JNW-DLC1 BIT STUFFING EXERCISE
Version 1.008

PURPOSE

This purpose of this exercise is for you to demonstrate your understanding of bit stuffing by writing methods to stuff and unstuff a bit set.

INSTALLATION PROCESS

Install the JNWS as described in the Users Guide, and, as described in the Users Guide, set *main()* to *Exercises.JDLC1*. Compile and run the program. If you have installed everything correctly, you will get the message, "You have successfully installed JNWS!" You can now program the needed algorithm.

EXERCISE PROCESS

You are provided with a template for your program in class *Exercises.JDLC1*. The template also contains the algorithm for bit stuffing, from *Understanding Internet Protocols* pages 35 to 38. The class has four methods

```
/**Reads a single line from standard input*/
    private static String ReadSingleLineFromStandardIO (  )

/**Stuffs a string of bits and returns the stuffed bit set*/
    private static Utility.JNWBitSet StuffBitSet ( Utility.JNWBitSet unstuffedBits )

/**Unstuffs a string of bits, and returns the unstuffed bit set.*/
    private static Utility.JNWBitSet UnstuffBitSet ( Utility.JNWBitSet stuffedBits )

 /**Runs the bitstuffing.*/
     public static void main ( String [ ] args )
```

Your assignment is to complete the two methods in BitStuffing which are needed to stuff the String and unstuff it.

Suggestions:

1. If you have a bit set of length n bits, what is the maximum possible size of the stuffed bit set? Start by making a stuffed bit set of that size.

2. Be careful to note that the bit frame will have a flag "01111110" at each end, which must not be stuffed.

3. Since even simple Java IO can become very complicated, use method *ReadSingleLineFromStandardIO* ( ) to read a String representing the email from the standard input. Use setJNWBitSetToString (String characterString) to convert the String to a bit set.

4. The methods of Utility.JNWBitSet will convert and display bit sets, which are java vectors that hold a set of bits (true or false). A JNWBitSet has length JNWLength, and it runs from the left-most bit at index JNWLength -1 to the right-most bit at index 0. Utility.JNWBitSet is described in greater detail in Appendix A of the Users Guide

5. Methods that you may need are:
   - setJNWLength (int length) set the length of the JNWBitSet
   - getJNWLength () gets the length of the JNWBitSet
   - setBitValue (int bitindex , int value) sets the bit at index bitIndex to 0 or 1
   - getBitValue (int bitindex) gets the bit at index bitIndex as a 0 or 1
   - shift (int numberOfBitsToShift) shifts the BitSet left (positive) or right (negative) by numberOfBitsToShift bits
   - setJNWBitSetToString (String characterString) converts a character String into a JNWBitSet
   - toString ( ) converts a JNWBitSet to a String of 0s and 1s, 40 bits per row

6. A good test is to use the input String "0111111000011111111000111111O" in the method main without converting it to an email. The stuffed output must be "011111100001111101110001111110" (note that the first and last 8 bits are not stuffed).

   JNWBitSet has a constructor

   public JNWBitSet(String stringOfZerosAndOnes)

   that creates a bit set from a String of 0s and 1s. For example,

   ```
   Utility. JNWBitSet aBitSet =
     new  Utility. JNWBitSet  ("01111110000011111111001101111110" ) ;
   ```

will create the bit set 01111110000011111111001101111110. So you should be able to create the test bit set very easily as a variation on your program.

7. If you use NetBeans, review the advice in the Users Guide about setting the *main ( )* method to be run.

EXERCISE

Write methods *Exercises.JDLC1.StuffBitSet( )*, and *Exercises.JDLC1. UnstuffBitSet ( )* . The stuff and unstuff do what their names imply and the main does the following using *Utility.Err.println()* to display:

- reads an email String
- displays the String
- converts it to a JNWBitSet
- displays the result
- stuffs the bit set
- displays the result
- unstuffs the result
- converts it back to text
- displays the text.

Run the program for the email String "Chicken Little was right~" which is the text that will be used by the grader. This string contains a pattern that will cause stuffing/unstuffing to happen. You should inspect your output to make sure your code works properly, before submitting.

DEMONSTRATIONS

Two demonstrations are available for you to see the effect of bit stuffing. One demonstration accepts an email String as input. A second demonstration accepts strings of 0s and 1s rather than a String of characters.

Put the file *JDLC1_BitStuffEMail.jar.jar* in any folder and use the command prompt

```
java –jar JDLC1_BitStuffEMail.jar.jar
```

to run the demonstration. Enter an email String in the standard IO; the demonstration should run.

SUBMITTING

Cut and paste the output from the debug window to a text file. Send your output and source file *Exercises.JDLC1.java* as email attachments to the grader.