```java
import java.util.Random;
import java.util.Timer;

public class KMP {
    private final int R;       // the radix
    private int[][] dfa;       // the KMP automoton

    private char[] pattern;    // either the character array for the pattern
    private String pat;        // or the pattern string

    public int NUM_COMPARISONS;

    /**
     * Preprocesses the pattern string.
     *
     * @param pat the pattern string
     */
    public KMP(String pat) {
        this.R = 256;
        this.pat = pat;

        // build DFA from pattern
        int m = pat.length();
        dfa = new int[R][m];
        dfa[pat.charAt(0)][0] = 1;
        NUM_COMPARISONS++;
        for (int x = 0, j = 1; j < m; j++) {
            for (int c = 0; c < R; c++)
                dfa[c][j] = dfa[c][x];     // Copy mismatch cases.
            dfa[pat.charAt(j)][j] = j+1;   // Set match case.
            x = dfa[pat.charAt(j)][x];     // Update restart state.
            NUM_COMPARISONS+=2;
        }
    }

    /**
     * Preprocesses the pattern string.
     *
     * @param pattern the pattern string
     * @param R the alphabet size
     */
    public KMP(char[] pattern, int R) {
        this.R = R;
        this.pattern = new char[pattern.length];
        for (int j = 0; j < pattern.length; j++)
            this.pattern[j] = pattern[j];

        // build DFA from pattern
        int m = pattern.length;
        dfa = new int[R][m];
        dfa[pattern[0]][0] = 1;
        NUM_COMPARISONS++;
        for (int x = 0, j = 1; j < m; j++) {
            for (int c = 0; c < R; c++)
                dfa[c][j] = dfa[c][x];     // Copy mismatch cases.
            dfa[pattern[j]][j] = j+1;      // Set match case.
            x = dfa[pattern[j]][x];        // Update restart state.
            NUM_COMPARISONS+=2;
        }
    }
```

```
61
62      /**
63       * Returns the index of the first occurrrence of the pattern string
64       * in the text string.
65       *
66       * @param  txt the text string
67       * @return the index of the first occurrence of the pattern string
68       *         in the text string; N if no such match
69       */
70      public int search(String txt) {
71
72          // simulate operation of DFA on text
73          int m = pat.length();
74          int n = txt.length();
75          int i, j, count=0;
76          for (i = 0, j = 0; i < n; i++) {
77              j = dfa[txt.charAt(i)][j];
78              NUM_COMPARISONS++;      // +1 array inspection per for loop
interation
79              if (j==m)  {
80                  count++;
81                  j=0;
82              }
83          }
84          return count;                   // returns count, number of times we
found the word
85      }
86
87      /**
88       * Returns the index of the first occurrrence of the pattern string
89       * in the text string.
90       *
91       * @param  text the text string
92       * @return the index of the first occurrence of the pattern string
93       *         in the text string; N if no such match
94       */
95      public int search(char[] text) {
96
97          // simulate operation of DFA on text
98          int m = pattern.length;
99          int n = text.length;
100         int i, j, count=0;
101         for (i = 0, j = 0; i < n; i++) {
102             j = dfa[text[i]][j];
103             NUM_COMPARISONS++; // +1 array inspection per for loop interation
104             if (j==m)  {
105                 count++;
106                 j=0;
107             }
108         }
109         return count;                    // returns count, number of times we
found the word
110     }
111
112
113     /**
114      * Takes a pattern string and an input string as command-line arguments;
115      * searches for the pattern string in the text string; and prints
116      * the first occurrence of the pattern string in the text string.
117      *
```

```
     * @param args the command-line arguments
     */
    public static void main(String[] args) {
        //Pattern and text. Can modify these to see different results.
        String pat = "dsgwadsgz";
        String txt = "adsgwadsxdsgwadsgz";

        char[] pattern = pat.toCharArray();
        char[] text    = txt.toCharArray();

        //string implementation of KMP
        KMP kmp1 = new KMP(pat);
        int count1 = kmp1.search(txt);

        //char array implementation of KMP
        KMP kmp2 = new KMP(pattern, 256);
        int count2 = kmp2.search(text);

        // print results
        System.out.println("text: " + txt + " pattern: " + pat);
        System.out.println("For the string version of KMP, it found it " + count1 + " times.");
        System.out.println("For the string version of KMP, it made " + kmp1.NUM_COMPARISONS + " comparisons.");
        System.out.println("For the char version of KMP, it found it " + count2 + " times.");
        System.out.println("For the char array version of KMP, it made " + kmp2.NUM_COMPARISONS + " comparisons.");
    }


}

```