**DOKUZ EYLUL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME2101 – PROJECT BASED LEARNING 3**

**LOGIC EXPRESSION**

**SIMPLIFICATION TOOL**

**by**

**Özgür Hepsağ 2014510043**

**Barış Bayramoğlu 2014510010**

**December, 2016**

**İZMİR**

**Page**

**CONTENTS**

**INTRODUCTION 1**

**COMPLEMENT REPORT 2**

**CLASS DIAGRAMS 3**

**EXPLANATION OF CLASSES AND IMPLEMENTATION 4**

**SCREENSHOTS 5**

**CONCLUSION AND FUTURE WORK 6**

**REFERENCES 7**

**INTRODUCTION**

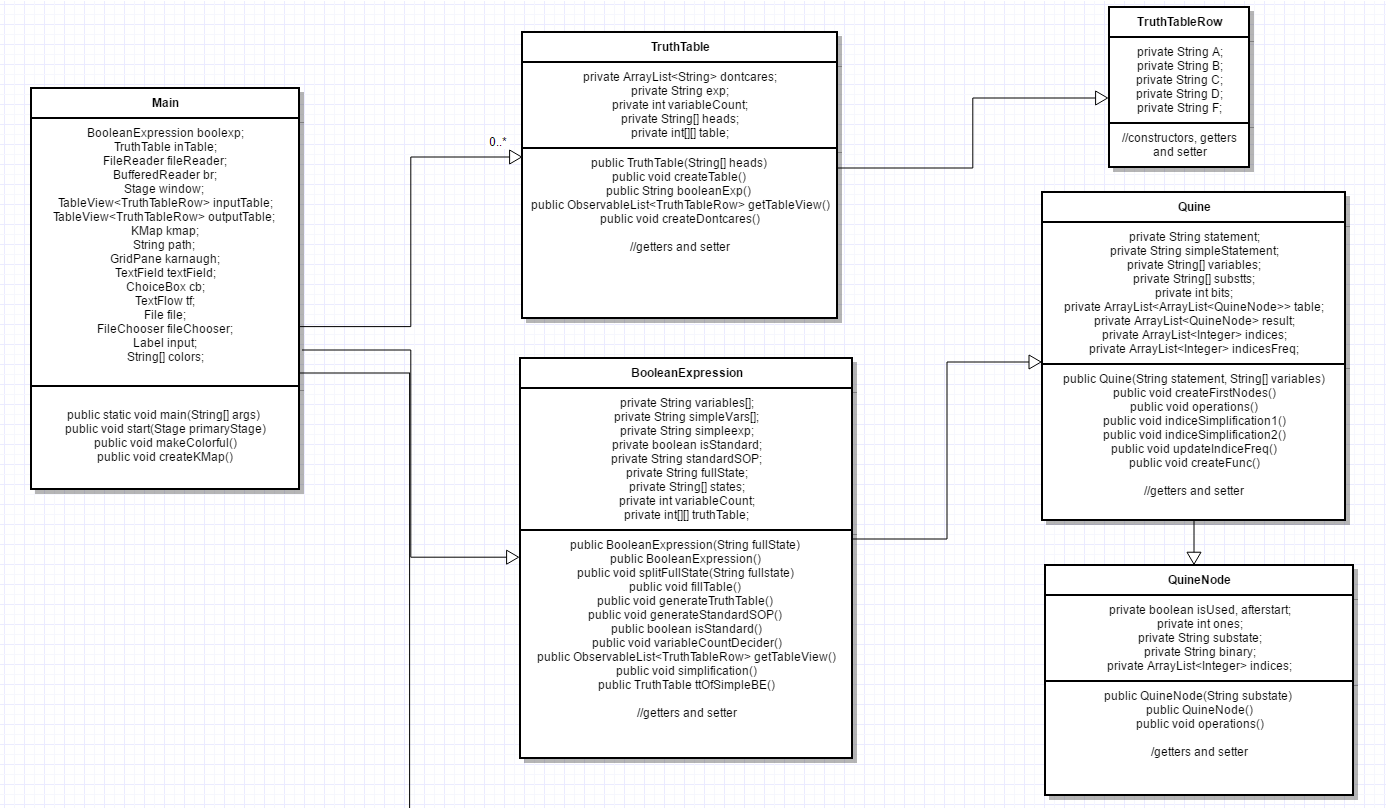
Purpose of this project is to simplify sum-of-product logic expressions. JavaFX is used for GUI and designed well for userfriendliness. User can easily type in expression and edit truth table. Also, user can get read own files (.be or .tt) to our program and his/her files are used as an input. There is two type inputs; one is using Quine McCluskey Algorithm, other is using karnaugh map. Two of them give almost same result but two of them absolutely correct answers. There would not be any wrong result. However, user can get some errors if he make mistake in writing their expression or edit truth table wrongfully.

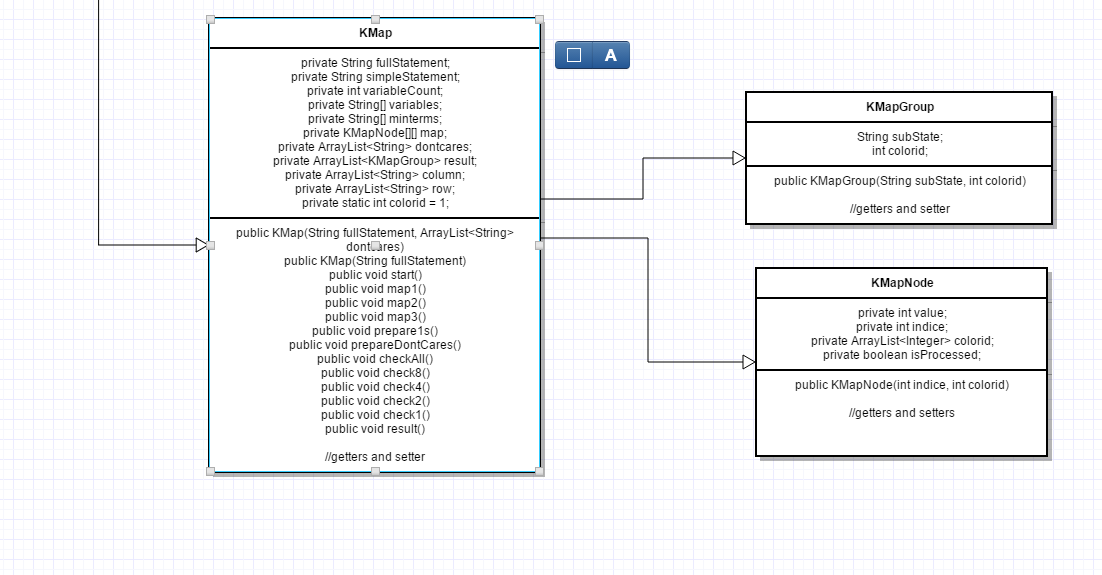
**COMPLETION REPORT**

All milestones are completed errorless and project has not any missing part at end of the deadline. Also some extra features are added to program like simplification with don’t cares and simplification not standard sum-of-product expressions.

Simplification part is done with two ways. But, there could be different solutions. For example, one of the other solutions is to use different tree data structures. On the other hand, string operations could have used for SOP expression in a recursion method. Except that, we use Quine McCluskey algorithm for boolean simplification. Distinct linked lists and stacks structures would have used in karnaugh map solving part. But, we use two dimensional array for that.

**CLASS DIAGRAM**





**EXPLANATION OF CLASSES AND IMPLEMENTATION**

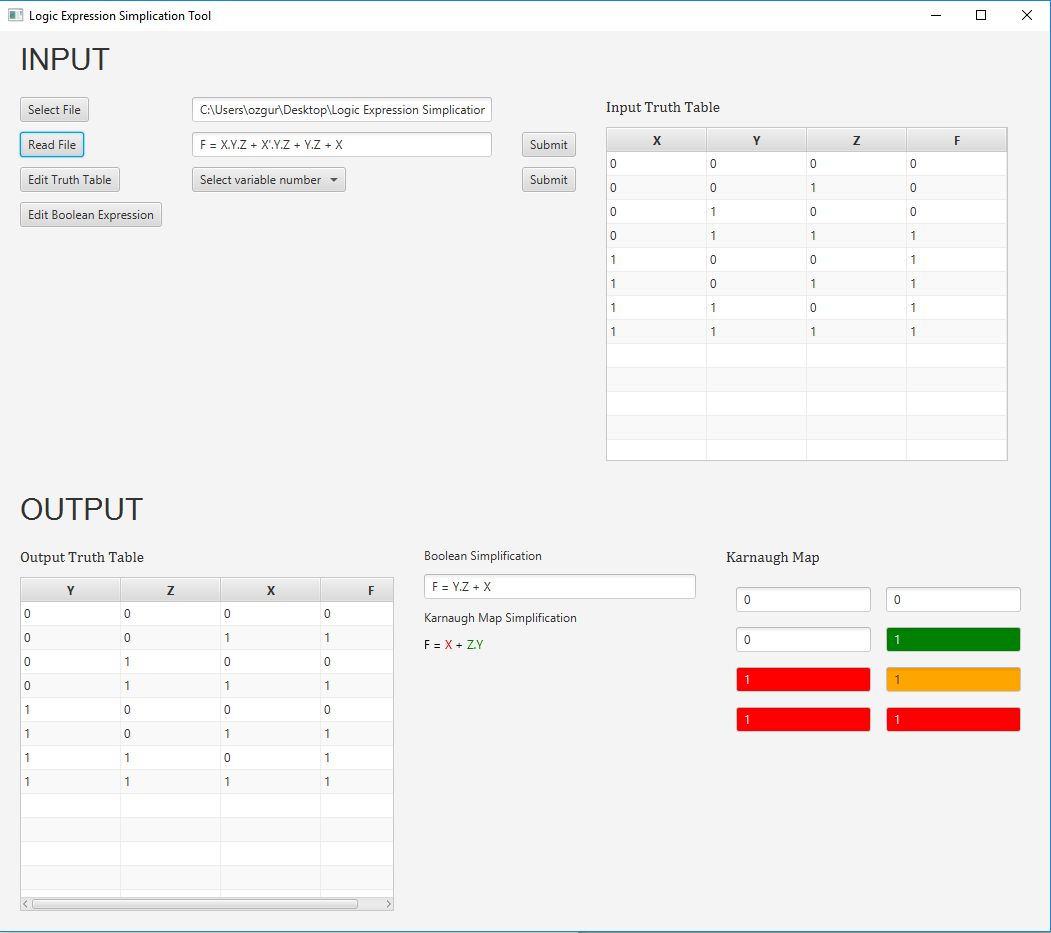
There is 9 class in our project. One of them is main class and tree of them big and very essential classes.

TruthTable class is available for any truth table input or converting boolean expressions to truth table. We have also TruthTableRow class. TruthTableRow is useful to display our table to GUI as tableview using observable list.

BooleanExpression class is written for any SOP input or converting truth tables to boolean expressions. This class is also uses the Quine class ,which was created to implement Quine McCluskey algorithm, for first simplification part and Quine uses QuineNode class for own simplification method.

KMap class is improved for either truth table or boolean expression inputs. Besides, its own purpose is to solve karnaugh maps correctly using KMapGrooup and KMapNode classes.

**SCREENSHOTS**



User can read any .be or .tt file using Select File and Read File button. Moreover, user can edit own truth table before select variable number or type in own expression using Edit Boolean Expression button. When user click that button, expression textfield becomes editable. User click Submit button, after typing is over. Same is necessary for editing truth table. Then, user might easily observe their outputs with colorful display. Outputs are a truth table, two simple expression and one karnaugh map.

**CONCLUSION AND FUTURE WORK**

Logic Expression Simplification Tool Project is developed completely. There is not any missing part.

We have already improved our project. These improvements ate simplification with don’t cares and simplification with not standard sum-of-product expressions.

Additionally, if there would more time and our lecturers would ask to us product-of-sum expression operations, we may have extended our program with doing that.

**REFERENCES**

1. <http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm>
2. <http://www.allaboutcircuits.com/textbook/digital/chpt-8/dont-care-cells-karnaugh-map/>
3. <https://en.wikipedia.org/wiki/Quine%E2%80%93McCluskey_algorithm>
4. <https://arxiv.org/ftp/arxiv/papers/1404/1404.3349.pdf>