请事先一个函数用来匹配包括'.'和'*'的正则表达式。模式中的字符'.'表示任意一个字符，而'*'表示它前面的字符可以出现任意次(包含0次)。在本题中，匹配是指字符串的所有字符匹配整个模式。例如，字符串"aaa"与模式"aaa"和"ab*ac*a"匹配，但是与"aa.a"和"ab*a"均不匹配。

```
public class B19 {
    public boolean match(String str, String pattern) {
        if(str == null || pattern == null)
            return false;
        return matchCore(new StringBuilder(str), 0, new StringBuilder(pattern), 0);
    }
    /*
     * '.'相当于一个万能字符，正常匹配就行，但'*'的匹配会涉及到前一个字符。
     * 所以要分两种，第一是后一个字符不是*,或者没有最后一个字符。 那么就看模式串和匹配串是都一样，不一样返回false
     *           第二是后一个字符是*,那么看模式串和匹配串是否一样（包括前面为.和正常匹配），一样则有三种情况，第一是*与匹配串多个相匹配，则为strIndex + 1
     *                                                                  第二是*与匹配串一个相匹配，则为strIndex + 1，patternIndex + 2
     *                                                                  第三是*与匹配串0个相匹配，则为patternIndex + 2
     *                              如果不匹配，则如a 与 b*a,那么strIndex，patternIndex + 2
     *   解释，只有三种，第一没有后一个字符，第二有后一个字符，并且后一个字符不为*,第三有后一个字符，并且后一个字符为*
     */
    public static boolean matchCore(StringBuilder str, int strIndex, StringBuilder pattern, int patternIndex) {
        //如果匹配串和模式串 匹配结束
        if(strIndex == str.length() && patternIndex == pattern.length())
            return true;
        if(strIndex != str.length() && patternIndex == pattern.length())
            return false;
        if(strIndex == str.length() && patternIndex != pattern.length()) {
            if(patternIndex + 1 < pattern.length() && pattern.charAt(patternIndex + 1) == '*')
                return matchCore(str, strIndex, pattern, patternIndex + 2);
            else
                return false;
        }
        //如果模式串的第二个字符不是*,或者已经只剩一个字符了
        if(patternIndex == pattern.length() - 1 || pattern.charAt(patternIndex + 1) != '*') {
            if(pattern.charAt(patternIndex) == '.' || pattern.charAt(patternIndex) == str.charAt(strIndex)) {
                return matchCore(str, strIndex + 1, pattern, patternIndex + 1);
            }else {
                return false;
            }
        }else {
            if(pattern.charAt(patternIndex) == '.' || pattern.charAt(patternIndex) == str.charAt(strIndex))
                return matchCore(str, strIndex + 1, pattern, patternIndex) ||
                        matchCore(str, strIndex + 1, pattern, patternIndex + 2)
                    || matchCore(str, strIndex, pattern, patternIndex + 2);
            else
                 return matchCore(str, strIndex, pattern, patternIndex + 2);
        }
    }
```