi về dữ liệu

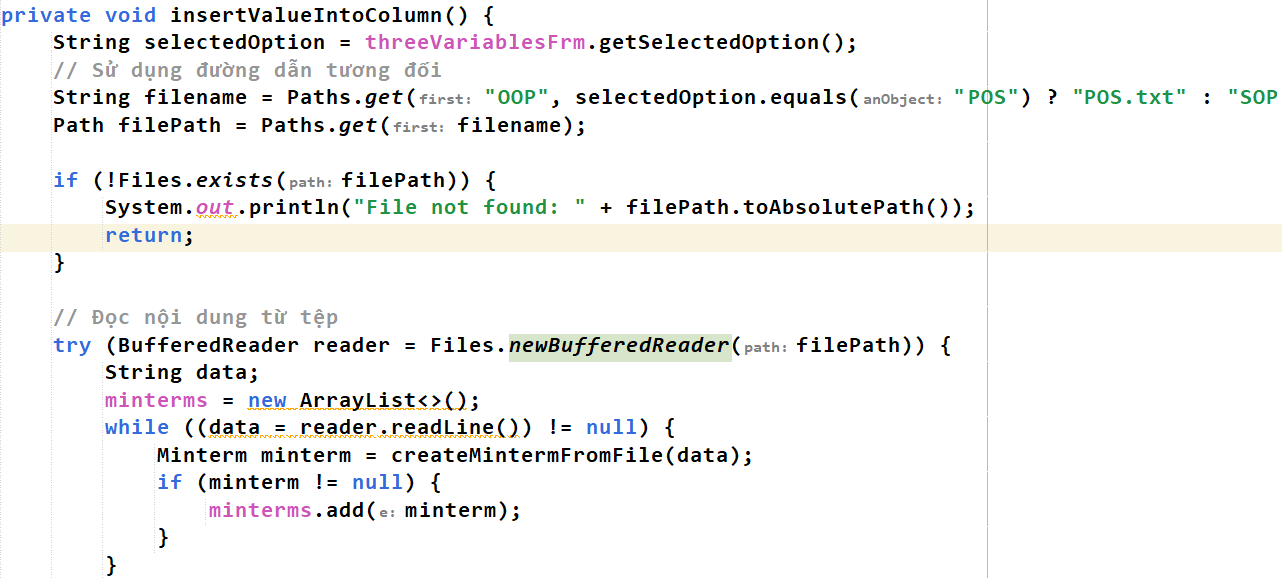


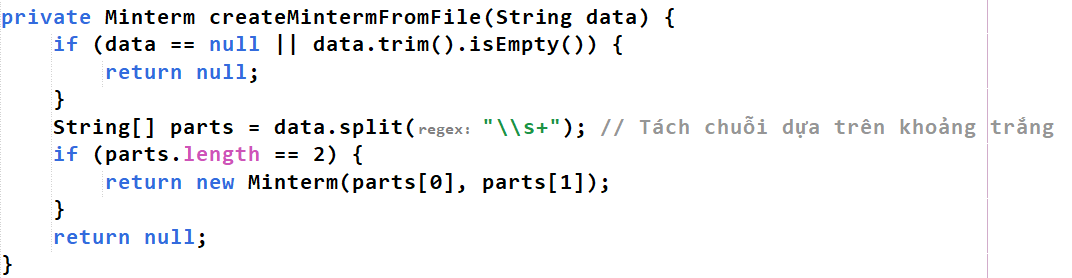
*Figure 2. The connection between Model – View - Controller*





*Figure 3. Extract data from View*

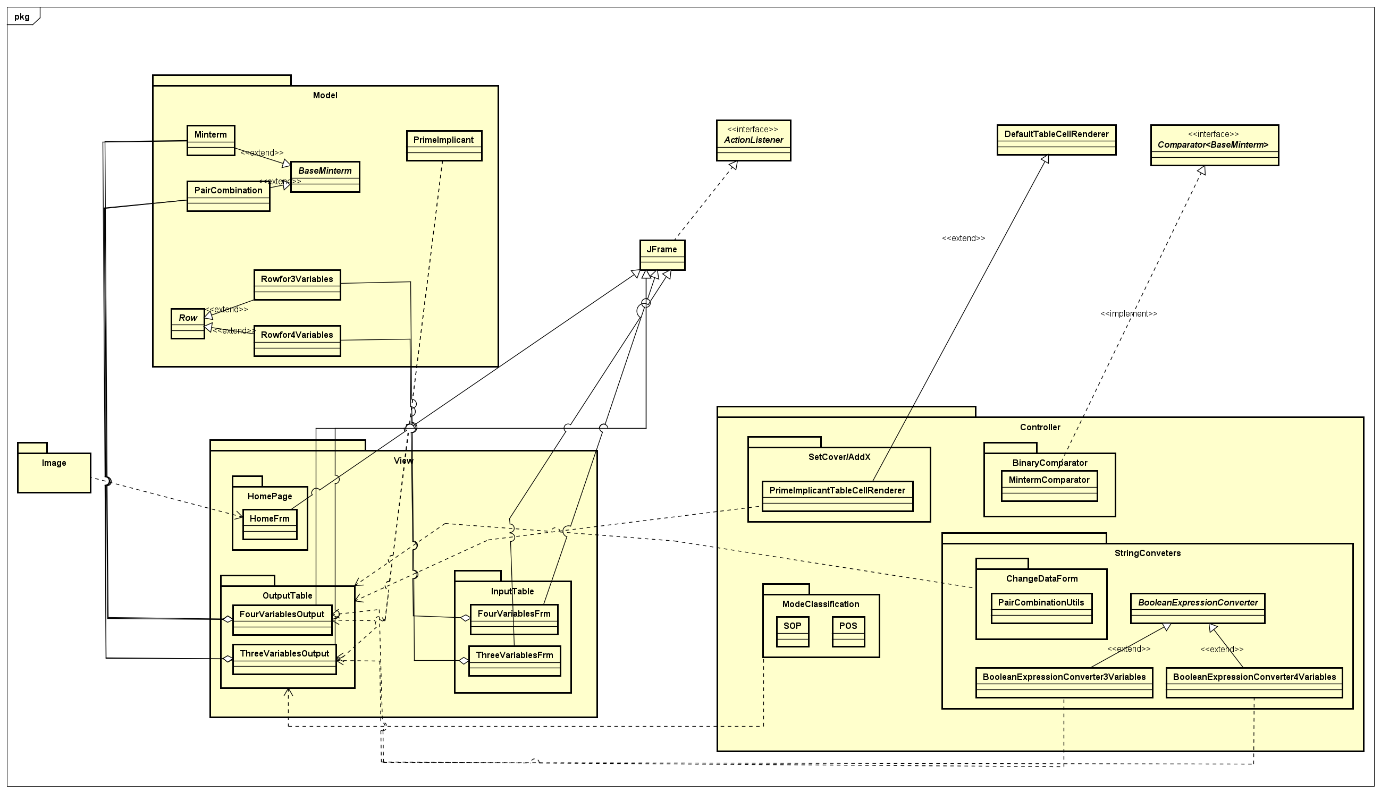
**

**

*Figure 3. Process data and Direct results and trigger View*

*to show data visualization*

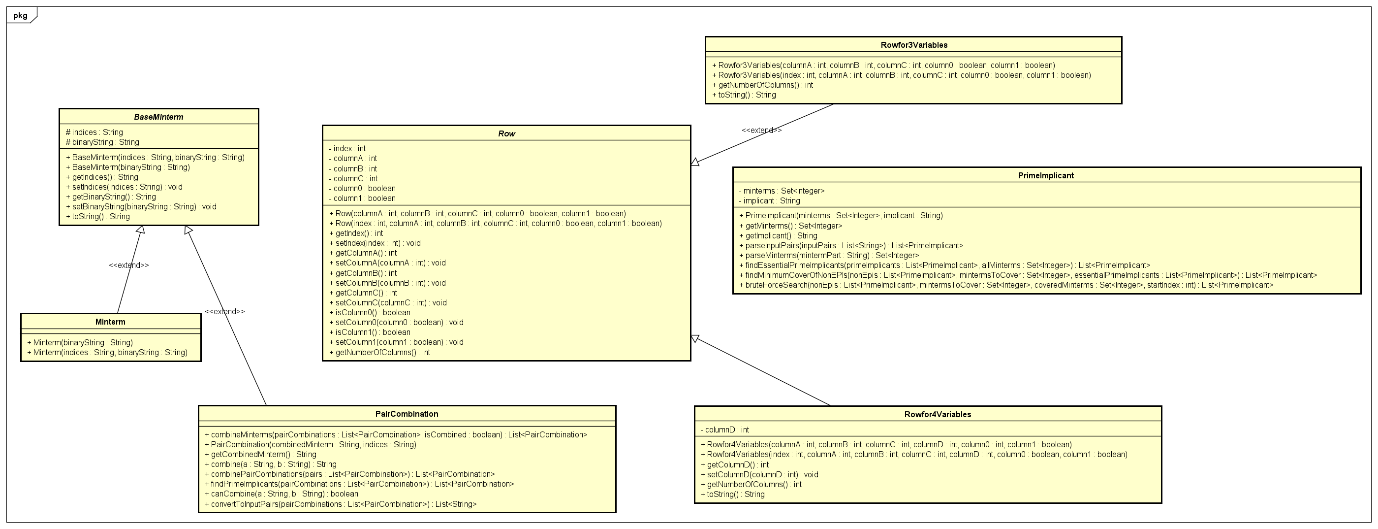
## General class diagram



*Figure 4. Logic expression normalizer class diagram*

## Class diagrams and explanation

### Model package:



*Figure 5. Model package’s class diagram*

Phần này sẽ có các lớp như sau:

**Lớp BaseMinterm:** Lớp này là một lớp trừu tượng đại diện cho các minterm (hoặc các kết hợp của minterm) bởi một thuật toán nhất định. Dưới đây là phân tích chi tiết về các thành phần của lớp này. Thuộc tính "indices" biểu thị chuỗi biểu diễn các chỉ số của các minterm đã được kết hợp. Thuộc tính "binaryString" biểu thị dạng nhị phân của minterm. Lớp cũng bao gồm các phương thức getter và setter cho "indices" và "binaryString", cùng một phương thức override là "toString()" trả về một chuỗi biểu diễn đối tượng, kết hợp các chỉ số và chuỗi nhị phân, hữu ích cho việc gỡ lỗi và đầu ra. Lớp này có các constructor như "BaseMinterm(String indices, String binaryString)" khởi tạo đối tượng với cả hai thuộc tính indices và một chuỗi nhị phân.

Lớp này là lớp trừu tượng và cung cấp cấu trúc cơ bản cho các triển khai cụ thể hơn như "Minterm" và "PairCombination".

**Lớp Minterm:** Lớp Minterm mở rộng từ BaseMinterm và đại diện cho một minterm đơn trong thuật toán Quine-McCluskey. Nó kế thừa tất cả các thuộc tính và phương thức từ BaseMinterm nhưng được tinh chỉnh để xử lý từng minterm cá nhân. Hiện tại, lớp Minterm không thêm các phương thức bổ sung nào, nhưng nó được sử dụng để sắp xếp "binaryString" và sau đó kết quả sẽ được chèn vào cột đầu tiên của IntermediateTable.

**Lớp PairCombination:** Lớp PairCombination mở rộng từ BaseMinterm và được thiết kế để đại diện cho các kết hợp của minterm. Nó đóng vai trò quan trọng trong việc kết hợp các minterm để đơn giản hóa biểu thức. Thuộc tính "combinedMinterm" tham chiếu đến chuỗi nhị phân biểu thị dạng kết hợp của các minterm. Dưới đây là chi tiết về các phương thức của lớp này:

* isCombined(): Kiểm tra xem minterm đã được kết hợp bằng cách tìm khoảng trắng trong chuỗi indices.
* combineMinterms(): Phương thức tĩnh cố gắng kết hợp một danh sách các đối tượng PairCombination. Nó xác định các cặp minterm khác nhau chỉ bởi một bit và kết hợp chúng.
* combinePairCombinations(): Phương thức tĩnh tiếp tục cố gắng kết hợp các minterm đã được kết hợp trước đó thành các nhóm lớn hơn.
* findPrimeImplicants(): Phương thức tĩnh xác định các prime implicant từ một danh sách các pair combination. Prime implicants là những implicant không thể kết hợp thêm.
* canCombine(): Phương thức tĩnh để kiểm tra xem hai chuỗi nhị phân khác nhau đúng một bit, xác định liệu chúng có thể được kết hợp hay không.
* convertToInputPairs(): Phương thức tĩnh chuyển đổi một danh sách các đối tượng PairCombination thành một danh sách các chuỗi phù hợp cho việc xử lý hoặc đầu ra sau.

**Lớp PrimeImplicant:** Lớp PrimeImplicant rất quan trọng trong quá trình quản lý prime implicants (những hàm biểu diễn chính) trong quá trình đơn giản hóa. Những hàm biểu diễn chính là các thành phần quan trọng trong biểu thức boolean đã được đơn giản hóa. Lớp này có các thuộc tính như sau:

* "Set<Integer> minterms": Lưu trữ các chỉ số của các minterm mà prime implicant này bao phủ.
* "String implicant": Biểu diễn những hàm biểu diễn chính dưới dạng chuỗi đã được đơn giản hóa.

Dưới đây là mô tả chi tiết về mục đích chính của lớp này:

* PrimeImplicant(Set<Integer> minterms, String implicant): Khởi tạo một PrimeImplicant với một tập hợp các minterm và chuỗi hàm tương ứng.
* parseInputPairs(List<String> inputPairs): Đầu tiên, nó chuyển đổi một danh sách các chuỗi biểu diễn hàm thành một danh sách các đối tượng PrimeImplicant. Mỗi chuỗi đầu vào được phân tách thành các phần minterm và hàm, phân tích thành các dạng tương ứng và sau đó được sử dụng để tạo các đối tượng PrimeImplicant.
* parseMinterms(String mintermPart): Chuyển đổi một chuỗi biểu diễn các chỉ số minterm thành một tập hợp các số nguyên và loại bỏ bất kỳ dấu ngoặc nào, phân tách chuỗi bằng dấu phẩy, loại bỏ khoảng trắng và phân tích từng phần thành một số nguyên.
* findEssentialPrimeImplicants(List<PrimeImplicant> primeImplicants, Set<Integer> allMinterms): Xác định các prime implicant cần thiết từ một danh sách các prime implicant. Các prime implicant cần thiết là những implicant bao phủ các minterm mà không có prime implicant nào khác bao phủ. Phương thức này xây dựng một bản đồ từ các minterm đến các prime implicant bao phủ chúng và xác định các minterm được bao phủ bởi đúng một prime implicant.
* findMinimumCoverOfNonEPIs(List<PrimeImplicant> nonEpis, Set<Integer> mintermsToCover, List<PrimeImplicant> essentialPrimeImplicants): Tìm tập con nhỏ nhất của prime implicant không cần thiết (non-EPIs) để bao phủ tất cả các minterm còn lại mà không được bao phủ bởi prime implicant cần thiết.
* bruteForceSearch(List<PrimeImplicant> nonEpis, Set<Integer> mintermsToCover, Set<Integer> coveredMinterms, int startIndex): Một phương thức đệ quy được sử dụng trong findMinimumCoverOfNonEPIs để khám phá tất cả các tổ hợp có thể của non-EPIs để tìm tập con nhỏ nhất bao phủ tất cả các minterm còn lại.
* Nó liên tục kết hợp các minterm được bao phủ bởi non-EPIs hiện tại và kiểm tra xem chúng có bao phủ tất cả các minterm cần thiết hay không.

**Lớp Row:** Lớp Row là một lớp cơ sở trừu tượng được thiết kế để đại diện cho một hàng trong bảng dữ liệu. Nó được sử dụng để chèn giá trị vào ThreeVariablesFrm và FourVariablesFrm và lấy giá trị của minterm bằng cách kiểm tra giá trị boolean trong hai cột cuối cùng. Lớp này có các thuộc tính như sau:

* "index", "columnA", "columnB", "columnC": Các cột dữ liệu để biểu diễn các biến A, B, C.
* "column0", "column1": Các cột dữ liệu cho biểu diễn trong chế độ SOP hoặc POS mà người dùng muốn.

**Lớp Rowfor3Variables và Rowfor4Variables:** Lớp Rowfor3Variables mở rộng từ lớp Row và được đặc biệt hóa cho trường hợp ba biến. Lớp Rowfor4Variables cũng mở rộng từ lớp Row, nó kế thừa tất cả các thuộc tính của lớp Row và có thêm một thuộc tính là "columnD" để phù hợp với số cột của bảng. Cả hai lớp này đều có phương thức toString() được ghi đè (@Override).

### View package:



*Figure 6. View package’s class diagram*

* Here is a general method I use for 5 classes:

- **Phương thức initComponents():**

* Phương thức này thiết lập giao diện và khởi tạo các thành phần của UI.
* Nó định nghĩa bố cục và các thuộc tính của các nhãn (labels), nút (buttons), và bảng điều khiển (panel).
* Nó cũng đặt vị trí của các thành phần bằng cách sử dụng tọa độ tuyệt đối, cung cấp vị trí chính xác trong cửa sổ.

**Phương thức addActionListener():**

* Gắn các bộ lắng nghe sự kiện (action listener) cho mỗi nút.
* Điều này đảm bảo rằng khi người dùng nhấp vào nút, phương thức actionPerformed sẽ được gọi.

**Phương thức actionPerformed(ActionEvent e):**

* Phương thức này xử lý các hành động được kích hoạt bởi các nút được nhấp vào.

### Class HomeFrm

Trong lớp HomeFrm, khi người dùng nhấp vào btn3Variables, một ThreeVariablesFrm mới được khởi tạo và hiển thị. Tương tự cho btn4Variables. Đối với btnTips, một hộp thoại tin nhắn được hiển thị với một lời khuyên hữu ích.

### Class Three/FourVariablesFrm

Lớp này chịu trách nhiệm hiển thị biểu mẫu nhập liệu, nơi người dùng có thể nhập một biểu thức Boolean ba biến và lựa chọn định dạng đầu ra (Tổng các Tích (SOP) hoặc Tích các Tổng (POS)). Nó quản lý việc thu thập dữ liệu đầu vào và khởi động quá trình đơn giản hóa.

**Các phương thức chính hỗ trợ:**

* addActionListener(): Gắn bộ lắng nghe sự kiện cho các nút và nút radio.
* addButtonGroup(): Nhóm các nút radio để chỉ cho phép chọn duy nhất một lựa chọn vào một thời điểm.
* insertValueIntoTable(): Điền dữ liệu vào JTable với tất cả các kết hợp của 3 biến.
* setCellRenderers(): Căn giữa văn bản trong các ô của bảng.
* writeDataToFile1(): Ghi dữ liệu SOP vào một tệp.
* writeDataToFile2(): Ghi dữ liệu POS vào một tệp.

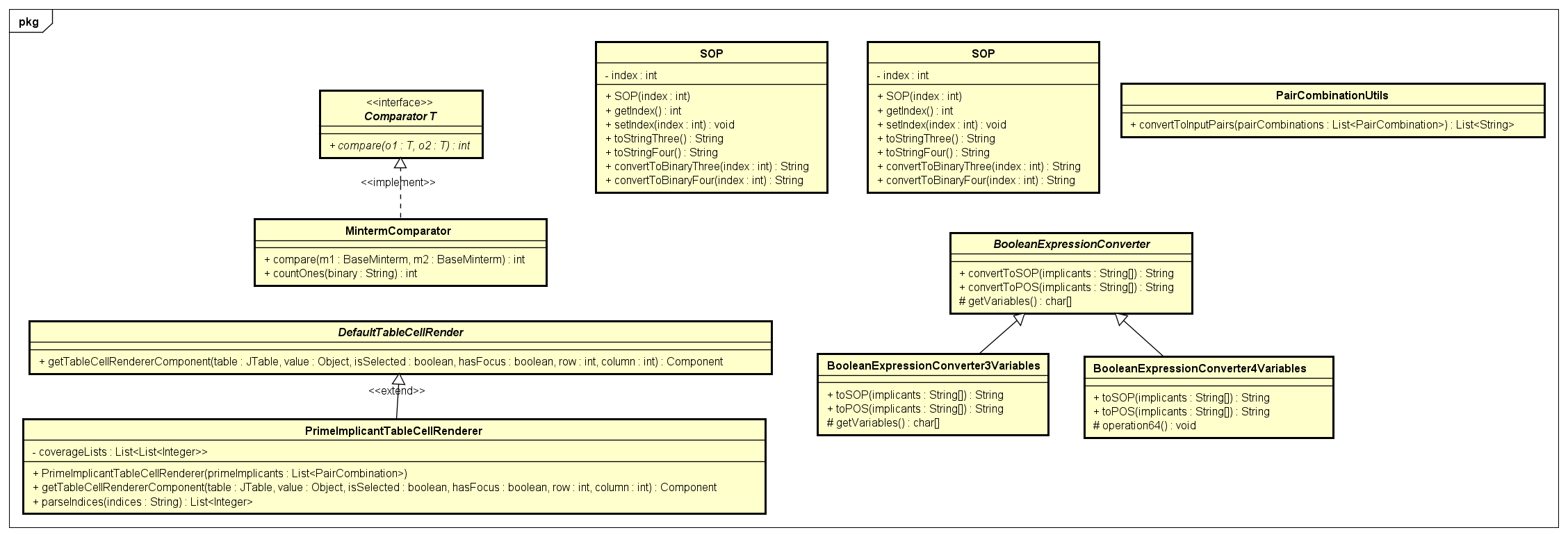
### Class Three/FourVariableOutput

Lớp này hiển thị kết quả của việc đơn giản hóa biểu thức Boolean, bao gồm các bảng trung gian và biểu thức đơn giản cuối cùng. Nó minh họa các bước trung gian của thuật toán Quine-McCluskey.

**Các phương thức chính hỗ trợ:**

* addActionListener(): Gắn bộ lắng nghe sự kiện cho các nút.
* insertValueIntoColumn(): Đọc dữ liệu đầu vào từ tệp, xử lý các minterm, kết hợp chúng thành các prime implicants, và điền vào các bảng.
* setCellRenderers(): Căn giữa văn bản trong các ô của bảng.
* clearContentsOfFile(): Xoá nội dung của tệp được chỉ định sau khi đọc xong.
* createMintermFromFile(): Phân tích một minterm từ dữ liệu tệp.
* insertIntoPiTable(): Chèn các Prime Implicant vào bảng PI.
* applyPrimeImplicantRenderer(): Áp dụng định dạng tùy chỉnh cho bảng PI để làm nổi bật prime implicants.
* insertValueIntoTable(): Xử lý các prime implicants để tìm ra các PI cần thiết và không cần thiết và chuyển đổi chúng thành biểu thức Boolean cuối cùng.

Controller package:



*Figure 7. Controller package’s class diagram*

### Lớp PairCombinationUtils

Lớp tiện ích này được thiết kế để chuyển đổi một danh sách các đối tượng PairCombination thành một danh sách các chuỗi. Mỗi chuỗi trong danh sách kết quả được hình thành bằng cách nối hai thuộc tính của mỗi đối tượng PairCombination: indices và combinedMinterm, cách nhau bởi một dấu cách. Điều này hữu ích cho các tác vụ yêu cầu xử lý các giá trị kết hợp này như chuỗi đầu vào. Giá trị trả về sẽ là một danh sách các chuỗi, trong đó mỗi chuỗi đại diện cho một đối tượng PairCombination.

### Lớp MintermComparator

Lớp so sánh này được sử dụng để sắp xếp các đối tượng "BaseMinterm" dựa trên biểu diễn chuỗi nhị phân của chúng. Nó chủ yếu sắp xếp các đối tượng dựa trên số lượng bit '1' trong chuỗi nhị phân của chúng, và nếu hai đối tượng có cùng số lượng bit '1', nó sẽ sắp xếp chúng dựa trên giá trị nhị phân của chúng. Lớp này rất quan trọng cho các nhiệm vụ liên quan đến việc sắp xếp các đối tượng "BaseMinterm" theo một thứ tự cụ thể để xử lý tiếp theo.

### Lớp SOP/POS

Lớp này đại diện cho việc phân loại chế độ Product of Sums (POS) hoặc Sum of Products (SOP) trong đại số Boolean. Nó đóng gói một chỉ số và cung cấp các phương thức để chuyển đổi chỉ số thành biểu diễn nhị phân của nó cho cả ba biến và bốn biến.

### Lớp BooleanExpressionConverter

Lớp trừu tượng này cung cấp một khung cho việc chuyển đổi biểu thức Boolean thành dạng Sum of Products (SOP) và Product of Sums (POS). Nó định nghĩa các phương thức để thực hiện các chuyển đổi này dựa trên biểu diễn nhị phân của các biểu thức.

### Lớp BooleanExpressionConverter3/4Variables

Lớp con này của BooleanExpressionConverter được thiết kế đặc biệt để chuyển đổi các biểu thức Boolean với ba biến (A, B, và C) và bốn biến (bao gồm biến D) thành dạng SOP hoặc POS. Nó triển khai phương thức trừu tượng để trả về tập hợp các biến được sử dụng trong quá trình chuyển đổi.

Đây là một tổng quan về các lớp và tiện ích trong gói controller của dự án Java của bạn, chú trọng vào xử lý đầu vào và xử lý biểu thức Boolean bằng phương pháp Quine-McCluskey.

### OOP-Principle in our Java Application:

 **Trừu tượng (Abstraction):**

* **Định nghĩa:** Trừu tượng là khái niệm ẩn đi các chi tiết triển khai phức tạp và chỉ hiển thị các đặc tính cần thiết của đối tượng.
* **Ứng dụng:**
  + **Abstract Classes:**
    - **BaseMinterm:** Lớp này trừu tượng hóa các thuộc tính (indices và binaryString) và hành vi cơ bản (phương thức để lấy và thiết lập các thuộc tính này) của các minterm.
    - **Row:** Lớp này trừu tượng hóa khái niệm của một hàng trong ngữ cảnh của biến và biểu thức Boolean, xác định các thuộc tính chung và các phương thức trừu tượng như getNumberOfColumns().
    - **BooleanExpressionConverter:** Lớp này cung cấp một sự trừu tượng hóa cho việc chuyển đổi biểu thức Boolean thành dạng SOP và POS mà không chi tiết hóa cách xử lý biến, điều này để lại cho các lớp con xử lý.

 **Đóng gói (Encapsulation):**

* **Định nghĩa:** Đóng gói là việc gói gọn dữ liệu và các phương thức hoạt động trên dữ liệu trong một đơn vị như lớp, và hạn chế quyền truy cập vào một số thành phần của đối tượng.
* **Ứng dụng:**
  + **Thành viên private và protected:** Các lớp như BaseMinterm, Row và PrimeImplicant đóng gói dữ liệu của chúng (ví dụ như indices, binaryString, minterms, v.v.) và cung cấp các phương thức công khai để truy cập và thay đổi dữ liệu này. Điều này đảm bảo quyền truy cập và sửa đổi kiểm soát vào trạng thái nội bộ.
  + **Các trình truy cập và trình thay đổi:** Các phương thức getter và setter (ví dụ như getIndices(), setIndices(), getMinterms(), setMinterms()) được sử dụng để điều khiển cách truy cập và thay đổi thuộc tính trong các lớp như BaseMinterm và PrimeImplicant.

 **Ẩn dữ liệu (Data Hiding):**

* **Định nghĩa:** Ẩn dữ liệu là nguyên tắc ẩn các chi tiết đối tượng nội bộ khỏi thế giới bên ngoài, chỉ tiết lộ các phần cần thiết thông qua các phương thức.
* **Ứng dụng:**
  + **Biến private:** Hầu hết các trường lớp (ví dụ như indices, binaryString trong BaseMinterm; minterms trong PrimeImplicant) được khai báo là private, đảm bảo rằng chúng không thể được truy cập trực tiếp từ bên ngoài lớp.
  + **Kiểm soát quyền truy cập:** Việc truy cập vào các trường private này được kiểm soát thông qua các phương thức công khai hoặc protected, duy trì tính toàn vẹn của dữ liệu.

 **Khởi tạo và Sử dụng đối tượng (Object Initialization & Usage):**

* **Định nghĩa:** Khởi tạo đối tượng đề cập đến việc tạo ra và thiết lập đối tượng để sử dụng. Sử dụng liên quan đến tương tác với các phương thức và thuộc tính của đối tượng.
* **Ứng dụng:**
  + **Constructor:** Mỗi lớp đều có các constructor để khởi tạo các đối tượng của nó. Ví dụ, Minterm và PairCombination sử dụng constructor để khởi tạo các thuộc tính của chúng (indices, binaryString).
  + **Các phương thức giống như Factory:** Các phương thức như PairCombinationUtils.convertToInputPairs được sử dụng để tạo ra và điều khiển các bộ sưu tập đối tượng một cách hiệu quả.

 **Kế thừa (Inheritance):**

* **Định nghĩa:** Kế thừa cho phép một lớp kế thừa các trường và phương thức từ một lớp khác.
* **Ứng dụng:**
  + **Các thứ bậc lớp:** Minterm và PairCombination kế thừa từ BaseMinterm, cho phép chúng chia sẻ các thuộc tính và hành vi chung trong khi cũng có các chức năng cụ thể riêng.
  + **BooleanExpressionConverter3Variables và các lớp kế thừa khác (không được hiển thị nhưng được giả định):** Kế thừa từ BooleanExpressionConverter, lớp này nhận các phương thức chung của chuyển đổi và thêm xử lý biến cụ thể.

 **Đa hình (Polymorphism):**

* **Định nghĩa:** Đa hình cho phép các đối tượng của các lớp khác nhau được xem như là đối tượng của một lớp siêu phổ biến, thường cho phép ghi đè phương thức và phân giải phương thức động.
* **Ứng dụng:**
  + **Các phương thức trừu tượng và ghi đè:** Các phương thức trong BooleanExpressionConverter là trừu tượng và được thực hiện khác nhau trong các lớp con của nó, cho phép hành vi khác nhau trong khi xử lý chúng như kiểu BooleanExpressionConverter.
  + **Gọi phương thức động:** Sử dụng các phương thức ghi đè theo cách đa hình, trong đó phương thức thực sự được gọi được xác định tại thời điểm chạy dựa trên lớp của đối tượng (ví dụ convertToSOP và convertToPOS trong các lớp con BooleanExpressionConverter khác nhau).

 **Mối quan hệ (Association):**

* **Định nghĩa:** Mối quan hệ là một liên kết giữa hai lớp cho phép một lớp liên kết với lớp khác.
* **Ứng dụng:**
  + **HomeFrm và Bảng nhập:** Lớp HomeFrm có mối quan hệ với các biểu mẫu nhập như ThreeVariablesFrm và FourVariablesFrm, mà nó khởi tạo và tương tác dựa trên hành động của người dùng.
  + **ThreeVariablesOutput và ThreeVariablesFrm:** ThreeVariablesOutput có tham chiếu đến ThreeVariablesFrm, chỉ ra rằng nó sử dụng biểu mẫu này để đọc lựa chọn nhập của người dùng.

 **Tổng hợp (Aggregation):**

* **Định nghĩa:** Tổng hợp là một dạng đặc biệt của mối quan hệ, trong đó một lớp là một bộ sưu tập hoặc chứa các lớp khác.
* **Ứng dụng:**
  + **Các danh sách đối tượng:** ThreeVariablesFrm tổng hợp các đối tượng Rowfor3Variables trong một danh sách. Tương tự, ThreeVariablesOutput chứa các bộ sưu tập các đối tượng Minterm và PairCombination.
  + **Các cấu trúc dữ liệu phức hợp:** Các lớp như PrimeImplicant tổng hợp các bộ sưu tập các chỉ số (minterms) để biểu diễn mối quan hệ của chúng trong việc đơn giản hóa Boolean.

 **Tổng quát hóa (Generalization):**

* **Định nghĩa:** Tổng quát hóa là quá trình rút trích các đặc điểm chung từ hai hoặc nhiều lớp và kết hợp chúng thành một lớp siêu tổng quát.
* **Ứng dụng:**
  + **Quan hệ lớp cha và lớp con:** BaseMinterm tổng quát hóa các thuộc tính chung của minterms, và các lớp con của nó (Minterm, PairCombination) đặc biệt hóa khái niệm này.
  + \*\*BooleanExpressionConverter tổng quát hóa phương pháp chuyển đổi của biểu thức Boolean, và các lớp con xử lý các trường hợp cụ thể cho số lượng biến khác nhau.

 **Nạp chồng phương thức (Method Overloading):**

* **Định nghĩa:** Nạp chồng phương thức là khả năng định nghĩa nhiều phương thức cùng tên nhưng có danh sách tham số khác nhau.
* **Ứng dụng:**
  + **Các constructor khác nhau:** Trong các lớp như Rowfor3Variables và POS, các constructor được nạp chồng để cho phép khởi tạo các đối tượng theo nhiều cách khác nhau (với số lượng tham số khác nhau).

 **Giao diện và Lớp trừu tượng (Interface and Abstract Classes):**

* **Định nghĩa:** Giao diện định nghĩa một hợp đồng các phương thức mà không có cài đặt, trong khi lớp trừu tượng cung cấp một lớp cơ sở với một số cài đặt chung.
* **Ứng dụng:**
  + **Lớp trừu tượng:** BooleanExpressionConverter và BaseMinterm là các lớp trừu tượng cung cấp chức năng cơ bản và các phương thức trừu tượng để được cài đặt bởi các lớp con.
  + **Giao diện có thể có:** Mặc dù không được hiển thị rõ ràng, các giao diện có thể được sử dụng để định nghĩa các hành vi như "Convertible" cho các đối tượng có thể chuyển đổi thành các dạng khác nhau (SOP/POS).

### Workflow:

1. **Nhập dữ liệu từ người dùng:**
   1. Người dùng tương tác với giao diện HomeFrm để chọn làm việc với biểu thức logic 3 hoặc 4 biến.
   2. Điều này mở ra các giao diện ThreeVariablesFrm hoặc FourVariablesFrm, nơi người dùng nhập các thuật ngữ biểu thức logic vào một bảng và chọn loại đầu ra (SOP hoặc POS).
2. **Xử lý dữ liệu:**
   1. Khi người dùng nhấn nút submit, giao diện (ThreeVariablesFrm hoặc FourVariablesFrm) thu thập dữ liệu từ người dùng và ghi vào tệp (SOP.txt hoặc POS.txt).
   2. Sau đó, nó mở giao diện kết quả tương ứng (ThreeVariablesOutput hoặc FourVariablesOutput).
3. **Kết hợp các minterm:**
   1. Giao diện kết quả đọc các minterm từ tệp, tạo đối tượng Minterm và sắp xếp chúng.
   2. Sau đó, nó sử dụng lớp PairCombination để kết hợp các minterm, xác định những minterm nào có thể kết hợp dựa trên biểu diễn nhị phân của chúng.
   3. Các kết hợp này được xử lý tiếp để tìm ra các prime implicants.
4. **Xác định prime implicants:**
   1. Các minterm đã kết hợp được phân tích để xác định các prime implicants bằng cách sử dụng các phương pháp trong lớp PairCombination.
   2. Prime implicants là những kết hợp minterm mà không thể kết hợp thêm nữa và phủ hết các minterm cần thiết.
5. **Rút gọn cuối cùng:**
   1. Các prime implicants được xác định được chuyển đổi thành biểu thức logic đơn giản ở dạng SOP hoặc POS bằng cách sử dụng lớp chuyển đổi phù hợp (BooleanExpressionConverter3Variables hoặc BooleanExpressionConverter4Variables).
   2. Biểu thức logic rút gọn cuối cùng được hiển thị trên giao diện kết quả.
6. **Tương tác với người dùng:**
   1. Người dùng có thể xem các bước trung gian và biểu thức logic rút gọn cuối cùng trên giao diện kết quả.
   2. Họ có thể chọn để rút gọn lại hoặc quay lại menu chính để thực hiện các thao tác khác.