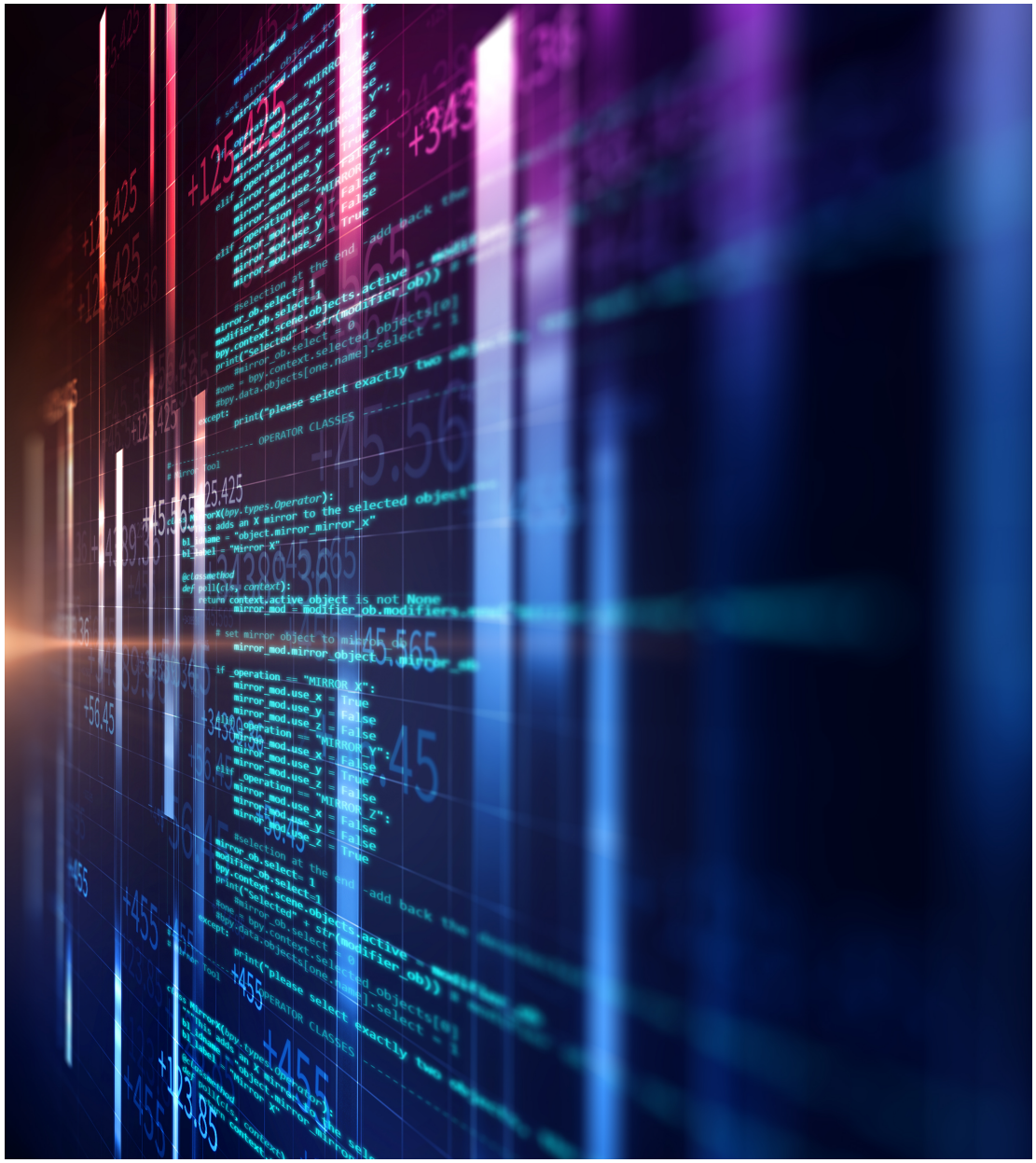
Regular Expressions

Brandon Krakowsky

Penn Engineering



Regular Expressions

Regular Expressions

- A regular expression (or regex) is a special sequence of characters that describes a pattern used for *searching*, *editing*, and *manipulating* text and data
- For example, regular expressions are widely used to define the constraint on Strings in *password* and *email validation*
- The most basic regular expression consists of a literal String that matches the first occurrence of that String

About Regex: https://www.regular-expressions.info/quickstart.html

Java Specific Tutorials: https://docs.oracle.com/javase/tutorial/essential/regex/

Regex Examples

RegexClass

RegexClass.java

```java
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * Contains various methods for parsing Strings based on regular expressions.
 * @author brandonkrakowsky
 *
 */
public class RegexClass {

    /**
     * Prints an array of tokens (Strings).
     * @param arr of tokens to print
     */
    public static void printTokens(String[] arr) {
        System.out.println("Printing tokens:");
        for (String s : arr) {
            System.out.print(s + " ");
        }
        System.out.println("\n");
    }

```

Split a String

```
22
23      /**
24       * Splits given string based on given regex pattern.
25       * @param str to split
26       * @param regex to match
27       * @return String array of tokens (Strings)
28       */
29      public static String[] splitString(String str, String regex) {
30          //split the given string str based on the given regex
31          return str.split(regex);
32      }
33
```

Split a String

```
161
162     public static void main(String[] args) {
163
164         String str = "the cow jumped over the moon";
165         //split the String based on a single space
166         String[] tokens = RegexClass.splitString(str, " ");
167         RegexClass.printTokens(tokens);
168
169         //split the String based on "the"
170         tokens = RegexClass.splitString(str, "the");
171         RegexClass.printTokens(tokens);
172
```

Split a String

```
186
187         /*
188          * split the String based on various amounts of whitespace
189          * \s matches a single whitespace character
190          * \s+ matches 1 or more whitespace characters
191          */
192
193         /*
194          * \ (backslash) is a special character.
195          * if you want to use it as a literal in a regex,
196          * you need to escape it with another backslash,
197          * so we use \\s+ to match 1 or more whitespace characters
198          */
199         str = "the              cow     jumped over                 the\n"
200                 + "   moon";
201         tokens = RegexClass.splitString(str, "\\s+");
202         RegexClass.printTokens(tokens);
203
```

Replace All with a Pattern

```
34    /**
35     * Replaces all instances of the given pattern
36     * with the given replacement in the given str.
37     * @param str to replace values in
38     * @param pattern to replace
39     * @param replace updated value
40     * @return Updated str
41     */
42    public static String replaceAllWithPattern(String str, String pattern, String replacement) {
43        //replace the given pattern with the given replacement in str
44        return str.replaceAll(pattern, replacement);
45    }
```

Replace All with a Pattern

```
189
190         //replace multiple whitespace characters with a single whitespace character
191         String updatedStr = RegexClass.replaceAllWithPattern(str, "\\s+", " ");
192         System.out.println("Replace whitespace: " + updatedStr);
193         System.out.println("");
194
```

Get Parts of a Phone Number

```java
47    /**
48     * Parses and returns various part of a phone number.
49     * @param phone number to parse
50     * @param part of phone number to return: 1 (area code), 2 (prefix) or 3 (number)
51     * @return Part of phone number
52     */
53    public static String getPhonePart(String phone, int part) {
54        if (part < 1 || part > 3) {
55            throw new IllegalArgumentException("Part must be 1, 2 or 3.");
56        }
57
58        //parenthesis() indicate groups
59        //\b matches an empty string or non-word character,
60        //at the beginning or end of pattern
61
62        //[-.\\s]+ indicates a character class,
63        //matching one of several characters (with repetition): -, ., whitespace
64        String regex = "\\b(\\d{3})[-.\\s]+(\\d{3})[-.\\s]+(\\d{4})\\b";
65
66        Pattern p = Pattern.compile(regex);
67        Matcher m = p.matcher(phone);
68
69        String phonePart = "";
70        while (m.find()) {
71            //get designated group
72            phonePart = m.group(part);
73        }
74
75        //return group
76        return phonePart;
77    }
```

Get Parts of a Phone Number

```
189
190         //get parts of phone number
191         String areaCode = RegexClass.getPhonePart("123-982-6342", 1); //get area code
192         String prefix = RegexClass.getPhonePart("800 787   2394", 2); //get prefix
193         String number = RegexClass.getPhonePart(" 508.717.0989   ", 3); //get line number
194         System.out.println("Phone number parts: " + areaCode + " " + prefix + " " + number);
195         System.out.println("");
196
```

Replace an Area Code

```
77      /**
78       * Replaces the area code in the given phone number with the given new area code.
79       * @param phone to replace area code in
80       * @param newArea for phone
81       * @return Updated phone number
82       */
83      public static String replaceAreaCode(String phone, String newArea) {
84          //[0-9] indicates a character class,
85          //matching one of several characters: 0 - 9
86          //{3} indicates a specific amount of repetition
87          return phone.replaceFirst("[0-9]{3}", newArea);
88      }
```

Replace an Area Code

```
196
197         //replace area code
198         String phone = "123-982-6342";
199         String updatedPhone = RegexClass.replaceAreaCode(phone, "888");
200         System.out.println("Updated phone: " + updatedPhone);
201         System.out.println("");
202
```