CS5332 – Homework1 | Message Buffer and Response Connector (MBRC)

Name: Phornsawan Roemsri

R#: R11685149

The results of running this program with Apache NetBeans 12.4.

```
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ MBRCJavaProject_HW1 ---
========================================================================
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 program will compute addition and multiplication operations on integer 10
follow the instruction in the message that sent from the Sender to the Receiver.
========================================================================
The Sender (Thread-0) is about to send the Sender's public key to the Reciever.
The Receiver (Thread-1) receives the Sender's public key.
The Sender (Thread-0) acknowledges that The Receiver received the Sender's public key.
               ----------------------------------------

The Sender (Thread-0) is about to send the message: ("add", 4) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 4 added 10, which is 14
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 14.

The Sender (Thread-0) is about to send the message: ("multiply", 1) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 1 multiply by 10, which is 10
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 10.

The Sender (Thread-0) is about to send the message: ("multiply", 8) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 8 multiply by 10, which is 80
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 80.

The Sender (Thread-0) is about to send the message: ("add", 2) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 2 added 10, which is 12
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 12.

The Sender (Thread-0) is about to send the message: ("add", 3) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 3 added 10, which is 13
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 13.

The Sender (Thread-0) is about to send the message: ("add", 99) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 99 added 10, which is 109
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 109.

The Sender (Thread-0) is about to send the message: ("multiply", 53) to the Reciever.
   The Receiver (Thread-1) received the message and the signature verification result is true.
The Receiver (Thread-1) is about to reply the result of 53 multiply by 10, which is 530
The Sender (Thread-0) receives the response (or the result) that valid and not corrupted from the Receiver, which is 530.

               ----------------------------------------
The program finished
========================================================================
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.073 s
Finished at: 2021-06-14T03:53:33-05:00
------------------------------------------------------------------------
```

Source code: https://github.com/rinriko/CS5332-Homework1-MBRC

The additional classes and methods are shown below in the list of classes and methods with the yellow highlights. Also, the descriptions are blue color and shown below the list of classes and methods.

Note* some methods may not show the description since it is obvious by its name or already shown in the homework description.

The list of classes and methods:

1) Sender class: assigned a thread
   - Sender()
   - setKeyAndSendToReceiver()
   - sendEndMessage()
   - printMessage()
   - printReceivedResult()
   - sign()
   - verify()
   - run()
2) Receiver class: assigned a thread
   - Receiver()
   - getPKeyAndInit()
   - printResult()
   - verify()
   - generate()
   - run()
3) AddCalculation class
   - add()
4) MultiplyCalculation class
   - multiply()
5) MBRC class
   - MBRC()
   - send()
   - receive()
   - reply()
6) KeyExchange class
   - KeyExchange()
   - send()
   - receive()
   - reply()
7) Message class
   - Message()
   - setSignature()
   - getName()
   - getContent()
   - getSignature()
   - getMessageByte()

8) Response class
   - Response()
   - setHashValue()
   - getStrResult()
   - getHashValue()
9) MBRCJavaProject_HW1 class
   - main()

The list of classes and methods with descriptions:

1) Sender class: assigned a thread
   - Sender(MBRC Obj, KeyExchange k)
     To initializing an object.
   - setKeyAndSendToReceiver().
     Generating keys and sending the public key to the receiver through the key buffer or KeyExchange class.
   - sendEndMessage()
     Send the endMessage to let the receiver thread know that there is no message left.
     If we do not use the endMessage, the receiver thread will wait for the message from the sender thread forever because the sender thread already finished its job. And the main thread will wait for the receiver thread to finish its job forever too. So we need it.
   - printMessage(Message message)
   - printReceivedResult(Response response)
   - sign(byte message[], PrivateKey prvk)
     Generate a digital signature.
   - verify(byte[] hashValue, String msg)
     Verify a message digest.
   - run()
     Start the sender thread by generating keys and sending the public key to the receiver through the key buffer or KeyExchange class. Then, show the message and send the message with the digital signature to the buffer. Waiting for the response from the receiver through the buffer, verify the integrity of the result, show the results, and send the endMessage to let the receiver thread know that there is no message left. If we do not use the endMessage, the receiver thread will wait for the message from the sender thread forever because the sender thread already finished its job. And the main thread will wait for the receiver thread to finish its job forever too. So we need it.
     *The run() method of thread class is called if the thread was constructed using a separate Runnable object otherwise this method does nothing and returns. When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times. (https://www.javatpoint.com/java-thread-run-method)

2) Receiver class: assigned a thread
   - Receiver(MBRC Obj, KeyExchange k)
     To initializing an object.
   - getPKeyAndInit()
     Getting the public key from the key buffer or Key Exchange class. And initializing the operation objects (add, multiply).
   - printResult(String name, int content, String result)
   - verify(byte message[], PublicKey pubk, byte[] signature)
     Verify a digital signature
   - generate(String msg)
     Generate a message digest
   - run()
     Started the receiver thread by getting the public key from the key buffer or KeyExchange class. Then, get the message from the buffer, verifies the digital signature of the message content, compute the result, show the results, generate a message digest for each result, and sends both the result and its message digest back to the sender through the buffer. Repeated the processes except the first one (getting the public key from the key buffer or KeyExchange class) till got the endMessage from the sender thread.
     *The run() method of thread class is called if the thread was constructed using a separate Runnable object otherwise this method does nothing and returns. When the run() method calls, the code specified in the run() method is executed. You can call the run() method multiple times. (https://www.javatpoint.com/java-thread-run-method)
3) AddCalculation class
   - add(int num)
4) MultiplyCalculation class
   - multiply(int num)
5) MBRC class
   - MBRC()
     To initializing an object.
   - send(Message message)
   - receive()
   - reply(Response response)
6) KeyExchange class
   Similar to MBRC class but change the message to the public key.
   - KeyExchange()
     To initializing an object.
   - send(PublicKey pk)
   - receive()
   - reply(String response)

7) Message class

This class model a message that will send from the sender object to the receiver object. The message contains the message name, message content, and a digital signature of the message content.

- Message(String name, int content)

  To initializing an object.
- setSignature(byte[] sign)
- getName()
- getContent()
- getSignature()
- getMessageByte()

8) Response class

This class model a response that will send from the receiver object to the sender object. The response contains the result and its message digest.

- Response(String res)

  To initializing an object.
- setHashValue(byte[] hashVal)
- getStrResult()
- getHashValue()

9) MBRCJavaProject_HW1 class

- main(String[] args)

  Create all objects and start all threads.

  *A Java program needs to start its execution somewhere. A Java program starts by executing the main method of some class. (http://tutorials.jenkov.com/java/main-method.html)