LAPORAN TUGAS BESAR

TEORI BAHASA & AUTOMATA

“Simple Parser for Proposition Logic Formula”



Anggota Kelompok :

1. Dion Pratama Putra 1301180181
2. Fajar Widhi Ardiyanto 1301180267
3. Ahmed Lasca Rasheeda 1301180333
4. Athalla Rizky Arsyan 1301184451

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK INFORMATIKA**

**UNIVERSITAS TELKOM**

**BANDUNG**

**2019**

1. **Anggota Kelompok**

|  |  |  |
| --- | --- | --- |
| No | NIM | Nama |
| 1 | 1301180181 | Dion Pratama Putra |
| 2 | 1301180267 | Fajar Widhi Ardiyanto |
| 3 | 1301180333 | Ahmed Lasca Rasheeda |
| 4 | 1301184451 | Athalla Rizky Arsyan |

1. **Spesifikasi Program**
2. Parser Sederhana untuk Formula Logika Proposisi

Program tahap ini merupakan program Lexycal Analyzer dengan menerapkan Finite Automata untuk mengenali Lexic yang dituliskan dalam ekspresi, menggunakan Bahasa program Java. Program ini akan mendefinisikan, menganalisa dan mengenali logika proposisi untuk kemudian menghasilkan Sequence Token Lexic berdasarkan inputan user. Pembuatan program ini terdiri atas dua tahapan yaitu :

1. Lexic Analyzer

Pada tahap satu, program yang dibangun berupa Lexical *Analyzer* dengan menerapkan Finite Automata untuk mengenali setiap Lexic yang dituliskan dalam Formula. Program ini akan mendefinisikan, menganalisa dan mengenali setiap String inputan user. Hasil dari program tersebut berupa *Sequence* Token Lexic berdasarkan String Formula yang dibaca. Namun pembuatan program pada tahap satu hanya dapat mendeteksi besaran dan token lexic tetapi tidak menentukan logika proposisi tersebut valid apa tidak.

1. Token Lexic Parser

Pada tahap dua, program yang dibuat lebih dikembangkan lagi dengan menambahkan fungsi parser untuk melakukan validasi terhadap String Formula Logika Proposisi yang dimasukkan. Dalam membangun fungsi parser tersebut, akan diimplementsikan *Context Free Grammar* atau *Push Down* Automata dalam memvalidasi sebuah String Formula Logika Prposisi.

Program ini dikembangan menggunakan *development tools* Netbeans dan dapat dijalankan pada *Operating System* yang memiliki Java *Runtime Environment*.

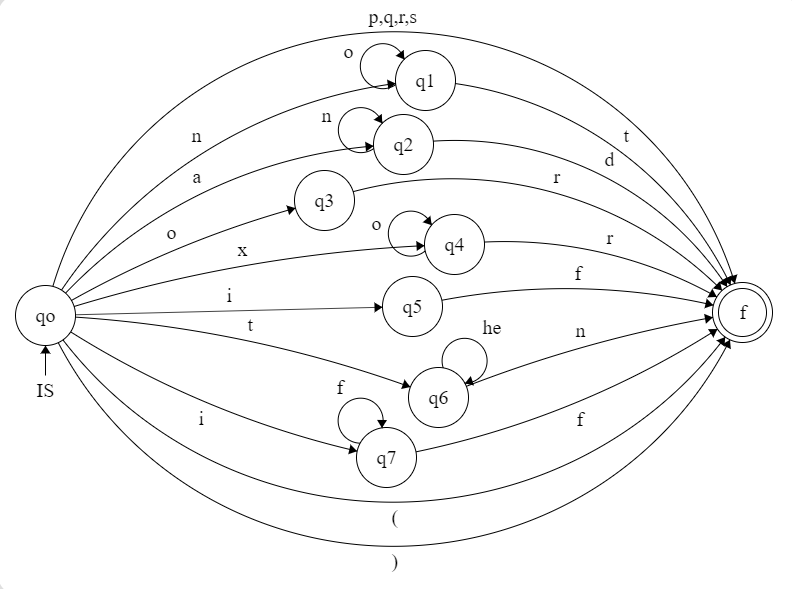
1. Batasan Masalah

Dalam pembangunan Lexical Analyzer dan Parser Program, terdapat beberapa batasan - batasan sebagai berikut :

1. Formula harus case sensitive dan hanya dapat menggunakan lowercase.
2. Antar satu lexic sebelum atau sesudahnya harus dipisahkan dengan spasi.
3. Program hanya dapat mengenali jenis lexic sebagai berikut :

|  |  |  |  |
| --- | --- | --- | --- |
| **String Lexic** | **Jenis** | **Token** | **Keterangan** |
| Proposisi:  p, q, r, s | Operand | 1 | Hanya 1 simbol (di antara p, q, r, s) yang dikenal sebagai 1 proposisi. |
| not | Operator | 2 | Contoh penulisan yang diterima:   * not proposisi * not(proposisi) |
| and | Operator | 3 | Contoh penulisan yang diterima:   * proposisi and proposisi * (proposisi)and(proposisi) * (formula)and(formula) |
| or | Operator | 4 | Contoh penulisan yang diterima:   * proposisi or proposisi * (proposisi)or(proposisi) * (formula)or(formula) |
| xor | Operator | 5 | Contoh penulisan yang diterima:   * proposisi xor proposisi * (proposisi)xor(proposisi) * (formula)xor(formula) |
| if | Operator | 6 | Aturan penulisan yang diterima:   * if proposisi then proposisi * if(proposisi)then(proposisi) * if(formula)then(formula) |
| then | Operator | 7 |
| iff | Operator | 8 | Aturan penulisan yang diterima:   * proposisi iff proposisi * (proposisi)iff(proposisi) * (formula)iff(formula) |
| ( | *Grouping* | 9 | Jumlah kurung buka dan tutup harus sama dan dituliskan di tempat-tempat yang tepat. |
| ) | *Grouping* | 10 |

1. Spesifikasi Kebutuhan Program
   * 1. Program ini dikembangkan menggunakan *Development Tools* Netbeans.
     2. Program ini dapat *Running* pada *Operating System* yang memiliki Java *Runtime Environment.*
2. **Design Program Tahap 1**
3. Rancangan Finite Automata terhadap String Lexic

****

1. Definisi Normal

M= ( Q, ∑, δ, q0)

Q = {q0,q1,q2,q3,q4,q5,q6,q7,f}

∑ = {pqrs,not,and,xor,or,if,then,iff,(,)}

q0 = {q0}

δ = δ digambarkan sebagai table transisi

1. Tabel Transisi

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| δ | P,q,r,s | N,o,t | A,n,d | O,r | X,o,r | I,f | T,h,e,n | I,f,f | ( | ) |
| q0 | f | {f} | {f} | {f} | {f} | {f} | {f} | {f} | f | f |
| q1 | {} | {f} | {} | {} | {} | {} | {} | {} | {} | {} |
| q2 | {} | {} | {f} | {} | {} | {} | {} | {} | {} | {} |
| q3 | {} | {} | {} | {f} | {} | {} | {} | {} | {} | {} |
| q4 | {} | {} | {} | {} | {f} | {} | {} | {} | {} | {} |
| q5 | {} | {} | {} | {} | {} | {f} | {} | {} | {} | {} |
| q6 | {} | {} | {} | {} | {} | {} | {f} | {} | {} | {} |
| q7 | {} | {} | {} | {} | {} | {} | {} | {f} | {} | {} |
| F | {} | {} | {} | {} | {} | {} | {} | {} | {} | {} |

1. **Design Program Tahap 2**
2. *Context Free Grammar*

S -> H | <NULL>

A -> p | q | r | s

B -> and | or | xor | iff

C -> not

D -> if

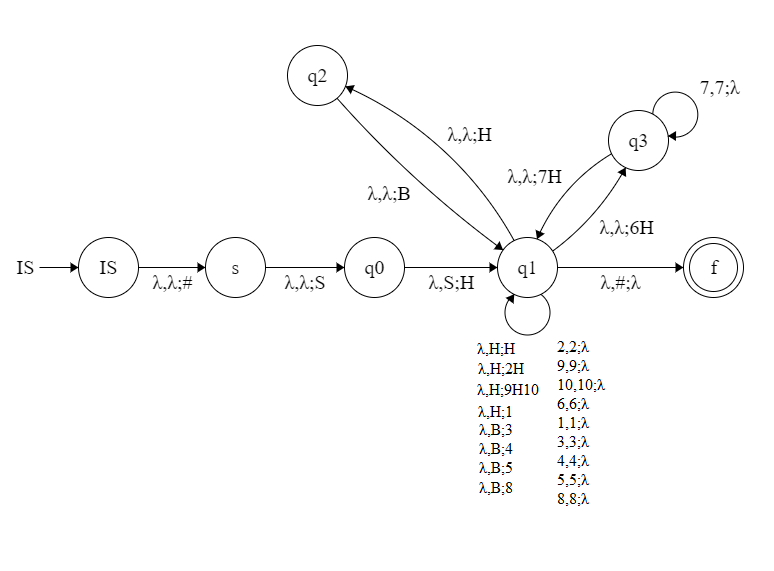
E -> then

F -> (

G -> )

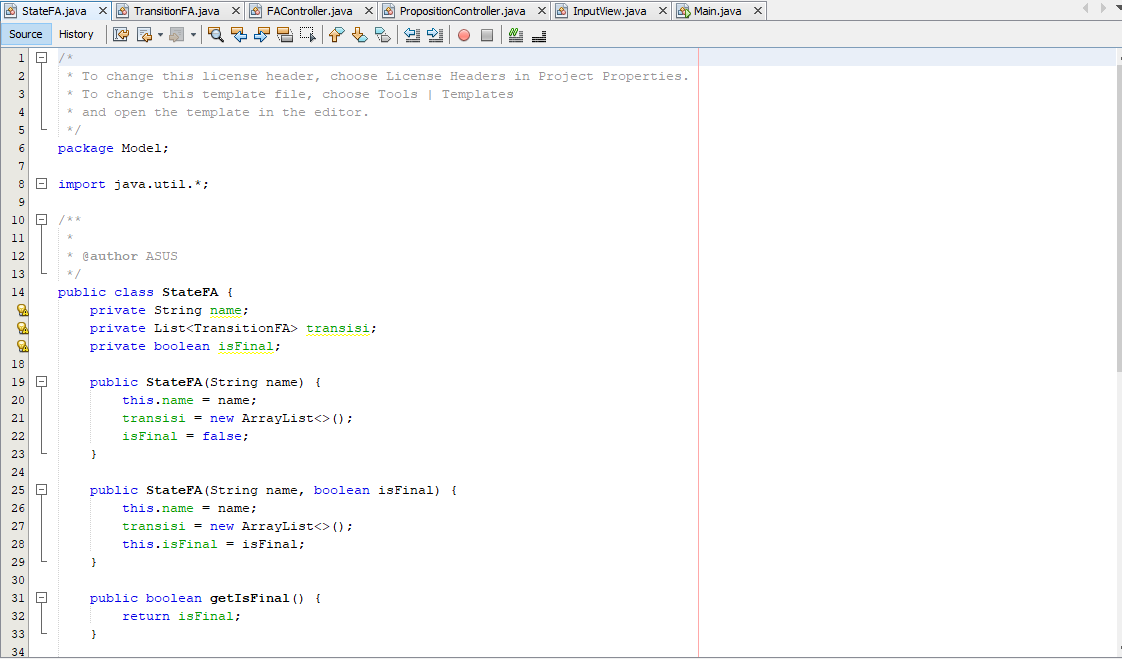
H -> A | HBH | CH | FHG | DHEH

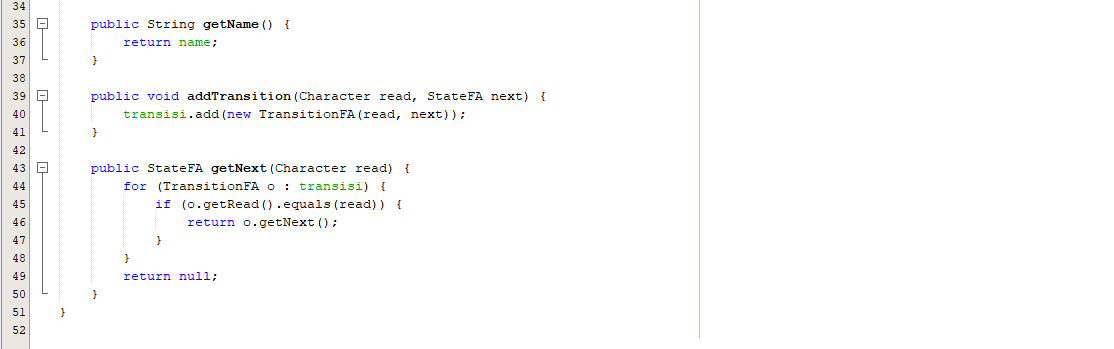
1. *Push Down* Automata

****

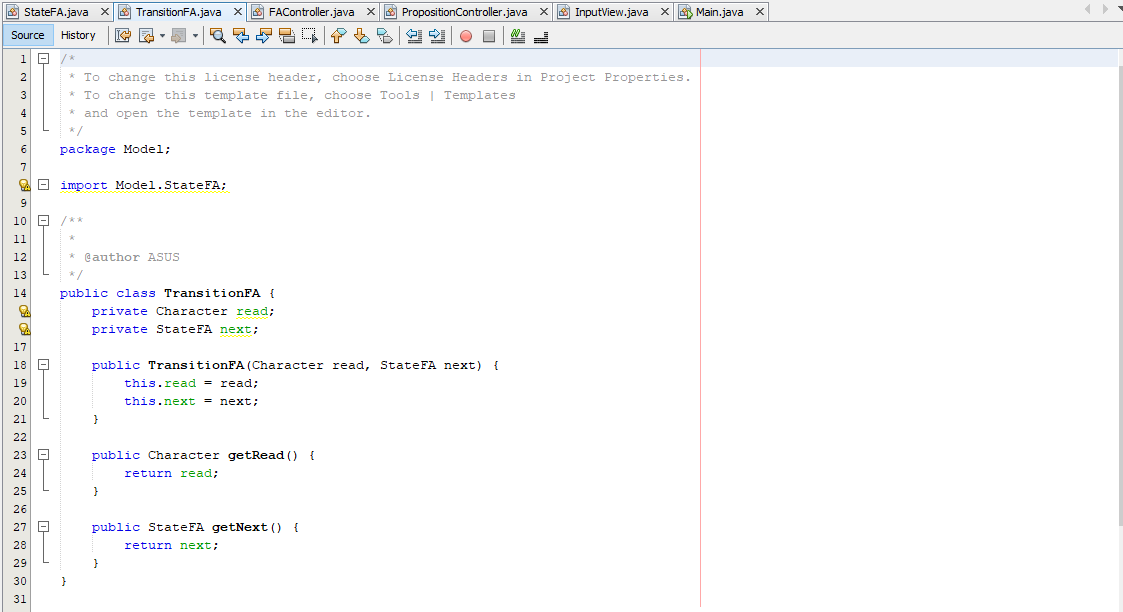
1. Kode Program

Kelas “StateFA” :

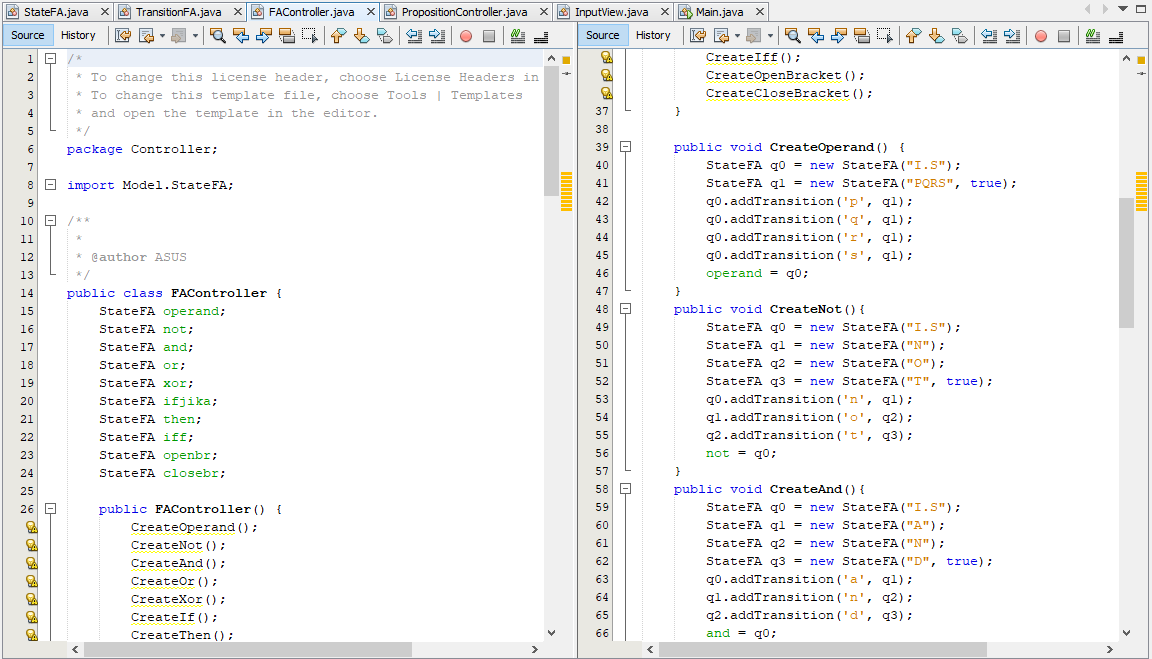


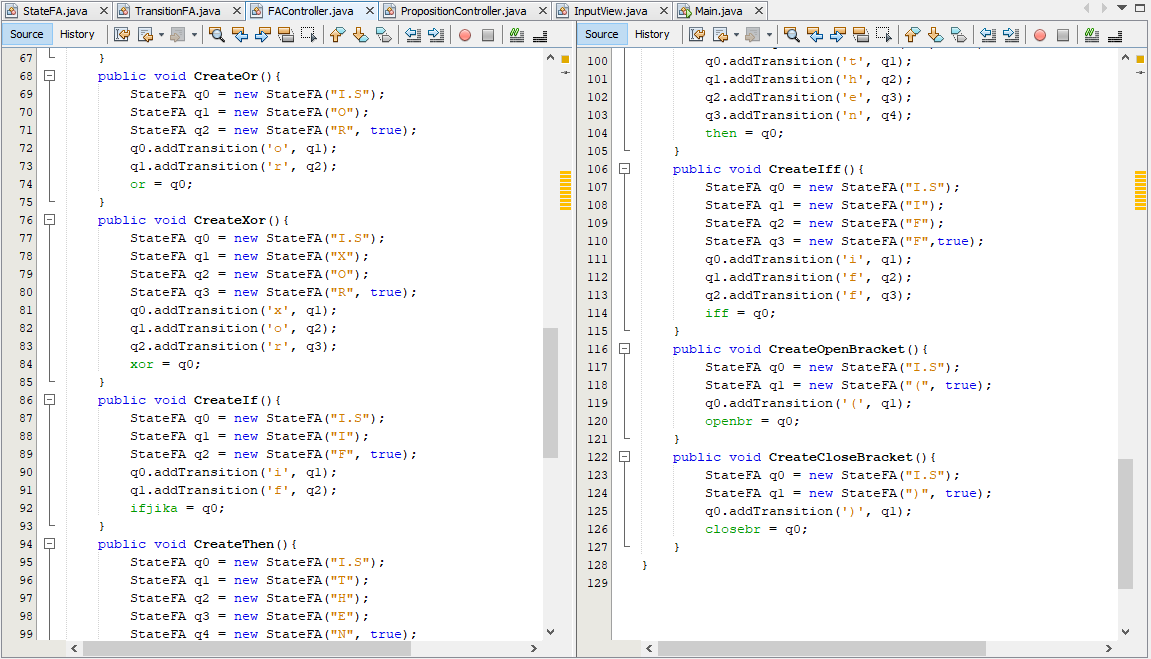


Kelas “TransitionFA”

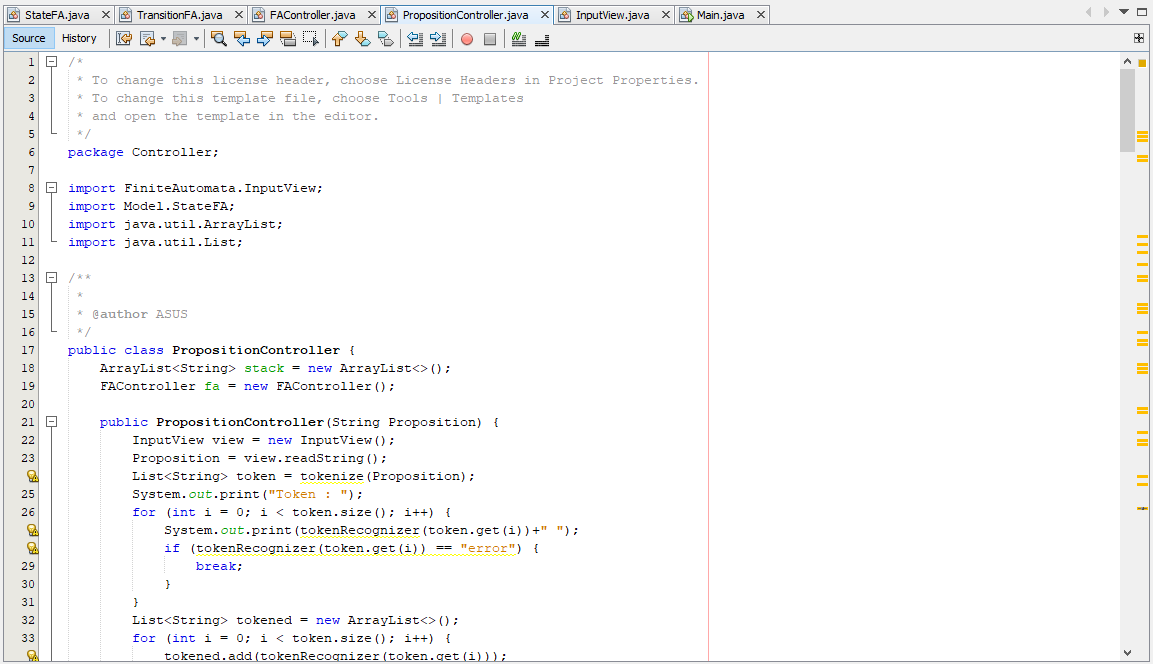


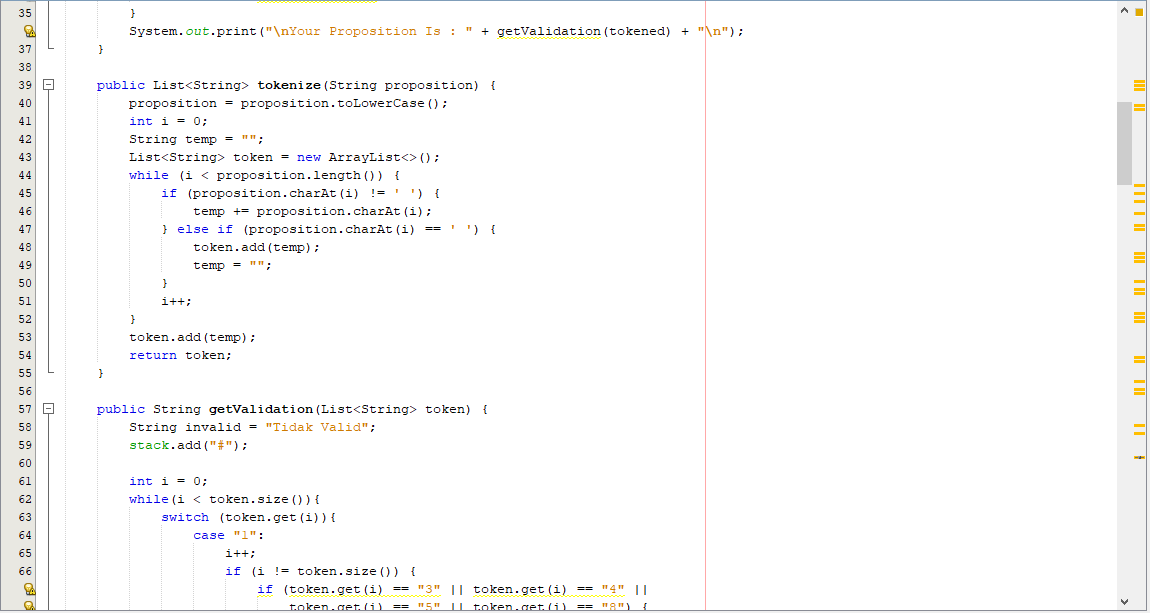
Kelas “FAController”

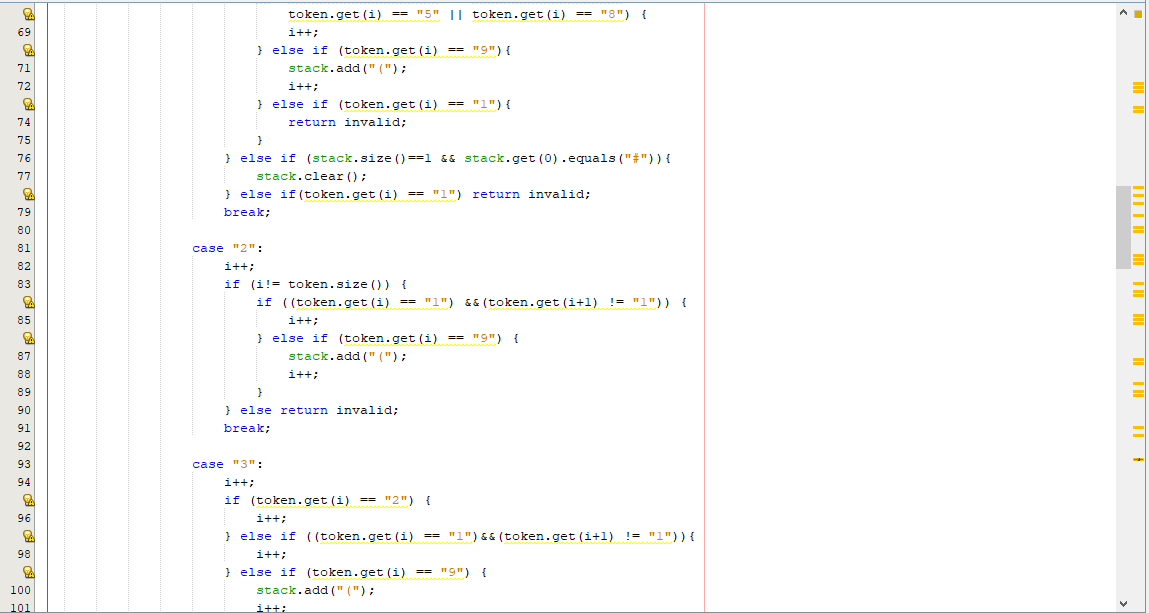


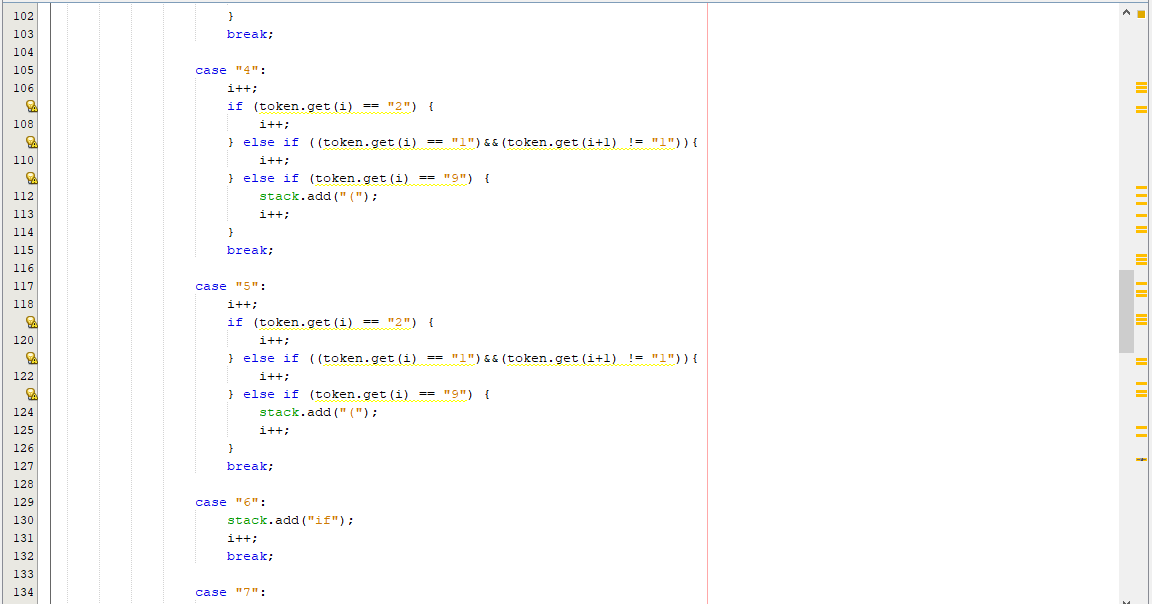


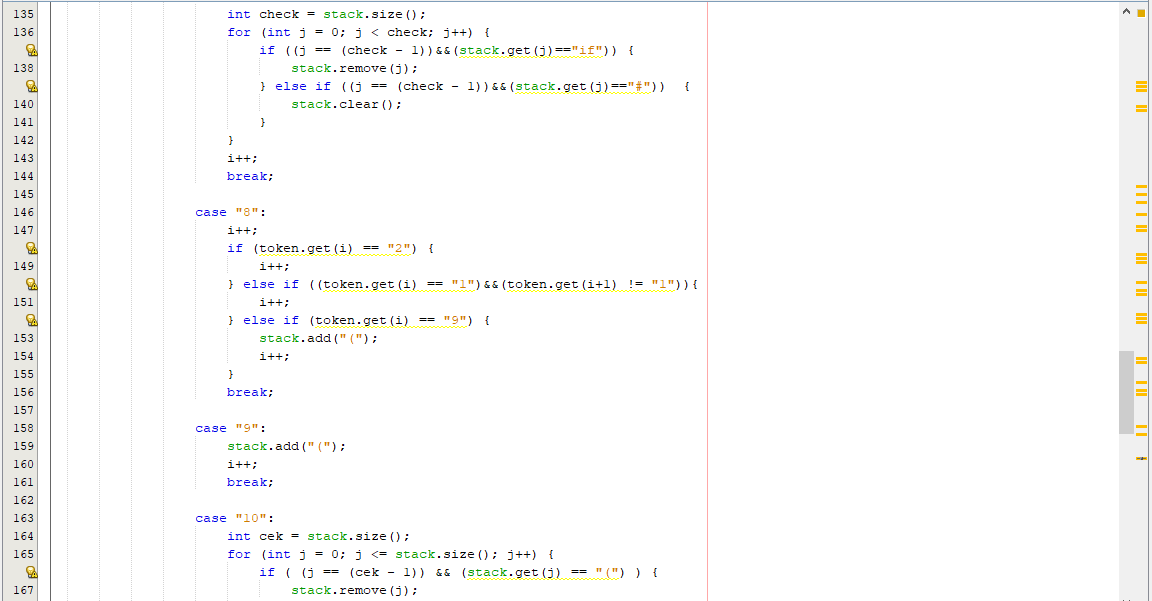
Kelas “PropositionController”

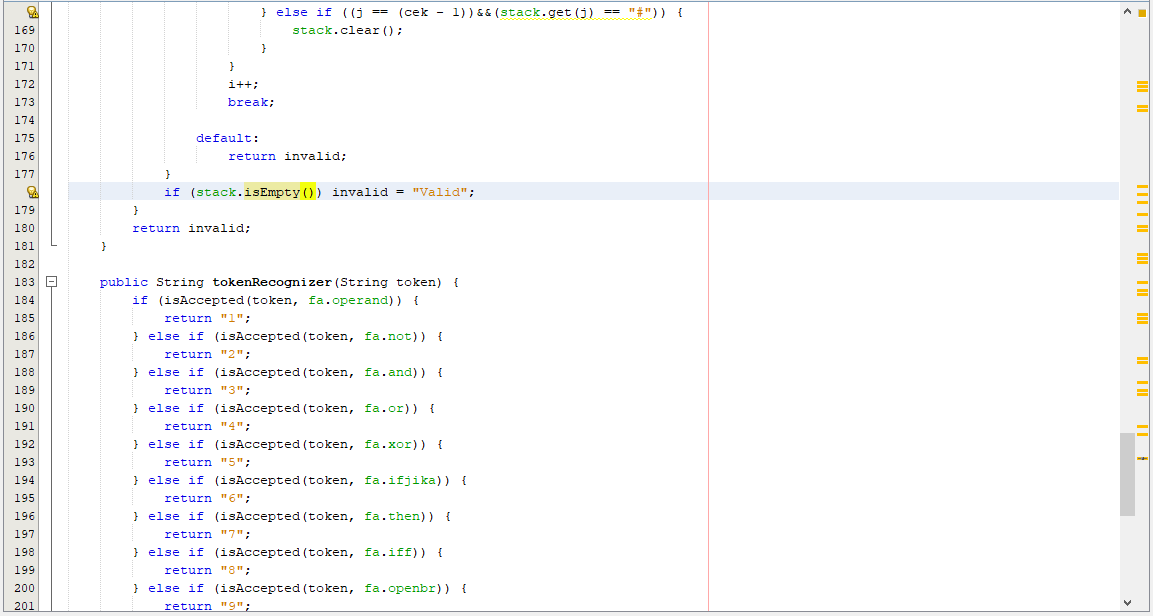


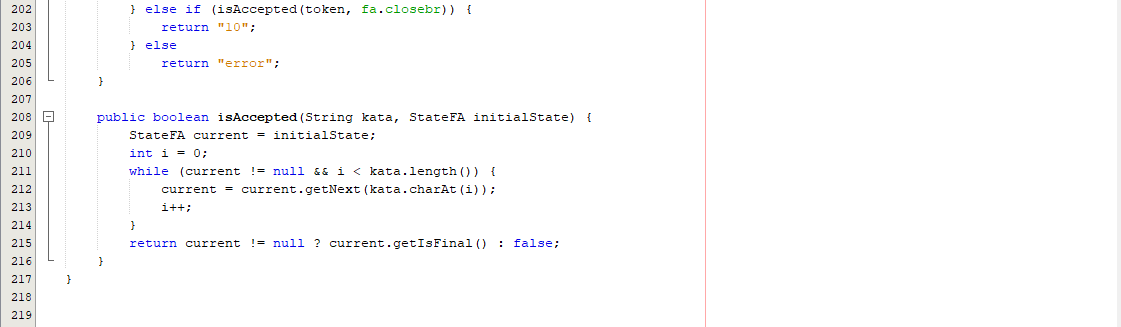




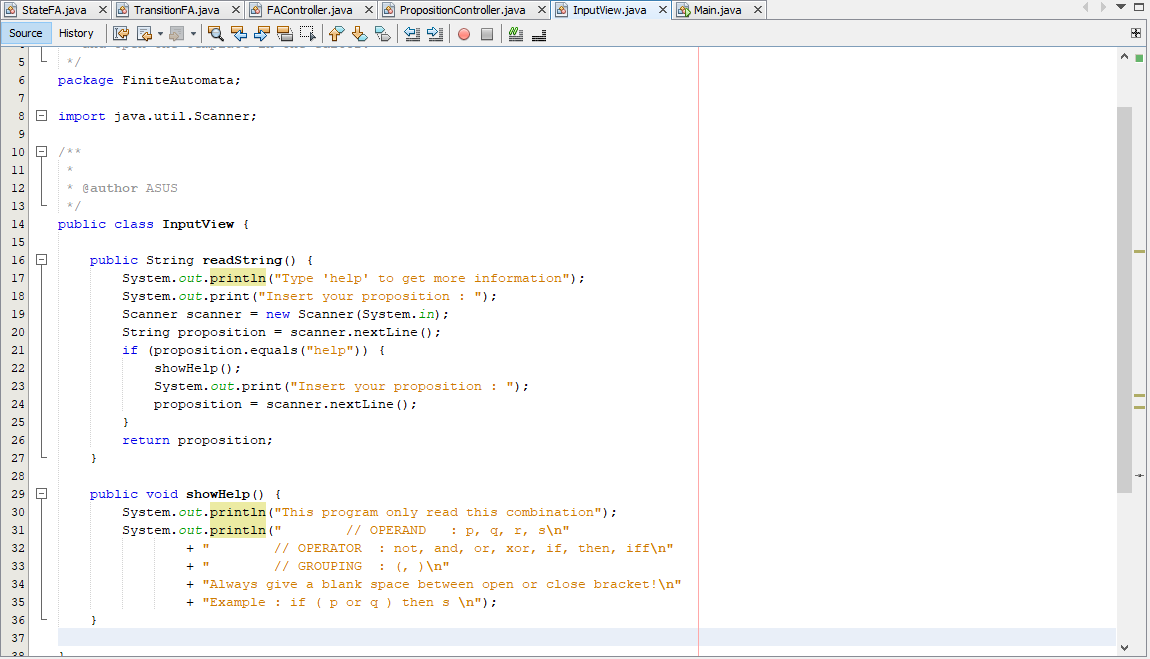




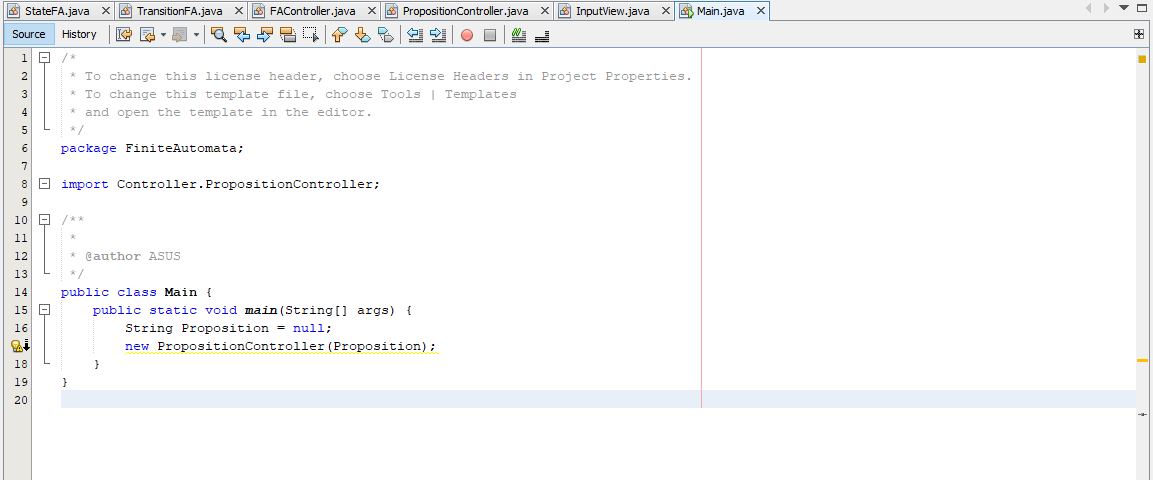




Kelas ”InputView”



Kelas “Main”



1. **Cara Kerja Program**

Ketika program dijalankan, terdapat sebuah field inputan pada tampilan output program yang dapat diisikan dengan Logika proposisi. Inputan tersebut oleh program dibaca sebagai string. Setelah field diisikan dengan inputan string dan menekan , maka program akan membaca kata per kata dan memasukkannya ke dalam Array.

Langkah-langkah konversi yang dilakukan oleh program Parser Logika Proposisi ini adalah : pertama program akan membaca huruf per huruf dari kata. Langkah kedua, program akan melakukan analisa terhadap susunan huruf tersebut berdasarkan aturan lexic yang ada dengan mengkonversi ke token lexic berdasarkan tipe lexicnya (lexic, tipe dan token lexicnya) dengan tetap memperhatikan urutan karakter yang muncul berikutnya. Setelah semua karakter dianalisa dan ditentukan besaran/tipe dan token lexicnya, maka semua informasi tersebut akan ditampilkan pada output.

1. **Pengujian Program**

