How to build and run:

1. Navigate to the project file base directory
2. Run the command “ant”
    a. This will place built files into a file named “wfong_p1.jar”
3. Run the command “java -jar wfong_p1.jar”

Files:

- bin
    - Contains the built files
- src
    - Contains all the .java source files
- build.xml
    - Contains the Java ant build file
- Java Classes
    - Test.java
        - The main class which runs the entire program
    - Node.java
        - This is a super class which contains methods for transmitting and sending data as well as initializing sockets.
    - ServerNode.java
        - This is a subclass of Node and its constructors allow for the instantiated object to be ran on a separate thread.
        - This class’s threadable run method allows this Node to act as a server, waiting to receive client connections.
    - SenderNode.java
        - This is a subclass of Node and its constructors allow for the instantiated object to be ran on a separate thread.
        - This class’s theadable run method allows this Node to act as a client, sending data as soon as a connection is established
    - RelayNode.java
        - This is a subclass of Node and its constructors allow for the instantiated object to be ran on a separate thread.
        - This class’s threadable run method allows this node to act as a relay. It waits for client connections and waits until all data has been received. It then transmits the received data as well as its own to another node.

Feature		Description
Project code has appropriate comments	x	Complete
The project compiles and builds without errors	x	Complete

The program starts	x	Complete
Creates and runs all the threads	x	Complete (only threads are used)
Loads the configuration files	x	Partial. This program assumes all port data is correct.
Sends data through sockets	x	Complete
Receives data through sockets	x	Complete
Prints out received data	x	Partial, the Nodes print out data they receive, but not the data's origin
Program exits after data delivery/reception	x	Complete

Bug Report:

1. Valgrind shows that ~5KB of RAM is definitely lost. However I am unsure if this is the result of incorrectly closing all my sockets and datastreams or from a function within the java libraries I used.
2. If the nodes are instantiated with incorrect port numbers, the client nodes may attempt to connect forever (there is no maximum timeout)

Credit:

- I used Eclipse Version: Kepler Service Release 2 as my Java environment and GNU's Valgrind for memory leak testing.
- All the libraries used were within those supplied by Java
- Thanks to Skyler Manzanares for helping me with Ant build files