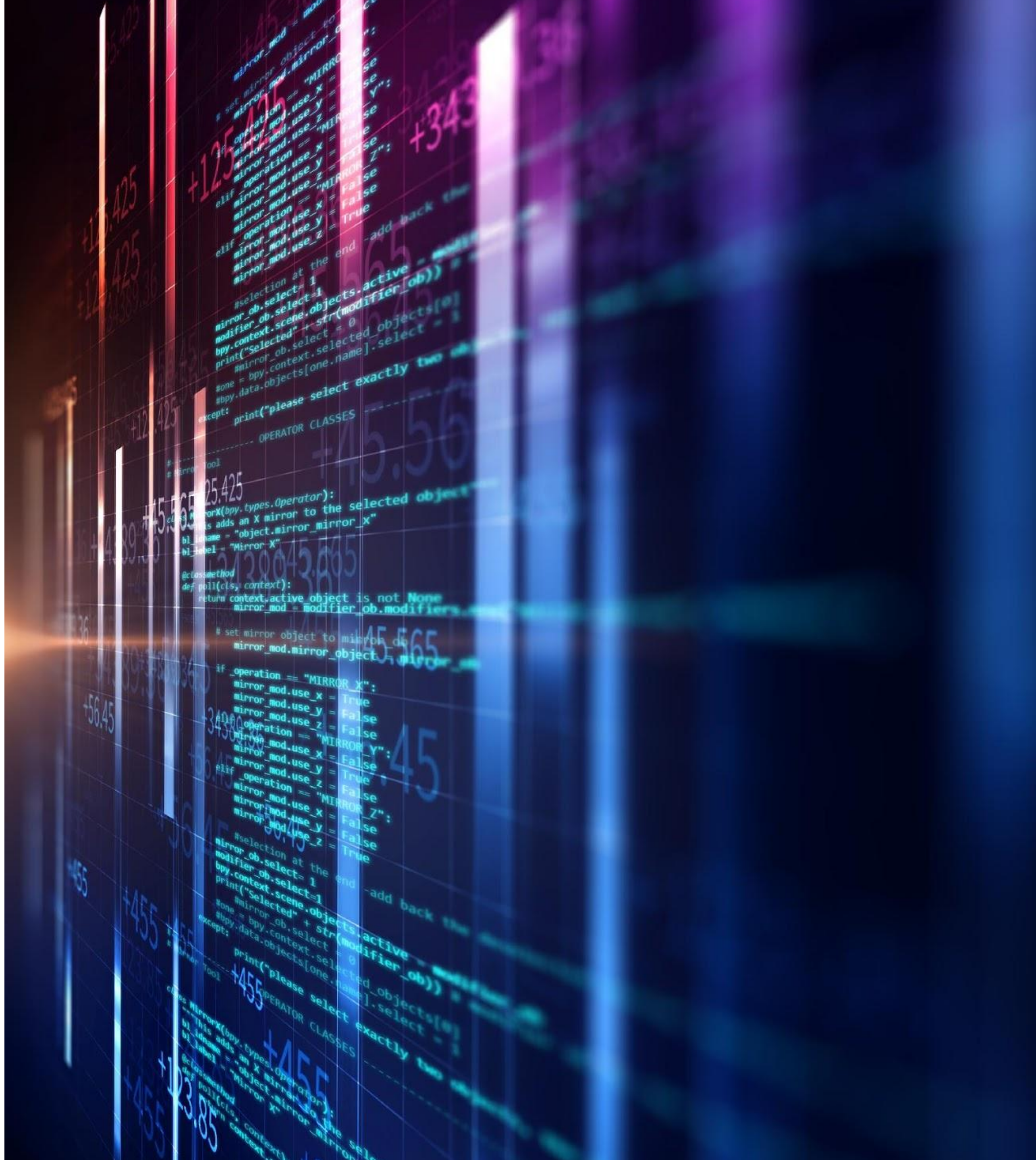
Regular Expressions

Brandon Krakowsky



Regular Expressions

Regular Expressions

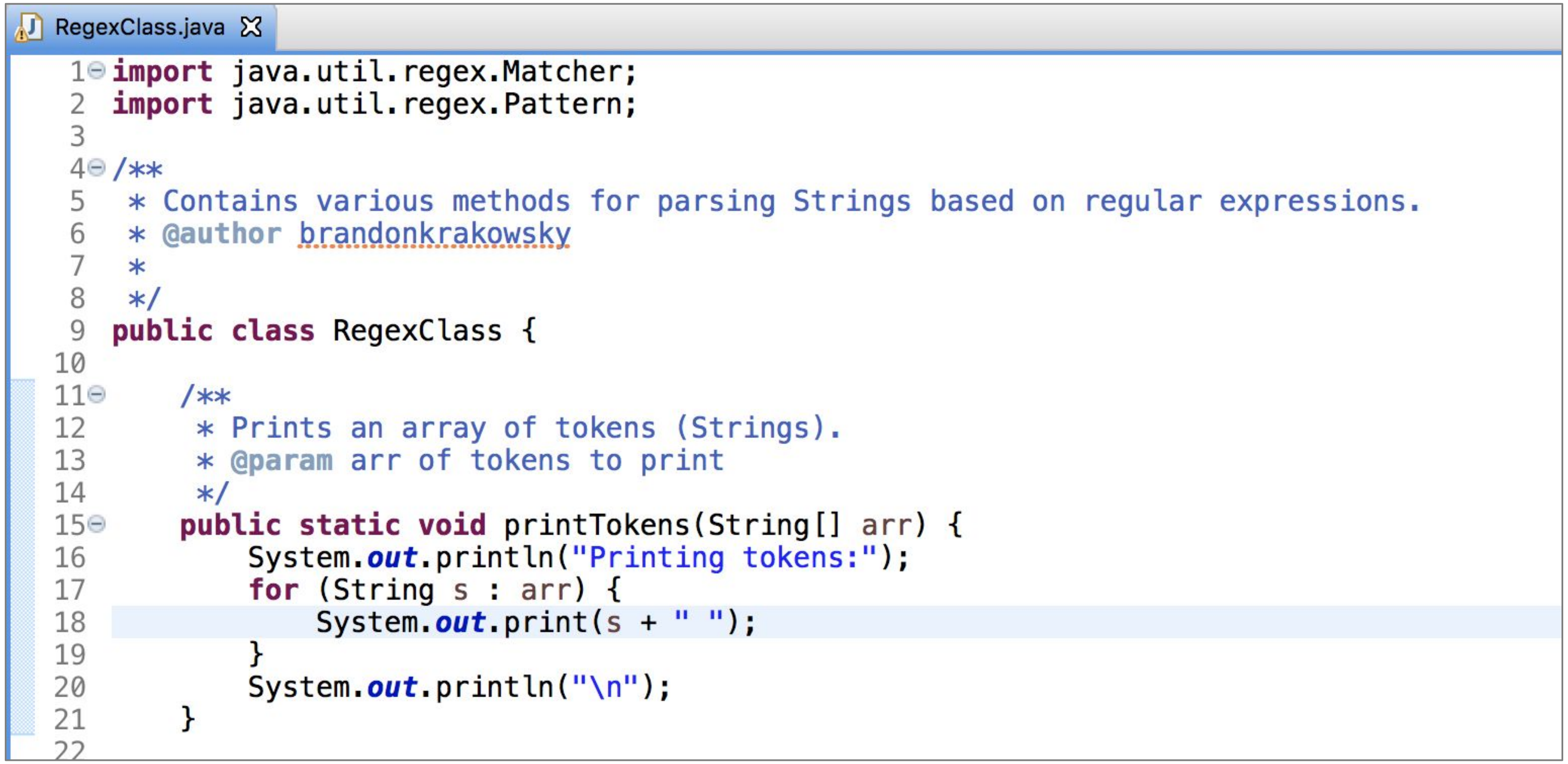
- A regular expression (or regex) is a special sequence of characters that describes a pattern used for *searching*, *editing*, and *manipulating* text and data
- For example, regular expressions are widely used to define the constraint on Strings in *password* and *email validation*
- The most basic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

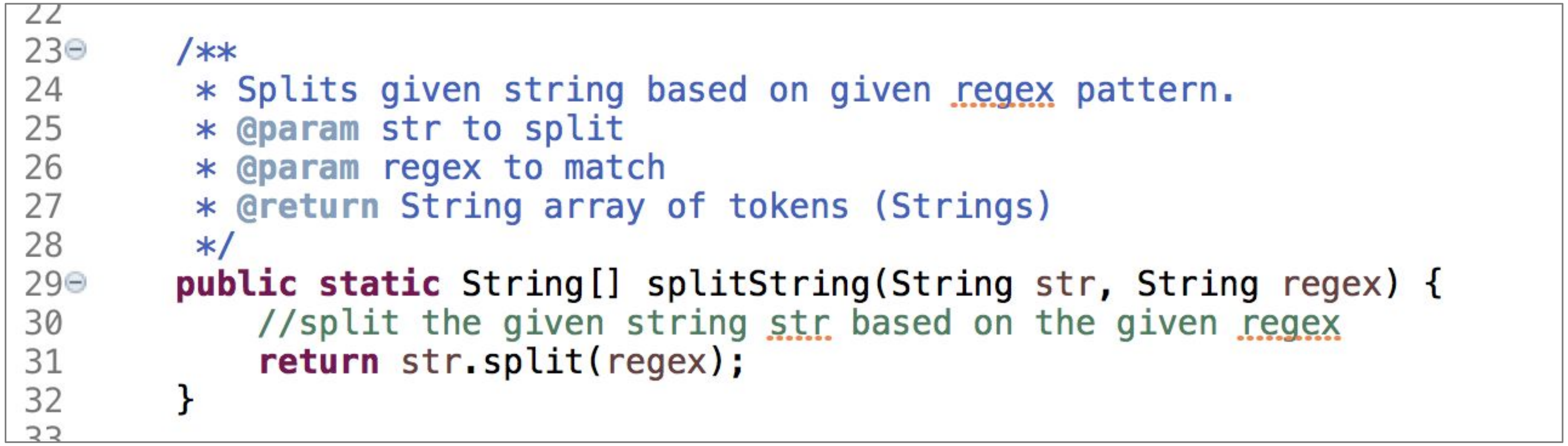
Regex Examples

RegexClass


RegexClass.java

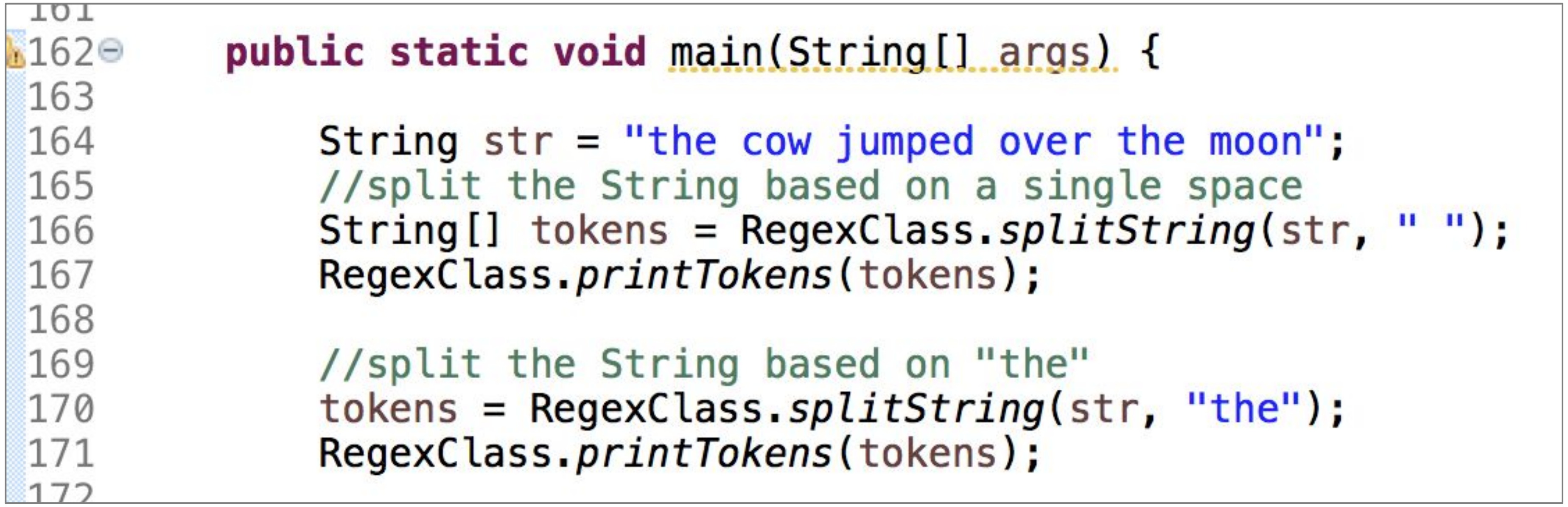
```java
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * 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

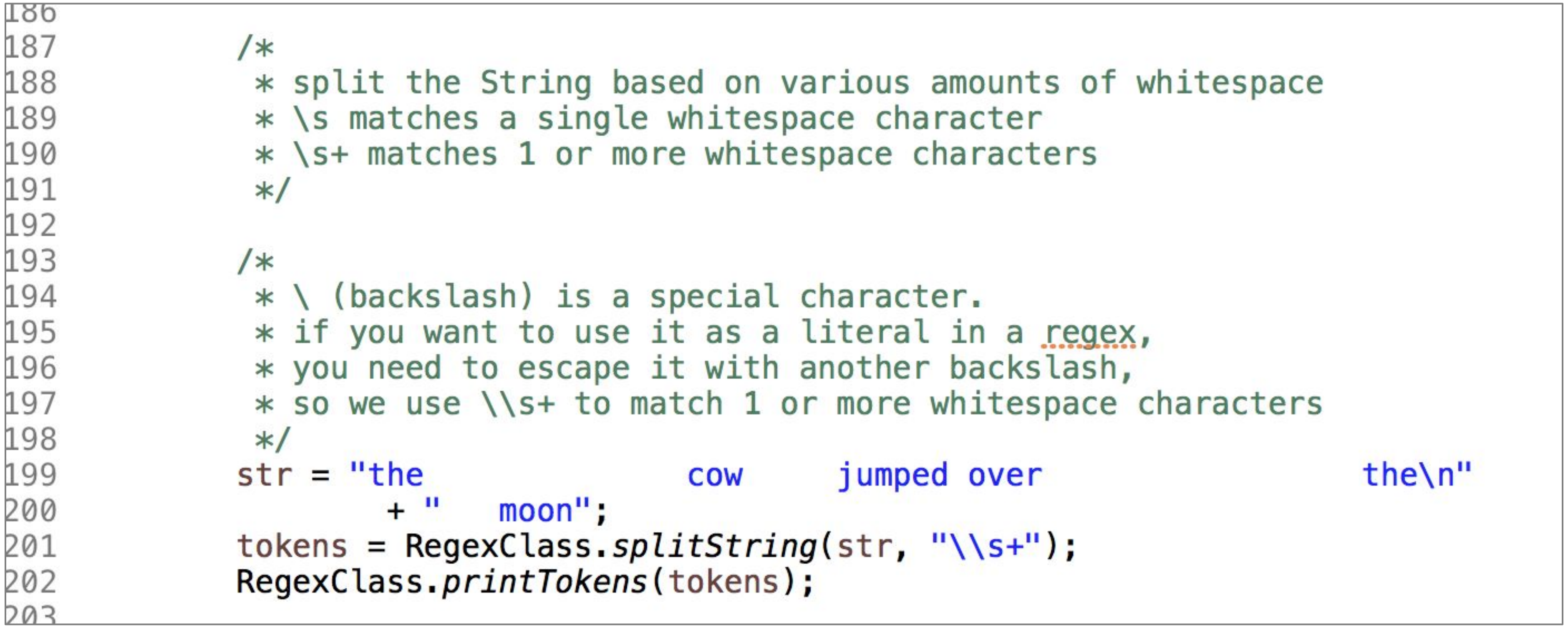


```
    /**
     * Splits given string based on given regex pattern.
     * @param str to split
     * @param regex to match
     * @return String array of tokens (Strings)
     */
    public static String[] splitString(String str, String regex) {
        //split the given string str based on the given regex
        return str.split(regex);
    }
```

Split a String

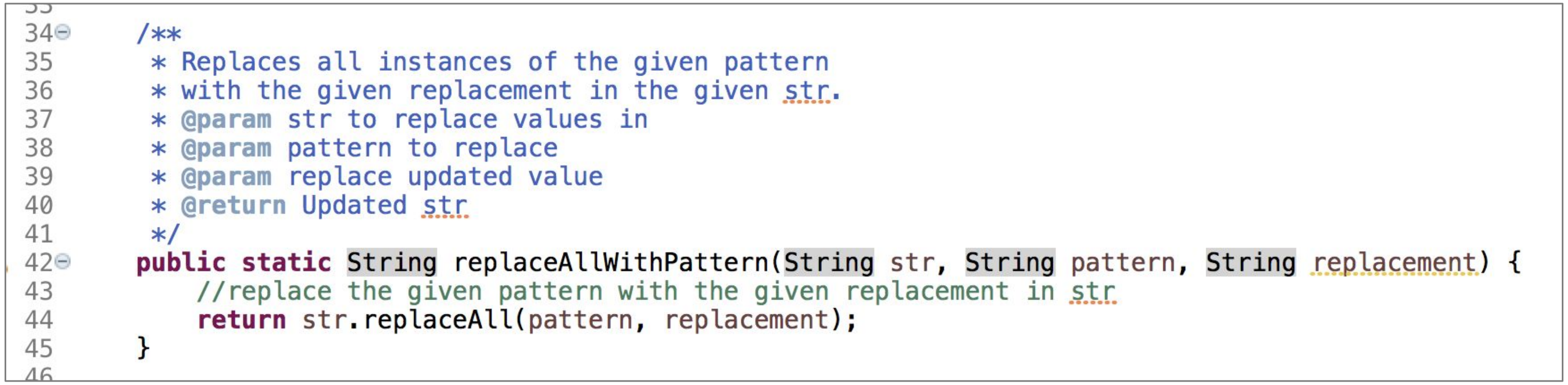

```
    public static void main(String[] args) {

        String str = "the cow jumped over the moon";
        //split the String based on a single space
        String[] tokens = RegexClass.splitString(str, " ");
        RegexClass.printTokens(tokens);

        //split the String based on "the"
        tokens = RegexClass.splitString(str, "the");
        RegexClass.printTokens(tokens);
```

Split a String



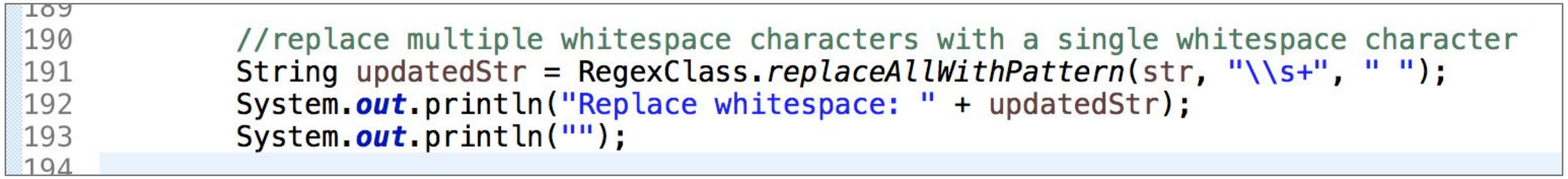
```
        /*
         * split the String based on various amounts of whitespace
         * \s matches a single whitespace character
         * \s+ matches 1 or more whitespace characters
         */

        /*
         * \ (backslash) is a special character.
         * if you want to use it as a literal in a regex,
         * you need to escape it with another backslash,
         * so we use \\s+ to match 1 or more whitespace characters
         */
        str = "the              cow     jumped over                 the\n"
                + "   moon";
        tokens = RegexClass.splitString(str, "\\s+");
        RegexClass.printTokens(tokens);
```

Replace All with a Pattern



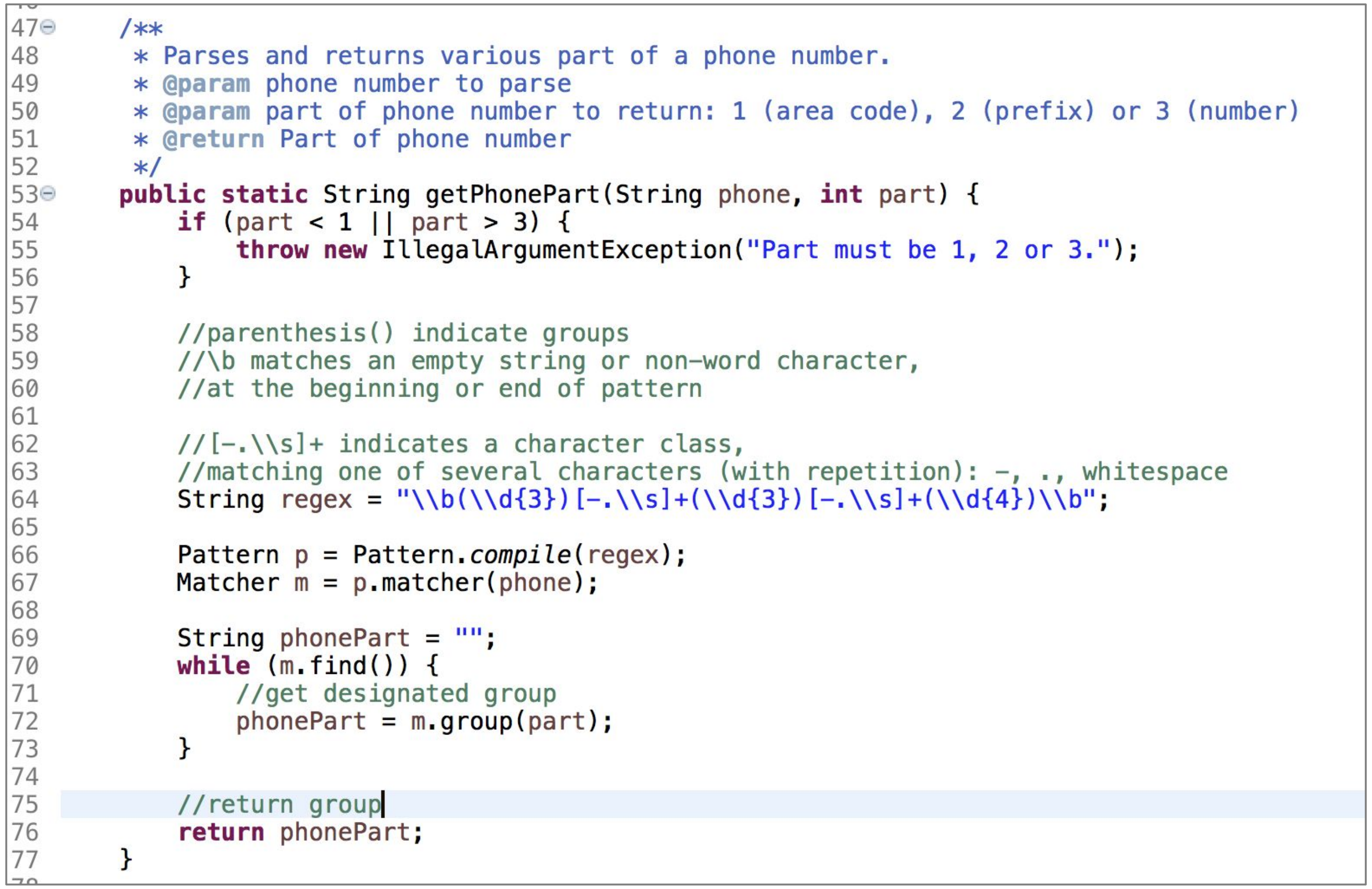
```
    /**
     * Replaces all instances of the given pattern
     * with the given replacement in the given str.
     * @param str to replace values in
     * @param pattern to replace
     * @param replace updated value
     * @return Updated str
     */
    public static String replaceAllWithPattern(String str, String pattern, String replacement) {
        //replace the given pattern with the given replacement in str
        return str.replaceAll(pattern, replacement);
    }
```

Replace All with a Pattern



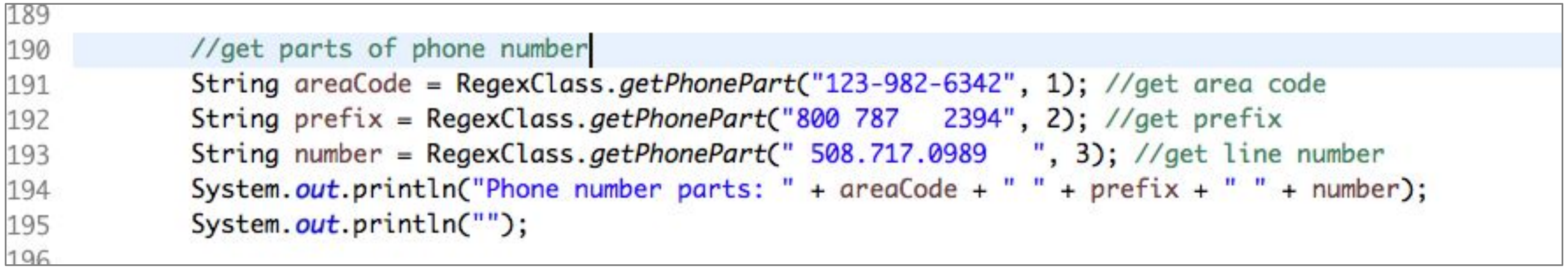
```java
        //replace multiple whitespace characters with a single whitespace character
        String updatedStr = RegexClass.replaceAllWithPattern(str, "\\s+", " ");
        System.out.println("Replace whitespace: " + updatedStr);
        System.out.println("");
```

Get Parts of a Phone Number



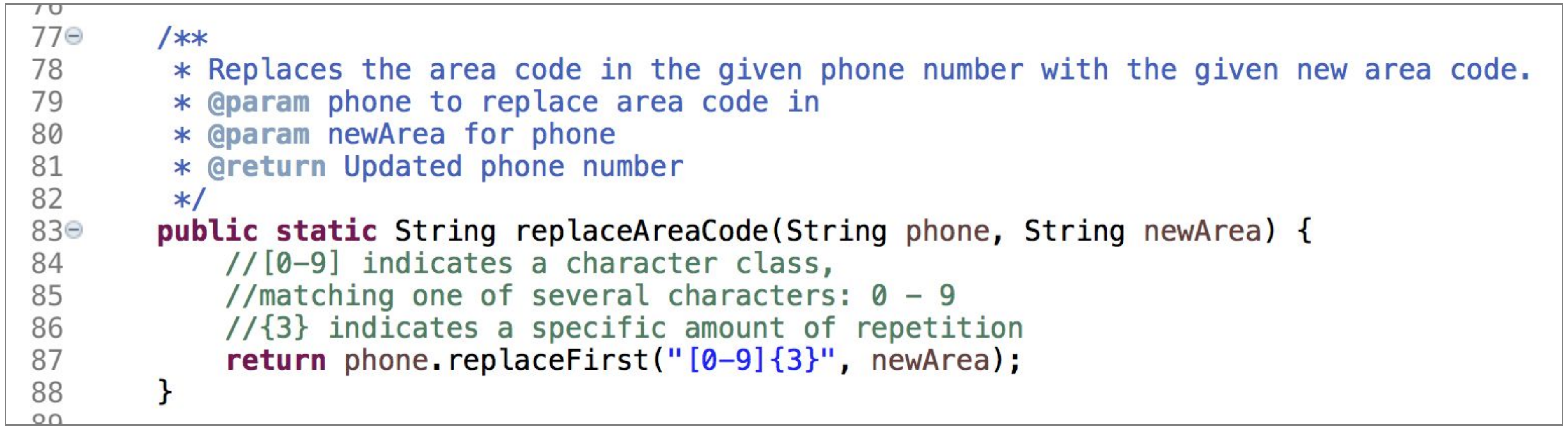
```java
    /**
     * Parses and returns various part of a phone number.
     * @param phone number to parse
     * @param part of phone number to return: 1 (area code), 2 (prefix) or 3 (number)
     * @return Part of phone number
     */
    public static String getPhonePart(String phone, int part) {
        if (part < 1 || part > 3) {
            throw new IllegalArgumentException("Part must be 1, 2 or 3.");
        }

        //parenthesis() indicate groups
        //\b matches an empty string or non-word character,
        //at the beginning or end of pattern

        //[-.\\s]+ indicates a character class,
        //matching one of several characters (with repetition): -, ., whitespace
        String regex = "\\b(\\d{3})[-.\\s]+(\\d{3})[-.\\s]+(\\d{4})\\b";

        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(phone);

        String phonePart = "";
        while (m.find()) {
            //get designated group
            phonePart = m.group(part);
        }

        //return group
        return phonePart;
    }
```

Get Parts of a Phone Number



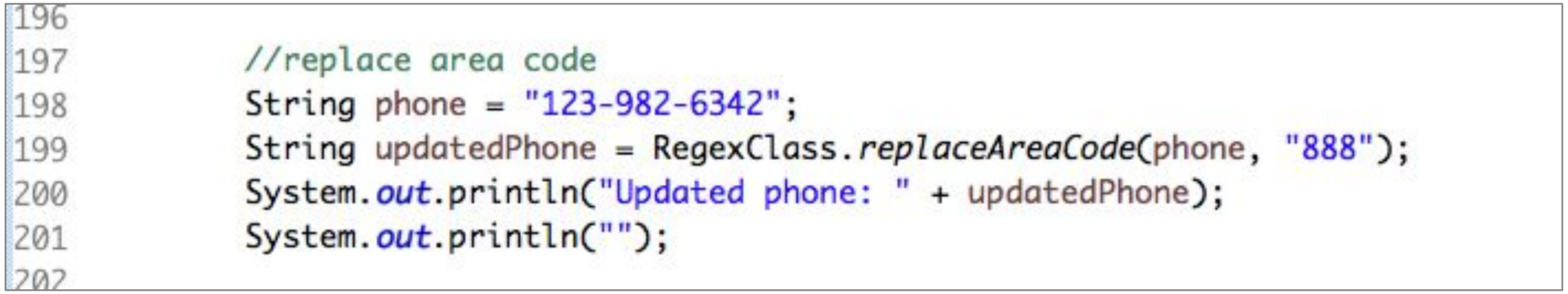
```

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

```

Replace an Area Code



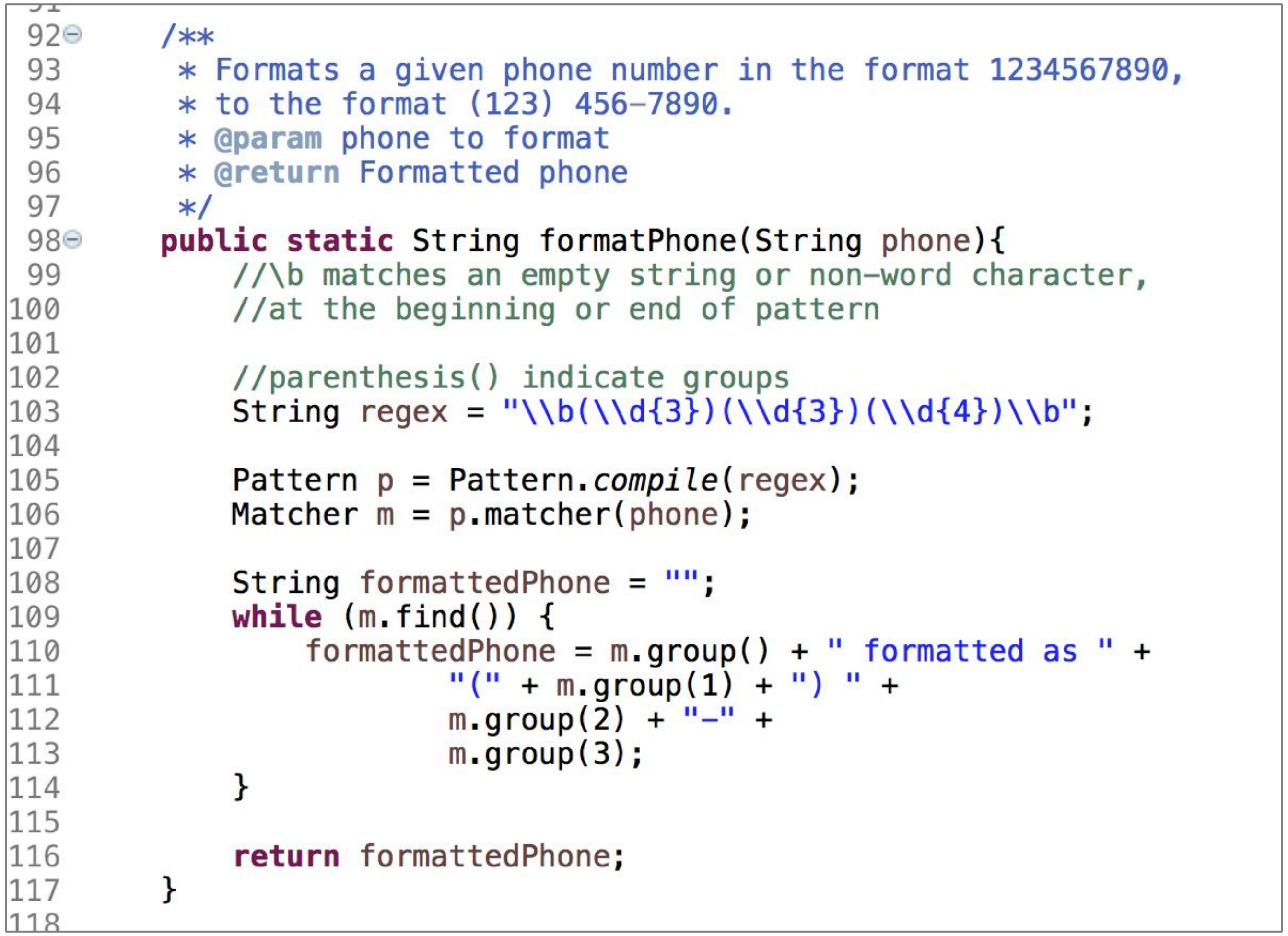
```java
    /**
     * Replaces the area code in the given phone number with the given new area code.
     * @param phone to replace area code in
     * @param newArea for phone
     * @return Updated phone number
     */
    public static String replaceAreaCode(String phone, String newArea) {
     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

Replace an Area Code



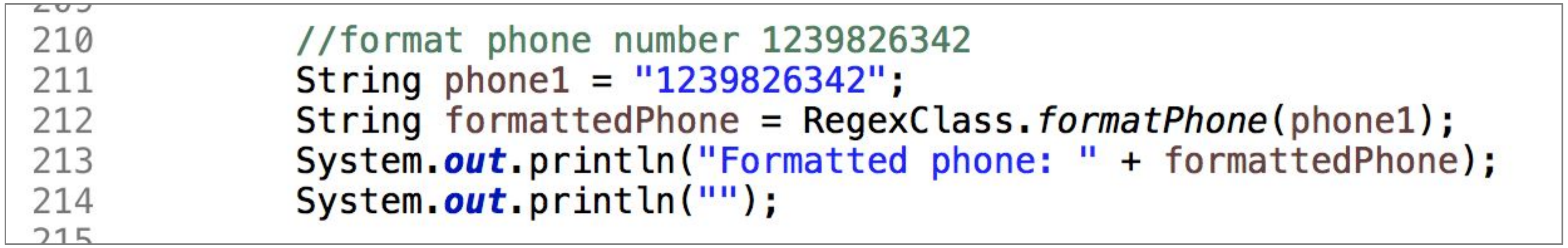
```
        //replace area code
        String phone = "123-982-6342";
        String updatedPhone = RegexClass.replaceAreaCode(phone, "888");
        System.out.println("Updated phone: " + updatedPhone);
        System.out.println("");
```

Format a Phone Number



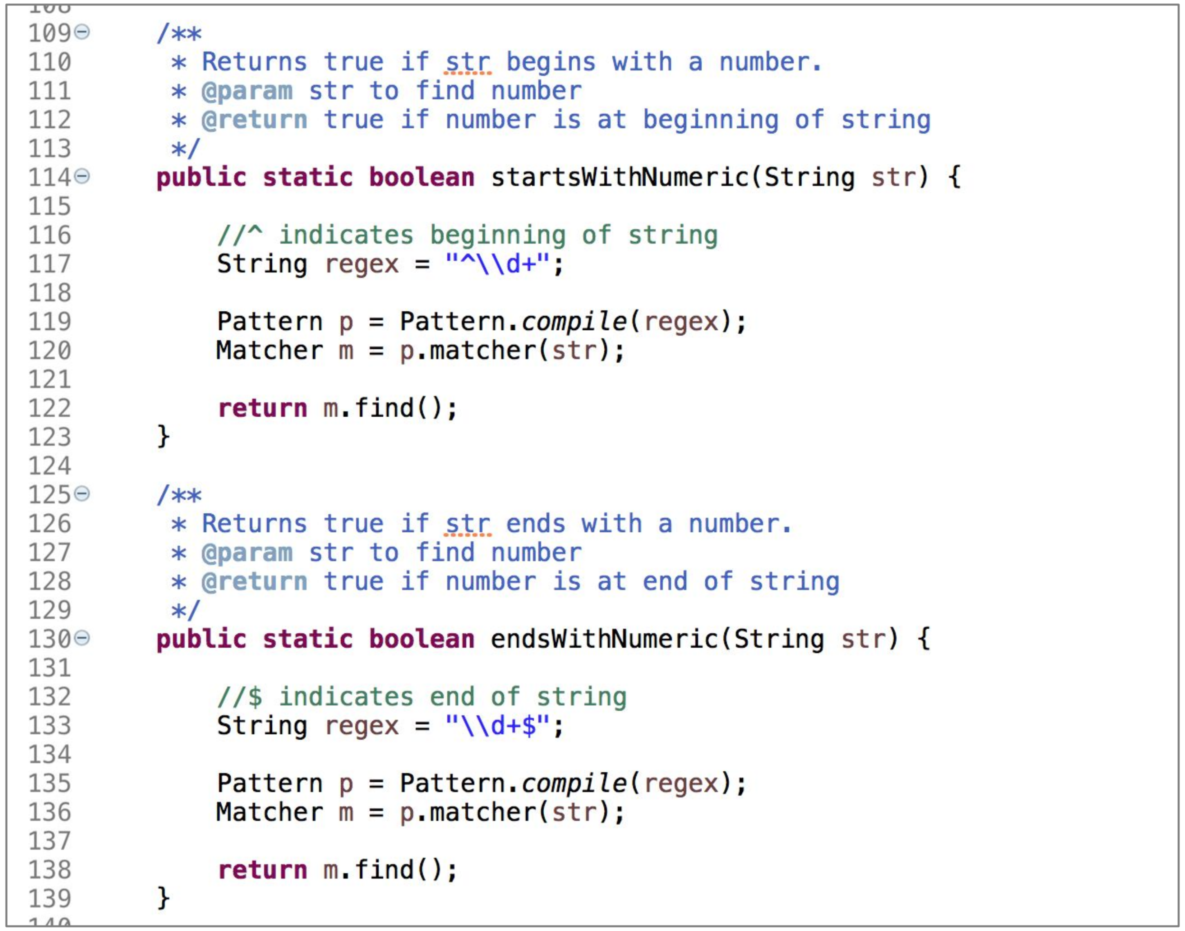
```
    /**
     * Formats a given phone number in the format 1234567890,
     * to the format (123) 456-7890.
     * @param phone to format
     * @return Formatted phone
     */
    public static String formatPhone(String phone){
        //\b matches an empty string or non-word character,
        //at the beginning or end of pattern

        //parenthesis() indicate groups
        String regex = "\\b(\\d{3})(\\d{3})(\\d{4})\\b";

        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(phone);

        String formattedPhone = "";
        while (m.find()) {
            formattedPhone = m.group() + " formatted as " +
                    "(" + m.group(1) + ") " +
                    m.group(2) + "-" +
                    m.group(3);
        }

        return formattedPhone;
    }
```

Format a Phone Number



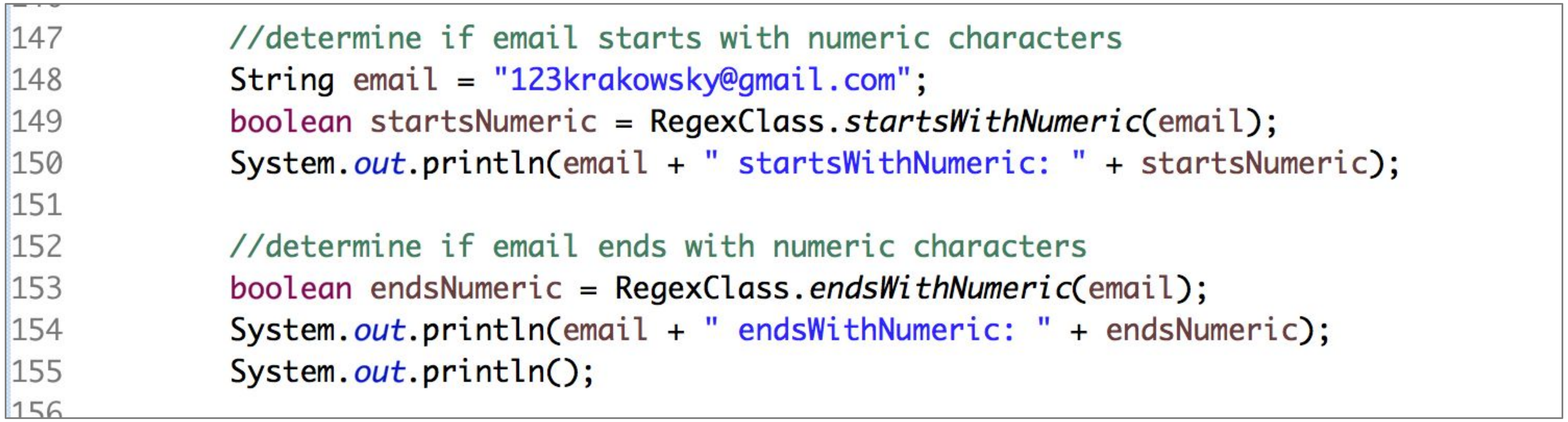
```
//format phone number 1239826342
String phone1 = "1239826342";
String formattedPhone = RegexClass.formatPhone(phone1);
System.out.println("Formatted phone: " + formattedPhone);
System.out.println("");
```

Starts/Ends with a Number



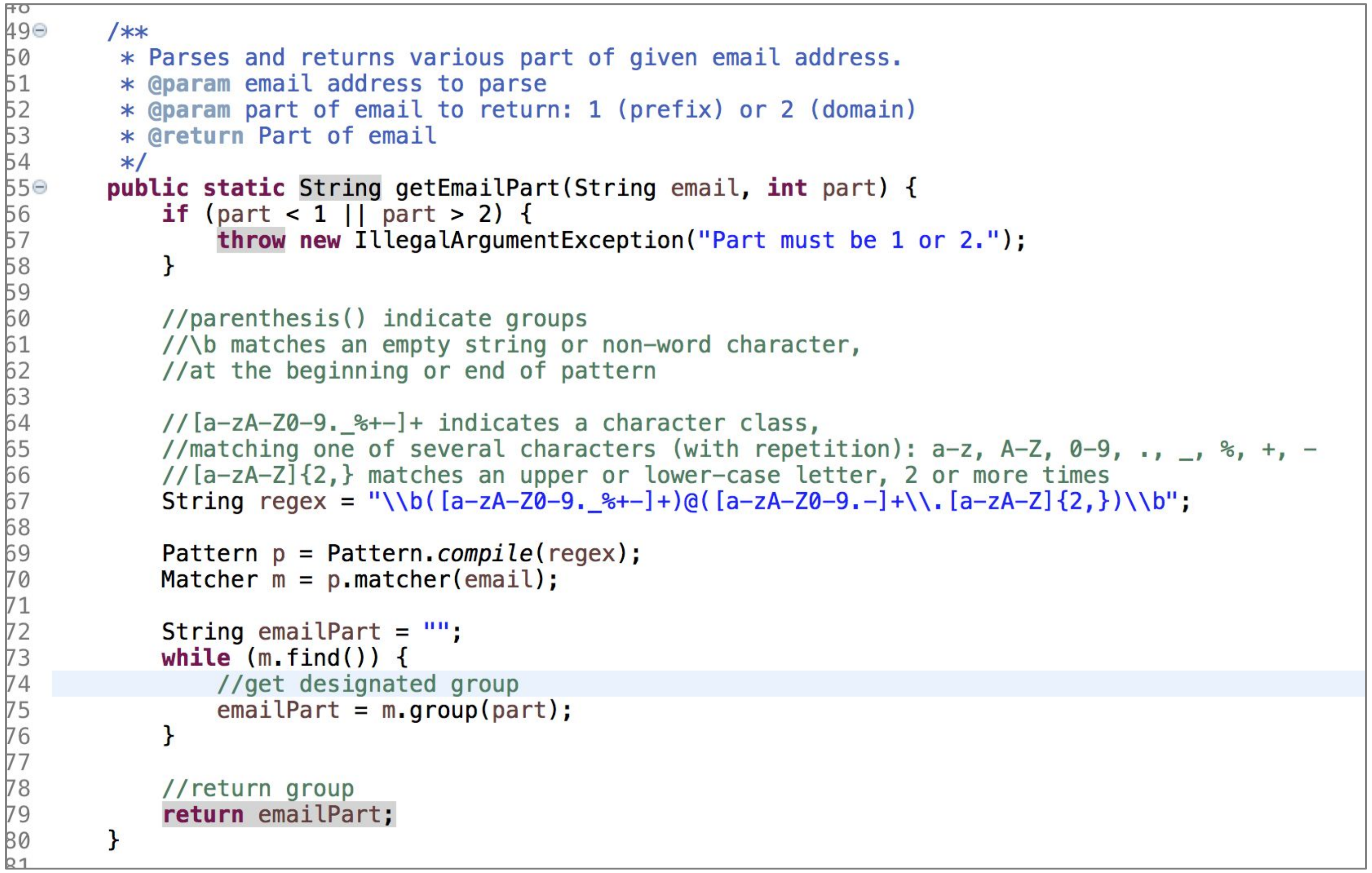
```
    /**
     * Returns true if str begins with a number.
     * @param str to find number
     * @return true if number is at beginning of string
     */
    public static boolean startsWithNumeric(String str) {

        //^ indicates beginning of string
        String regex = "^\\d+";

        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(str);

        return m.find();
    }

    /**
     * Returns true if str ends with a number.
     * @param str to find number
     * @return true if number is at end of string
     */
    public static boolean endsWithNumeric(String str) {

        //$ indicates end of string
        String regex = "\\d+$";

        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(str);

        return m.find();
    }
```

Starts/Ends with a Number



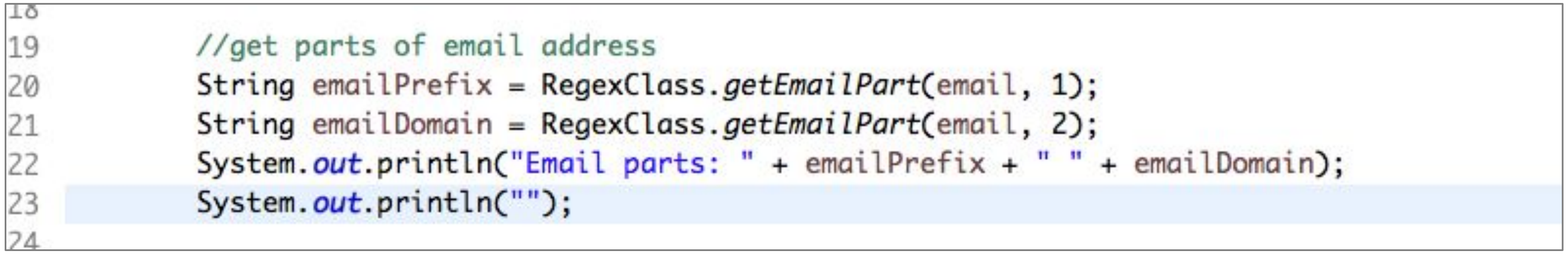
```
        //determine if email starts with numeric characters
        String email = "123krakowsky@gmail.com";
        boolean startsNumeric = RegexClass.startsWithNumeric(email);
        System.out.println(email + " startsWithNumeric: " + startsNumeric);

        //determine if email ends with numeric characters
        boolean endsNumeric = RegexClass.endsWithNumeric(email);
        System.out.println(email + " endsWithNumeric: " + endsNumeric);
        System.out.println();
```

Get Parts of an Email Address



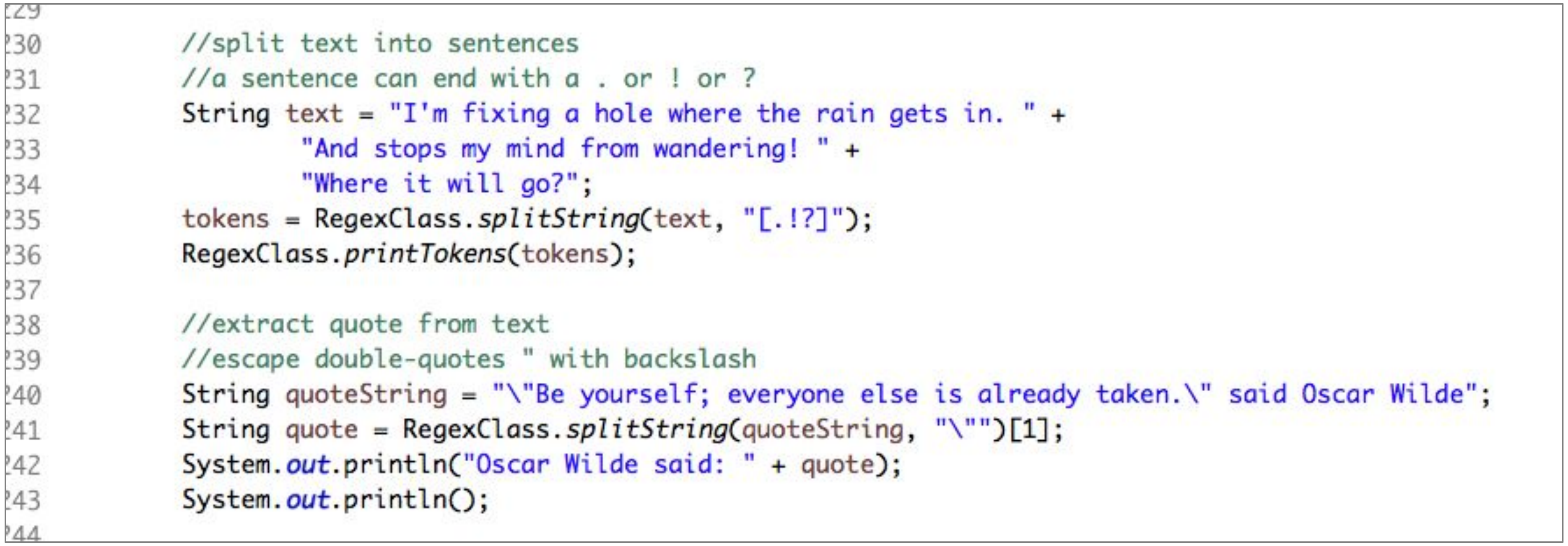
```java
    /**
     * Parses and returns various part of given email address.
     * @param email address to parse
     * @param part of email to return: 1 (prefix) or 2 (domain)
     * @return Part of email
     */
    public static String getEmailPart(String email, int part) {
        if (part < 1 || part > 2) {
            throw new IllegalArgumentException("Part must be 1 or 2.");
        }

        //parenthesis() indicate groups
        //\b matches an empty string or non-word character,
        //at the beginning or end of pattern

        //[a-zA-Z0-9._%+-]+ indicates a character class,
        //matching one of several characters (with repetition): a-z, A-Z, 0-9, ., _, %, +, -
        //[a-zA-Z]{2,} matches an upper or lower-case letter, 2 or more times
        String regex = "\\b([a-zA-Z0-9._%+-]+)@([a-zA-Z0-9.-]+\\.[a-zA-Z]{2,})\\b";

        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(email);

        String emailPart = "";
        while (m.find()) {
            //get designated group
            emailPart = m.group(part);
        }

        //return group
        return emailPart;
    }
```

Get Parts of an Email Address



```
        //get parts of email address
        String emailPrefix = RegexClass.getEmailPart(email, 1);
        String emailDomain = RegexClass.getEmailPart(email, 2);
        System.out.println("Email parts: " + emailPrefix + " " + emailDomain);
        System.out.println("");
```

Split String into Sentences & Extract Quote



```
        //split text into sentences
        //a sentence can end with a . or ! or ?
        String text = "I'm fixing a hole where the rain gets in. " +
                "And stops my mind from wandering! " +
                "Where it will go?";
        tokens = RegexClass.splitString(text, "[.!?]");
        RegexClass.printTokens(tokens);

        //extract quote from text
        //escape double-quotes " with backslash
        String quoteString = "\"Be yourself; everyone else is already taken.\" said Oscar Wilde";
        String quote = RegexClass.splitString(quoteString, "\"")[1];
        System.out.println("Oscar Wilde said: " + quote);
        System.out.println();
```