Ex. No. 5 Simulation of a DFA

AIM

To simulate the concept of Deterministic Finite Automata.
Video Link : https://youtu.be/prQZkR2NGMc

DESCRIPTION

In theory of computation, a branch of theoretical computer science, a deterministic finite automaton (DFA)—also known as deterministic finite state machine—is a finite state machine that accepts/rejects finite strings of symbols and only produces a unique computation (or run) of the automaton for each input string.
A deterministic finite automaton M is a 5-tuple, $(Q, \Sigma, \delta, q_0, F)$, consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function ($\delta : Q \times \Sigma \rightarrow Q$)
- an initial or start state ($q_0 \in Q$)
- a set of accept states ($F \subseteq Q$)

ALGORITHM

Input: An input string x terminated by an eof character. A DFA D with start state S0 and set of accepting states F
Output: The answer "yes" if DFA accepts the string; "no" otherwise.
Method:

```
s=s0
c=nextchar
while c!=eof do
    s=move(s,c);
    c=nextchar;
end
if s is in F then
   return "yes"
else return "no"
```

SAMPLE INPUT & OUTPUT

DFA Transition Table

	a	b
A	B	A
B	B	C
C	B	D
D	B	A

Input String : aabb

Output: yes

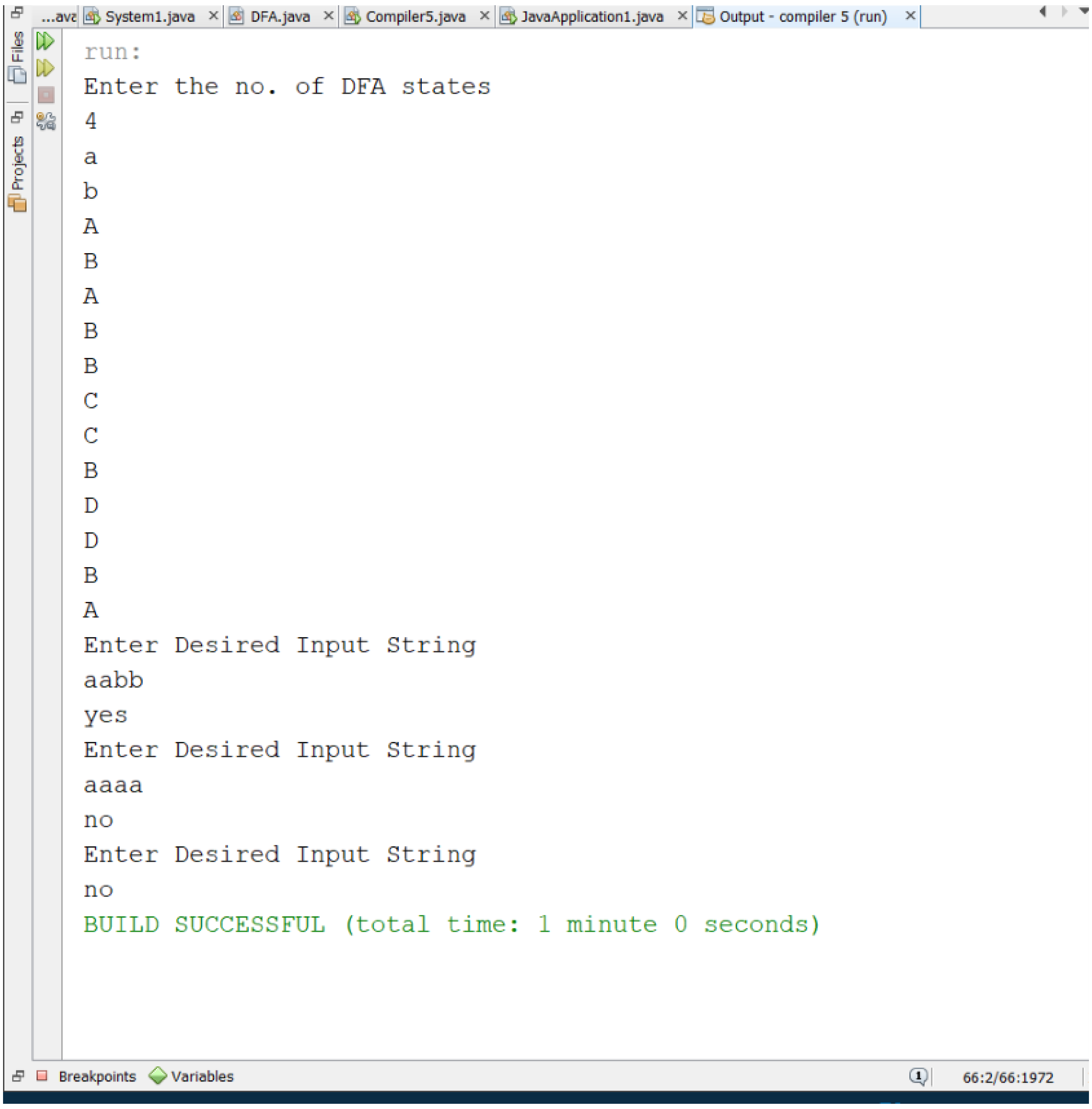
QUESTION SET

1. Write a program to check whether the given input string is accepted by the given DFA.
2. Write a program to simulate deterministic finite automata.

   Working code:

```
package compiler.pkg5;

import java.util.Scanner;

public class Compiler5 {
static char currentChar;
static char currentState;
static char[] LineCharArray;
static int LineCharArrayCount = 0;

  public static void main(String[] args) {

    //Scanner obj's
    Scanner inputScanner = new Scanner(System.in);
    //reading input
    DFA.GetDFATable();

    boolean userSatisfied = false;
    while( ! userSatisfied){
      System.out.println("Enter Desired Input String");
      String inputString = inputScanner.nextLine();
      if(inputString.contentEquals("no"))
        break;
      else{
        //converting into readable format
        LineCharArray = inputString.toCharArray();
        LineCharArrayCount = 0;
        //transitions start
        Algorithm();
      }
    }
  }

  private static void Algorithm() {
      //Algorithm used in the experiment
    currentState = DFA.InitialState();    //denotes current state
    currentChar = NextChar();      //current character from the input
string
    while(currentChar != '$'){
      currentState = Move(currentState,currentChar);
```

```
            currentChar = NextChar();
        }
        //checking if input is correct or not
        System.out.println(FinalStateChecker());
    }

    private static char NextChar() {
        if(LineCharArray == null || LineCharArrayCount  ==
LineCharArray.length){
            return '$';
        }else{
            return LineCharArray[LineCharArrayCount++];
        }
    }

    private static char Move(char currentstate,char currentchar) {
        return DFA.NextState(currentstate,currentchar);
    }

    private static String FinalStateChecker() {
            if(DFA.FinalState()==currentState) {
                return "yes";
            }else {
                return "no";
            }
        }
}


package compiler.pkg5;

import java.util.Scanner;

public class DFA {
    static char DfaTable[][];
    static int noOfStates;

    public static void GetDFATable() {
        //local scanner
        Scanner DFAScanner = new Scanner(System.in);
        Scanner getStates = new Scanner(System.in);
        //getting the DFA states to initialize the array
        System.out.println("Enter the no. of DFA states");
        noOfStates = getStates.nextInt()+1;
        DfaTable = new char[noOfStates][3];
//      CurrentState(InitialState());
        //storing elements into the DFA table

        for(int i=0;i<noOfStates;i++){
            for(int j=0;j<3;j++){
                if(i==0 && j==0){DfaTable[0][0] = ' ';continue;}
                DfaTable[i][j] = DFAScanner.next().charAt(0);
            }
        }
    }

    public static char InitialState() {

        return DfaTable[1][0];
    }

    public static char NextState(char currentstate,char currentchar){
        for(int i=0;i<noOfStates;i++){
            if(currentstate == DfaTable[i][0]){
                if(currentchar == DfaTable[0][1]){
                    return DfaTable[i][1];
                }else if(currentchar == DfaTable[0][2]){
                    return DfaTable[i][2];
                }
            }
        }
        return '$';
    }

    public static char FinalState() {
        return DfaTable[noOfStates-1][0];
    }
}
```



```
run:
Enter the no. of DFA states
4
a
b
A
B
A
B
B
C
C
B
D
D
B
A
Enter Desired Input String
aabb
yes
Enter Desired Input String
aaaa
no
Enter Desired Input String
no
BUILD SUCCESSFUL (total time: 1 minute 0 seconds)
```