Code :

```
import java.util.Scanner;

public class StringMatchingAlgorithms {

    // Naive String Matching Algorithm
    static void naiveSearch(String text, String pattern) {
        int n = text.length();
        int m = pattern.length();
        int stepCount = 0;

        System.out.println("\nNaive String Matching:");
        for (int i = 0; i <= n - m; i++) {
            int j;
            for (j = 0; j < m; j++) {
                stepCount++;
                if (text.charAt(i + j) != pattern.charAt(j))
                    break;
            }
            if (j == m) {
                System.out.println("Pattern found at index " + i);
            }
        }
        System.out.println("Total character comparisons (steps): " + stepCount);
    }

    // Rabin–Karp String Matching Algorithm
    static void rabinKarpSearch(String text, String pattern, int q) {
```

```
int d = 256; // number of characters in input alphabet
int n = text.length();
int m = pattern.length();
int p = 0; // hash value for pattern
int t = 0; // hash value for text window
int h = 1;
int stepCount = 0;

// The value of h would be "pow(d, m-1) % q"
for (int i = 0; i < m - 1; i++)
   h = (h * d) % q;

// Calculate hash value for pattern and first window of text
for (int i = 0; i < m; i++) {
   p = (d * p + pattern.charAt(i)) % q;
   t = (d * t + text.charAt(i)) % q;
}

System.out.println("\nRabin–Karp String Matching:");
// Slide the pattern over text
for (int i = 0; i <= n - m; i++) {
   stepCount++;

   // Check hash values of pattern and current window
   if (p == t) {
      // Check characters one by one to confirm
      int j;
      for (j = 0; j < m; j++) {
         stepCount++;
```

```
            if (text.charAt(i + j) != pattern.charAt(j))
                break;
        }
        if (j == m)
            System.out.println("Pattern found at index " + i);
    }

    // Calculate hash value for next window
    if (i < n - m) {
        t = (d * (t - text.charAt(i) * h) + text.charAt(i + m)) % q;
        if (t < 0)
            t += q;
    }
  }
  System.out.println("Total hash comparisons (steps): " + stepCount);
}

// Main Function
public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);

  System.out.print("Enter text: ");
  String text = sc.nextLine();

  System.out.print("Enter pattern: ");
  String pattern = sc.nextLine();

  naiveSearch(text, pattern);
  rabinKarpSearch(text, pattern, 101); // 101 is a prime number for hashing
```

```
        sc.close();
    }
}
```

Output :

Enter text: ABCCDDAEFG

Enter pattern: DDA

Naive String Matching:

Pattern found at index 4

Total character comparisons (steps): 16

Rabin–Karp String Matching:

Pattern found at index 4

Total hash comparisons (steps): 10