

```
package nmc.platform;

import java.io.IOException;

import javax.microedition.io.Datagram;
import javax.microedition.io.DatagramConnection;
import javax.microedition.io.Connector;

public class DatagramIoPort {
    private int dgmDataSize = 1024;
    public void changeRcvBufferSize(int newSize) {dgmDataSize = newSize; }
    public int getRcvBufferDfltSize() { return dgmDataSize; }
    private DatagramConnection dgmConnection;
    private int portNumber;
    private String ipAddr;

    static public DatagramIoPort GetRcvPort(int rxPortNumber)
        throws IOException {
        return new DatagramIoPort(rxPortNumber);
    }
    private DatagramIoPort(int rxPortNumber) throws IOException {
        this.portNumber = rxPortNumber;
        String localURL = "datagram://:" + rxPortNumber;
        this.dgmConnection = (DatagramConnection) Connector.open(
            localURL, Connector.READ_WRITE, true);
    }
    private DatagramIoPort(String destIpAddr, int destPortNumber)
        throws IOException {
        this.portNumber = destPortNumber;
        this.ipAddr = destIpAddr;

        // Try to open the connection; it will be read and written and the
        // code here will handle timeouts.
        String addrAsUrl = "datagram://" + destIpAddr + ":" + destPortNumber;
        this.dgmConnection = (DatagramConnection) Connector.open(
            addrAsUrl, Connector.READ_WRITE, true);
    }
    static public DatagramIoPort GetTxPort(String destIpAddr,
        int destPortNumber)
        throws IOException {
        return new DatagramIoPort(destIpAddr, destPortNumber);
    }

    public byte[] blockingRx() throws IOException {
        Datagram incoming = dgmConnection.newDatagram(dgmDataSize);
        dgmConnection.receive(incoming);
        byte[] payload = incoming.getData();
        return payload;
    }
    public void send(byte[] payload) throws IOException {
        Datagram dgm = dgmConnection.newDatagram(dgmDataSize);
        dgm.setData(payload, 0, payload.length);
        dgmConnection.send(dgm);
    }
}
```

```
package nmc.platform;

import java.util.Vector;

public class FifoQueue {

    private com.ajile.util.FifoQueue q;

    public FifoQueue(int capacity) {
        this.q = new com.ajile.util.FifoQueue(capacity);
    }

    public int enqueue(Object o) {
        return q.enqueue(o);
    }

    public Object blockingDequeue() {
        return q.blockingDequeue();
    }

    public boolean isEmpty() {
        return q.isEmpty();
    }
}
```