# CS410-Project1 Phase1 Design Report

By Berkecan Koçyiğit

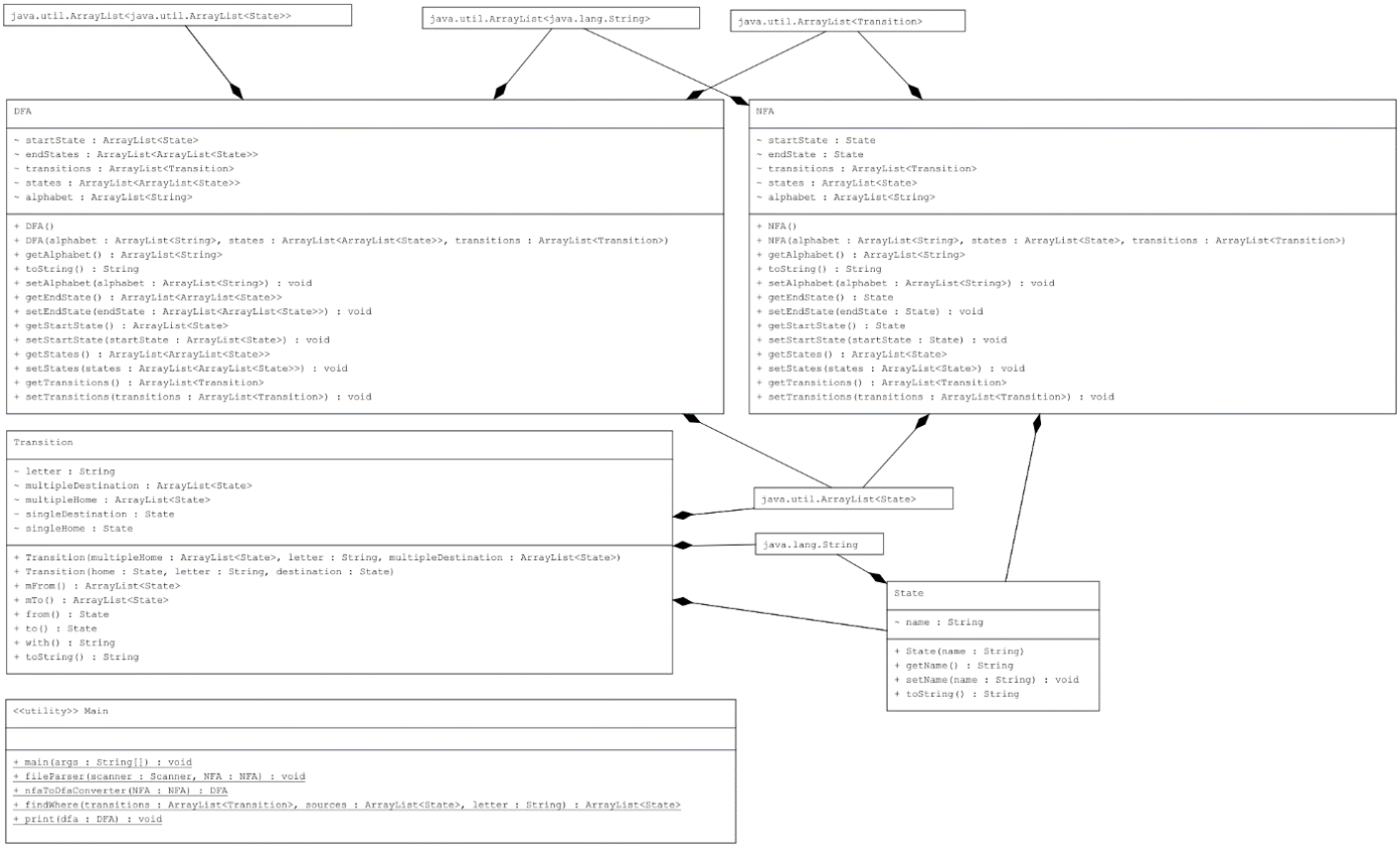
## 1. Understanding the concept

In order to convert NFA to DFA I used the algorithm that we learned in the class. But the entire project was not about conversion. I had to decide how to parse the text file and which data structure to use. After initializing the NFA I had to convert NFA to DFA. Following steps will explain each step in detail.

## 2. Data structures and Classes

I have following classes in order to represent my data:

- Main
- Transition
- State
- NFA
- DFA




Above you can see my UML diagram.

# 3. Classes and algorithm in detail

## 3.1. Classes

### 3.1.1. NFA and DFA Classes

NFA and DFA objects are very similar and can be the subset of the same object which can be called as FA. But I didn't see any benefit from it. NFA object has startState (State) and endState(Array of States), Transitions array which holds Transition objects, states array that holds State objects. DFA object has same parameters that only different thing is states array holds array of State objects (ArrayList<ArrayList<State>>).

### 3.1.2. Transition Class

Transition object has 2 versions for home and destination objects. They are separated as singular and multiple. singularHome and singularDestination stores one State object. multipleHome and multipleDestination objects are storing an array of State objects. They are separated like this because DFA's states are sets of states so an array of State objects used to represent it.

### 3.1.3. State Class

State object is a simple class with just name String (Only for 1 state name).

## 3.2. Algorithm

### 3.2.1. Main Function and fileParser

Main function takes the txt file with the Scanner object and sends it to the fileParser method. File parser method starts with an "ALPHABET" string and continuously takes each line as an alphabet letter (String) and puts it in an ArrayList called alphabet until it sees a "STATES" string. Then it takes each line as the name of a new State object and puts it into states ArrayList. Then the process goes similar for every step of text parsing. Then all that ArrayLists, startState, endState used to create NFA object and that NFA object returned to main.

### 3.2.2. nfaToDfaConverter

Then the main function puts that NFA object in the nfaToDfaConverter method. Then the nfaToDfaConverter method returns a DFA object. That object is used as a parameter of print function.

nfaToDfaConverter method (will be called converter) takes an NFA object as a parameter. Then creates states, alphabet, transitions List from NFA object's getters. Then it creates an empty DFA object. At this stage there are a lot of sequential steps. Because of that, I will explain the process with bullet points in order.

- DFAstates and DFAtransitions Lists are created. (Note that DFAstates objects is an ArrayList that holds an ArrayList that contains State objects because DFA uses sets in states. Also DFAtransitions object uses array variables of Transition object (will be explained with detail))
- startStateList created and added startState of NFA objects with getters.
- startStateList added into DFAstates.
- deadState created with transitions. Transitions added into DFAtransitions.

After this stage there is a while(True) loop. Inside of that loop there are 2 nested for loops that look for every DFAstate's for each alphabet. There is a method called findWhere that finds which states are reachable by source states and letters for every transition (NFA transition). That method simply contains an if statement inside of the for loop, then returns the state array. After the findWhere method returns the destination state array, transition objects can be created using letter and state objects that come from the for loop and findWhere method returned. That destination state array object is added to DFAstates Array if it isn't inside of the array. Then the same logic applied to the DFAtransitions array. After completing both loops a if checks DFAstates and DFAtransitions' sizes changed or not. If they changed, while(True) loop will be exited using break.

In clear form, inside the while(True) infinite loop two for loops checks for every state array object that inside the DFAstates object for every alphabet. Then if there is a transition that includes that letter and source state, destination state will be added in the DFAstates array. Then the Transition object is created and added to the DFAtransitions array. Both additions are checked if there is the same object in the array. So, in each while loop cycle one or more state arrays are added to the DFAstates array. That state array will be used next while cycle and cycle continues until there are no changes to DFAstates array. With that DFA conversion is done.

There is also deadstate state object that has a Transition to itself for each alphabet. If a state goes nowhere, that state has transition to deadstate.