1. Weather Station Simulator
   1. Description of Sample Application

This example simulates a typical weather station consists of three main components i.e., WeatherStationSensor, Transmitter and a Receiver.

WeatherStationSensor, runs in a thread, generates weather-data readings at random intervals and temporary stores them in a queue, accessible to the Transmitter. On receiving a request weather-data from the Receiver in random intervals, the Transmitter sends all of the data available in the queue, one weather-data reading at a time and in order, to Receiver. Receiver periodically sends more requests for weather data if it don’t receive any data for some time period.

* 1. Design Enhancements

Figure 1 shows an overview of the current architecture for WeatherStationSimulator and protocol messages. The system contains three main classes i.e., WeatherStationSensor, Transmitter and Receiver. WeatherStationSensor generates WeatherDataVector(s) (weather-sensitive observations). Transmitter collects WeatherDataVector(s) and sends them to the Receiver. Figure 2 describes the WeatherStationSensor design. The UML Sequence Diagram in Figure 3 shows the transmitter/receiver interactions in more details.

Application runs two instances of Transmitter and one instance of Receiver. Each transmitter starts its own WeatherStationSensor thread. The sensor combines the readings from its various sub-components (Figure 2) into a WeatherDataReading object. It then generates an instance of WeatherDataVector message, and populates it with four WeatherDataReading instances, at random intervals, and stores in a temporary data structure.

There would be two instancesReceiver can requests the Transmitter to either SEND, PAUSE or STOP WeatherDataVector(s) as shown in Figure 1.

* If Receiver sends a WeatherDataRequest of type SEND to each Transmitter, Transmitters receive the request, and starts sending the stored WeatheDataVector(s), one at a time. After transferring all the WeatherDataVector(s), a transmitter sleeps unless the receiver notifies it again. When Receiver receives WeatherDataVector, it saves to a file and returns to the listening state. Receiver resends WeatheDataRequest of any value after random time interval.
* If Receiver sends PAUSE request, Transmitter interrupts sending of WeatherDataVector(s) and sleeps.
* If Receiver sends STOP request, Transmitter terminates itself.

## 

Figure 1: Interaction diagram between Transmitter (two threads) and Two Receivers