```
import java.util.Arrays;

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) {
    char[] Punctuations = { '\'', ',', '.', ';', ':', '!', '?' };
    for (int i = 0; i < Punctuations.length; i++) {
      if (Punctuations[i] == c) {
        return true;
      }
    }
    return false;
  }

  /*
   * 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) {

    int count = 0;
    for (int i = 0; i < s.length(); i++) {
      if (isPunct(s.charAt(i)) == true) {
        count++;
      }
    }
    return count;
  }

  /*
   * 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) {
    int count = 0;
    for (int i = startIndex; i < s.length(); i++) {
      if (isPunct(s.charAt(i)) == true) {
        count++;
      }
    }
    return count;
  }

  /*
   * 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) {
      for (int i = startPosition; i < s.length(); i++) {
        if (isPunct(s.charAt(i)) == true) {
          return i;
        }
      }
      return -1;
    }

    /*
     * 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) {
      for (int i = 0; i < s.length(); i++) {
        if (isPunct(s.charAt(i)) == true) {
          return i;
        }
      }
      return -1;

    }

    /*
     * 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) {
      for (int i = s.length() - 1; i >= 0; i--) {
        if (isPunct(s.charAt(i)) == true) {
          return i;
        }
      }
      return -1;
    }

    /*
     * 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) {
      for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) == oldChar) {
          s = s.replace(oldChar, newChar);
        }
      }
      return s;
```

```
  }

  /*
   * 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) {
    for (int i = 0; i < s.length(); i++) {
      if (isPunct(s.charAt(i)) == true) {
        s = s.replace(s.charAt(i), ' ');
      }
    }
    return s;
  }

  /*
   * 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) {
    for (int i = s.length() - 1; i >= 0; i--) {
      System.out.println("s = " + s.charAt(i));
      if (isPunct(s.charAt(i)) == true) {
        s = s.replace(Character.toString(s.charAt(i)), "");
      }
    }
    return s;
  }

  /* returns true if c is found in s or false otherwise */
  public static boolean foundIn(String s, char c) {
    for (int i = 0; i < s.length(); i++) {
      if (s.charAt(i) == c) {
        return true;
      }
    }
    return false;
  }

  /*
   * Returns true of s contains none of the characters
   * found in chars or false otherwise.
   */
  public static boolean containsNone(String s, String chars) {
    char[] ch = chars.toCharArray();
    char[] s_to_char = s.toCharArray();
    System.out.println(ch);
    System.out.println(Arrays.toString(s_to_char));

    for (int i = 0; i < ch.length; i++) {
      if (Arrays.toString(s_to_char).contains(String.valueOf(ch[i]))) {
        System.out.println(ch[i]);
        return false;
      }
    }
```

```
    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) {
    for (int i = 0; i < s.length(); i++) {
      if (isPunct(s.charAt(i)) == false) {
        return false;
      }
    }
    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) {
    for (int i = 0; i < s.length(); i++) {
      if (isPunct(s.charAt(i)) == true) {
        return false;
      }
    }
    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) {
    if (s.length() > 1) {
      for (int i = 0; i < s.length() - 1; i++) {
        if (isPunct(s.charAt(i)) == true && isPunct(s.charAt(i + 1))) {
          return true;
        }
      }
    }
    return false;
  }
}
```