Programming Assignment #2 - RDT 3.0

Name: Michelle Emamdie

UFID: 7985-3119

Email: memamdie@ufl.edu

How to Compile

- Download network.java, receiver.java, sender.java, and Packets.java
- Switch into the directory that contains all these files
- Enter the command javac *.java to compile all the files
- Prior to running the code you will need to create a test text file in the same directory. Let's assume you've named this file **message.txt**

Running the Code

- On thunder.cise.ufl.edu: java network [Port Number]
- On sand.cise.ufl.edu: java receiver thunder.cise.ufl.edu [Same Port Number]
- On storm.cise.ufl.edu: java sender thunder.cise.ufl.edu [Same Port Number] [Whatever you named the test file]

Description of Code Structure

network class

- Class: network
    - Functions:
        - main - Creates a socket on the server on the designated port and then waits for a receiver and sender to join.
    - Static Classes:
        - MessageThread extends Thread

MessageThread class

- Class: MessageThread
    - Helper class that spins off new threads so that sender and receiver run on isolated threads

but the same network

- Functions:
  - Constructor - creates a new Message Thread
  - run() - Will be the driving force to allow the sender and the receiver to communicate.
  - send(String) - Sends a messasge to the output socket
  - toDifferentThread(String) - Sends message to the other thread executing on the network
- Properties:
  - String ACK2 - constant
  - String newline - constant
  - Socket socket - The socket that the network has opened
  - int id - used to identify which thread this is

receiver class

- Class: receiver
  - Functions:
    - Constructor- Makes a receiver based on the host name and port number
    - run()- Handles the reception of packets from the network server.
    - closeAll()- Close out all the open connections
    - main()- Takes in two inputs, the host and the port
  - Properties:
    - Socket socket -The socket that will be used to connect to the server socket
    - PrintWriter writer - Used to output to the server socket so that messages can be sent to the sender
    - BufferedReader buffer - The input from the socket

sender class

- Class: sender
  - Functions:
    - Constructor - Makes a valid connection to a port
    - sendPackets() - Creates packets that will be sent to the server and sends them to the server.
    - Also receives responses from the server
    - closeAll() - Close all the open connections
    - main() - Used to create a sender that will read a file and send the message as packets to the server
  - Properties:
    - Socket socket - Connects this thread to the socket currently running on the server
    - PrintWriter writer - Used to output to the server socket so that messages can be sent to

the receiver
    - BufferedReader buffer - The input from the file
    - BufferedReader bufferInput - The input from the socket
    - DataInputStream data - The stream of data coming in from the file
    - FileInputStream stream - Used to read the file from the directory
    - newline - New line character to be used when the end of the line has been located.

Packets class

- Class: Packets
- Functions:
    - Constructor - Initialize the properties
    - createPacket(String) - Creates a new packet with the content passed to it
    - generateChecksum(String) - Counts the sum of the ascii values in the content string
    - validateAck(String) - Determines if the acknowledgement received is valid
    - validate() - Validates the checksum to make sure it has not changed from what it should be
    - parse(String) - Take the string and set the properties for a packet
    - generateMessage() - Takes the packet and turns it into as string so that it can be sent to the network
- Properties:
    - int sequenceNum - alternating 0 or 1 for each packet
    - int packetID - the current number of packets
    - int checkSum - sum of the ascii character values of the string
    - String content - The data that will be part of the message
    - boolean last - True when the last packet is being sent
    - String newline - Constant new line character

Execution results

message.txt

```
1.
You are my sunshine.
2.
My only sunshine
```

Run on thunder

```
> javac *.java
> java network 8080
```

Network output

```
Waiting... connect receiver
Get connection from: /127.0.0.1:56767
Get connection from: /127.0.0.1:56778
Received: Packet0, 1, DROP
Received: ACK1, PASS
Received: ACK1, PASS
Received: ACK0, PASS
Received: ACK0, PASS
Received: Packet1, 4, DROP
Received: ACK0, PASS
Received: ACK0, PASS
Received: ACK1, PASS
Received: ACK1, PASS
Received: Packet0, 7, CORRUPT
Received: ACK0, PASS
Received: Packet1, 8, CORRUPT
Received: ACK1, PASS
Received: ACK0, PASS
Received: ACK0, PASS
Received: Packet1, 10, DROP
Received: ACK0, PASS
Received: ACK0, PASS
Received: ACK1, PASS
Received: ACK1, PASS
Received: Packet0, 13, CORRUPT
Received: ACK0, PASS
Received: Packet1, 14, CORRUPT
Received: ACK1, PASS
Received: Packet0, 15, CORRUPT
Received: ACK0, PASS
Received: ACK1, PASS
Received: ACK1, PASS
Received: ACK-1, PASS
```

You'll notice that it connects and then begins transmitting the packets. One interesting note here is that the last log message is that the ACK-1 this is becuase I send a response of -1 when the last piece of data has been sent.

Run on sand

```
> javac *.java
> java receiver thunder.cise.ufl.edu 8080
```

Receiver output

```
Waiting... connect sender
```

```
Waiting 1, 1, 1 2 95 1., ACK1
Message: 1.
Waiting 0, 2, 0 3 23 , ACK0
Waiting 0, 3, 0 5 317 You, ACK0
Waiting 1, 4, 1 6 312 are, ACK1
Waiting 0, 5, 0 7 231 my, ACK1
Waiting 1, 6, 1 8 924 sunshine., ACK0
Message:
 You are my sunshine.
Waiting 0, 7, 0 9 23 , ACK0
Waiting 0, 8, 0 11 96 2., ACK0
Message:
 2.
Waiting 1, 9, 1 12 23 , ACK1
Waiting 0, 10, 0 13 199 My, ACK1
Waiting 1, 11, 1 14 451 only, ACK0
Waiting 0, 12, 0 15 878 sunshine, ACK1
Waiting 1, 13, 1 16 23 , ACK1
```

This output shows us that the message is outputted whenever there is a period. Then there is a new line which looks like a packet with no content. Then the next message to be outputted will be until the next period. Notice at the end there is no period so the message is not outputted.

Run on storm

```
> javac *.java
> java sender hxr.cise.ufl.edu 8080 message.txt
```

Sender output

```
Waiting: ACK2, 0, DROP, resend packet0.
Waiting: ACK1, 0, ACK1, no more packets to send.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK2, 0, DROP, resend packet1.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK1, 0, ACK1, no more packets to send.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK1, 0, ACK1, no more packets to send.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK2, 0, DROP, resend packet1.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK1, 0, ACK1, no more packets to send.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK1, 0, ACK1, no more packets to send.
Waiting: ACK0, 0, ACK0, no more packets to send.
Waiting: ACK1, 0, ACK1, no more packets to send.
```

The sender's activity is to be expected. You will notice some of the packets are dropped and when

there is a drop the packet must be sent again

Bugs and Missing Items

I do not believe I have any bugs but I may not have properly interpreted the instructions. I am purposely outputting ACK-1 when the last packet has been sent because I did not see anywhere that it said the ACK had to have a certain value on this case. If I am mistaken, I do understand. Also, as my code is, it does not handle blank lines so if the message were instead:

```
1.
You are my sunshine.

2.
My only sunshine
```

My code would stop at the blank line. I had this changed but then I had an extra new line in the output message because of the way that I add a new line to the end of each read line from the sender.