Regular Expressions

COLLABORATORS			
	TITLE : Regular Expressions		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Andrew Butcher	09/15/10	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Some Examples

1.1 Simple Character Classes

Brackets denote a character class. Simply surrounding your class with brackets will match your input once. Supplying a + will signify that your class will be repeated one or more times.

```
[a]   Just a, once.

[a-z]+        Any lower case letter one or more times.

[a-zA-Z]+ Any upper or lower case letter one or more times.

[a-zA-Z0-9]+  Any upper or lowercase letters or numbers one or more times.
```

1.2 Matching Strings

```
word         This will match the word "word"

(word)               Allows you to match "word" as well but placing it
                     in parens will allow you to reference it like
                     this...

(word)\1       Will match "wordword" so you could do this if you were matching  ↩
   html
                     tags

<(html)>.*</\1>      Will allow you to match the <html> and it's closing tag </ ↩
   html>.
```

1.3 The Kleene Star or Wildcard *

The wildcard * means that the class will be repeated zero or more times.

The . stands for absolutely anything, once.

```
[a-z]*[0-9]+       Zero or more a-z characters followed by one or more numbers.

.*         Meaning anything at all zero or more times.

.at                  This will match hat, cat, bat, fat, etc.
```

1.4 Searching At the Beginning or End of Strings

A caret ^ means match this class at the beginning of my string. The ^ must preceed your class.

A dollar sign $ means match this class at the end of my string. The $ must follow your class.

```
^[A-Z].*      Caret means 'begins with' so... any upper case letter followed by
                    anything zero or more times.

.*;$        Anything ending in a semicolon.
```

1.5 Logical Expressions

Placing a bar | in your expression will signify a logical OR.

```
([a-b]+|[0-1]+)     One or more a-b's OR one or more 1-0's.
```

1.6 Practical Applications

1.6.1 Matching an IP Address

This regular expression will match an IP Address. It's not a valid IP Address but it looks like one, at least.

```
([0-9]{1,3}[.]){3}([0-9]{1,3})
```

Curly braces signify the amount of times this class will occur. Also note that you have to wrap special character such as the period in braches [.] although you could also terminate them with a back-slash like, \:

1.6.2 Matching a MAC Address

See if you can do this yourself, using the IP Address example. Valid MAC Addresses include:

```
C0:FF:EE:BA:BE:EE
DE:AD:BE:EF:BA:BE
```

1.6.3 sed: Stream Editor

sed stands for stream editor and allows you to apply regular expressions to files to make quick fixes.

```
[23:36]abutcher@beta:~$ echo "biscuit mix" | sed 's/biscuit/beef/'
beef mix

[23:36]abutcher@beta:~$ echo "     <---- Notice the leading whitespace." | sed -r  ↩
   's/^[ ]+//'
<---- Notice the leading whitespace.

abutcher@shell003:~$ echo "     <---- Notice the leading whitespace." > whitespace ↩
   .txt

abutcher@shell003:~$ cat whitespace.txt
     <---- Notice the leading whitespace.

abutcher@shell003:~$ sed -r 's/^[ ]+//' whitespace.txt
<---- Notice the leading whitespace.

abutcher@shell003:~$ sed -ri 's/^[ ]+//' whitespace.txt

abutcher@shell003:~$ cat whitespace.txt
<---- Notice the leading whitespace.
```

Chapter 2

Regular Expressions in Java

2.1 RegexTest.java

```
package edu.wvu.mix.abutcher;

import java.io.IOException;
import java.util.regex.Pattern;

import jline.ConsoleReader;

public class RegexTest {
    public static void main(String[] args) throws IOException {

        ConsoleReader reader = new ConsoleReader();

        String line;
        while ((line = reader.readLine("$> ")) != null) {

            if (Pattern.matches(".*;$", line)) {
                System.out.println("\'" + line + "\' ends in a semicolon.");
            }
            if (Pattern.matches("([0-9]{1,3}[.]){3}([0-9]{1,3})", line)) {
                System.out.println("\'" + line + "\' looks like an IP address.");
            }
            if (Pattern.matches("^[A-Z].*", line) ) {
                System.out.println("\'" + line + "\' begins with a capital letter ↩
                    .");
            }
        }
    }
}
```