Homework Turnin

Email:	rgalanos@fcps.edu
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	01-07
Receipt ID:	411aa8533a491ce8bdb0fdc32ddc0030

Turnin Successful!

The following file(s) were received:

StringCoderDriver.java (4776 bytes)

```
//Name:    Date:

import java.util.*;
public class StringCoderDriver
{
   public static void main(String[] args)
   {
      StringCoder sc = new StringCoder("sixtyzipperswerequicklypickedfromthewovenjutebag");
      StringPart[] sp = sc.encodeString("overeager");
      for(int i=0; i<sp.length; i++)
         System.out.print(sp[i]+", ");   // (37, 3), (14, 2), (46, 2), (9, 2),
      System.out.println();
      String s = sc.decodeString(sp);
      System.out.println(s);             // overeager

      StringPart[] sp2 = sc.encodeString("kippers");
      for(int i=0; i<sp2.length; i++)
         System.out.print(sp2[i]+", ");  // (20, 1), (6, 6),
      System.out.println();
      String s2 = sc.decodeString(sp2);
      System.out.println(s2);            // kippers

      StringPart[] sp3 = sc.encodeString("colonials");
      for(int i=0; i<sp3.length; i++)
         System.out.print(sp3[i]+", ");  // (19, 1), (31, 1), (21, 1), (31, 1), (40, 1), (1, 1), (46, 1), (21, 1), (0,
      System.out.println();
      String s3 = sc.decodeString(sp3);
      System.out.println(s3);            // colonials

      StringPart[] sp4 = sc.encodeString("werewolf");
      for(int i=0; i<sp4.length; i++)
         System.out.print(sp4[i]+", ");   // (12, 4), (36, 2), (21, 1), (29, 1),
      System.out.println();
      String s4 = sc.decodeString(sp4);
      System.out.println(s4);             // werewolf
   }
}


class StringCoder
{
   private String masterString;
/** @param master the master string for the StringCoder
* Precondition: the master string contains all the letters of the alphabet
*/
   public StringCoder(String master)
   {
      masterString = master;
   }

/** @param parts an array of string parts that are valid in the master string
```

```
* Precondition: parts.length > 0
* @return the string obtained by concatenating the parts of the master string
*/
   //PART A:
   public String decodeString(StringPart[] parts)
   {
      String decoded = "";
      for(int i = 0; i < parts.length; i++)
      {
         decoded = decoded + masterString.substring(parts[i].getStart(),parts[i].getStart()+parts[i].getLength());
      }
      return decoded;
   }


/** @param str the string to encode using the master string
* Precondition: all of the characters in str appear in the master string;
* str.length() > 0
* @return a string part in the master string that matches the beginning of str.
* The returned string part has length at least 1.
*/
   private StringPart findPart(String str)
   {
      int x = 0;
      String s = str.substring(0, x);
      while( masterString.contains(s) )
      {
         x++;
         if(x > str.length())
            break;
         s = str.substring(0, x);
      }
      s = str.substring(0, x - 1);
      int start = masterString.indexOf(s);
      StringPart sp = new StringPart(start, s.length());
      return sp;
   }


/** @param word the string to be encoded
* Precondition: all of the characters in word appear in the master string;
* word.length() > 0
* @return an array of string parts of the master string that can be combined
* to create word
*/
// Part B
   public StringPart[] encodeString(String word)
   {
      int i = 0;
      int length = 0;
      final int wordLength = word.length();
      StringPart[] temp = new StringPart[100];
      while(length<wordLength)
      {
         temp[i] = findPart(word);
         length = length + temp[i].getLength();
         word = word.substring(temp[i].getLength());
         i++;
      }

      StringPart[] encoded = new StringPart[i];
      int x = 0;
      for(int a = 0;a<temp.length;a++)
      {
         if(temp[a]!=null)
         {
            encoded[x] = temp[a];
            x++;
         }
      }
      return encoded;
   }
}


class StringPart
{
//private data fields--what does a StringPart know?
   int start;
   int length;
```

```
/** @param start the starting position of the substring in a master string
* @param length the length of the substring in a master string
*/
   public StringPart(int start, int length)
   {
      this.start = start;
      this.length = length;
   }

/** @return the starting position of the substring in a master string
*/
   public int getStart()
   {
      return start;
   }

/** @return the length of the substring in a master string
*/
   public int getLength()
   {
      return length;
   }
   public String toString()
   {
      return "(" + start + ", " + length + ")";
   }
}


/****************  output  ****************
(37, 3), (14, 2), (46, 2), (9, 2),
overeager
(20, 1), (6, 6),
kippers
(19, 1), (31, 1), (21, 1), (31, 1), (40, 1), (1, 1), (46, 1), (21, 1), (0, 1),
colonials
(12, 4), (36, 2), (21, 1), (29, 1),
werewolf

*****************************************/
```