0 Introduction

§8.6, “Regular Expressions,” in the text introduced regular expressions in the matches, replaceAll, replaceFirst, and split methods in the String class. These methods are shorthand for using the methods in the Pattern and Matcher classes. Using the methods in the Pattern and Matcher classes gives you more options and flexibilities to process strings using regular expressions. These two classes are in the java.util.regex package. The supplement introduces the Pattern and Matcher classes.

1 The Pattern and Matcher Classes

The matches, replaceAll, replaceFirst, and split methods in the String class are shorthand for using the methods in the Pattern and Matcher classes. Using the methods in the Pattern and Matcher classes gives you more options and flexibilities to process strings using regular expressions. These two classes are in the java.util.regex package.

The Pattern class represents a compiled regular expression. To create a Pattern object using the static method Pattern.compile(regex). The resulting pattern can then be used to create a Matcher object that can match arbitrary character sequences against the regular expression. All of the state involved in performing a match resides in the matcher, so many matchers can share the same pattern.

For example, the following statements create a pattern for regular expression a*b in Line 1, create a matcher for string aaaaabbb using the pattern in Line 2, and invokes the Matcher’s matches method to check whether the string aaaaabbb matches the pattern.

```java
Pattern pattern = Pattern.compile("a*b*");
Matcher matcher = pattern.matcher("aaaaabbb");
boolean isMatched = matcher.matches();
```

If a regular expression is used only once, you can simply use the static Pattern.matches(regex, string) method, as follows:

```java
Pattern pattern = Pattern.compile("a*b*");
boolean isMatched = pattern.matches("aaaaabbb");
```
boolean isMatched = Pattern.matches("a*b*", "aaaabbb");

This is also the same as
"aaaabbb".matches("a*b*").

If a regular expression will be used more than once, it is more efficient to first create a Pattern object for the regular expression once and reuse it to match various strings by creating Matcher objects. The Matcher class contains useful methods for matching whole string, replacing, and locating substrings.

<table>
<thead>
<tr>
<th>java.util.regex.Matcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>+find(): boolean</td>
</tr>
<tr>
<td>+find(index: int): boolean</td>
</tr>
<tr>
<td>+group(): String</td>
</tr>
<tr>
<td>+matches(): boolean</td>
</tr>
<tr>
<td>+replaceAll(regex: String, replacement: String): String</td>
</tr>
<tr>
<td>+replaceFirst(regex: String, replacement: String): String</td>
</tr>
</tbody>
</table>

Finds the next subsequence of the input that matches the pattern.
Resets this matcher and finds the next subsequence of the input sequence that matches the pattern, starting at the specified index.
Returns the previous matched subsequence.
Matches the entire string with the pattern.
Returns a new string that replaces every matching substring with the replacement.
Returns a new string that replaces the first matching substring with the replacement.

Figure 1.1
The Matcher class contains the methods for matching whole string, replacing, and locating substrings.

Listing 1.1 presents an example that uses the Pattern and Matcher classes to process strings using regular expressions. Line 5 creates a Pattern object and Line 9 creates a Matcher object. The find() method in Line 11 finds the next matching subsequence for the pattern and the group() method Line 12 returns previous matched subsequence. Figure 1.2 shows the output of the program.

Listing 1.1 RegexDemo.java (Pattern and Matcher Classes)

***PD: Please add line numbers (including space lines) in the following code***

***Layout: Please layout exactly. Don’t skip the space. AU.***

.Side Remark on Line 5: create a Pattern
.Side Remark on Line 9: create a Matcher
.Side Remark on Line 11: find a match
.Side Remark on Line 12: matched substring

```
import java.util.regex.*;

public class RegexDemo {
    public static void main(String[] args) {
        Pattern pattern = Pattern.compile("[\d]{3}-[\d]{2}-[\d]{4}"),
        String string = "George's social is 223-34-34343" +
                        "Peter's social is 111-35-2222";
```
Matcher matcher = pattern.matcher(string);

while (matcher.find())
    System.out.println(matcher.group());

Figure 1.2
You can use the `Matcher` class to find all matched substrings.