```
import java.util.StringTokenizer;
public class Conversion {
 String Nonterminal[];
 String Terminal[];
 String TT[][];
 String STT[][];
 String Prod[][];
 int STTsize;
public Conversion(String Nonterminal[],String Terminal[],String TT[][],String Prod[][],String STT[][])
   {
     this.Nonterminal=Nonterminal;
     this.Terminal=Terminal;
     this.TT =TT;
     this.Prod=Prod;
     this.STT=STT;
   }
public void PrintTransistionTable(String TT[][], String P[][],String T[],String NT[])
{
      int i, j, k, m;
      TT[0][0] = "-";
      for (i = 1; i < 4; i++)
      {
         TT[0][i] = T[i - 1];
         STT[0][i] = T[i - 1];
      }
      for (i = 1; i < 5; i++)
      {
         TT[i][0] = NT[i - 1];
         STT[i][0] = NT[i - 1];
      }
      for (i = 1; i <= 4; i++)//for TT row cpunt
      {
         for (j = 0; j <= 6; j++)//for colum Production row count
         {
            if (TT[i][0] == P[j][0])
            {
               for (k = 1; k <= 3; k++)
               {
                  for (m = 0; m <= 6; m++)
                  {
                     if (TT[0][k] == P[m][1] && TT[i][0] == P[m][0])
                     {
                        if (TT[i][k] == "-")
                        {
                           TT[i][k] = P[m][2];
                        }
                        else if (TT[i][k] != P[m][2])
                        {
                           String str = TT[i][k]; //take previous values stored in transistion table
                           if (str.length() < 2)
                           {
                              TT[i][k] = TT[i][k] + "," + P[m][2];
                           }
                        }
                     }
                  }
               }
            }
         }
      }
```

```
    System.out.println("\n");
    System.out.println("\t" + "===Transistion Table===");
    System.out.println("\t" + "-----------------------");
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < 4; j++)
        {
            System.out.print("\t" + TT[i][j]);
        }
        System.out.println("\n");
    }

    STT[0][0] = "-";
    for (i = 0; i < 4; i++)
    {
        STT[1][i] = TT[1][i];
    }
    STTsize = 2;

}
public void subTransTable()
{
    boolean addRow = false;
    for (int i = 1; i <= 3; i++)
    {
        for (int j = 1; j < STTsize; j++)
        {
            if (STT[j][0].equals(STT[j][i]) == false && STT[j][i].equals("-") == false)
            {
                String st = STT[j][i];
                for (int z = 1; z < STTsize; z++)
                {
                    if (st.equals(STT[z][0]))
                    {
                        addRow = false;
                        z = STTsize + 1;
                    } else
                    {
                        addRow = true;
                    }
                }
                if (addRow)
                {
                    String transt[][] = getRow(st);
                    for (int k = 0; k < 4; k++)
                    {
                        STT[STTsize][k] = transt[0][k];
                    }
                    addRow = false;
                }
            }
        }
    }
    STTsize++;
    if (STTsize < 11)
    {
        subTransTable();
    }
}
```

```
public void displaySTT()
{
   System.out.println("\n");
   System.out.println("\t" + "===Sub Transistion Table===");
   System.out.println("\t" + "-----------------------");
   for (int i = 0; i < 10; i++)
   {
      for (int j = 0; j < 4; j++)
      {
         System.out.print("\t" + STT[i][j]);
      }
      System.out.println("\n");
   }
}

public String[][] getRow(String ST)
{
   String transt[][] = new String[1][4];
   StringTokenizer StrTok = new StringTokenizer(ST, ",");
   ST = "";
   while (StrTok.hasMoreTokens())
   {
      ST = ST + StrTok.nextToken();
   }
   for (int i = 0; i < 4; i++)
   {
      transt[0][i] = "";
   }
   for (int i = 1; i < 5; i++)
   {
      for (int k = 0; k < ST.length(); k++)
      {
         if (TT[i][0].equals(ST.substring(k, k + 1)))
         {
            for (int j = 0; j < 4; j++) {
               if (k == 0 || transt[0][j].equals("-"))
               {
                  transt[0][j] = TT[i][j];
               } else if (TT[i][j].equals("-") == false)
               {
                  transt[0][j] = transt[0][j] + "," + TT[i][j];
               }

            }
            if (k == ST.length())
            {
               i = 5;
            }
         }
      }
   }
   return transt;
}

public void PrintProduction(String lProd[][])
{
   System.out.println("\n");
   System.out.println("\tConversion of NDFA to DFA:");
   System.out.println("\n");
   System.out.println("\tProduction are:");
```

```
        for (int i = 0; i < 7; i++)
        {
            System.out.println("\t" + lProd[i][0] + "=>" + lProd[i][1] + "" + lProd[i][2]);

        }
    }

    public static void main(String[] args) {
        String NonTerminal[] = {"A", "B", "C", "D"};

        String Terminal[] = {"a", "b", "c"};

        String TT[][] = {{"-", "-", "-", "-"},
        {"-", "-", "-", "-"},
        {"-", "-", "-", "-"},
        {"-", "-", "-", "-"},
        {"-", "-", "-", "-"}};
        String STT[][] = new String[10][4];
        String Prod[][] = {{"A", "a", "A"},
        {"A", "c", "B"},
        {"B", "b", "A"},
        {"B", "b", "B"},
        {"B", "a", "C"},
        {"C", "a", "D"},
        {"D", "c", "C"}};

        Conversion c = new Conversion(NonTerminal, Terminal, TT, Prod, STT);

        c.PrintProduction(Prod);
        c.PrintTransistionTable(TT, Prod, Terminal, NonTerminal);
        c.subTransTable();
        c.displaySTT();

    }

}
```