# Instructions for JNW2 Project DLC2 – Generating FCS

See Chapter 4 of *Understanding Internet Protocols*. Project DLC2 is basically the same as given at the end of Chapter 4, though the software context is somewhat different.

Your assignment is to complete the code in JNW2/message/CrcFcs.java to perform generateHdlcFcs(). You will test this using again the Chicken Little email1.txt file. You can continue to use configuration file SimpleWAN.txt for DLC2 (see Preliminary Student Guide for more detail).

You will be able to see both the Chicken Little input and the resulting binary codes, before and after stuffing and unstuffing, in the JNW2 text output. This includes the 16-bit HDLC FCS from CrcFcs.generateHdlcFcs(), which is at the end of the unstuffed and stuffed frames that are printed by JNW2. Before you program generateHdlcFcs(0), the FCS is there as 16 zeros.

Note that generateHdlcFcs() will have as input the whole frame, complete with 01111110 flags at start and end, so you must ignore the first and last 8 bits of the input.

It is likely to be easier to debug your work if you start with a shorter input BitSequence and you know what the result should be. There is a link on the course Moodle page to a page which shows a correct FCS calculation for input string "123". JNW2 classes, including CrcFcs, include a main() designed to test that class with input that you type. You should study the code in CrcFcs.main() to understand what it is doing. The test is good if the FCS comes out sixteen zeros when calculated on the frame containing a valid FCS, and the FCS produced for "123" matches that posted in Moodle. Note that you will need to set NetBeans to use the main() in CrcFcs, as before when you selected the main() in RunSimulation, and after you finish testing you'll have to put it back in order to make a run for submission.

You will need to use class BitSequence to program generateHdlcFcs(). (Note that position 0 of a BitSequence is viewed as the most significant bit by JNW2.) Here are some methods from that class that you may need to write your progam:

New BitSequence(int length) – makes an empty (all zeros) BitSequence of size *length*
int BitSequence.size() – returns length in bits
boolean BitSequence.getValue(int index) – returns the value at index position
void BitSequence.setValue(int index, boolean value) – updates the position with the value
void BitSequence.setOctet(int index, byte value) – updates 8 bits starting at index with the values in the byte

Warning: BitSequence relies on underlying class BitSet, which has a behavior you might not expect: if you modify a bit beyond the current index range of a BitSet, the BitSet will grow larger to include the new index, with values in between set to false.

When your CrcFcs is functional, you are to submit CrcFcs.java and a copy of the JNW2 output (select and copy from the output pane of NetBeans). Submit by upload through the course Moodle page. Be sure to include your name in the code comment provided.