Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '\*' where:

* '?' Matches any single character.
* '\*' Matches any sequence of characters (including the empty sequence).

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 = "\*"

**Output:** true

**Explanation:** '\*' matches any sequence.

**Example 3:**

**Input:** s = "cb", p = "?a"

**Output:** false

**Explanation:** '?' matches 'c', but the second letter is 'a', which does not match 'b'.

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() && (p.charAt(i) == '\*'); i++) { //case when aa = \*

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

}

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) == '\*') {

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\*)

);

}

}

}

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

}

}