```
package regextalk.password.toshow;

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

public class Password02RegexTwoRules {
   public static final String SPECIAL_CHARS = "><?.,!@#$%^&*+=_)(\\}\\{\\]\\[";
   public static final String LKA_LOWER = "(?=.*[a-z])"; //LKA=lookahead
   public static final String LKA_UPPER = "(?=.*[A-Z])";
   public static final String LKA_DIGIT = "(?=.*[0-9])";
   public static final String LKA_SPECIAL = "(?=.*[" + SPECIAL_CHARS + "])";
   private static final String REGEX = "" +
      "^" +                                //start of input
         "(?:" +                           //non capturing group
            LKA_LOWER + LKA_UPPER +              //Option 1
            "|" +                                   //or
            LKA_LOWER + LKA_DIGIT +              //Option 2
            "|" +                                   //or
            LKA_LOWER + LKA_SPECIAL +            //Option 3
            "|" +                                   //or
            LKA_UPPER + LKA_DIGIT +              //Option 4
            "|" +                                   //or
            LKA_UPPER + LKA_SPECIAL +            //Option 5
            "|" +                                   //or
            LKA_DIGIT + LKA_SPECIAL +            //Option 6
         ")" +                              //end group
      "[A-Za-z0-9" + SPECIAL_CHARS + "]" +  //All possible characters
      "{8,24}" +                            //8 to 24 chars
      "$";                                  //end of input

   public static void main(String[] ignored) {
      Matcher matcher = Pattern.compile(REGEX).matcher("ignored input");

      Arrays.stream(newDemoInputArray()).forEach(input -> {
         boolean valid = matcher.reset(input).find();
         System.out.printf("\"%s\" is %s password.%n",
                           input, (valid ? "a VALID" : "an invalid"));
      });
   }

   public static String[] newDemoInputArray() {
      return new String[]{
            "",                     //bad (bad rules, bad length)
            "abcdefghij",           //bad (bad rules, good length)
            "abc123",               //bad (good rules, bad length)
            "abc123abc",            //3 rules: bad, 2 rules: good
            "a1$A",                 //bad (good rules, bad length)
            "abc123$%^ABC",         //good
            "abcABC123$&*",         //good
            "abc ABC123$#$"         //bad (whitespace)
      };
   }
}
```