Directions

A. Due Date: 26 Dec 2024 at 11:55pm

B. The homework may be done in groups of up to two students.

What to turn in

C. Turn in all related source code (.java files) along with any headers or supplemental libraries needed to compile the code.

D. The GitHub repository includes several GitHub Actions scripts that will compile your code and perform test runs on it. You can see the output from the actions scripts in the Actions tab on GitHub. Those outputs will be the basis for your grade.

- Do not submit compiled .class or .jar files. The Actions scripts will build them for you.
- Do not change the package names in the starter code.
- Do not change the name of the class with the `main` methods in them (SlidingWindowReceiver.java and SlidingWindowSender.java). You made add more classes or packages as necessary, but you may not remove files.

E. Turn in all related source code along with any headers or build files needed to compile the code in the GitHub repository.

F. Modify the provided README.md file to include the following information:

- Names of all students in the group
- Total number of hours spent on each part of the assignment
- Date of submission
- Comments or feedback on the assignment's requirements or complexity (optional)

G. A complete README.md is worth 5 points on the assignment.

How to submit

H. Turn in your submission via the starter repositories opened for you on GitHub via GitHub Classroom.

I. All submissions **must** be made in the starter GitHub repository for the assignment that is opened via GitHub classroom. Code placed anywhere else or submitted in any other manner - including via email or in unrelated repositories - will not be graded.

J. Indicate that your work is complete by performing a commit with the comment "Submitted for grading" on each repository. The grade for the assignment will be based on the first commit with that comment, so create a submission with that comment only when you are finished.

CRC and Sliding Window

1 Sliding Window Simulator with CRC

Having prepared tools that can do simple sending on a network using UDP, we will now build a sliding window simulator based on UDP. I recommend that you use the tool you developed in the previous assignment as a basis for the sliding window tool. Adapt the programs to support sliding window sending along with CRC32 packet integrity checking. The tools must allow the user to configure the sliding window parameters. We can then use the sliding window simulator to compare theoretical sending times to actual ones on the network.

The sliding window simulator will consist of two programs:

1. A sending tool that sends a file to the receiving program using UDP via the sliding window algorithm.
2. A receiving tool the receives a file from the sender using UDP and the sliding window algorithm.

1.1 Sliding Window Protocol

The sending program will implement the *sliding window protocol* as follows. Let us assume the file needs n packets to be sent. The sender behaves as follows:

New Sending If there is a packet in the sending window that can be sent and hasn't been sent yet, it is sent. If there are multiple new packets that can be sent at once, they are sent in ascending order (*e.g.* Packet 2 then Packet 3 then Packet 4).

Time Out If a packet was sent more than $2 \times RTT$ time ago and a corresponding ACK wasn't received yet, the packet is resent. The timer is then reset for $2 \times RTT$ after the resending has completed.

ACK Arrives When an ACK arrives (*e.g.* ACK 5), the sending window is advanced (*e.g.* Packets 1 - 5 are acknowledged and are no longer in the sending window).

Completion If the ACK arrival causes us to finish the sending the data (*e.g.* $n = 5$ and there are 5 packets total), the sending program sends a final packet containing only the byte -1. After the ACK on the final packet arrives, the tool prints out the data listed below in Section 1.5.1, and quits.

1.2 Receiver Protocol

The receiving program will implement the sliding window protocol as follows.

Packet in the Receiving Window Arrives The receiving program checks what is the highest ACK value (*e.g.* X) that can be sent such that X and all previous packets have successfully arrived and sends ACK X to the sending program.

The packet's data is written out to the output file.

Packet not in the Receiving Window Arrives The receiving program <u>does not</u> enqueue the packet. It finds the highest ACK value (*e.g.* X) that can be sent such that X and all previous packets have successfully arrived and sends ACK X to the sending program.

Packet with invalid CRC arrives The receiving program <u>does not</u> enqueue the packet. It finds the highest ACK value (*e.g.* X) that can be sent such that X and all previous packets have successfully arrived and sends ACK X to the sending program.

Completion If the packet is the final packet (*i.e.* it contains only -1 (a single byte of value -1)), the receiving tool sends an appropriate ACK message, closes the output file, prints the data listed in Section 1.6.1, and quits.

1.3 Use of CRC32

Each packet sent by the sender must include a CRC32 value. To calculate the value, use the Crc32 class in Java (`https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/zip/CRC32.html`.

The class includes several useful methods that you can use to calculate and verify CRC remainders:

`update` Adds bytes to the calculation.

`getValue` Gets the remainder for the bytes provided so far. Note that the value returned is a `long` (8 bytes). You most likely will need to cast it to an `int` (4 bytes) to use it.

`reset` Resets the bytes in the calculation. Be sure to reset the CRC32 object between packets if you use a single Crc32 object for multiple packets.

1.4 Packet formatting

To make the tool writing easier, we will use the following packet format for all messages from the sender:

bytes	1 2 3 4	5 6 7 8	9 +...
content	Packet #	CRC32 value	Packet data

The packet # field is an int (4 bytes) that starts from 0 and represents the packet number. The CRC32 value is an int (4 bytes) that represents the CRC32 remainder received from the Crc32 class. The packet data is a byte array of the data in the packet.

You can use the `SlidingWindowPacket` class provided with the starter code to create and parse packets of that format.

ACK packets must have the following format:

bytes	1 2 3 4
content	Packet #

You can also wrap the int as a String if you wish.

1.5 Sending Tool

The sending tool shall be a command line tool that accepts the following command line parameters. The parameters may be in any order.

dest The IP address to send to (*e.g.* 127.0.0.1, 0.0.0.0, 10.1.201.15)

port The listening port the recipient uses. Must be [1025-65535] (*e.g.* 5000, 5656)

f The name of the file to send (*e.g.* tests/file1.txt, file2.pdf)

packetsize The data packet size (Bytes). Must be a positive integer. (*e.g.* 100)

sws The sending window size (Packets). Must be a positive integer. (*e.g.* 50, 5)

rtt The RTT to assume for timeout calculations (ms). Must be a positive integer. (*e.g.* 10, 100)

droplist Optional. A comma separated list of packets to intentionally drop. (*e.g.* 1,2,30 means drop packet 1, packet 2, and packet 30). Dropping is accomplished by sending an intentionally invalid CRC remainder with the packet. Packets with the numbers listed are intentionally given an invalid

CRC remainder the first time they are sent. Subsequent times, they must be sent with a correct CRC remainder. The drop list does not need to be provided in sorted order (*e.g.* 5,1,20 means drop packet 1, packet 5, and packet 20). If a packet number appears more than once, just ignore the duplicate values (*e.g.* 1,2,2,500,2 means drop packet 1, packet 2, packet 500. Each packet is dropped only the first time it is sent).

If any parameter is missing, quit with a usage message. See the tests below for the expected usage message.

If any parameter is illegal, quit with an error message and a usage message. See the tests below for expected error messages.

1.5.1 Sending Tool Output

The sending tool must output a log of events and actions that it performs, including:

- When a packet is sent. Example: Sent packet 0
- When an ACK is received. Example: Received ACK on 0
- When the final packet is sent. Include the name of the file sent. Example: Sent final packet for tests/file1.txt
- When a timeout occurs. Example: Timed out waiting for ACK
- When a packet is resent. Example: Resent packet 5

1.5.2 Test Cases (85 points total)

The following are test inputs and outputs assuming the tool is called `SlidingWindowSender-5785.jar` using the test files provided in the starter repository. Every test must begin with `java -jar`, but those commands are skipped for brevity in all items listed below except for the first one.

Some test outputs are shortened for brevity. Complete sample outputs can be found in the starter repository.

1. `java -jar SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5000 -f=tests/file1.txt -sws=3 -packetsize=100 -rtt=100`

```
Sent packet 0
Sent packet 1
Sent packet 2
Received ACK on 0
Sent packet 3
Received ACK on 1
Sent packet 4
Received ACK on 2
Sent packet 5
Received ACK on 3
Sent packet 6
Received ACK on 4
Sent packet 7
Received ACK on 5
Sent packet 8
Received ACK on 6
Sent packet 9
Received ACK on 7
Sent packet 10
Received ACK on 8
Sent packet 11
Received ACK on 9
Sent packet 12
Received ACK on 10
Sent packet 13
Received ACK on 11
Sent packet 14
Received ACK on 12
Sent packet 15
Received ACK on 13
Sent packet 16
Received ACK on 14
Sent packet 17
Received ACK on 15
Sent packet 18
Received ACK on 16
Sent packet 19
Received ACK on 17
Sent packet 20
Received ACK on 18
```

```
Sent packet 21
Received ACK on 19
Sent packet 22
Received ACK on 20
Sent packet 23
Received ACK on 21
Sent packet 24
Received ACK on 22
Sent packet 25
Received ACK on 23
Sent packet 26
Received ACK on 24
Sent packet 27
Received ACK on 25
Received ACK on 26
Received ACK on 27
Sent final packet for tests/file1.txt
Received ACK 28
```

2. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5001 -f=tests/file1.txt -sws=10 -packetsize=100 -rtt=150`

```
Sent packet 0
Sent packet 1
Sent packet 2
Sent packet 3
Sent packet 4
Sent packet 5
Sent packet 6
Sent packet 7
Sent packet 8
Sent packet 9
Received ACK on 0
Sent packet 10
Received ACK on 1
Sent packet 11
Received ACK on 2
Sent packet 12
Received ACK on 3
Sent packet 13
Received ACK on 4
Sent packet 14
Received ACK on 5
Sent packet 15
Received ACK on 6
Sent packet 16
Received ACK on 7
Sent packet 17
Received ACK on 8
Sent packet 18
Received ACK on 9
Sent packet 19
Received ACK on 10
Sent packet 20
Received ACK on 11
Sent packet 21
Received ACK on 12
Sent packet 22
Received ACK on 13
Sent packet 23
Received ACK on 14
Sent packet 24
Received ACK on 15
Sent packet 25
Received ACK on 16
Sent packet 26
Received ACK on 17
Sent packet 27
Received ACK on 18
Received ACK on 19
Received ACK on 20
Received ACK on 21
Received ACK on 22
Received ACK on 23
Received ACK on 24
Received ACK on 25
Received ACK on 26
Received ACK on 27
Sent final packet for tests/file1.txt
Received ACK 28
```

3. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5002 -f=tests/file1.txt -sws=20 -packetsize=50 -rtt=80`

```
Sent packet 0
Sent packet 1
Sent packet 2
Sent packet 3
Sent packet 4
Sent packet 5
Sent packet 6
Sent packet 7
Sent packet 8
Sent packet 9
Sent packet 10
Sent packet 11
Sent packet 12
Sent packet 13
```

```
Sent packet 14
Sent packet 15
Sent packet 16
Sent packet 17
Sent packet 18
Sent packet 19
Received ACK on 0
Sent packet 20
...
Sent packet 54
Received ACK on 35
Received ACK on 36
Received ACK on 37
Received ACK on 38
Received ACK on 39
Received ACK on 40
Received ACK on 41
Received ACK on 42
Received ACK on 43
Received ACK on 44
Received ACK on 45
Received ACK on 46
Received ACK on 47
Received ACK on 48
Received ACK on 49
Received ACK on 50
Received ACK on 51
Received ACK on 52
Received ACK on 53
Received ACK on 54
Sent final packet for tests/file1.txt
Received ACK 55
```

4. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5003 -f=tests/file2.txt -sws=5 -packetsize=1000 -rtt=200 -droplist=10`

```
Sent packet 0
Sent packet 1
Sent packet 2
Sent packet 3
Sent packet 4
Received ACK on 0
Sent packet 5
...
Received ACK on 1048
Sent final packet for tests/file2.txt
Received ACK 1049
```

5. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5004 -f=tests/file2.txt -sws=40 -packetsize=1020 -rtt=250 -droplist=10,550`

```
Sent packet 0
...
Sent packet 10
...
Received ACK on 0
Sent packet 40
Received ACK on 1
Sent packet 41
Received ACK on 2
Sent packet 42
Received ACK on 3
Sent packet 43
Received ACK on 4
Sent packet 44
Received ACK on 5
Sent packet 45
Received ACK on 6
Sent packet 46
Received ACK on 7
Sent packet 47
Received ACK on 8
Sent packet 48
Received ACK on 9
Sent packet 49
Received ACK on 9
Received ACK on 9
...
Timed out waiting for ACK
Resent packet 10
Resent packet 11
...
Resent packet 49
Received ACK on 29
Sent packet 50
Sent packet 51
...
Sent packet 550
Received ACK on 511
...
Sent packet 589
Received ACK on 549
Received ACK on 549
...
Timed out waiting for ACK
Resent packet 550
Resent packet 551
...
```

Sent final packet for tests/file2.txt
Received ACK 1029

6. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5005 -f=tests/file2.txt -packetsize=500 -sws=50 -rtt=11 -droplist=1,10,20,600`

   Sent packet 0
   Sent packet 1
   Sent packet 2
   Sent packet 3
   ...
   Sent final packet for tests/file2.txt
   Received ACK 2098

7. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5006 -f=tests/file3.pdf -sws=40 -packetsize=600 -rtt=100 -droplist=5,100,58,200`

   Sent packet 0
   ...
   Sent final packet for tests/file3.pdf
   Received ACK 685

8. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5007 -f=tests/file3.pdf -sws=50 -packetsize=700 -rtt=150 -droplist=1,2,3,100,200,300,400,500`

   Sent packet 0
   ...
   Sent final packet for tests/file3.pdf
   Received ACK 587

9. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5008 -f=tests/file3.pdf -sws=100 -packetsize=30 -rtt=200 -droplist=2,3,52,800,801,802,900,1024,1025,3003,4004,5005,6006,7007,13000`

   Sent packet 0
   ...
   Sent final packet for tests/file3.pdf
   Received ACK 13692

10. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5009 -f=tests/file3.pdf -sws=70 -packetsize=25 -rtt=150 -droplist=10,25,85,90,97,100,200,1000`

    Sent packet 0
    ...
    Sent final packet for tests/file3.pdf
    Received ACK 16430

11. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5010 -f=tests/file3.pdf -sws=85 -packetsize=900 -rtt=300 -droplist=50,450,1000`

    Sent packet 0
    ...
    Sent final packet for tests/file3.pdf
    Received ACK 456

12. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5011 -f=tests/file1.txt -sws=20 -packetsize=80 -rtt=400 -droplist=2,3,30,40`

    Sent packet 0
    ...

Sent final packet for tests/file1.txt
Received ACK 34

13. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5012 -f=tests/file2.txt -sws=120 -packetsize=95 -rtt=200 -droplist=5,15,25,9,71,501,2000,10000,10500`

    Sent packet 0
    ...
    Sent final packet for tests/file2.txt
    Received ACK 11038

14. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5013 -f=tests/file1.txt -sws=10 -packetsize=122 -rtt=800 -droplist=7`

    Sent packet 0
    ...
    Sent final packet for tests/file1.txt
    Received ACK 23

15. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5014 -f=tests/file2.txt -sws=120 -packetsize=90 -rtt=10 -droplist=11,23,800,1200,4000,4002,5000,10000,10100,11000,11111`

    Sent packet 0
    ...
    Sent final packet for tests/file2.txt
    Received ACK 11651

16. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5014 -f=tests/file2.txt -sws=120 -packetsize=90 -droplist=11,23,800,1200,4000,4002,5000,10000,10100,11000,11111`

    Missing parameter. Expected output:

    ```
    Syntax:  SlidingWindowClient-5785 -dest=ip -port=p -f=filename -packetsize=bytes
    -sws=size -rtt=ms [-droplist=1,2]
    -droplist is optional.  Refers to packets that will be sent incorrectly the first
    time.  The list should be comma separated without spaces.  Example:  1,3,4
    ```

17. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5014 -f=tests/file2.txt -sws=120 -rtt=10 -droplist=11,23,800,1200,4000,4002,5000,10000,10100,11000,11111`

    Missing parameter. Expected output:

    ```
    Syntax:  SlidingWindowClient-5785 -dest=ip -port=p -f=filename -packetsize=bytes
    -sws=size -rtt=ms [-droplist=1,2]
    -droplist is optional.  Refers to packets that will be sent incorrectly the first
    time.  The list should be comma separated without spaces.  Example:  1,3,4
    ```

18. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=abc -f=tests/file2.txt -sws=120 -packetsize=90 -rtt=10 -droplist=11,23,800,1200,4000,4002,5000,10000,10100,11000,11111`

    Illegal parameter. Expected output:

    ```
    Error parsing port:  For input string:  "abc"
    Syntax:  SlidingWindowClient-5785 -dest=ip -port=p -f=filename -packetsize=bytes
    -sws=size -rtt=ms [-droplist=1,2]
    -droplist is optional.  Refers to packets that will be sent incorrectly the first
    time.  The list should be comma separated without spaces.  Example:  1,3,4
    ```

19. `SlidingWindowSender-5785.jar -dest=127.0.0.1 -port=5013 -f=tests/file1.txt -sws=-10 -packetsize=122 -rtt=800 -droplist=7`

    Illegal parameter. Expected output:

    ```
    Error:  SWS must be positive.
    Syntax:  SlidingWindowClient-5785 -dest=ip -port=p -f=filename -packetsize=bytes
    -sws=size -rtt=ms [-droplist=1,2]
    -droplist is optional.  Refers to packets that will be sent incorrectly the first
    time.  The list should be comma separated without spaces.  Example:  1,3,4
    ```

1.6 Receiving Tool

The receiving tool shall be a command line tool that accepts the following command line parameters. The parameters may be in any order.

ip The IP address to listen on (*e.g.* 127.0.0.1, 0.0.0.0, 10.1.201.15)

port The listening port to use. Must be [1025-65535] (*e.g.* 5000, 5656)

outfile A path or filename to write the output to. (*e.g.* outfiles/file1.txt, file2.pdf)

rws The receiving window size (Packets). Must be a positive integer. (*e.g.* 50, 5)

If any parameter is missing, quit with a usage message. See the tests below for the expected usage message.

If any parameter is illegal, quit with an error message and a usage message. See the tests below for expected error messages.

1.6.1 Receiving Tool Output

The receiving tool outputs a log of events and actions that it performs, including:

- An opening listening message. Example: Listening...
- When a packet is received (in the receiving window) with a valid CRC. Example: Received packet 0 valid CRC
- When a packet is received (in the receiving window) with an invalid CRC. Example: Received packet 10 invalid CRC
- When a packet is received but is outside the receiving window. Example: Received packet 608 valid CRC outside receive window
- When an ACK is sent. Example: Acked: 4
- When the final packet is received and the file is closed. Include the name of the file output and the highest value packet received. Example: File outfiles/Test4-file2-out.txt completed. Received 1049 packets

1.6.2 Test Cases (85 points total)

The following are test inputs and outputs assuming the tool is called `SlidingWindowReceiver-5785.jar` using the test files provided in the starter repository. Every test must begin with `java -jar`, but those commands are skipped for brevity in all items listed below except for the first one. The test outputs shown here are for the corresponding sliding window sender tests above (*e.g.* Test 1 here for Test 1 above, Test 2 here for Test 2 above, etc.)

Some test outputs are shortened for brevity. Complete sample outputs can be found in the starter repository.

1. `java -jar SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5000 -outfile=outfiles/Test1-file1-out.txt -rws=3`

```
Listening...
Received packet 0 valid CRC
Acked: 0
Received packet 1 valid CRC
Acked: 1
Received packet 2 valid CRC
Acked: 2
Received packet 3 valid CRC
Acked: 3
Received packet 4 valid CRC
Acked: 4
Received packet 5 valid CRC
Acked: 5
Received packet 6 valid CRC
Acked: 6
Received packet 7 valid CRC
Acked: 7
Received packet 8 valid CRC
Acked: 8
Received packet 9 valid CRC
Acked: 9
Received packet 10 valid CRC
Acked: 10
Received packet 11 valid CRC
Acked: 11
Received packet 12 valid CRC
Acked: 12
Received packet 13 valid CRC
Acked: 13
Received packet 14 valid CRC
Acked: 14
Received packet 15 valid CRC
Acked: 15
Received packet 16 valid CRC
Acked: 16
Received packet 17 valid CRC
Acked: 17
Received packet 18 valid CRC
Acked: 18
Received packet 19 valid CRC
Acked: 19
Received packet 20 valid CRC
Acked: 20
Received packet 21 valid CRC
Acked: 21
Received packet 22 valid CRC
Acked: 22
Received packet 23 valid CRC
Acked: 23
Received packet 24 valid CRC
Acked: 24
Received packet 25 valid CRC
Acked: 25
Received packet 26 valid CRC
Acked: 26
Received packet 27 valid CRC
Acked: 27
Acked: 28
File outfiles/Test1-file1-out.txt completed. Received 28 packets
```

2. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5001 -outfile=outfiles/Test2-file1-out.txt -rws=5`

```
Listening...
Received packet 0 valid CRC
Acked: 0
Received packet 1 valid CRC
Acked: 1
Received packet 2 valid CRC
Acked: 2
Received packet 3 valid CRC
Acked: 3
Received packet 4 valid CRC
Acked: 4
Received packet 5 valid CRC
Acked: 5
Received packet 6 valid CRC
Acked: 6
Received packet 7 valid CRC
Acked: 7
Received packet 8 valid CRC
Acked: 8
Received packet 9 valid CRC
Acked: 9
Received packet 10 valid CRC
Acked: 10
Received packet 11 valid CRC
Acked: 11
Received packet 12 valid CRC
Acked: 12
Received packet 13 valid CRC
Acked: 13
Received packet 14 valid CRC
Acked: 14
Received packet 15 valid CRC
Acked: 15
```

```
Received packet 16 valid CRC
Acked: 16
Received packet 17 valid CRC
Acked: 17
Received packet 18 valid CRC
Acked: 18
Received packet 19 valid CRC
Acked: 19
Received packet 20 valid CRC
Acked: 20
Received packet 21 valid CRC
Acked: 21
Received packet 22 valid CRC
Acked: 22
Received packet 23 valid CRC
Acked: 23
Received packet 24 valid CRC
Acked: 24
Received packet 25 valid CRC
Acked: 25
Received packet 26 valid CRC
Acked: 26
Received packet 27 valid CRC
Acked: 27
Acked: 28
File outfiles/Test2-file1-out.txt completed. Received 28 packets
```

3. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5002 -outfile=outfiles/Test3-file1-out.txt -rws=7`

   Listening...
   Received packet 0 valid CRC
   Acked: 0
   ...
   Acked: 54
   Acked: 55
   File outfiles/Test3-file1-out.txt completed. Received 55 packets

4. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -rws=10 -port=5003 -outfile=outfiles/Test4-file2-out.txt`

   Listening...
   Received packet 0 valid CRC
   Acked: 0
   ...
   Received packet 10 invalid CRC
   ...
   Acked: 1048
   Acked: 1049
   File outfiles/Test4-file2-out.txt completed. Received 1049 packets

5. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5004 -outfile=outfiles/Test5-file2-out.txt -rws=20`

   Listening...
   Received packet 0 valid CRC
   Acked: 0
   ...
   Received packet 10 invalid CRC
   ...
   Received packet 550 invalid CRC
   ...
   Acked: 1028
   Acked: 1029
   File outfiles/Test5-file2-out.txt completed. Received 1029 packets

6. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5005`

`-outfile=outfiles/Test6-file2-out.txt -rws=30`

```
Listening...
Received packet 0 valid CRC
Acked: 0
Received packet 1 invalid CRC
Received packet 1 invalid CRC
Received packet 10 invalid CRC
Received packet 20 invalid CRC
Received packet 31 valid CRC outside receive window
Received packet 32 valid CRC outside receive window
Received packet 33 valid CRC outside receive window
...
Received packet 2 valid CRC outside receive window
Received packet 3 valid CRC outside receive window
...
Received packet 600 invalid CRC
...
Received packet 630 valid CRC outside receive window
Received packet 631 valid CRC outside receive window
Received packet 632 valid CRC outside receive window
...
Acked: 2096
Received packet 2097 valid CRC
Acked: 2097
Acked: 2098
File outfiles/Test6-file2-out.txt completed. Received 2098 packets
```

7. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5006 -outfile=outfiles/Test7-file3-out.pdf -rws=40`

   Listening...
   Received packet 0 valid CRC
   Acked: 0
   Received packet 1 valid CRC
   Received packet 5 invalid CRC
   Received packet 6 valid CRC outside receive window
   ...
   Received packet 58 invalid CRC
   ...
   Received packet 59 valid CRC outside receive window
   ...
   Acked: 683
   Received packet 684 valid CRC
   Acked: 684
   Acked: 685
   File outfiles/Test7-file3-out.pdf completed. Received 685 packets

8. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5007 -outfile=outfiles/Test8-file3-out.pdf -rws=50`

   Listening...
   Received packet 0 valid CRC
   Acked: 0
   ...
   Acked: 585
   Received packet 586 valid CRC
   Acked: 586
   Acked: 587
   File outfiles/Test8-file3-out.pdf completed. Received 587 packets

9. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5008 -outfile=outfiles/Test9-file3-out.pdf -rws=60`

   Listening...
   Received packet 0 valid CRC
   Acked: 0

Received packet 1 valid CRC
...
Acked: 13690
Received packet 13691 valid CRC
Acked: 13691
Acked: 13692
File outfiles/Test9-file3-out.pdf completed. Received 13692 packets

10. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5009 -outfile=outfiles/Test10-file3-out.pdf -rws=70`

    Listening...
    Received packet 0 valid CRC
    Acked: 0
    Received packet 1 valid CRC
    ...
    Acked: 16428
    Received packet 16429 valid CRC
    Acked: 16429
    Acked: 16430
    File outfiles/Test10-file3-out.pdf completed. Received 16430 packets

11. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5010 -outfile=outfiles/Test11-file3-out.pdf -rws=80`

    Listening...
    Received packet 0 valid CRC
    Acked: 0
    ...
    Acked: 456
    Received packet 456 valid CRC outside receive window
    Acked: 456
    Acked: 457
    File outfiles/Test11-file3-out.pdf completed. Received 457 packets

12. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5011 -outfile=outfiles/Test12-file1-out.txt -rws=19`

    Listening...
    Received packet 0 valid CRC
    Acked: 0
    ...
    Acked: 34
    Received packet 34 valid CRC outside receive window
    Acked: 34
    Acked: 35
    File outfiles/Test12-file1-out.txt completed. Received 35 packets

13. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5012 -outfile=outfiles/Test13-file2-out.txt -rws=100`

    Listening...
    Received packet 0 valid CRC
    Acked: 0
    ...

Acked: 11036
Received packet 11037 valid CRC
Acked: 11037
Acked: 11038
File outfiles/Test13-file2-out.txt completed. Received 11038 packets

14. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5013 -outfile=outfiles/Test14-file1-out.txt -rws=110`

    Listening...
    Received packet 0 valid CRC
    Acked: 0
    ...
    Acked: 21
    Received packet 22 valid CRC
    Acked: 22
    Acked: 23
    File outfiles/Test14-file1-out.txt completed. Received 23 packets

15. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5014 -outfile=outfiles/Test15-file2-out.txt -rws=110`

    Listening...
    Received packet 0 valid CRC
    Acked: 0
    ...
    Acked: 11649
    Received packet 11650 valid CRC
    Acked: 11650
    Acked: 11651
    File outfiles/Test15-file2-out.txt completed. Received 11651 packets

16. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5014 -outfile=outfiles/Test16-file2-out.txt`

    Missing parameter. Expected output:

    ```
    Syntax:  SlidingWindowReceiver-5785 -ip=ip -port=p -outfile=f -rws=r
    ```

17. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5014 -rws=110`

    Missing parameter. Expected output:

    ```
    Syntax:  SlidingWindowReceiver-5785 -ip=ip -port=p -outfile=f -rws=r
    ```

18. `SlidingWindowReceiver-5785.jar -ip=127.0.0.1 -port=5014 -outfile=outfiles/Test18-file2-out.txt -rws=-5`

    Illegal parameter. Expected output:

    ```
    Error:  RWS must be positive
    Syntax:  SlidingWindowReceiver-5785 -ip=ip -port=p -outfile=f -rws=r
    ```

19. `SlidingWindowReceiver-5785.jar -ip=300.0.1 -port=5013 -outfile=outfiles/Test19-file1-out.txt -rws=110`

    Illegal parameter. Expected output:

```
Error parsing listening address:  300.0.1:  Name or service not known
Syntax:  SlidingWindowReceiver-5785 -ip=ip -port=p -outfile=f -rws=r
```

1.7 Documentation (10 points)

Add Javadoc documentation to every method and class. The documentation for methods must include:

- A 1-2 sentence summary of the method's purpose
- @param entries for all parameters, including what they are used for
- @return entry with a 1-2 sentence description of the return value
- @throws entries for any exceptions thrown.

The documentation for classes must includes:

- A 2-3 sentence description of the class' purpose
- @author the code's author
- @version a version for the class. Update on every commit to the repository.