The Main Class:

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// Major           : Computer Science
// Course          : Computer Theory - 601322 / First Semester 2016-2017
// Assignment No.  : 2
// Author          : Ibrahim Alhamad
// Author ID       : 201220475
// Description     : It is a dynamic Finite Automata for any language with any alphabet, you can test the
//                   machine to accept or reject any given word.
////////////////////////////////////////////////////////////////////////////////////////////////////
import java.util.Scanner;
public class Main {
   public static void main(String args[]) {

      Scanner input = new Scanner(System.in); // Scanner object to get the input from the user.

      System.out.print("Enter the alphabet in one string: ");
      String AlphabetString = input.next(); // get the alphabet string from the user.
      Character Alphabet[] = new Character[AlphabetString.length()]; // set the size of the Alphabet
      for (int i = 0; i < AlphabetString.length(); i++)
         Alphabet[i] = AlphabetString.charAt(i); // split the AlphabetString into Alphabet array.

      System.out.print("Enter the states in one string (*it have to be numeric numbers*, e.g.'01234'): ");
      String States[] = input.next().split(""); // split the states string into states array.

      System.out.print("Enter the start state: ");
      String startState = input.next(); // set the start state to the user input.

      System.out.print("Enter the final states in one string(it have to be like the states): ");
      String FinalStates[] = input.next().split(""); // split the states string into states array.

      DFA dfa = new DFA(Alphabet, States, startState, FinalStates);
```

```
        while (true) {
            System.out.print("Enter a word: ");
            String word = input.next();// get the word from the user.
            System.out.println(dfa.isItAccepted(word));
        }
    }
}
```

The DNF Class:

```
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
public class DFA {

    private String transitionsTable[][]; // transition table as a 2D array
    private List<Character> Alphabet;    // the Alphabet as a List Data Structure
    private String States[];                    // Array of states
    private String FinalStates[];               // array of the final states
    private String StartState;                  // the start state
    private String currentState;         // the current state

    DFA(Character Alphabet[],String States[], String startState, String FinalStates[]) {
        this.Alphabet = Arrays.asList(Alphabet);      // convert the Alphabet array into a List
        this.States = States;                       // set the states array
        this.StartState = startState;                // set the start state
        this.FinalStates = FinalStates;              // set the final states array
        this.transitionsTable = fillTransitionsTable();  // fill the transitions table by the user
    }
```

```
    String isItAccepted(String token) {
        this.currentState = this.StartState; // set the current state to the start state
            for(int index = 0;index < token.length();index++) {
                if (Alphabet.contains(token.charAt(index))) { // check if the current character is in the Alphabet
                    int csAsIndex = Integer.parseInt(this.currentState); // copy the currentState as an Index
                    int indexOfTheChar = this.Alphabet.indexOf(token.charAt(index)); // get the current char Index
                    this.currentState = this.transitionsTable[csAsIndex][indexOfTheChar]; // were the magic happen
                } else {
                    return "Rejected"; // Rejected if the word contains non-alphabet characters
                }
            }
            for(int i = 0; i < FinalStates.length; i++) {
                if (currentState.equals(FinalStates[i])) {
                    return "Accepted"; // Accepted if the current state is one of the final states
                }
            }
        return "Rejected";
    }
    private String[][] fillTransitionsTable() {
      int rows = States.length;   // set the rows number to the state numbers
      int cols = Alphabet.size(); // set the cols number to the alphabet numbers
      String[][] data = new String[rows][cols]; // set the transitions table size (rows * cols)
      Scanner input = new Scanner(System.in);

      for(int row = 0; row < data.length; row++) { // loop on each row
        for(int col = 0;col < data[row].length; col++) { // loop through each row
          System.out.print("Transition('" + States[row] + "', " + Alphabet.get(col) + ") = ");
          data[row][col] = input.next(); // Transition(x, y) = z
        }
        System.out.println("---------------------");
      }
      return data; // return the transitions table
    }
}
```