HOW TO RUN THIS PROGRAM:
Python scripts are included in "~/proj_files/". Generate input-files in that (~/proj_files/) directory and then do the following:

With apache-ant installed, **IN THE PROGRAM ROOT DIRECTORY**, do:

1) ant compile;
2) ant jar;
3) ant -Dprop1="some_integer" run;

Where "some_integer" is the number of generated input-files. If the flag is incorrect, or a bad string is passed, program usage output will be generated and the ring will exit.

Feature	Status/Description
Load input file	Complete
Parse input file	Complete
Generate token	Complete
Generate frame(s)	Complete
Node runs as an autonomous thread	Complete
All nodes terminate after all data in the network were transmitted	Complete
Randomly accept a frame	Complete. Chance of frame rejection is 20%
Extract all fields from frame	Complete
Switch between listen and transmit states	Complete
Pass the tokens and frames to the next neighbor	Complete
Print to the output file with correct format	Complete
Documentation and Comments	Complete
Executable files	Complete
Monitor detects lost token	Complete. Chance of losing a token is 2%. I modified this value as large rings (num nodes > 10) would take forever to transmit their data at the 5% value specified (tokens kept getting lost before reaching the last

	nodes).
Monitor cleans up garbled frame	Complete. Garbled frames are generated at a 1% rate.
Monitor detects orphaned frames	Complete
Ant build file	Complete
Terminal output	Complete. The program generates output that indicates frame reception. frame transmission, garbled frames, lost tokens, monitor clean-up, and network shutdown.
Nodes and monitor implement STPLP protocol.	Complete
All sockets and files are closed on exit	Complete
Extra Credit Features	
Binary Frame	Complete. Frames are bumped as binary strings i.e. “00101010101111…”
Network shuts down using inter-node communication	Complete. Once a node is finished transmitting all frames and all frames have been successfully ACK’d, it alerts the monitor using the FS bits on the TOKEN. When the monitor receives the alerted TOKEN, it ACK’s and resets the token. When all nodes have alerted the monitor finished, the Monitor shuts down the ring and exits.
Node Priority	INCOMPLETE. Frames include priority bits, but priority transmission is NOT implemented. This means that input-file generation and output-file grading requires the priority scripts, but priority Tx does not work.

Known Bugs:
None

Token Holding Time:
The token holding time is equal to the max size of a frame (in bits) x 5. I wanted to keep the Tx time under 10ms, and this value satisfied that condition. Nodes check that the total

number of bits sent + size of the next frame is less than or equal to the THT before transmitting the frame.

Garbled Frames:
Garbled frames are created by removing the FS byte. They are detected by comparing the size of the garbled frame with the size of a non-garbled frame (garbled frames have 8 less bits).

File descriptions:
build/ directory holds compiled classes and .jar executable.
src/ directory holds Main, Node, Frame, and Monitor classes.
proj_files/ directory holds the python scripts and generated files.

Misc:

1) python scripts are including in "~/proj_files/", and all generated input-files, output-files, and sorted-output-files live in that directory.
2) Priority implementation is not complete, but frame input and output require the priority bits, so priority python scripts are necessary.
3) Everything works