Regular Expressions

Exercises and Solutions

1. What is a regular expression?

Answer:

A regular expression is a way to describe a pattern using a sequence of characters.

1. What is a metacharacter? How do you use a metacharacter as an ordinary character in a regular expression?

**Answer:**

All characters that have special meanings inside a regular expression are called metacharacters. When a metacharacter is preceded by a backslash character, it is treated as an ordinary character.

1. What class do you use to compile a pattern?

**Answer:**

java.util.regex.Pattern

1. What class do you use to match a compiled pattern?

**Answer:**

java.util.regex.Matcher

1. What does the regular expression "[aieou]" mean? Will it match the string "Hello"?

**Answer:**

The regular expression "[aieou]" matches any vowel in a given string. Tis regular expression will not match the string “Hello”.

1. Write a regular expression that will match any word that starts with a consonant in lower case, followed by one or more vowels in lower case, and followed by a consonant in lowercase. For example, it should match cat, dog, cool, cot, doom, deem, etc., but not cola, Cat, fish, Cold, etc.

**Answer:**

"[a-z&&[^aeiou]][aeiou]+[a-z&&[^aeiou]]"

1. What will be the output of the following snippet of code?  
     
   String source = "I saw the rat running.";  
   String regex = "r..";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   while(m.find()) {  
    System.out.println(m.group());  
   }

**Answer:**

rat

run

1. Complete the following snippet of code that will match two words "cat" and "cot" in the input. When the code is run, it should print cat and cot in two separate lines.

String source = "cat camera can pen cow cab cot";  
String regex = /\* Your code goes here \*/;  
Pattern p = Pattern.compile(regex);  
Matcher m = /\* Your code goes here \*/;  
while(m.find()) {  
 System.out.println(m.group());  
}

**Solution:**

String source = "cat camera can pen cow cab cot";

String regex = "c[ao]t";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(source);

while(m.find()) {

System.out.println(m.group());

}

1. Complete the following snippet of code that will replace all 3-letter words that start with 'c' with their uppercase equivalent. The code should print "CAT camera CAN pen COW CAB COT".  
     
   String source = "cat camera can pen cow cab cot";  
   String regex = "/\* You code goes here\*/";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   String str = m.replaceAll(mr -> mr.group().toUpperCase());  
   System.out.println(str);

**Solution:**

String source = "cat camera can pen cow cab cot";

String regex = **"c[a-z]{2}"**;

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(source);

String str = m.replaceAll(mr -> mr.group().toUpperCase());

System.out.println(str);

1. Write the output of the following snippet of code.  
     
   String source = "ABXXXABB";  
   String regex = "AB\*";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   String str = m.replaceAll("Hello");  
   System.out.println(str);

**Answer:**

HelloXXXHello

1. Write the output of the following snippet of code.  
     
   String source = "ABXXXABB";  
   String regex = "AB?";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   String str = m.replaceAll("Hello");  
   System.out.println(str);

**Answer:**

HelloXXXHelloB

1. Write the output of the following snippet of code.  
     
   String source = "ABXXXABB";  
   String regex = "AB+";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   String str = m.replaceAll("Hello");  
   System.out.println(str);

**Answer:**

HelloXXXHello

1. Describe the intent of the following snippet of code and write the output.  
     
   String source = "I have 25 cents and 400 books.";  
   String regex = "\\b(\\d+)\\b";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   int sum = m.results()  
    .mapToInt(mr -> Integer.parseInt(mr.group()))  
    .sum();  
   System.out.println(sum);

**Answer:**

Regex matches group of digits in given string, maps each group to an integer, and sum of all mapped integers is assigned to variable sum.

425

1. How many groups are present in the following regular expression:  
   String regex = "\\b((\\d{3})(\\d{3})(\\d{4}))|((\\d{3})(\\d{4}))\\b";

**Answer:**

Seven groups are present in the regular expression.

1. Complete the following snippet of code that prints 7-digit and 10-digit phone numbers in xxx-xxxx and (xxx) xxx-xxxx formats. The output should be "(334) 233-0908, 233-7656, 234, (617) 908-6547, unknown".  
     
   String source = "3342330908, 2337656, 234, 6179086547, unknown";  
   String regex = "/\* Your code goes here\*/";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);  
   StringBuilder sb = new StringBuilder();  
   while(m.find()) {  
    String replacement =   
    m.group(1) != null ? /\* Your code goes here\*/;  
    m.appendReplacement(sb, replacement);  
   }  
     
   m.appendTail(sb);  
   System.out.println(sb.toString());

**Solution:**

String source = "3342330908, 2337656, 234, 6179086547, unknown";

String regex = "\\b(\\d{3}){0,1}(\\d{3})(\\d{4})\\b";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(source);

StringBuilder sb = new StringBuilder();

while(m.find()) {

String replacement =

m.group(1) == null ? **"" : "(" + m.group(1) + ") "**;

replacement += m.group(2) + "-" + m.group(3);

m.appendReplacement(sb, replacement);

}

m.appendTail(sb);  
System.out.println(sb.toString());

1. Complete the following snippet of code that will print each word in the source string on a separate line.  
     
   String source = "bug dug jug mug tug";  
   String regex = "/\*your code goes here\*/";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);

while(m.find()) {  
 System.out.println(m.group());  
}

**Solution:**

String source = "bug dug jug mug tug";

String regex = "([a-z])+\\b";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(source);

while(m.find()) {

System.out.println(m.group());

}

1. The following snippet of code attempts to count and print the number of questions marks (?) in the input string. Complete the following snippet of code, so the output is 3.  
     
   String source = "What? How? I do not know. Why?";  
   String regex = "/\* Your code goes here \*/";  
   Pattern p = Pattern.compile(regex);  
   Matcher m = p.matcher(source);   
   long questionMarkCount = m.results().count();  
   System.out.println(questionMarkCount);

**Solution:**

String source = "What? How? I do not know. Why?";

String regex = **"\\?"**;

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(source);

long questionMarkCount = m.results().count();

System.out.println(questionMarkCount);