# Regular Expressions - Examples

Week 14 – Presentation 5

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

public class PatternQuoteExample {
    public static void main (String[] args) {
        String input = "Math operators: +-*/. ";
        boolean result;

        String quoted = Pattern.quote("+-*/.");
        System.out.println(quoted);

        // regex using standard escaping
        result = input.matches(".*\\s+\\+-\\*/\\.\\s+.*");
        System.out.println(result);

        // regex Using Pattern.quote around our search string
        result = input.matches(".*\\s+" + quoted + "\\s+.*");
        System.out.println(result);

        // regex Using \Q and \E around our search string
        result = input.matches(".*\\s+\\Q+-*/.\\E\\s+.*");
        System.out.println(result);
    }
}
```

```
\Q+-*/.\E
true
true
true
```

```java
import java.util.*;
import java.util.regex.*;
public class PatternSplitExample {
    public static void main (String[] args) {
        final String input = "value1||value2||value3";
        final Pattern p = Pattern.compile( Pattern.quote( "||" ) );

        // call split and print each element from generated array
        // using stream API
        Arrays.stream( p.split(input) )  // p.splitAsStream(input)
              .forEach( System.out::println );
    }
}
```

```
H:\work\CS209A_19S\notes08>javac PatternSplitExample.java

H:\work\CS209A_19S\notes08>java PatternSplitExample
value1
value2
value3
```

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

public class AsPredicateExample {
   public static void main (String[] args) {
      final String[] monthsArr =
         { "10", "0", "05", "09", "12", "15", "00", "-1", "100" };
      final Pattern validMonthPattern =
         Pattern.compile( "^(?:0?[1-9]|1[0-2])$" );
      List<String> filteredMonths =
         Stream.of( monthsArr )
               .filter( validMonthPattern.asPredicate() )
               .collect( Collectors.toList() );
      System.out.println( filteredMonths );
   }
}
```

```
H:\work\CS209A_19S\notes08>javac AsPredicateExample.java

H:\work\CS209A_19S\notes08>java AsPredicateExample
[10, 05, 09, 12]
```

```java
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;
import java.util.regex.Pattern;
import java.util.stream.Collectors;

public class RegexPredicateExample {
    public static void main(String[] args) {
        // Compile regex as predicate
        Predicate<String> emailFilter =
            Pattern.compile("^(.+)@example.com$")
                   .asPredicate();

        // Input list
        List<String> emails = Arrays.asList(
            "alex@example.com", "bob@yahoo.com",
            "cat@google.com",   "david@example.com"
        );

        // Apply predicate filter
        List<String> desiredEmails =
            emails.stream()
                  .filter(emailFilter)
                  .collect(Collectors.<String>toList());

        // Now perform desired operation
        desiredEmails.forEach(System.out::println);
    }
}
```

```
alex@example.com
david@example.com
```

```
H:\work\CS209A_19S\notes08>java MatcherMatchesExample
[mastering regular expressions] => [mastering]: false
[mastering regular expressions] => [mastering.*]: true
[mastering regular expressions] => [regular.*]: false
```

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

public class MatcherMatchesExample {
   public static void main (String[] args) {
      final Pattern pattern1 = Pattern.compile( "mastering" );
      final Pattern pattern2 = Pattern.compile( "mastering.*" );
      final Pattern pattern3 = Pattern.compile( "regular.*" );

      String input = "mastering regular expressions";
      Matcher matcher = pattern1.matcher(input);
      System.out.printf( "[%s] => [%s]: %s%n",
         input, matcher.pattern(), matcher.matches());

      // update the matcher pattern with a new pattern
      matcher.usePattern(pattern2);
      System.out.printf( "[%s] => [%s]: %s%n",
         input, matcher.pattern(), matcher.matches());

      // update the matcher pattern with a new pattern
      matcher.usePattern(pattern3);
      System.out.printf( "[%s] => [%s]: %s%n",
         input, matcher.pattern(), matcher.matches());
   }
}
```

```
[1] => [value1]
[1] => [value2]
[1] => [value3]
[1] => [value1]
[1] => [value2]
[1] => [value3]
```

```java
import java.util.regex.*;
public class MatcherFindExample {
   public static void main (String[] args) {
      final String input =
         "some text <value1> anything <value2><value3> here";

      /* Part 1 */
      final Pattern pattern = Pattern.compile( "<([^<>]*)>" );
      Matcher matcher = pattern.matcher(input);
      while (matcher.find()) {
         System.out.printf( "[%d] => [%s]%n",
            matcher.groupCount(), matcher.group(1) );
      }

      /* Part 2 */
      // now use similar pattern but use a named group and reset the
      // matcher
      matcher.usePattern( Pattern.compile( "<(?<name>[^<>]*)>" ) );
      matcher.reset();
      while (matcher.find()) {
         System.out.printf( "[%d] => [%s]%n",
            matcher.groupCount(), matcher.group("name"));
      }
   }
}
```

```
H:\work\CS209A_19S\notes08>java MatcherAppendExample
<v1=n1 v2=n2 v3=n3> n1=v1 n2=v2 abc=123 <pq=v abc=id> v=pq
```

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

public class MatcherAppendExample {
   public static void main (String[] args) {
      final String input =
         "<n1=v1 n2=v2 n3=v3> n1=v1 n2=v2 abc=123 <v=pq id=abc> v=pq";

      // pattern1 to find all matches between < and >
      final Pattern pattern = Pattern.compile( "<[^>]+>" );

      // pattern1 to find each name=value pair
      final Pattern pairPattern = Pattern.compile( "(\\w+)=(\\w+)" );
      Matcher enclosedPairs = pattern.matcher(input);

      StringBuffer sbuf = new StringBuffer();
      // call find in a loop and call appendReplacement for each match
      while (enclosedPairs.find()) {
         Matcher pairMatcher = pairPattern.matcher( enclosedPairs.group());
         // replace name=value with value=name in each match
         enclosedPairs.appendReplacement(
            sbuf, pairMatcher.replaceAll( "$2=$1" )
         );
      }
      // appendTail to append remaining character to buffer
      enclosedPairs.appendTail( sbuf );
      System.out.println( sbuf );
   }
}
```

```
java.net.URL  url = new URL( sURL );
```

```
106     /**
107      * Initializes an input stream from a URL.
108      *
109      * @param  url the URL
110      * @throws IllegalArgumentException if cannot open {@code url}
111      * @throws IllegalArgumentException if {@code url} is {@code null}
112      */
113     public In(URL url) {
114         if (url == null) throw new IllegalArgumentException("url argument is null");
115         try {
116             URLConnection site = url.openConnection();
117             InputStream is     = site.getInputStream();
118             scanner            = new Scanner(new BufferedInputStream(is), CHARSET_NAME);
119             scanner.useLocale(LOCALE);
120         }
121         catch (IOException ioe) {
122             throw new IllegalArgumentException("Could not open " + url, ioe);
123         }
124     }
```