Practices for Lesson 7

Practices Overview

In these practices, you use regular expressions and `String.split()` to manipulate strings in Java. For each practice, a NetBeans project is provided for you. Complete the project as indicated in the instructions.

Practice 7-1: Summary Level: Parsing Text with `split()`

Overview

In this practice, parse comma-delimited text and convert the data into Shirt objects.

Assumptions

You have participated in the lecture for this lesson.

Summary

You have been given some comma-delimited shirt data. Parse the data, store it in shirt objects, and print the results. The output from the program should look like the following.

```
=== Shirt List ===
Shirt ID: S001
Description: Black Polo Shirt
Color: Black
Size: XL

Shirt ID: S002
Description: Black Polo Shirt
Color: Black
Size: L

Shirt ID: S003
Description: Blue Polo Shirt
Color: Blue
Size: XL

Shirt ID: S004
Description: Blue Polo Shirt
Color: Blue
Size: M

Shirt ID: S005
Description: Tan Polo Shirt
Color: Tan
Size: XL

Shirt ID: S006
Description: Black T-Shirt
Color: Black
Size: XL
```

```
Shirt ID: S007
Description: White T-Shirt
Color: White
Size: XL

Shirt ID: S008
Description: White T-Shirt
Color: White
Size: L

Shirt ID: S009
Description: Green T-Shirt
Color: Green
Size: S

Shirt ID: S010
Description: Orange T-Shirt
Color: Orange
Size: S

Shirt ID: S011
Description: Maroon Polo Shirt
Color: Maroon
Size: S
```

Tasks

Open the `StringsPractice01` project and make the following changes.

1. Edit the `main` method of the `StringSplitTest.java` file.
2. Parse each line of the `shirts` array.
3. Convert the shirt data into a `List` of `Shirt` objects.
4. Print out the list of shirts.
5. Run the `StringSplitTest.java` file and verify that your output is similar to that shown in the “Summary” section of this practice.

Practice 7-1: Detailed Level: Parsing Text with `split()`

Overview

In this practice, parse comma-delimited text and convert the data into Shirt objects.

Assumptions

You have participated in the lecture for this lesson.

Summary

You have been given some comma-delimited shirt data. Parse the data, store it in shirt objects, and print the results. The output from the program should look like the following.

```
=== Shirt List ===
Shirt ID: S001
Description: Black Polo Shirt
Color: Black
Size: XL

Shirt ID: S002
Description: Black Polo Shirt
Color: Black
Size: L

Shirt ID: S003
Description: Blue Polo Shirt
Color: Blue
Size: XL

Shirt ID: S004
Description: Blue Polo Shirt
Color: Blue
Size: M

Shirt ID: S005
Description: Tan Polo Shirt
Color: Tan
Size: XL

Shirt ID: S006
Description: Black T-Shirt
Color: Black
Size: XL
```

```
Shirt ID: S007
Description: White T-Shirt
Color: White
Size: XL

Shirt ID: S008
Description: White T-Shirt
Color: White
Size: L

Shirt ID: S009
Description: Green T-Shirt
Color: Green
Size: S

Shirt ID: S010
Description: Orange T-Shirt
Color: Orange
Size: S

Shirt ID: S011
Description: Maroon Polo Shirt
Color: Maroon
Size: S
```

Tasks

Open the `StringsPractice01` project and make the following changes.

1. Edit the `main` method of the `StringSplitTest.java` file.
2. Parse each line of the `shirts` array. Create a `Shirt` object for each line and add the `Shirt` to a `List`. A `for` loop to perform these steps could be as follows:

```
        for(String curLine:shirts){
            String[] e = curLine.split(",");
            shirtList.add(new Shirt(e[0], e[1], e[2], e[3]));
        }
```

3. Print out the list of shirts. A loop to do this could be like the following:

```
System.out.println("=== Shirt List ===");
for (Shirt shirt:shirtList){
    System.out.println(shirt.toString());
}
```

4. Run the `StringSplitTest.java` file and verify that your output is similar to that shown in the “Summary” section of this practice.

Practice 7-2: Summary Level: Creating a Regular Expression Search Program

Overview

In this practice, create a program that searches a text file by using regular expressions.

Assumptions

You have participated in the lecture for this lesson.

Summary

Create a simple application that will loop through a text file (`gettys.html`) and search for text by using regular expressions. If the desired text is found on a line, print out the line number and the line text. For example, if you performed a search for “<h4>” the output would be:

```
9  <h4>Abraham Lincoln</h4>
10  <h4>Thursday, November 19, 1863</h4>
```

Tasks

Open the `StringsPractice02` project and make the following changes. Please note that the code to read a file has been supplied for you.

Note: The `gettys.html` file is located in the root of the project folder. To examine the file, with the project open, click the Files tab. Double-click the file to open it and examine its contents.

1. Edit the `FindText.java` file.
2. Create a `Pattern` and a `Matcher` field.
3. Generate a `Matcher` based on the supplied `Pattern` object.
4. Search each line for the pattern supplied.
5. Print the line number and the line that has matching text.
6. Run the `FindText.java` file and search for these patterns.
   - All lines that contain: <h4>
   - All the lines that contain the word “to” (For example, line 17 should not be selected.)
   - All the lines that start with 4 spaces’
   - Lines that begin with “<p” or “<d”
   - Lines that only contain HTML closing tags (for example, “</div>”)
7. (Optional) Modify the program to accept the file name and regular expression pattern on the command line.

Practice 7-2: Detailed Level: Creating a Regular Expression Search Program

Overview

In this practice, create a program that searches a text file by using regular expressions.

Assumptions

You have participated in the lecture for this lesson.

Summary

Create a simple application that will loop through a text file and search for text by using regular expressions. If the desired text is found on a line, print out the line number and the line text. For example, if you performed a search for “<h4>” the output would be:

```
9  <h4>Abraham Lincoln</h4>
10  <h4>Thursday, November 19, 1863</h4>
```

Tasks

Open the `StringsPractice02` project and make the following changes. Please note that the code to read a file has been supplied for you.

Note: The `gettys.html` file is located in the root of the project folder. To examine the file, with the project open, click the Files tab. Double-click the file to open it and examine its contents.

1. Edit the `FindText.java` file.
2. Create fields for a `Pattern` and a `Matcher` object.

```
    private Pattern pattern;
    private Matcher m;
```

3. Outside the search loop, create and initialize your pattern object.

```
        pattern = Pattern.compile("<h4>");
```

4. Inside the search loop, generate a `Matcher` based on the supplied `Pattern` object.

```
        m = pattern.matcher(line);
```

5. Inside the search loop, search each line for the pattern supplied. Print the line number and the line that has matching text.

```
                if (m.find()) {
                   System.out.println(" " + c + "  "+ line);
                }
```

6. Run the `FindText.java` file and search for these patterns.
    - All the lines that contain: <h4>

      ```
              pattern = Pattern.compile("<h4>");
      ```

    - All the lines that contain the word “to” (For example, line 17 should not be selected.)

      ```
              pattern = Pattern.compile("\\bto\\b");
      ```

    - All the lines that start with 4 spaces

      ```
              pattern = Pattern.compile("^\\s{4}");
      ```

    - Lines that begin with “<p” or “<d”

      ```
              pattern = Pattern.compile("^<[p|d]");
      ```

    - Lines that only contain HTML closing tags (for example, “</div>”)

      ```
              pattern = Pattern.compile("^</.*?>$");
      ```

7. (Optional) Modify the program to accept the file name and regular expression pattern on the command line.

Practice 7-3: Summary Level: Transforming HTML by Using Regular Expressions

Overview

In this practice, use regular expressions to transform `<p>` tags into `<span>` tags.

Assumptions

You have participated in the lecture for this lesson.

Summary

You have decided that you want to change the formatting of the `gettys.html` file. Instead of using `<p>` tags, `<span>` tags should be used. In addition, you think that the value for class should be "sentence" instead of "line." Use regular expressions to find the lines that you want to change. Then use regular expressions to transform the tags and the attributes as described. The transformed lines should be output to the console. The output should look like the following:

```
13 <span class="sentence">Four score and seven years ago our
fathers brought forth on this continent a new nation, conceived
in liberty, and dedicated to the proposition that all men are
created equal.</span>
 14 <span class="sentence">Now we are engaged in a great civil
war, testing whether that nation, or any nation, so conceived
and so dedicated, can long endure.</span>
 15 <span class="sentence">We are met on a great battle-field of
that war.</span>
 16 <span class="sentence">We have come to dedicate a portion of
that field, as a final resting place for those who here gave
their lives that that nation might live.</span>
 17 <span class="sentence">It is altogether fitting and proper
that we should do this.</span>
 21 <span class="sentence">But, in a larger sense, we can not
dedicate, we can not consecrate, we can not hallow this
ground.</span>
 …
```

One approach to the problem could be to break the algorithm into three steps.

1. Break the line into three parts: the start tag, the content, and the end tag.
2. Replace the current tags with a new tag.
3. Replace the attribute value with a new attribute value.

Then return the newly formatted line.

The method signatures to replace the tag and attributes might look like this:

```
    public String replaceTag(String tag, String targetTag,
String replaceTag){ }
    public String replaceAttribute(String tag, String attribute,
String value){
```

Tasks

Open the `StringsPractice03` project and make the following changes. Please note that the code to read a file has been supplied for you.

1. Edit the `SearchReplace.java` file.
2. Create a Pattern object to match the entire line.
3. As you loop through the file, do the following:
   - Create a Matcher to match the current line.
   - Execute the `find()` method to find a match.
   - If there is a match, replace the start and end tags.
   - Replace the attribute
4. Create a method that will replace the contents of any tag.
5. Create a method that will replace a tag’s attribute.
6. Run the `SearchReplace.java` file and produce the output shown in the “Summary” section of this practice.

Practice 7-3: Detailed Level: Transforming HTML by Using Regular Expressions

Overview

In this practice, use regular expressions to transform <p> tags into <span> tags.

Assumptions

You have participated in the lecture for this lesson.

Summary

You have decided that you want to change the formatting of the `gettys.html` file. Instead of using <p> tags, <span> tags should be used. In addition, you think that the value for class should be “sentence” instead of “line.” Use regular expressions to find the lines that you want to change. Then use regular expressions to transform the tags and the attributes as described. The transformed lines should be output to the console. The output should look like the following:

```
13 <span class="sentence">Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.</span>
 14 <span class="sentence">Now we are engaged in a great civil war, testing whether that nation, or any nation, so conceived and so dedicated, can long endure.</span>
 15 <span class="sentence">We are met on a great battle-field of that war.</span>
 16 <span class="sentence">We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live.</span>
 17 <span class="sentence">It is altogether fitting and proper that we should do this.</span>
 21 <span class="sentence">But, in a larger sense, we can not dedicate, we can not consecrate, we can not hallow this ground.</span>
 …
```

One approach to the problem could be to break the algorithm into three steps.

1. Break the line into three parts: the start tag, the content, and the end tag.
2. Replace the current tags with a new tag.
3. Replace the attribute value with a new attribute value.

Then return the newly formatted line.

The method signatures to replace the tag and attributes might look like this:

```
    public String replaceTag(String tag, String targetTag, String replaceTag){ }
    public String replaceAttribute(String tag, String attribute, String value){
```

Tasks

Open the `StringsPractice03` project and make the following changes. Please note that the code to read a file has been supplied for you.

1. Edit the `SearchReplace.java` file.
2. Create a Pattern object to match the entire line.

```
Pattern pattern1 = Pattern.compile("(<" + targetTag +
".*?>)(.*?)(</" + targetTag + ".*?>)");
```

3. As you loop through the file, do the following:
   - Create a Matcher to match the current line.

```
Matcher m = pattern1.matcher(line);
```

   - Execute the `find()` method to find a match. If there is a match, replace the start and end tags. Replace the attribute

```
if (m.find()) {
  String newStart = replaceTag(m.group(1), targetTag,
replaceTag);
  newStart = replaceAttribute(newStart, attribute, value);
  String newEnd = replaceTag(m.group(3), targetTag, replaceTag);

  String newLine = newStart + m.group(2) + newEnd;
  System.out.printf("%3d %s\n", c, newLine);
}
```

4. Create a method that will replace the contents of any tag.

```
public String replaceTag(String tag, String targetTag, String
replaceTag){
    Pattern p = Pattern.compile(targetTag);   // targetTag is
regex
    Matcher m = p.matcher(tag);   // tag is text to replace
    if (m.find()){
        return m.replaceFirst(replaceTag); // swap target with
replace
    }
    return targetTag;
}
```

5. Create a method that will replace a tag’s attribute.

```
public String replaceAttribute(String tag, String attribute,
String value){
    Pattern p = Pattern.compile(attribute + "=" + "\".*?\"");
    Matcher m = p.matcher(tag);  // tag is text to replace
    if (m.find()){
        return m.replaceFirst(attribute + "=" + "\"" + value +
"\"");
    }
    return tag;
}
```

6. Run the `SearchReplace.java` file and produce the output shown in the “Summary” section of this practice.