a) is_right (a right triangle)

b) is_scalene (no two sides are the same length)

c) is_isosceles (exactly two sides are the same length)

d) is_equilateral (all three sides are the same length)

```
//*******************************************************************
//
//   Three_Sides.java      Programming Project      Application
//
//   Authors:  Lewis and Loftus
//
//   Classes:  Three_Sides
//             Triangle
//
//*******************************************************************

//-------------------------------------------------------------------
//
//  Class Three_Sides
//
//  Methods:
//
//     public static void main (String[] args)
//
//-------------------------------------------------------------------

class Three_Sides {

   //============================================================
   // Create and determine properties of various triangles.
   //============================================================
   public static void main (String[] args) {

      Triangle right = new Triangle (3, 4, 5);
      Triangle equal = new Triangle (6, 6, 6);
      Triangle isosceles = new Triangle (3, 7, 7);
      Triangle scalene = new Triangle (4, 5, 6);

      // check the right triangle
      System.out.println(right.print_sides() + " triangle:");

      if (right.is_right())
         System.out.println("\tIt is a right triangle");
      else
   System.out.println("\tIt is not a right triangle");

if (right.is_isosceles())
   System.out.println("\tIt is isosceles");
else
   System.out.println("\tIt is not isosceles");

if (right.is_equilateral())
   System.out.println("\tIt is equilateral");
else
   System.out.println("\tIt is not a equilateral");

if (right.is_scalene())
   System.out.println("\tIt is scalene");
else
   System.out.println("\tIt is not scalene");

System.out.println();

// check the equilateral triangle
System.out.println(equal.print_sides() + " triangle:");

if (equal.is_right())
   System.out.println("\tIt is a right triangle");
else
   System.out.println("\tIt is not a right triangle");

if (equal.is_isosceles())
   System.out.println("\tIt is isosceles");
else
   System.out.println("\tIt is not isosceles");

if (equal.is_equilateral())
   System.out.println("\tIt is equilateral");
else
   System.out.println("\tIt is not a equilateral");

if (equal.is_scalene())
   System.out.println("\tIt is scalene");
else
   System.out.println("\tIt is not scalene");

System.out.println();
```

```
// check the isosceles triangle
System.out.println(isosceles.print_sides() + " triangle:");

if (isosceles.is_right())
   System.out.println("\tIt is a right triangle");
else
   System.out.println("\tIt is not a right triangle");

if (isosceles.is_isosceles())
   System.out.println("\tIt is isosceles");
else
   System.out.println("\tIt is not isosceles");

if (isosceles.is_equilateral())
   System.out.println("\tIt is equilateral");
else
   System.out.println("\tIt is not equilateral");

if (isosceles.is_scalene())
   System.out.println("\tIt is scalene");
else
   System.out.println("\tIt is not scalene");

System.out.println();

// check the scalene triangle
System.out.println(scalene.print_sides() + " triangle:");

if (scalene.is_right())
   System.out.println("\tIt is a right triangle");
else
   System.out.println("\tIt is not a right triangle");

if (scalene.is_isosceles())
   System.out.println("\tIt is isosceles");
else
   System.out.println("\tIt is not isosceles");

if (scalene.is_equilateral())
   System.out.println("\tIt is equilateral");
else
   System.out.println("\tIt is not equilateral");

if (scalene.is_scalene())
   System.out.println("\tIt is scalene");
else
```

```
         System.out.println("\tIt is not scalene");

   }  // method main

}  // class Three_Sides

//---------------------------------------------------------------------
//
//  Class Triangle
//
//  Constructors:
//
//     public Triangle (int s1, int s2, int s3) {
//
//  Methods:
//
//     private int largest ()
//     private int shortest ()
//     public boolean is_right ()
//     public boolean is_equilateral ()
//     public boolean is_isosceles ()
//     public boolean is_scalene ()
//
//---------------------------------------------------------------------

class Triangle {

   int  side1, side2, side3;

   //===========================================================
   //  Sets up a triangle with the specified side lengths.
   //===========================================================
   public Triangle (int s1, int s2, int s3) {
      side1 = s1;
      side2 = s2;
      side3 = s3;
   }  // constructor Triangle

   //===========================================================
   //  Returns the length of the longest side of the triangle.
   //===========================================================
   private int largest () {
      int  max = side1;
      if (side2 > max)
         max = side2;
      if (side3 > max)
         max = side3;

      return max;
   }  // method largest

   //===========================================================
   //  Returns the length of the shortest side of the triangle.
   //===========================================================
   private int shortest () {
      int  min = side1;

      if (side2 < min)
         min = side2;
      if (side3 < min)
         min = side3;

      return min;
   }  // method shortest

   //===========================================================
   //  Determines whether the triangle is a right triangle.
   //===========================================================
   public boolean is_right () {
      return ((side1 * side1 + side2 * side2) == (side3 * side3));
   }

   //===========================================================
   //  Determines whether a triangle is equilateral.  If the
   //  longest side is equal to the shortest side, then the
   //  triangle is equilateral.
   //===========================================================
   public boolean is_equilateral () {

      int longest_side, shortest_side;

      longest_side = largest();
      shortest_side = shortest();

      return (shortest_side == longest_side);
   }  // method is_equilateral

   //===========================================================
   //  Determines whether a triangle is isosceles.  Any (and
//  at least) two sides must be equal.
//=============================================================
public boolean is_isosceles () {

   boolean   answer;

   if (side1 == side2)
      answer = true;
   else if (side1 == side3)
      answer = true;
   else if (side2 == side3)
      answer = true;
   else
      answer = false;

   return answer;
}  // is_isosceles

//=============================================================
//  Determines whether a triangle is scalene.
//=============================================================
public boolean is_scalene() {

   boolean answer;

   if (side1 == side2)
      answer = false;
   else if (side1 == side3)
      answer = false;
   else if (side2 == side3)
      answer = false;
   else
      answer = true;

   return answer;
}  // method is_scalene

//=============================================================
//  Prints the sides of the triangle.
//=============================================================
public String print_sides() {
   return (side1 + " " + side2 + " " + side3);
}  // method print_sides
}  // class Triangle
```

4-20 Write a class called `String_Analyzer`, which stores a string and provides several methods which determine and return the following characteristics. The string may contain several sentences. Each word in a sentence is separated by a single space character and each sentence is terminated with a period. One space separates each sentence.

a) number of sentences in the string

b) number of words in the entire string

c) number of characters in the entire string

d) average number of words per sentence

e) average number of characters per word

f) length of the longest word (in characters)

g) length of the longest sentence (in words)

Hint: use the `charAt` and `lastIndexOf` methods from the `String` class in the Java API.

```
//*******************************************************************
//
//   String_Evaluation.java    Programming Project    Application
//
//   Authors:  Lewis and Loftus
//
//   Classes:  String_Evaluation
//             String_Analyzer
//
//*******************************************************************

//-------------------------------------------------------------------
//
//  Class String_Evaluation
//
//  Methods:
//
//     public static void main (String[] args)
//
//-------------------------------------------------------------------

class String_Evaluation {

   //============================================================
   //  Creteates a String_Analyzer object and exercises it.
   //==========================================================
   public static void main (String[] args) {

      String_Analyzer  dick = new String_Analyzer
         ("See Dick. See Jane. See Dick and Jane.");
      String_Analyzer  spot = new String_Analyzer
        ("Take cognizance of Spot. Observe Spot progress expeditiously.");

      System.out.println("See Dick. See Jane. See Dick and Jane.");
      System.out.print("\tThe number of sentences in above string ");
      System.out.println("is: " + dick.num_sentences());

      System.out.print("\tThe number of words in above string ");
      System.out.println("is: " + dick.num_words());

      System.out.print("\tThe number of characters in above string ");
      System.out.println("is: " + dick.num_chars());

      System.out.print("\tThe average number of words per sentence in ");
      System.out.println("above string is: " + dick.words_per_sent());

      System.out.print("\tThe average number of characters per word in ");
      System.out.println("above string is: " + dick.chars_per_word());

      System.out.println();
      System.out.print("Take cognizance of Spot. ");
      System.out.println("Observe Spot progress expeditiously.");
      System.out.print("\tThe longest sentence in the above string is ");
      System.out.println(spot.longest_sent() + " words.");

      System.out.print("\tThe longest word in the above string is ");
      System.out.println(spot.longest_word() + " characters.");

   }  // method main

}  // class String_Evalation

//-------------------------------------------------------------------
//
//  Class String_Analyzer
//
//  Constructors:
//
```

```
//     public String_Analyzer (String str)
//
//  Methods:
//
//     private boolean end_of_word (int pos)
//     public int num_sentences ()
//     public int num_words ()
//     public int num_chars ()
//     public int words_per_sent ()
//     public int chars_per_word ()
//     public int longest_word ()
//     public int longest_sent ()
//
//---------------------------------------------------------------------

class String_Analyzer {

   final private char PERIOD = '.', SPACE = ' ';
   private String base_string;

   //===========================================================
   //  Sets up a new object with the specified string.
   //===========================================================
   public String_Analyzer (String str) {
      base_string = str;
   }  // constructor String_Analyzer

   //===========================================================
   //  Determines if the character at the specified position
   //  is an end-of-word marker (period or space).
   //===========================================================
   private boolean end_of_word (int pos) {

      boolean answer = false;

      if (pos != base_string.length()) {
         if (base_string.charAt(pos) == PERIOD)
            answer = true;
         else if (base_string.charAt(pos) == SPACE)
            answer = true;
      }

      return answer;

   }  // method end_of_word
```

```
//==========================================================
//  Determines the number of sentences in the base string.
//==========================================================
public int num_sentences () {

   int  position = 0, count = 0;

   while (position < base_string.length()){
      if (base_string.charAt(position) == PERIOD)
         count = count + 1;
      position = position + 1;
   }

   return count;

}  // method num_sentences

//==========================================================
//  Determines the number of words in the base string.
//==========================================================
public int num_words () {

   int position = 0;
   int count = 0;

   if (base_string.length() != 0) {
      while (position < base_string.length()){
         if (base_string.charAt(position) == SPACE)
            count = count + 1;
         position = position + 1;
      }
      count = count + 1;
   }

   return count;

}  // method num_words

//==========================================================
//  Determines the number of characters in the base string.
//==========================================================
public int num_chars () {
   return base_string.length();
}  // method num_chars
```

```
//===========================================================
//  Determines the average number of words per sentence.
//===========================================================
public int words_per_sent () {
   return (num_words() / num_sentences());
}  // method words_per_sent

//===========================================================
//  Determines the average number of characters per word.
//===========================================================
public int chars_per_word() {
   return (num_chars() / num_words());
}  // method chars_per_word

//===========================================================
//  Determines the length of the longest word, in chars.
//===========================================================
public int longest_word () {

   int  letter, max = 0, position = 0;

   while (position < base_string.length()) {
      letter = 0;      // start of a new word
      while (end_of_word(position) == false) {
         letter = letter + 1;
         position = position + 1;
      }

      // move past period & leading space
      while (end_of_word(position))
         position = position + 1;

      if (letter > max)
         max = letter;
   }

   return max;

}  // method longest_word

//===========================================================
//  Determines the length of the longest sentence, in words.
//===========================================================
public int longest_sent () {
```

```
         int  word, max = 0, position = 0;

         while (position < base_string.length()) {
            word = 0;      // start of a new sentence
            while (base_string.charAt(position) != PERIOD) {
               if (base_string.charAt(position) == SPACE)
                  word = word + 1;
                  position = position + 1;
            }

            word = word + 1;         // count last word in sentence
            position = position + 2; // move past period & leading space

            if (word > max)
               max = word;
         }

         return max;

      }  // method longest_sent

   }  // class String_Analyzer
```

4-21 Write a class that uses the StringTokenizer class to identify the parts of a phone number. Assume that the format of the phone number is (nnn) nnn-nnnn. For example, given the phone number of (610) 555-1212, 610 is the areacode, 555 is the exchange, and 1212 is the extension. The class should have at least three public methods one that returns the areacode, one that returns the exchange, and one that returns the extension.

Not Provided