# CS 3240 - Project I

Names

November 3, 2011

# Chapter 1

# The Project

## 1.1 Introduction

A scanner generator was designed and implemented in JAVA. An object-oriented approach was followed, however use of libraries and high level tools was avoided. The application is capable of parsing a lexical specification in an established format and, in a minimum number of steps, output a functional table-driven deterministic finite automaton- (DFA-)driven scanner. This scanner's functionality is focused upon the task of analyzing plaintext input, which may or may not follow a recognized syntax. Tokens are identified on the fly, validated, and stored within a predefined XML schema. While no semantic actions or compilation are actually being performed at this stage, refinement of methods demonstrates clear potential for this project.

### 1.1.1 The Goal

Despite the clear inefficiency of writing lexical analysis software in a high-level language, especially in the just-in-time interpreter realm of Java, it is important to note the elegance in approach which can be achieved. While the pursuit of perfection by means of preproccessing and optimization are important to language, there remains value in alternative routes. A Java-based scanner may suffer in performance and flexibility, but the potential for rapid prototyping, availability of a large library and countless APIs can offer significant benefit.

## 1.2 Architecture

The application is powered by a main driver class which is responsible for making procedural and functional calls to the other modules. It runs as a console application and provides realtime feedback on parsing and lexical analysis tasks. The driver maintains interaction with the user and handles file i/o.

### 1.2.1 Automata

All scanner-related logic within the application relies upon finite automata of various classifications and structures. The fundamental building block is the State object. This contains attributes describing an individual state of an automaton. In the interest of minimizing memory overhead, transitions between these states exist in their respective parent automata. This also maintains an organized set of object references and lends a greater degree of control over traversal.

At the top of the inheritance hierchy is the FiniteAutomata interface, a framework around which more specific objects are designed. The interface itself guarantees sets of characters (representing the chosen alphabet), and states, with separate designations for initial and final states. From this extends the remaining two interfaces, named DFA, for Deterministic Finite Automata, and NFA, for the Nondeterministic variety. These build upon the existing structure by defining transitions. Specifically, the NFA interface makes allowances for epsilon transitions and multiple starting states. The final level in the hierarchy are the actual automata classes: the MapBasedDFA, TableDrivenDFA, MapBasedNFA, and a customized NFA structure, MinimalNFA, to be used by the parser integrated within the scanner generator.

## 1.3 Use Case

Upon running the application, a file containing a plaintext lexical specification is loader by the driver class. Such a file is required to take the following form:

```
%% Definitions for character classes will be contained in this section.
$DIGIT     [0-9]
$NON-ZERO [^0]  IN  $DIGIT
$SMALLCASE    [a-z]
$LETTER       [A-Za-z]
%% Token definitions will be contained in this section using regexes
$IDENTIFIER   $LETTER ($LETTER| $DIGIT)*
```

A hard-coded, recursive descent parser traverses the input file from top to bottom, using regular expressions to identify and define character classes. Due to this approach, the lexical specification is required to have been written in such a way as to exclude left-recursion in definitions, as this could cause looping and will not be interpreted correctly. It then continues on to read each of the lines containing an Identifier, instantiating and populating a dedicated micro-NFA. Upon successful completion of parsing, these miniature NFAs (as demonstrated by small numbers of states) are combined, forming a master NFA containing the entire language's grammatical and lexical specifications. While this crude NFA is technically sufficiently well-developed to function as a scanner on its own, it is then converted into a map-based DFA, greatly simplifying execution. This product is then further refined into a table-driven DFA, at which point it is minimized, removing any duplicate paths or unnecessarily complex routes. At this point in exectution, the automaton has been sufficiently optimized and can be deemed a customized scanner, built to recognize the chosen regular language.

# Chapter 2

# Documentation

# Table of Contents

# Model Documentation

## Model Detail

This document provides a complete overview of all element details. For simpler and more focused reports, simply copy this initial template and turn off the sections not required.

## Model

| | |
|---|---|
| *Type:* | **Package** |
| *Status:* | Proposed. Version . Phase 1.0. |
| *Package:* | |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011* |
| *GUID:* | {A2CAB543-AE41-41b2-AA74-9A3C4473AC21} |

## automata

| | |
|---|---|
| *Type:* | **Package** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | Model |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011* |
| *GUID:* | {6F93D628-E594-4eb0-B698-1B4C7C821F09} |

**automata** - *(Logical diagram)*

| | |
|---|---|
| *Created By:* | Paul *on* 11/3/2011 |
| *Last Modified:* | 11/3/2011 |
| *Version:* | 1.0. *Locked:* False |
| *GUID:* | {9662D210-B41A-41c1-BD80-877881D0FA1A} |

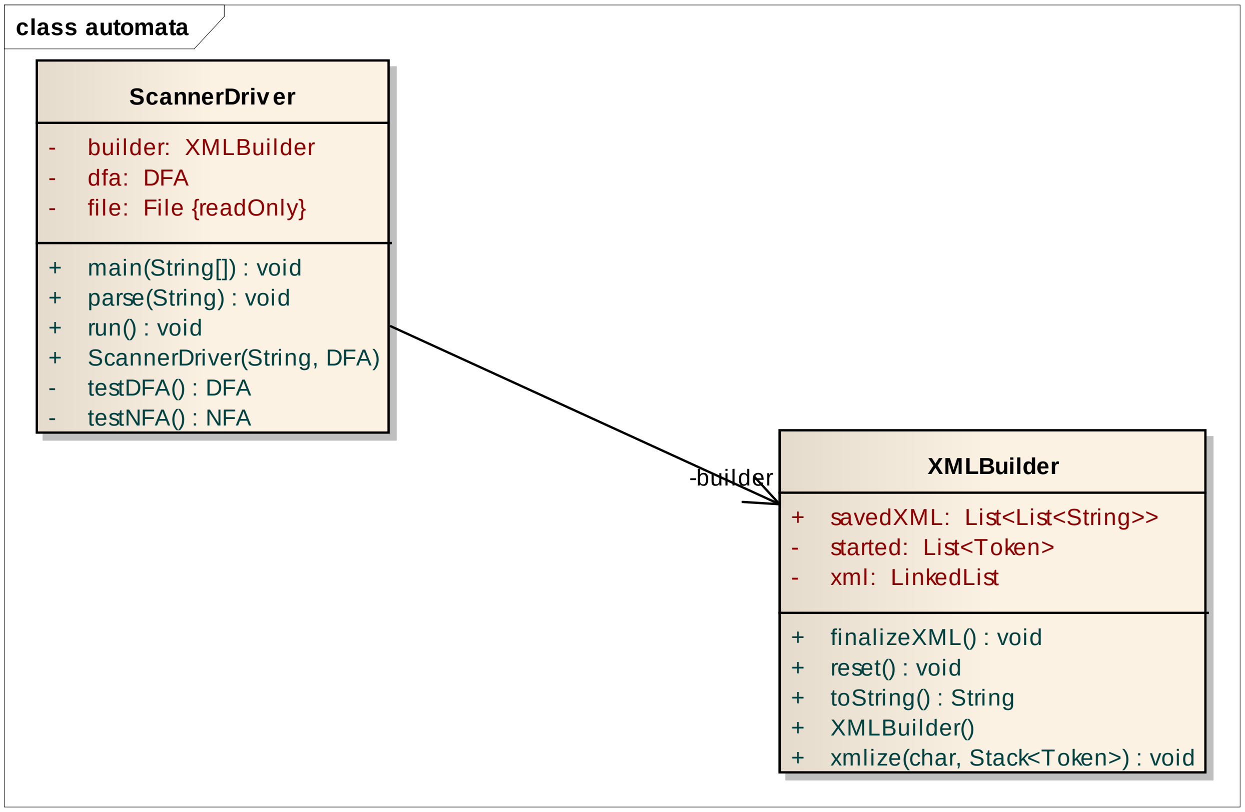

Figure: 1

## ScannerDriver

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {70B6573E-416E-4bbe-9BAF-8306F46B146E} |

The Main driver class. Creates a DFA based on given classes and regular expressions, then verifies that a given file contains valid tokens. It can also generate an XML-structure of the tokens, showing how the characters form into compounding tokens.

### Custom Properties

- isActive = False

### Connections

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association**<br>Source -> Destination | Public<br>ScannerDriver | Private dfa<br>DFA | |
| **Association** | Public | Private builder | |

| **Connector** | **Source** | **Target** | **Notes** |
|---|---|---|---|
| Source -> Destination | ScannerDriver | XMLBuilder | |

***Attributes***

| **Attribute** | **Notes** | **Constraints and tags** |
|---|---|---|
| **builder** XMLBuilder<br>Private | | *Default:* |
| **dfa** DFA<br>Private | | *Default:* |
| **file** File<br>Private<br>Const | | *Default:* |

***Operations***

| **Method** | **Notes** | **Parameters** |
|---|---|---|
| Static **main()** void<br>Public | Creates the DFA and ScannerDriver using the files containing the regexes and the tokens. If no arguments are supplied, the sample cases are used and are expected to be in the same directory. | **String[]** [in] args<br>1st argument is used as the name of the file containing all of the tokens |
| **parse()** void<br>Public | | **String** [in] word |
| **run()** void<br>Public | | |
| **ScannerDriver()**<br>Public | Creates and initializes the ScannerDriver | **String** [in] fileName<br>The name of the file containing all of the tokens<br>**DFA** [in] dfa<br>The DFA to use while scanning |
| Static **testDFA()** DFA<br>Private | A temporary DFA used for testing. | |

| Method | Notes | Parameters |
|---|---|---|
| Static **testNFA()** NFA Private | A temporary NFA used for testing. | |

## XMLBuilder

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {7C0CB7EA-2CB5-41f6-9FD1-566802DE43D0} |

Generates and stores XML-like structures to represent the tokens that are read and parsed.

### *Custom Properties*

- isActive = False

### *Connections*

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association** Source -> Destination | Public ScannerDriver | Private builder XMLBuilder | |

### *Attributes*

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **savedXML** List<List<String>> Public | | *Default:* |
| **started** List<Token> Private | | *Default:* |

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **xml** LinkedList<br>Private | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **finalizeXML()** void<br>Public | Finalizes the current xml tree and saves it internally that can be printed or saved later | |
| **reset()** void<br>Public | Resets all of the temporary structures. Still keeps the finalized xml structures used for printing. | |
| **toString()** String<br>Public | | |
| **XMLBuilder()**<br>Public | | |
| **xmlize()** void<br>Public | Adds the given tokens and character to the current xml tree | **char** [in] character<br>The character within the tokens<br>**Stack<Token>** [in] tokens<br>The tokens associated with the character. |

## automata

| | |
|---|---|
| *Type:* | **Package** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011* |
| *GUID:* | {99C05C17-A01D-4002-B163-2C73A3C6C287} |

**automata** - *(Logical diagram)*

| | |
|---|---|
| *Created By:* | Paul *on* 11/3/2011 |
| *Last Modified:* | 11/3/2011 |
| *Version:* | 1.0. *Locked:* False |
| *GUID:* | {24718268-7E22-400f-B962-4665FCEB15DB} |

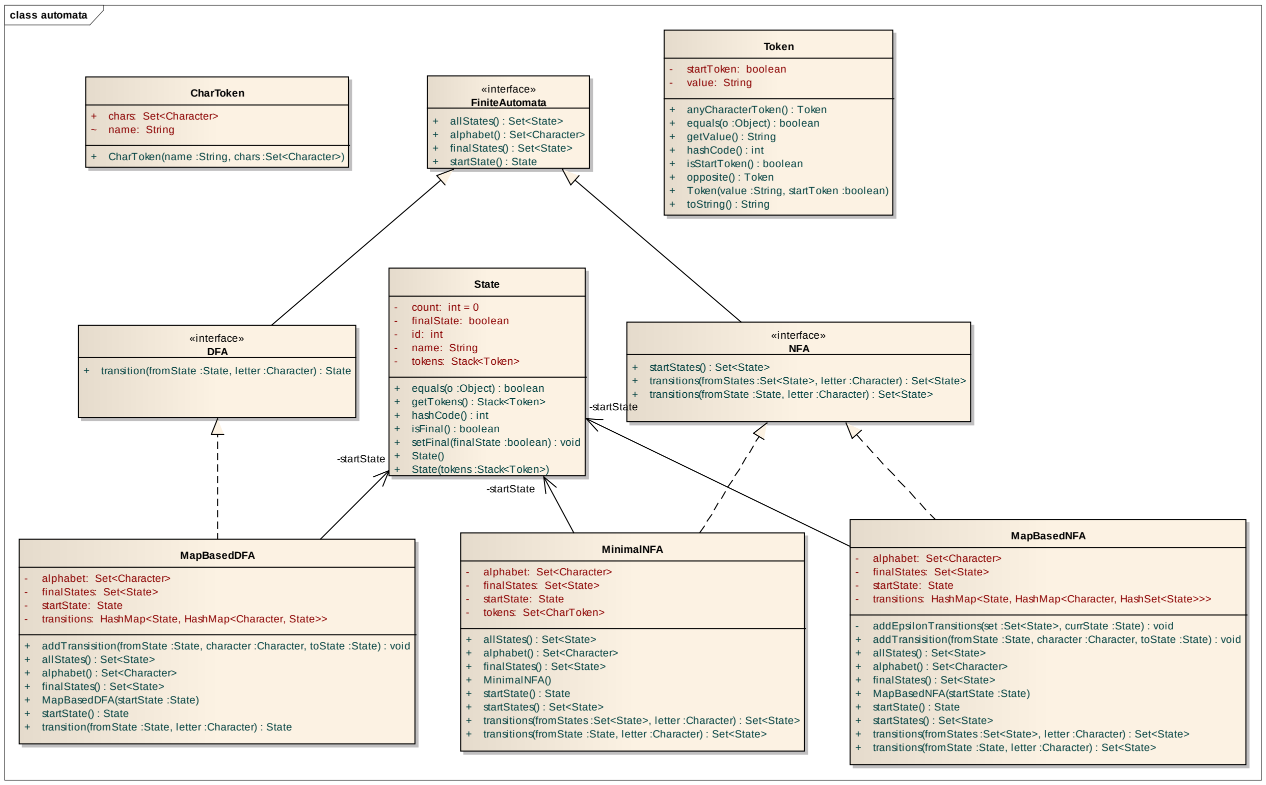
class automata
CharToken
+ chars: Set<Character>
~ name: String
+ CharToken(name :String, chars :Set<Character>)
«interface»
FiniteAutomata
+ allStates() : Set<State>
+ alphabet() : Set<Character>
+ finalStates() : Set<State>
+ startState() : State
Token
- startToken: boolean
- value: String
+ anyCharacterToken() : Token
+ equals(o :Object) : boolean
+ getValue() : String
+ hashCode() : int
+ isStartToken() : boolean
+ opposite() : Token
+ Token(value :String, startToken :boolean)
+ toString() : String
State
- count: int = 0
- finalState: boolean
- id: int
- name: String
- tokens: Stack<Token>
+ equals(o :Object) : boolean
+ getTokens() : Stack<Token>
+ hashCode() : int
+ isFinal() : boolean
+ setFinal(finalState :boolean) : void
+ State()
+ State(tokens :Stack<Token>)
«interface»
DFA
+ transition(fromState :State, letter :Character) : State
«interface»
NFA
+ startStates() : Set<State>
+ transitions(fromStates :Set<State>, letter :Character) : Set<State>
+ transitions(fromState :State, letter :Character) : Set<State>
-startState
-startState
-startState
MapBasedDFA
- alphabet: Set<Character>
- finalStates: Set<State>
- startState: State
- transitions: HashMap<State, HashMap<Character, State>>
+ addTransisition(fromState :State, character :Character, toState :State) : void
+ allStates() : Set<State>
+ alphabet() : Set<Character>
+ finalStates() : Set<State>
+ MapBasedDFA(startState :State)
+ startState() : State
+ transition(fromState :State, letter :Character) : State
MinimalNFA
- alphabet: Set<Character>
- finalStates: Set<State>
- startState: State
- tokens: Set<CharToken>
+ allStates() : Set<State>
+ alphabet() : Set<Character>
+ finalStates() : Set<State>
+ MinimalNFA()
+ startState() : State
+ startStates() : Set<State>
+ transitions(fromStates :Set<State>, letter :Character) : Set<State>
+ transitions(fromState :State, letter :Character) : Set<State>
MapBasedNFA
- alphabet: Set<Character>
- finalStates: Set<State>
- startState: State
- transitions: HashMap<State, HashMap<Character, HashSet<State>>>
- addEpsilonTransitions(set :Set<State>, currState :State) : void
+ addTransisition(fromState :State, character :Character, toState :State) : void
+ allStates() : Set<State>
+ alphabet() : Set<Character>
+ finalStates() : Set<State>
+ MapBasedNFA(startState :State)
+ startState() : State
+ startStates() : Set<State>
+ transitions(fromStates :Set<State>, letter :Character) : Set<State>
+ transitions(fromState :State, letter :Character) : Set<State>

Figure: 2

## *CharToken*

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {04C3816C-8356-4ff5-B382-CDF5691420B8} |

### ***Custom Properties***

- isActive = False

### ***Attributes***

| Attribute | Notes | Constraints and tags |
|---|---|---|

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **chars** Set<Character><br>Public | | *Default:* |
| **name** String<br>Package | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **CharToken()**<br>Public | | **String** [in] name<br><br>**Set<Character>** [in] chars |

## *MapBasedDFA*

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {758A5AE4-2DC7-455c-8967-8BE425D61693} |

A HashMap based DFA. Used mainly for temporary testing before we have actual table-based DFA.

***Custom Properties***

- isActive = False

***Connections***

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association**<br>Source -> Destination | Public<br>NFAConverter | Private dfa<br>MapBasedDFA | |
| **Realization**<br>Source -> Destination | Public<br>MapBasedDFA | Public<br>DFA | |
| **Association** | Public | Private startState | |

| Connector | Source | Target | Notes |
|---|---|---|---|
| Source -> Destination | MapBasedDFA | State | |

### *Attributes*

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **alphabet** Set<Character> Private | | *Default:* |
| **finalStates** Set<State> Private | | *Default:* |
| **startState** State Private | | *Default:* |
| **transitions** HashMap<State, HashMap<Character, State>> Private | | *Default:* |

### *Operations*

| Method | Notes | Parameters |
|---|---|---|
| **addTransisition()** void Public | Adds a transition to the DFA. If the character has never been seen before, the internal DFA alphabet will be expanded to include the character. If there is already a transition on the given character from the given from state, this new transition will overwrite the old one. The from state does not have to already exist in the DFA, however, the transition will be useless if it does not. | **State** [in] fromState<br>The state the transition goes from<br>**Character** [in] character<br><br>**State** [in] toState<br>The state the transition goes to |

| Method | Notes | Parameters |
|---|---|---|
| | @param letter The character the transition is over | |
| **allStates()** Set<State> Public | | |
| **alphabet()** Set<Character> Public | | |
| **finalStates()** Set<State> Public | | |
| **MapBasedDFA()** Public | Must start out with a given start state. | **State** [in] startState The start state to begin with |
| **startState()** State Public | | |
| **transition()** State Public | | **State** [in] fromState **Character** [in] letter |

## *MapBasedNFA*

*Type:* **Class**
*Status:* Proposed. Version 1.0. Phase 1.0.
*Package:* automata *Keywords:*
*Detail:* *Created on 11/3/2011. Last modified on 11/3/2011.*
*GUID:* {F634EDE2-878C-4714-9257-F0B4F21AF56F}

A HashMap based NFA. Used mainly for temporary testing before we have actual table-based NFA. An epsilon transition is represented by a transition over a "null" character

### *Custom Properties*

- isActive = False

### *Connections*

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Realization** Source -> Destination | Public MapBasedNFA | Public NFA | |
| **Association** Source -> Destination | Public MapBasedNFA | Private startState State | |

### *Attributes*

| Attribute | Notes | Constraints and tags |
|---|---|---|

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **alphabet** Set<Character><br>Private | | *Default:* |
| **finalStates** Set<State><br>Private | | *Default:* |
| **startState** State<br>Private | | *Default:* |
| **transitions**<br>HashMap<State,<br>HashMap<Character,<br>HashSet<State>>><br>Private | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **addEpsilonTransitions()**<br>void<br>Private | A recursive helper function that adds all the states coming from a given state over epsilon transitions. | **Set<State>** [in] set<br>The set to add all the states to<br>**State** [in] currState<br>The state to start from (it is assumed that this was already added to the states) |
| **addTransisition()** void<br>Public | Adds a transition to the NFA. If the character has never been seen before, the internal NFA alphabet will be expanded to include the character. If there is already a transition on the given character from the given from state, the toState will be added to the set of states going from the fromState over the character. The from state does not have to already exist in the DFA, however, the transition will be useless if | **State** [in] fromState<br>The state the transition goes from<br>**Character** [in] character<br><br>**State** [in] toState<br>The state the transition goes to |

| Method | Notes | Parameters |
|---|---|---|
| | it does not.<br>@param letter The character the transition is over | |
| **allStates()** Set<State><br>Public | | |
| **alphabet()** Set<Character><br>Public | | |
| **finalStates()** Set<State><br>Public | | |
| **MapBasedNFA()**<br>Public | Must start out with a given start state. | **State** [in] startState<br>The start state to begin with |
| **startState()** State<br>Public | | |
| **startStates()** Set<State><br>Public | | |
| **transitions()** Set<State><br>Public | | **Set<State>** [in] fromStates<br><br>**Character** [in] letter |
| **transitions()** Set<State><br>Public | Returns all the states that can occur after transitioning over a given character on a state. This also includes epsilon transitions that occur after the transition. If there is no such transition, null is returned.<br>@return All states that occur after the transition (+epsilon trans) | **State** [in] fromState<br>The state to look from<br>**Character** [in] letter<br>The letter to transition over |

## *MinimalNFA*

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {DB074A81-8DA1-414c-BB02-0080365D7A51} |

### ***Custom Properties***

- isActive = False

### ***Connections***

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Realization**<br>Source -> Destination | Public<br>MinimalNFA | Public<br>NFA | |

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association**<br>Source -> Destination | Public<br>MinimalNFA | Private startState<br>State | |

***Attributes***

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **alphabet** Set<Character><br>Private | | *Default:* |
| **finalStates** Set<State><br>Private | | *Default:* |
| **startState** State<br>Private | | *Default:* |
| **tokens** Set<CharToken><br>Private | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **allStates()** Set<State><br>Public | | |
| **alphabet()** Set<Character><br>Public | | |
| **finalStates()** Set<State><br>Public | | |
| **MinimalNFA()**<br>Public | | |
| **startState()** State<br>Public | | |

| Method | Notes | Parameters |
|---|---|---|
| **startStates()** Set<State> Public | | |
| **transitions()** Set<State> Public | | **Set<State>** [in] fromStates<br><br>**Character** [in] letter |
| **transitions()** Set<State> Public | | **State** [in] fromState<br><br>**Character** [in] letter |

## State

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {0D22CBB8-D246-41c4-8707-32B907111EA8} |

Tokens are stored in the state in a stack. The top token is the most general token applying to the State, and the bottom token is the most specific/smallest token.

### *Custom Properties*

- isActive = False

### *Connections*

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association** Source -> Destination | Public NFAConverter | Private startState State | |
| **Association** Source -> Destination | Public MinimalNFA | Private startState State | |
| **Association** Source -> Destination | Public MapBasedNFA | Private startState State | |
| **Association** Source -> Destination | Public MapBasedDFA | Private startState State | |

### *Attributes*

| Attribute | Notes | Constraints and tags |
|---|---|---|

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **count** int<br>Private<br>Static | | *Default:* 0 |
| **finalState** boolean<br>Private | | *Default:* |
| **id** int<br>Private | | *Default:* |
| **name** String<br>Private | | *Default:* |
| **tokens** Stack<Token><br>Private | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **equals()** boolean<br>Public | | **Object** [in] o |
| **getTokens()**<br>Stack<Token><br>Public | | |
| **hashCode()** int<br>Public | | |
| **isFinal()** boolean | | |

| Method | Notes | Parameters |
|---|---|---|
| Public | | |
| **setFinal()** void<br>Public | | **boolean** [in] finalState |
| **State()**<br>Public | | |
| **State()**<br>Public | | **Stack<Token>** [in] tokens |

## *Token*

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {AC47BE95-6D8A-4682-9B15-70C4E69E6E58} |

### ***Custom Properties***

- isActive = False

### ***Attributes***

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **startToken** boolean<br>Private | | *Default:* |
| **value** String<br>Private | | *Default:* |

### ***Operations***

| Method | Notes | Parameters |
|---|---|---|
| Static<br>**anyCharacterToken()**<br>Token | | |

| Method | Notes | Parameters |
|---|---|---|
| Public | | |
| **equals()** boolean<br>Public | | **Object** [in] o |
| **getValue()** String<br>Public | | |
| **hashCode()** int<br>Public | | |
| **isStartToken()** boolean<br>Public | | |
| **opposite()** Token<br>Public | | |
| **Token()**<br>Public | | **String** [in] value<br><br>**boolean** [in] startToken |
| **toString()** String<br>Public | | |

## DFA

*Type:* **Interface** **FiniteAutomata**
*Status:* Proposed. Version 1.0. Phase 1.0.
*Package:* automata *Keywords:*
*Detail:* *Created on 11/3/2011. Last modified on 11/3/2011.*
*GUID:* {686D66DC-86C5-41f7-A1B6-E53E605F232A}

### *Connections*

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association**<br>Source -> Destination | Public<br>ScannerDriver | Private dfa<br>DFA | |
| **Association**<br>Source -> Destination | Public<br>DFAMinimizer | Private originalDFA<br>DFA | |
| **Realization**<br>Source -> Destination | Public<br>MapBasedDFA | Public<br>DFA | |
| **Generalization**<br>Source -> Destination | Public<br>DFA | Public<br>FiniteAutomata | |

### *Operations*

| Method | Notes | Parameters |
|---|---|---|
| **transition()** State<br>Public | Returns the list of state resulting in moving from the fromState to the | **State** [in] fromState<br><br>**Character** [in] letter |

| Method | Notes | Parameters |
|---|---|---|
| | | |

## *FiniteAutomata*

| | |
|---|---|
| *Type:* | **Interface** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {41729338-720C-4f60-871B-F190FDA05205} |

### *Connections*

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Generalization** Source -> Destination | Public NFA | Public FiniteAutomata | |
| **Generalization** Source -> Destination | Public DFA | Public FiniteAutomata | |

### *Operations*

| Method | Notes | Parameters |
|---|---|---|
| **allStates()** Set<State> Public | Returns a copy of the list of all the states in the FA | |
| **alphabet()** Set<Character> Public | Returns a copy of the alphabet of the finite automata | |
| **finalStates()** Set<State> Public | Returns a copy the set of all final states in the FA | |
| **startState()** State Public | The start state of this FA | |

## *NFA*

| | |
|---|---|
| *Type:* | **Interface** **FiniteAutomata** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {B84DF54E-CCF4-4770-B1BD-D5219C35014F} |

### *Connections*

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association**<br>Source -> Destination | Public<br>NFAConverter | Private nfa<br>NFA | |
| **Generalization**<br>Source -> Destination | Public<br>NFA | Public<br>FiniteAutomata | |
| **Realization**<br>Source -> Destination | Public<br>MinimalNFA | Public<br>NFA | |
| **Realization**<br>Source -> Destination | Public<br>MapBasedNFA | Public<br>NFA | |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **startStates()** Set<State><br>Public | Returns the start state, in addition to any states reachable from the start state after epislon transitions. | |
| **transitions()** Set<State><br>Public | Returns the list of possible states that come out the given fromStates across a letter transition | **Set<State>** [in] fromStates<br><br>**Character** [in] letter |
| **transitions()** Set<State><br>Public | Returns the list of possible states that come out the given fromState across a letter transition | **State** [in] fromState<br><br>**Character** [in] letter |

## conversion

| | |
|---|---|
| *Type:* | **Package** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011* |
| *GUID:* | {1E56445C-9030-46fb-BB3C-1BAC845A6C49} |

**conversion** - *(Logical diagram)*

| | |
|---|---|
| *Created By:* | Paul *on* 11/3/2011 |
| *Last Modified:* | 11/3/2011 |
| *Version:* | 1.0. *Locked:* False |
| *GUID:* | {3A9DBADA-AB23-494c-8A94-4232124DE06F} |

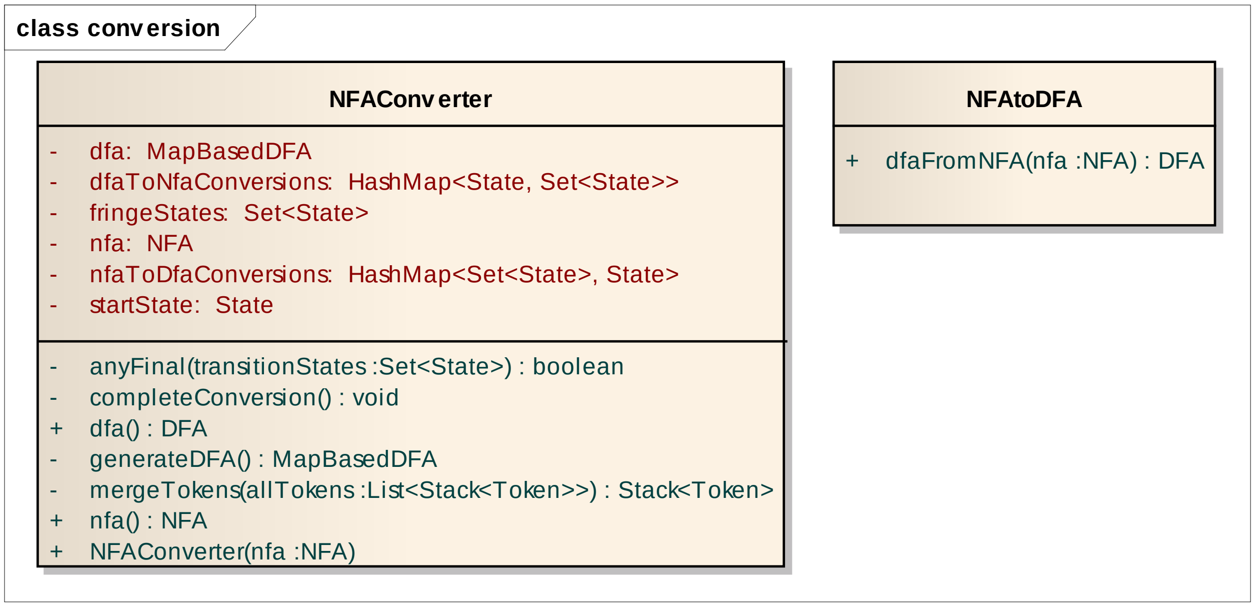
class conversion
NFAConverter
- dfa: MapBasedDFA
- dfaToNfaConversions: HashMap<State, Set<State>>
- fringeStates: Set<State>
- nfa: NFA
- nfaToDfaConversions: HashMap<Set<State>, State>
- startState: State
- anyFinal(transitionStates :Set<State>) : boolean
- completeConversion() : void
+ dfa() : DFA
- generateDFA() : MapBasedDFA
- mergeTokens(allTokens :List<Stack<Token>>) : Stack<Token>
+ nfa() : NFA
+ NFAConverter(nfa :NFA)
NFAtoDFA
+ dfaFromNFA(nfa :NFA) : DFA

Figure: 3

## *NFAConverter*

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | conversion *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {D7618FE3-50D4-4381-91BB-31D98716A95E} |

A utility class used by NFAtoDFA to perform the conversion.

### ***Custom Properties***

- isActive = False

### ***Connections***

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association** Source -> Destination | Public NFAConverter | Private startState State | |
| **Association** Source -> Destination | Public NFAConverter | Private nfa NFA | |
| **Association** Source -> Destination | Public NFAConverter | Private dfa MapBasedDFA | |

### ***Attributes***

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **dfa** MapBasedDFA<br>Private | | *Default:* |
| **dfaToNfaConversions** HashMap<State, Set<State>><br>Private | | *Default:* |
| **fringeStates** Set<State><br>Private | | *Default:* |
| **nfa** NFA<br>Private | | *Default:* |
| **nfaToDfaConversions** HashMap<Set<State>, State><br>Private | | *Default:* |
| **startState** State<br>Private | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **anyFinal()** boolean<br>Private | | **Set<State>** [in] transitionStates |
| **completeConversion()**<br>void<br>Private | | |
| **dfa()** DFA<br>Public | | |
| **generateDFA()**<br>MapBasedDFA<br>Private | | |
| **mergeTokens()**<br>Stack<Token><br>Private | | **List<Stack<Token>>** [in]<br>allTokens |
| **nfa()** NFA<br>Public | | |
| **NFAConverter()**<br>Public | | **NFA** [in] nfa |

## *NFAtoDFA*

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | conversion *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {35162A47-4115-414e-AC2E-14D7B7438AD0} |

### ***Custom Properties***

- isActive = False

### ***Operations***

| Method | Notes | Parameters |
|---|---|---|
| Static **dfaFromNFA()**<br>DFA<br>Public | | **NFA** [in] nfa |

## minimization

| | |
|---|---|
| *Type:* | **Package** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | automata |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011* |

*GUID:* {65E318F1-2EAA-4d0a-BA82-6B7BDECA51B9}

**minimization** - *(Logical diagram)*

| | |
|---|---|
| *Created By:* | Paul *on* 11/3/2011 |
| *Last Modified:* | 11/3/2011 |
| *Version:* | 1.0. *Locked:* False |
| *GUID:* | {ABFB90EE-FDE7-4fe4-B5DE-391906D4CFD9} |

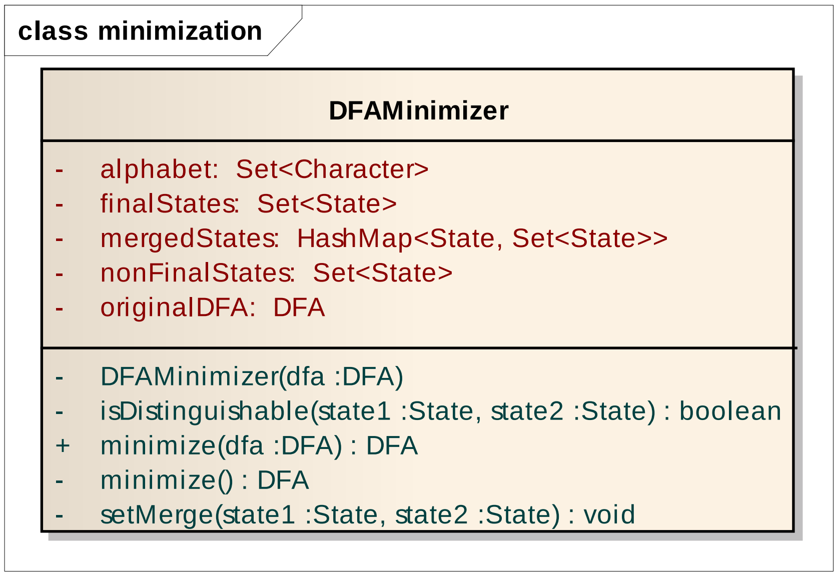
class minimization
DFAMinimizer
- alphabet: Set<Character>
- finalStates: Set<State>
- mergedStates: HashMap<State, Set<State>>
- nonFinalStates: Set<State>
- originalDFA: DFA
- DFAMinimizer(dfa :DFA)
- isDistinguishable(state1 :State, state2 :State) : boolean
+ minimize(dfa :DFA) : DFA
- minimize() : DFA
- setMerge(state1 :State, state2 :State) : void

Figure: 4

## DFAMinimizer

| | |
|---|---|
| *Type:* | **Class** |
| *Status:* | Proposed. Version 1.0. Phase 1.0. |
| *Package:* | minimization *Keywords:* |
| *Detail:* | *Created on 11/3/2011. Last modified on 11/3/2011.* |
| *GUID:* | {769920DC-FC7E-499e-A97E-37A54437639F} |

### Custom Properties

- isActive = False

### Connections

| Connector | Source | Target | Notes |
|---|---|---|---|
| **Association** Source -> Destination | Public DFAMinimizer | Private originalDFA DFA | |

***Attributes***

| Attribute | Notes | Constraints and tags |
|---|---|---|
| **alphabet** Set<Character> Private | | *Default:* |
| **finalStates** Set<State> Private | | *Default:* |
| **mergedStates** HashMap<State, Set<State>> Private | | *Default:* |
| **nonFinalStates** Set<State> Private | | *Default:* |
| **originalDFA** DFA Private | | *Default:* |

***Operations***

| Method | Notes | Parameters |
|---|---|---|
| **DFAMinimizer()** Private | | **DFA** [in] dfa |
| **isDistinguishable()** boolean Private | | **State** [in] state1<br><br>**State** [in] state2 |

| **Method** | **Notes** | **Parameters** |
|---|---|---|
| | | |
| Static **minimize()** DFA<br>Public | | **DFA** [in] dfa |
| **minimize()** DFA<br>Private | | |
| **setMerge()** void<br>Private | | **State** [in] state1<br><br>**State** [in] state2 |