CIS 1068 Assignment 5
practice with static methods and strings

Due: February 15

70 points

description

The purpose of this assignment is to gain additional practice with sequences, functions, and testing.

import starter code and tests

Please download this zipped Eclipse project. Import it into Eclipse:

- Choose `File -> Import -> General -> Existing Projects into workspace.`
- Click `Next`
- Select archive file. Click the Browse button and navigate to the zip file you've just downloaded.
- Click Finish

Once you've done this, you'll find the project and its included source file `StringPractice.java`.

The contents:

```
public class StringPractice {
  /* returns true if c is a punctuation mark or false otherwise
   *
   * Punctuation mark is defined as:
   * apostrophe '
   * comma ,
   * period .
   * semicolon ;
   * colon :
   * exclamation point !
   * question mark ?
   *
   * (You don't have to worry about any others)
   */
  public static boolean isPunct(char c) {
    return true;
  }

  /*
   * returns the number of punctuation marks
   * found in s.
   *
   * call your isPunct( ) method. Do not compy and paste
   * the same logic into this function */
  public static int numPunct(String s) {
    return 0;
  }

  /*
   * returns the number of punctuation marks
   * found in s starting at the given index.
   *
   * call your isPunct( ) method. Do not compy and paste
   * the same logic into this function */
  public static int numPunct(String s, int startIndex) {
    return 0;
```

```
    }

    /*
     * returns the index of the first occurrence
     * of a punctuation mark in s starting
     * from index startPosition or -1 if there are
     * none at index startPosition or later.
     *
     * When implementing this function, call your isPunct( ) method.
     * Do not simply copy and paste the body of isPunct( ) into this method.
     */
    public static int indexOfFirstPunct(String s, int startPosition) {
      return 0;
    }

    /*
     * returns the index of the first punctuation mark in s or
     * -1 if s contains no punctuation marks
     *
     * use your solution to indexOfFirstPunct(String s, int startPosition)
     * in this function. Do not repeat the same logic.
     *
     * Notice that this method has the same name as the
     * previous one, but that it takes a different number of arguments. This is
     * perfectly legal in Java. It's called "method overloading"
     *
     */
    public static int indexOfFirstPunct(String s) {
      return 0;
    }

    /*
     * returns the index of the last occurrence of a punctuation
     * mark in s or -1 if s contains no punctuation
     *
     * When implementing this function, call your isPunct( ) method.
     * Do not simply copy and paste the body of isPunct( ) into this method.
     */
    public static int indexOfLastPunct(String s) {
      return 0;
    }

    /*
     * returns a new String which is the same as s but with
     * each instance of oldChar replaced with newChar
     */
    public static String substitute(String s, char oldChar, char newChar) {
      return "";
    }

    /*
     * returns a new String which is the same as s, but
     * with each instance of a punctuation mark replaced
     * with a single space character
     *
     * Use at least one of your other functions in your
     * solution to this.
     *
     */
    public static String substitutePunct(String s) {
      return "";
    }

    /*
     * returns a new String which is the same as s,
     * but with all of the punctuation
     * marks removed.
     *
     * Use at least one of your other functions
     * in your solution to this one.
     *
```

```
   */
  public static String withoutPunct(String s) {
    return "";
  }

  /* returns true if c is found in s or false otherwise */
  public static boolean foundIn(String s, char c) {
    return true;
  }

  /*
   * Returns true of s contains none of the characters
   * found in chars or false otherwise.
   */
  public static boolean containsNone(String s, String chars) {
    return true;
  }

  /*
   * Returns true of s is comprised of only punctuation or
   * false otherwise
   *
   * Use at least one of your other
   * functions in this one.
   */
  public static boolean onlyPunct(String s) {
    return true;
  }

  /*
   * Returns true of s contains no punctuation or
   * false otherwise
   *
   * Use at least one of your other
   * functions in this one.
   */
  public static boolean noPunct(String s) {
    return true;
  }

  /*
   * returns true if s has two punctuation marks
   * right next to each other or false otherwise
   *
   * use at least one of your other methods
   * in your solution to this method
   */
  public static boolean consecutivePuncts(String s) {
    return false;
  }
}
```

You see that it contains several methods that have not yet been finished. Fill in your own implementation. For now, the methods contain only a placeholder return value so that the code will compile. You'll replace the lines with the return statements with returns that are appropriate.

Do not modify any other files in the project. Do not change the return types, names, or parameters of any of these functions or of the name of the class itself. There is also no need in this assignment for any global variables. In short, you will be modifying the body of each of these functions.

You do not have to write a `main()` to test your code. Though it may not be obvious, all of the test code has been written for you. If you're curious, you can find it by looking at the file `StringPracticeTest.java` included in the same directory as the code you need to complete. These tests use the popular JUnit testing framework. For now, you are not required to understand how they work. If you're curious, by all means look, but please do not modify the contents of this file.

In order to test your methods using the provided tests, open the `StringPracticeTest.java` file, and the click the run button. Initially, you'll see a red bar, indicating that not all of the tests have passed. Each of your functions whose

tests did not pass will have an X through its name. Eventually, you'll have perfectly debugged code, and the bar will turn green.

what to submit

There are two options: because you're only modifying `StringPractice.java`, you may submit just this, or you can upload a zip of your entire modified Eclipse project directory. Please send your work through Canvas.

for computers running older versions of java

If you're running an old version of Java and you have trouble with the zipped project file (but *only* if you're having trouble), here is an alternate version containing the same code, but is configured to use older versions of Java and JUnit (Java 8 and JUnit 4).