Summer 2021 Due by 6/13/2021

CS4331/CS5332 – Homework1

You are required to implement a multi-threaded program for communicating messages between a sender object thread and a receiver object thread using a "message buffer and response connector (MBRC)" object in an object-oriented language (Java, C++, or C#).

The figure below depicts the MBRC class, followed by the detailed specifications of operations. The MBRC class provides the send(), receive(), and reply() operations and has a message buffer and a response buffer where each buffer can hold only one message.

Your program must create two separate objects, a sender object, and a receiver object, each having its thread in multi-threaded programming and communicate with each other. They are concurrently running to communicate messages synchronously. The sender object can send the next message to the receiver object only after receiving the response of the previous message from the receiver object.

Also, the message Content of each message from a sender object to a receiver object requires a digital signature (**non-repudiation security service)**. In contrast, each response from the receiver object to the sender object requires a message digest (**integrity security service)**.

The messages from the sender object to the receiver object have a structure (message name, message content), where the message name is a string and the message content is an integer, e.g., ("add", 3) or ("multiply", 7). Before sending a message to the receiver object, the sender object prints the message (message name and message content) using its printMessage() operation on the screen. The sender object signs on the message content (meaning to generate a digital signature of the message content) and sends a message and its digital signature to the receiver object via a MBRC object. The sender object calls the send() operation of MBRC object to send a message to the receiver object. The send() operation stores the message in the message buffer of MBRC object.

The receiver object receives a message from the sender object by calling the receive() operation of MBRC object. The receive() operation reads the message stored in the message buffer.

AddCalculation class has an add() operation, and MultiplyCalculation class has a multiply() operation. When the receiver object receives a message from the sender object via the receive() operation of MBRC object, it verifies the digital signature of the message content. If the digital signature is valid, the receiver object calls an operation on the AddCalculation or MultiplyCalculation class depending on the message name. For example, the receiver object calls the add() operation of AddCalucation class if it receives a message like ("add", 3). In contrast, it calls the multiply() operation of MultiplyCalculation class if it receives a message like ("multiply", 3). The add() operation adds 10 to the integer in a message and returns the result.

The multiply() operation multiplies the integer by 10 and returns the result. The receiver object prints the result using a printResult() operation on the screen.

The receiver object returns the result back to the sender object by calling the reply() operation of MBRC object. The reply() operation stores the result in the response buffer of MBRC object.

The result requires **integrity security**, so the receiver object must generate a message digest for each result using its generate() operation before sending it to the sender object. The receiver object sends both the result and its message digest to the sender object.

When the sender object receives the result, it must verify the integrity of the result using its verify() operation. If the integrity of the result is not breached, the sender object prints the result using the printReceivedResult() on the screen

The messages are ("add", 4), ("multiply", 1), ("multiply", 8), ("add", 2), ("add", 3), ("add", 99), ("multiply", 53) that are sent by the sender object to the receiver object. When all messages are communicated one by one, your program terminates.

The classes and their operations that must be implemented:

1) Sender class: assigned a thread
    - printMessge()
    - printReceivedResult()
    - sign() //generate a digital signature
    - verify() //verify a message digest

2) Receiver class: assigned a thread
    - printResult()
    - verify() //verify a digital signature
    - generate() //generate a message digest

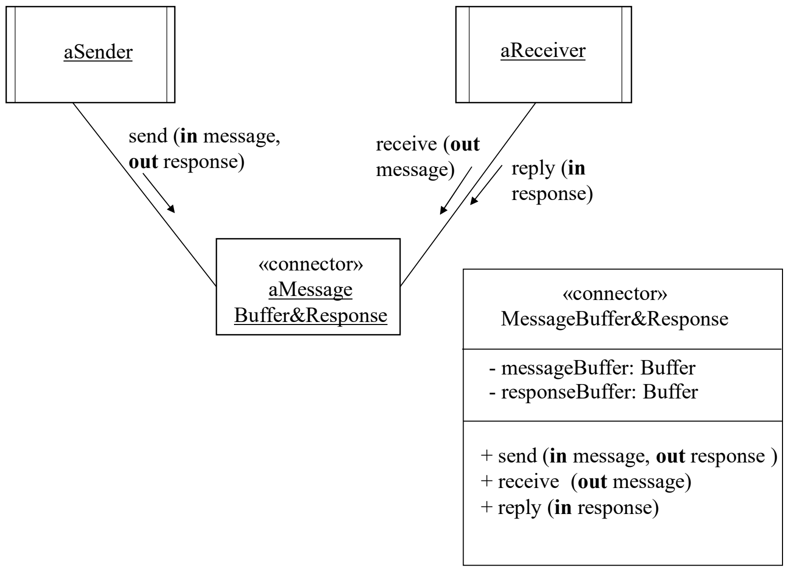
3) AddCalculation class
    - add()

4) MultiplyCalculation class
    - multiply()

5) MBRC class
    - send()
    - receive()
    - reply()

Describe additional classes you defined and assumptions if needed. Please make a zip file containing all source code files for HW1 and submit it to the blackboard.

message buffer and response connector



Message buffer & response connector

```
monitor MessageBuffer&Response
 -- Encapsulates a message buffer that holds at most one message
 -- and a response buffer that holds at most one response.
 -- Monitor operations are executed mutually exclusively.
private messageBufferFull : Boolean = false;
private responseBufferFull : Boolean = false;
public send (in message, out response)
  place message in buffer;
  messageBufferFull := true;
  signal;
  while responseBufferFull = false do wait;
  remove response from response buffer;
  responseBufferFull := false;
end send;

public receive (out message)
  while messageBufferFull = false do wait;
  remove message from buffer;
  messageBufferFull := false;
end receive;

public reply (in response)
  Place response in response buffer;
  responseBufferFull := true;
   signal;
end reply;
end MessageBuffer&Response;
```