OOP Mini-Project Report

A. Assignment of members

Members

- Phạm Đinh Gia Dũng - 20194428
- Phùng Quốc Việt - 20194463
- Vũ Quốc Việt - 20194464

Assignment

GUI

- ThreeVariableScreen & Controller: Phạm Đinh Gia Dũng
- FourVariableScreen & Controller: Phạm Đinh Gia Dũng
- Output Screen: Phạm Đinh Gia Dũng
- ColumnBlock: Phạm Đinh Gia Dũng & Phùng Quốc Việt
- PIBlock: Vũ Quốc Việt

Report

- Equally distributed

Classes

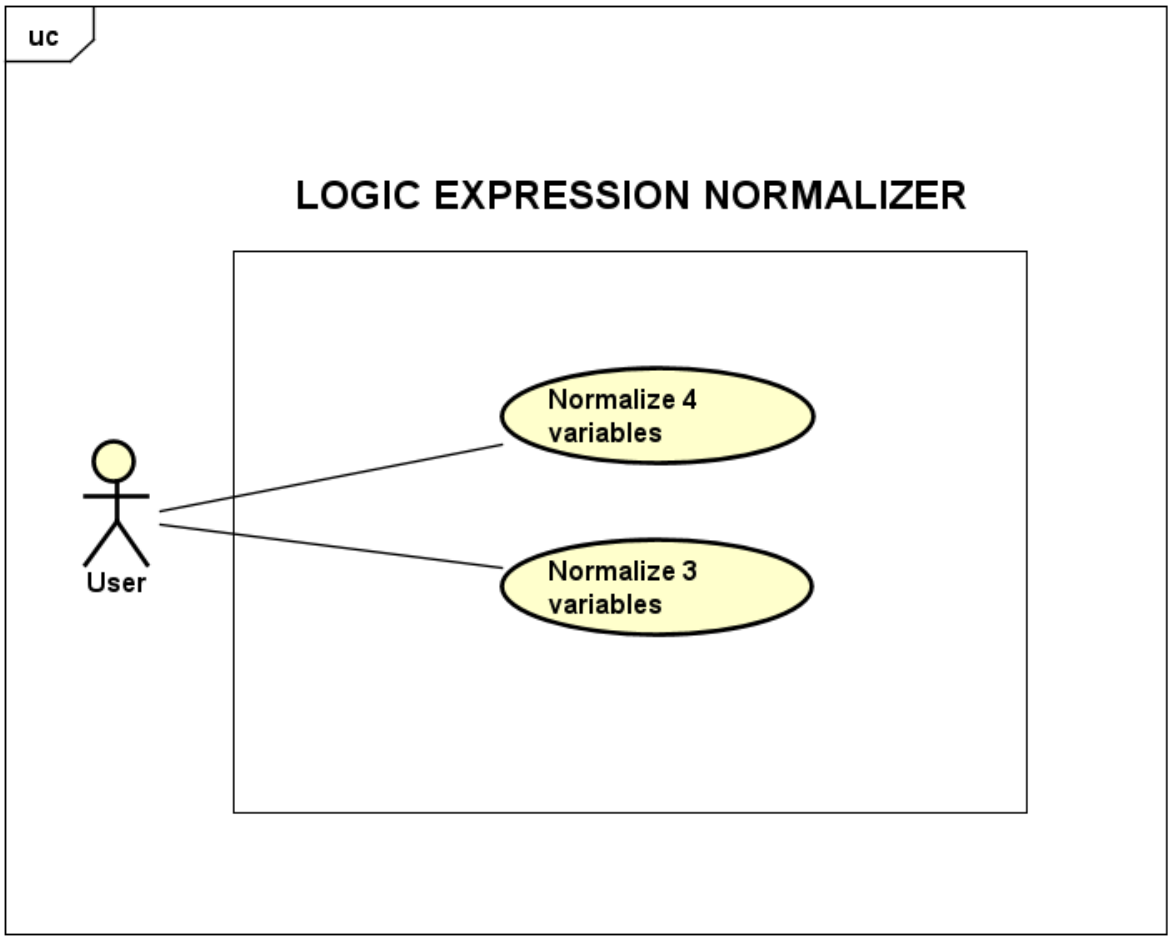
- Table & TruthTable: Phùng Quốc Việt & Phạm Đinh Gia Dũng
- IntermediateColum: Phùng Quốc Việt
- IntermediateColumnContainer: Phùng Quốc Việt
- Group: Phùng Quốc Việt
- Term & MinTerm & CombinedTerm: Phùng Quốc Việt
- PITable : Vũ Quốc Việt

B. Project description

Mini-Project requirement

- This application is a logic expression normalizer using the Quine-McCluskey algorithm. It takes a boolean expression as an input.
- User can choose two cases of input (3 variables or 4 variables expression) and two cases of output (POS or SOP canonical form)
- Application outputs: The minimal boolean expression, intermediate columns and PI Table

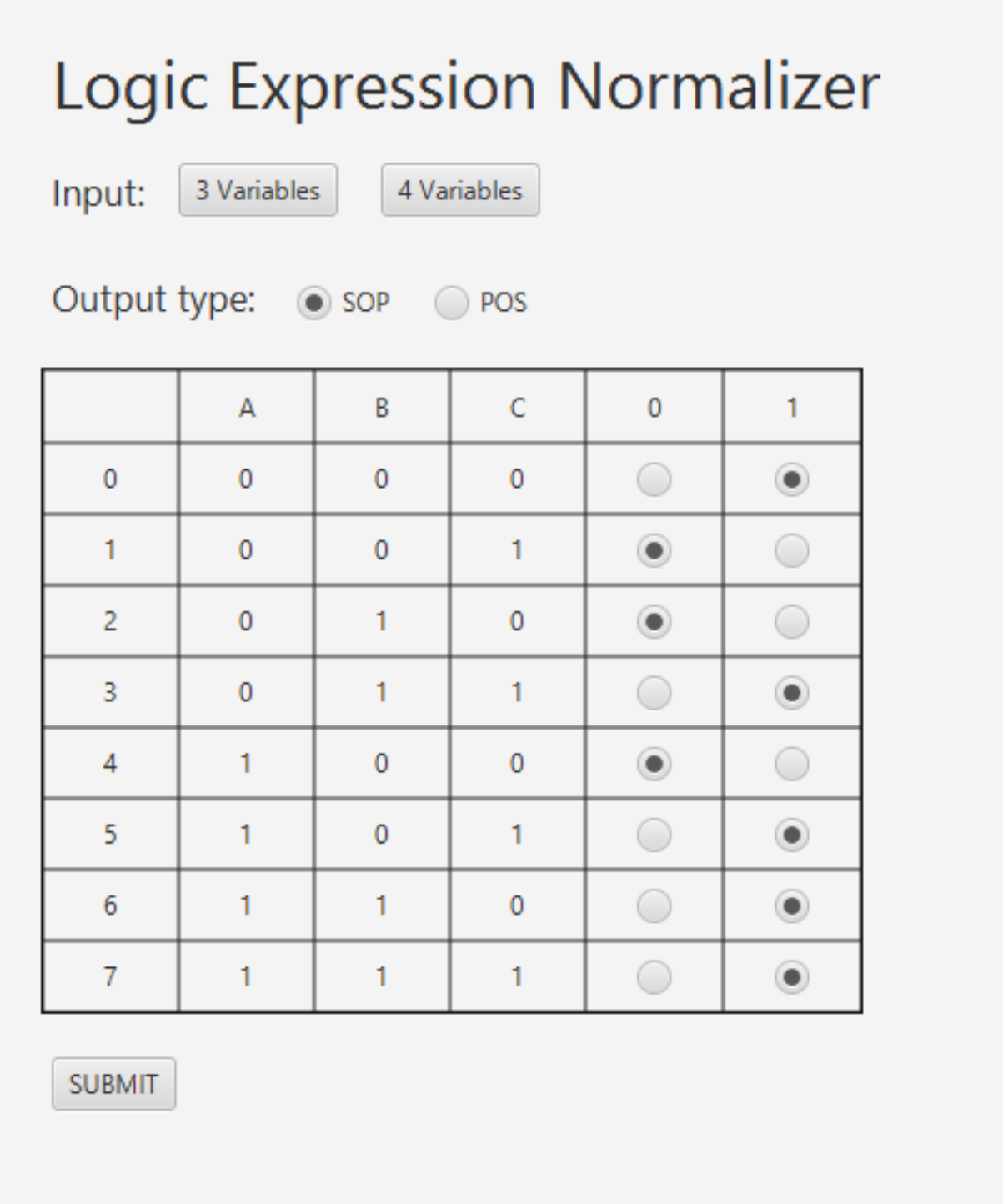
Use case & explanation



1. **Normalize 3 variables:** This is the layout of one of two main screens.

▼ This use case include these events

- Choose number of variables for input: Default option upon starting the application is 3 variables, users can choose the number of variables by pressing “3 Variables” button or“4 Variables” button, upon choosing a new window will appear with the corresponding user interface.
- Choose output type: Default option upon starting the application is SOP canonical form, user can choose the number of variables by pressing button “SOP” or button “POS”.
- Submit input: The output screen will appear, containing the simplified expression (SOP or POS depend on user previous choice) , intermediate columns and PI Table



3 Variables Input screen

2. **Normalize 4 variables:** This is the layout of one of two main screens.

▼ This use case include these events

- Choose number of variables for input: Default option upon starting the application is 3 variables, users can choose the number of variables by pressing “3 Variables” button or“4 Variables” button, upon choosing a new window will appear with the corresponding user interface.

- Choose output type: Default option upon starting the application is SOP canonical form, user can choose the number of variables by pressing button “SOP” or button “POS”.
- Submit input: The output screen will appear, containing the simplified expression (SOP or POS depend on user previous choice) , intermediate columns and PI Table

Logic Expression Normalizer

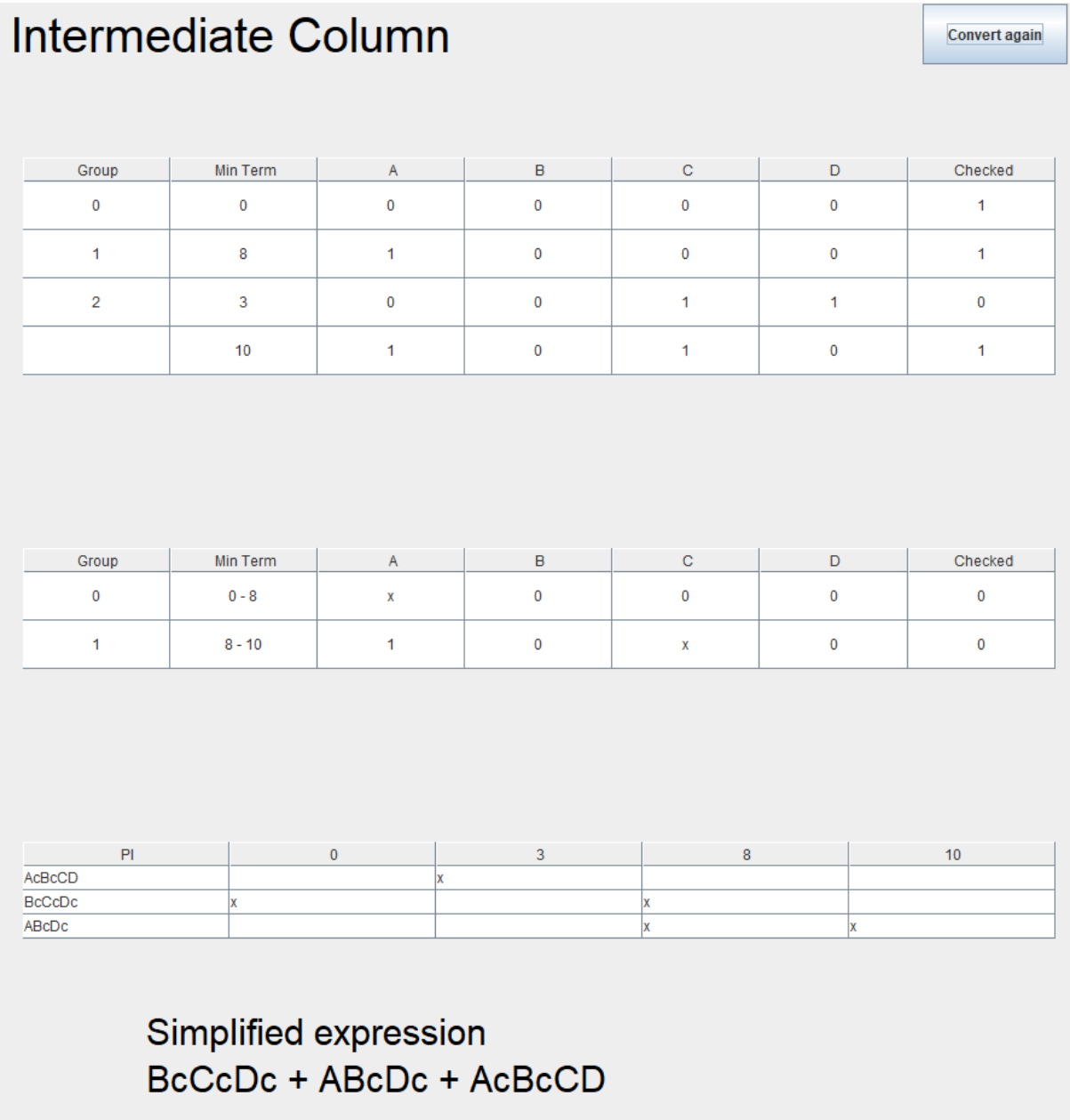
Input: 3 Variables 4 Variables

Output type: SOP POS

	A	B	C	D	0	1
0	0	0	0	0	○	●
1	0	0	0	1	●	○
2	0	0	1	0	●	○
3	0	0	1	1	○	●
4	0	1	0	0	●	○
5	0	1	0	1	●	○
6	0	1	1	0	●	○
7	0	1	1	1	●	○
8	1	0	0	0	○	●
9	1	0	0	1	●	○
10	1	0	1	0	○	●
11	1	0	1	1	●	○
12	1	1	0	0	●	○
13	1	1	0	1	●	○
14	1	1	1	0	●	○
15	1	1	1	1	●	○

SUBMIT

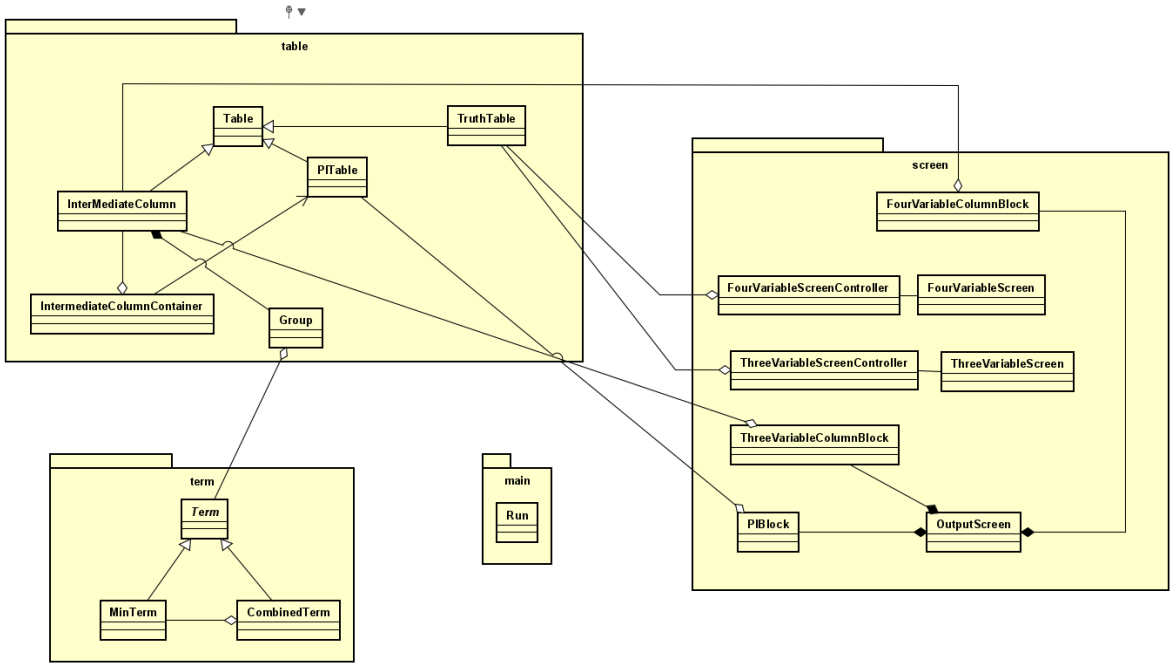
4 Variables Input screen



Output Screen

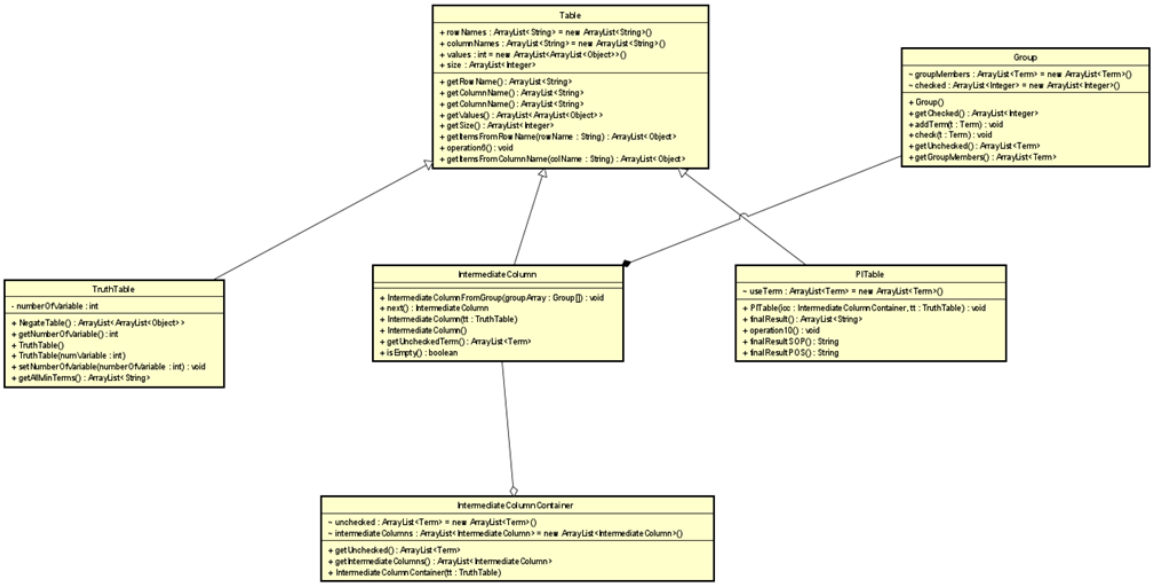
C. Design

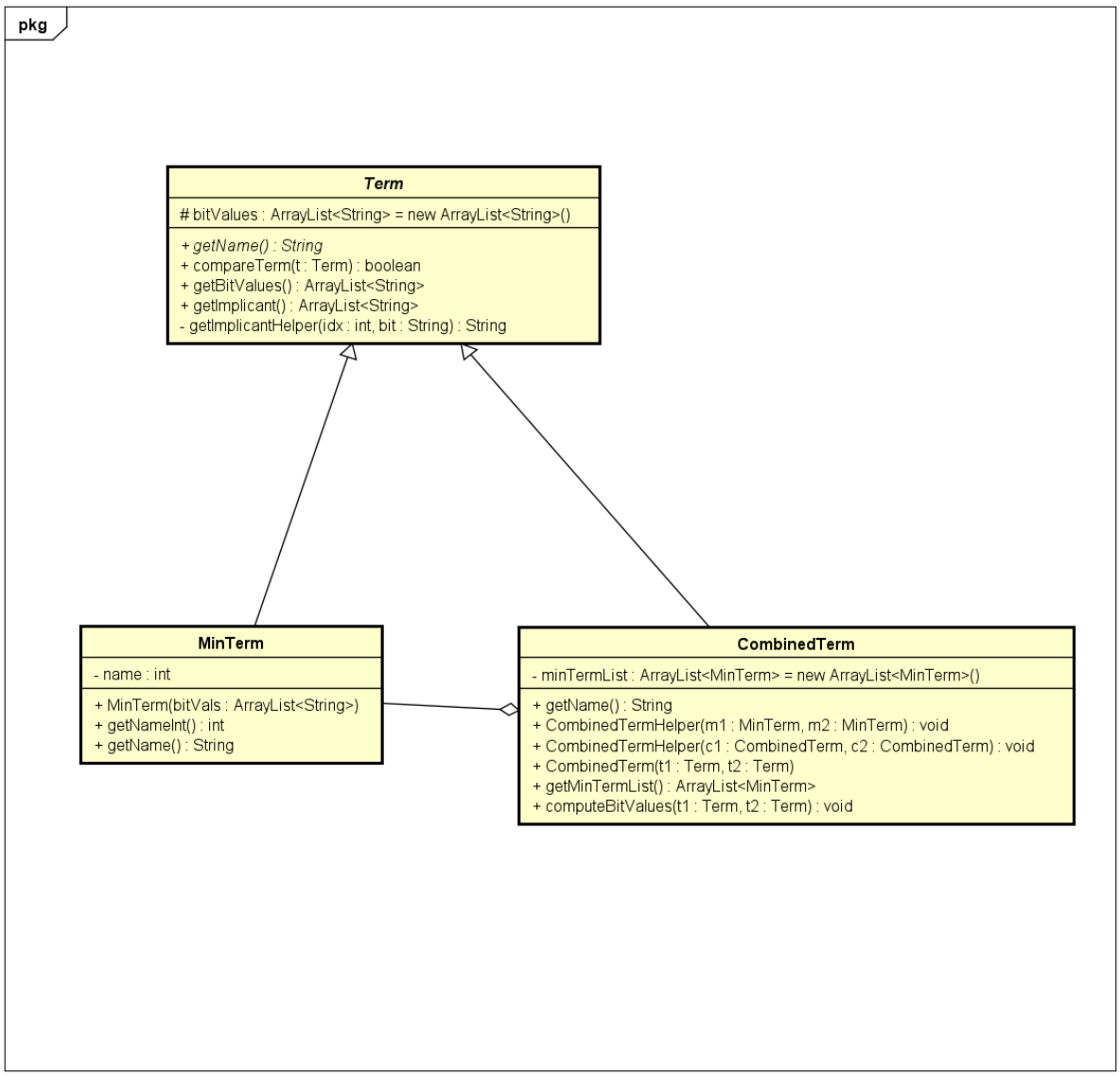
General class diagram



General Diagram

Package Diagram





Package: term

Explanation

1. Term Package

- Contains an abstract class Term and 2 child classes MinTerm and CombinedTerm.
- MinTerm is MinTerm
- CompareTerm is the representation of a matched pair in the intermediate columns. It stores a list of minterms as well as the combined binary representation.
- Important methods

💡 `getBitValues()` : retrieve the binary representation of the MinTerm or CombinedTerm (e.g [0, 1, 0, 1])

💡 `compareTerm(Term t)` : return true if one term's binary representation differs from another by exactly 1 variable, false otherwise

💡 `computeBitValues()` : compute the binary representation after combining

2. Table package

- Class: "Group", "IntermediateColumnContainer", a "Table" parent class with 3 child classes "TruthTable", "IntermediateColumn" & "PITable",
- The basic idea is that the logical expression will be stored as a TruthTable. This TruthTable object will then be used to construct an initial IntermediateTable. At this point, all the unchecked terms are PIs (prime implicants). These terms will be used to build the PI Table.
- Important methods

💡 `getAllMinTerms()` : get all the minterms within TruthTable

💡 `NegateTable()` : Replace all the "1" entries in the truth table with "0", and vice versa

💡 `next()` : generate the next Intermediate Column. This is done by iterating through all elements in each two adjacent groups and calling compareTerm() on each pair to check if they are combinable. If they are combinable, the terms are marked as 1 within their corresponding groups.

💡 `IntermediateColumn(TruthTable tt)` : construct IntermediateColumn from a TruthTable. This is done by dividing the minterms obtained from TruthTable into groups corresponding to the number of "1" in bit-value.

- `IntermediateColumnContainer(TruthTable tt)` : this will execute the whole intermediate columns building process. It will call next() until no more IntermediateColumn can be constructed.
- `getUnchecked()` : return all the unchecked terms from every IntermediateColumn
- `PITable(IntermediateColumnContainer icc, TruthTable tt)` : construct PITable from all IntermediateColumns and the User-input TruthTable
- `finalResult()` : produce the final simplified expression.
- `finalResultSOP()` and `finalResultPOS()` : convert to SOP and POS canonical form respectively

3. Screen Package

- Contain 2 input screen and 2 controller class, 1 output screen with 2 “Block” class, used to populate Intermediate Column and PI Table.
- Important methods
  - `CreateTruthTable()` : Populate the TruthTable by creating all of the possible binary equivalents of Minterms in the form of an array list of “0” & “1”. Take user input to complete the truth table, depending on the radio button, each binary equivalent will append “0” or “1” to their end
  - `Three/FourVariableColumnBlock()` : Populate the output’s Intermediate column tables with the result returned by the algorithm
  - `PIBlock()` : Populate the output’s PITable with the result returned by the algorithm

4. Usage of Object-oriented design principles

▼ **Aggregation**

- Object of the CombinedTerm class contains multiple objects of the MinTerm class. Object of Group class contains multiple objects of Term class, which means it can be MinTerm as well as CombinedTerm
- IntermediateColumnContainer is designed to includes all IntermediateColumns associating with each step of Quine-McCluskey algorithm, thus objects of IntermediateColumnContainer class contains multiple objects of IntermediateColumn class

▼ **Inheritance**

- Since TruthTable, IntermediateColumn and PITable all have in common attributes rowNames, columnNames, Values and Size, as well as methods getRowNames, getColumnNames, getValues... We create Table class from which TruthTable, IntermediateColumn and PITable inherit
- Similarly, MinTerm and CombinedTerm have in common bitValues attribute, we create the Term class from which both of these classes inherit

▼ **Association**

- In order to construct the screen for each table, we need all the information about rows, columns and values of those tables. Thus, there's an association between each table with its GUI

▼ **Polymorphism**

- Class Term has an abstract method called getName(), which is further implemented by its child classes (MinTerm and CombinedTerm) in different manners. When we iterate through the list of PI to retrieve their names, we call the getName() method, and depending on the object's class, it will execute the corresponding method.