```java
package regextalk.password.toshow;

import java.util.Arrays;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Password03LogicTwoRules {

  public static final int MIN_LENGTH = 8;
  public static final int RULE_COUNT = 2;
  public static final boolean WHITESPACE_OK = false;

  private Matcher lowerCaseMatcher =  matcherForRegex("[a-z]");
  private Matcher upperCaseMatcher =  matcherForRegex("[A-Z]");
  private Matcher digitMatcher =      matcherForRegex("[0-9]");
  private Matcher symbolMatcher =     matcherForRegex(
      "[><?.,!@#$%^&*+=_)(\\}\\{\\]\\[]");
  private Matcher whitespaceMatcher = matcherForRegex("\\s");

  public boolean isPasswordValid(String password) {

    int rulesFollowed = getSpecialRulesFollowedCount(password);
    if(rulesFollowed < RULE_COUNT) { return false; }

    boolean longEnough = (password.length() >= MIN_LENGTH);
    if (!longEnough) { return false; }

    if (WHITESPACE_OK) {
      return true;
    }

    boolean whitespaceWasFoundAndIsBad = whitespaceMatcher.
        reset(password).find();
    return whitespaceWasFoundAndIsBad;
  }

  private int getSpecialRulesFollowedCount(String password) {
    int count = (lowerCaseMatcher.reset(password).find() ? 1 : 0) +
            (upperCaseMatcher.reset(password).find() ? 1 : 0);

    //Short circuiting, for when RULE_COUNT happens to be
    //two or greater.
    if (count < RULE_COUNT) {
      count = (digitMatcher.reset(password).find() ? 1 : 0);
    }

    if (count < RULE_COUNT) {
      count = (symbolMatcher.reset(password).find() ? 1 : 0);
    }

    return count;
  }

  private Matcher matcherForRegex(String regex) {
    return Pattern.compile(regex).matcher("ignored input");
  }

  public static void main(String[] ignored) {

```

```
60      String[] inputs = new String[] {
61          "",                 //bad (bad rules, bad length)
62          "abcdefghij",       //bad (bad rules, good length)
63          "abc123",           //bad (good rules, bad length)
64          "abc123abc",        //3 rules: bad, 2 rules: good
65          "a1$A",             //bad (good rules, bad length)
66          "abc123$%^ABC",     //good
67          "abcABC123$&*",     //good
68          "abc ABC123$#$"     //bad (whitespace)
69      };
70
71      Password03LogicTwoRules validator = new Password03LogicTwoRules();
72
73      Arrays.stream(inputs).forEach(input -> {
74
75        boolean valid = validator.isPasswordValid(input);
76        System.out.printf("\"%s\" is %s password.%n", input,
77                    (valid ? "a VALID" : "an invalid"));
78      });
79    }
80  }
81
```