Given an input string s and a pattern p, implement regular expression matching with support for '.' and '\*' where:

* '.' Matches any single character.​​​​
* '\*' Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

**Example 1:**

**Input:** s = "aa", p = "a"

**Output:** false

**Explanation:** "a" does not match the entire string "aa".

**Example 2:**

**Input:** s = "aa", p = "a\*"

**Output:** true

**Explanation:** '\*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

**Example 3:**

**Input:** s = "ab", p = ".\*"

**Output:** true

**Explanation:** ".\*" means "zero or more (\*) of any character (.)".

**Solution:**

class Solution {

public boolean isMatch(String s, String p) {

if (s == null || p == null) {

return false;

}

boolean[][] dp = new boolean[s.length()+1][p.length()+1];

dp[0][0] = true;

for (int i = 0; i < p.length(); i++) {

if (p.charAt(i) == '\*') { // AA = A\*

dp[0][i+1] = dp[0][i-1];

}

}

for (int i = 0 ; i < s.length(); i++) {

for (int j = 0; j < p.length(); j++) {

if (p.charAt(j) == '.') { // when AB = A.

dp[i+1][j+1] = dp[i][j];

}

if (p.charAt(j) == s.charAt(i)) {

dp[i+1][j+1] = dp[i][j];

}

if (p.charAt(j) == '\*') {

if (p.charAt(j-1) != s.charAt(i) && p.charAt(j-1) != '.') { // When A = AB\*

dp[i+1][j+1] = dp[i+1][j-1];

} else {

dp[i+1][j+1] = (dp[i+1][j] // in this case, a\* counts as single a (XA = XA\*)

|| dp[i][j+1] //in this case, a\* counts as multiple a (XAA = XA\*)

|| dp[i+1][j-1] // in this case, a\* counts as empty (A = AX\*)

);

}

}

}

}

return dp[s.length()][p.length()];

}

}