Metro State University of Denver, Department of Mathematics and Computer Science
CS 3700-001: Computer Networks, Spring 2013, Dr. Weiying Zhu
Homework 11, Due Date: 10:00am 5/2/2013, Submission: Upload source code (.java) files to Blackboard

Write a program to simulate how the CSMA/CD algorithm at the Link Layer on ONE GIVEN host eventually sends one link-layer frame successfully. The user inputs in Steps 4 and 5 is used to simulate the information that NIC obtains from the shared "channel". (**Hint**: You may use the Timer and TimerTask classes for scheduling a task to be performed later. See http://enos.itcollege.ee/~jpoial/docs/tutorial/essential/threads/timer.html and JavaDoc. Or feel free to use any class to implement such functions.)

Your CSMA/CD program needs to

1. Display prompt messages to collect two user inputs (to keep it simple, both inputs are positive integers):
   `How many seconds from now until this frame is ready to be sent out?`
   `How long does it take for the host to transmit this frame entirely? (in seconds)`
2. Initialize the value of a variable for *total-number-of-collisions-so-far* as **zero**.
3. Use a timer to schedule a *sensing* task to be performed ***x*** seconds later, where ***x*** is the first user input collected in Step 1.
4. Perform the *sensing* task scheduled in Step 3, Step 4.1, or Step 5.2.3. Display a message for user input:
   `NIC senses channel. Is channel idle?`
   - 4.1 If the user input is "No", use the timer to schedule a new *sensing* task to be performed **2** seconds later.
   - 4.2 If the user input is "Yes",
     - 4.2.1 Initialize the value of a variable for *leftover-transmitting-duration* as ***y*** seconds, where ***y*** is the second user input collected in Step 1.
     - 4.2.2 Display a message, "**`NIC starts transmitting the frame. The leftover-transmission-duration is z seconds.`**", where ***z*** is the current value of the variable for *leftover-transmitting-duration*.
     - 4.2.3 Use the timer to schedule a *collision-detecting* task to be performed **2** second later.
5. Perform the *collision-detecting* task scheduled in Step 4.2.3 or Step 5.1.2.4. Display a prompt message to ask for user input:
   `NIC detects collision on channel. Is there any other transmission on channel?`
   - 5.1 If the user input is "No",
     - 5.1.1 If the value of the variable for *leftover-transmitting-duration* (See Step 4.2.1.) is **zero**, DISPLAY "**`Done with transmitting this frame!`**" and TERMINATE this program.
     - 5.1.2 If the value of the variable for *leftover-transmitting-duration* is **positive**,
       - 5.1.2.1 Subtract ***min*(2,** the value of the variable for *leftover-transmitting-duration***)** seconds from the value of the variable for *leftover-transmitting-duration*,
       - 5.1.2.2 DISPLAY a message, "**`NIC keeps transmitting the frame. The leftover-transmission-duration is z seconds.`**", where ***z*** is the current value of the variable for *leftover-transmitting-duration*.
       - 5.1.2.3 Use the timer to SCHEDULE a *collision-detecting* task to be performed ***min*(2,** the value of the variable for *leftover-transmitting-duration***)** seconds later.
   - 5.2 If the user input is "Yes",
     - 5.2.1 Add **1** to the value of the variable for *total-number-of-collisions-so-far* (see Step 2)
     - 5.2.2 Use binary backoff to determine ***k*y***, where ***y*** is the second user input collected in Step 1, ***k*** is randomly chosen from $\{0, 1, 2, \ldots, 2^{m-1}\}$, and ***m*** is the current value of the variable for *total-number-of-collisions-so-far*.
     - 5.2.3 Display a message, "**`NIC detects a collision and aborts transmitting the frame and will sense the channel for re-transmission k*y seconds later.`**", where k*y is the product calculated in Step 5.2.2.
     - 5.2.4 Use a timer to schedule a *sensing* task to be performed ***k*y*** seconds later.

A version with three processes: Host A, Host B, and Channel for simulating CSMA/CD (total points of 120):

- Each one of Host A and Host B has a frame to be sent out. The Channel keeps track of the data transmission procedures on both Host A and Host B and responds to sensing and collision detection requests coming from Host A and Host B.
- The Channel process may use UDP datagrams to communicate with Host A and Host B. Whenever Host A or Host B starts transmitting data or aborts data transmitting, it MUST notify the Channel process.
- The Channel process may take only ONE user input as below for collision detection:
  `What is the propagation delay (in seconds) between Host A and Host B?`
  This user input, e.g., 3 seconds, is used for possible data collision. For example, when Host A starts transmitting its frame at time 15 seconds and Host B asks Channel whether it's idle at time 17 seconds, Channel responds "Yes". Then when Host B asks Channel at time 19 seconds whether there is any other transmission on Channel, Channel responds "Yes".
- Each one of Host A and Host B may take only TWO user inputs as below:
  `How many seconds from now until this frame is ready to be sent out?`
  `How long does it take for the host to transmit this frame entirely? (in seconds)`
- Each one of Host A and Host B MUST implement the ENTIRE logic defined in the basic version including those displays. However, all the user inputs defined in Step 4 and Step 5 MUST be REPLACED by queries to and responses from the Channel process. And the Channel process needs to provide appropriate responses according to its knowledge on data transmission on Host A and Host B.