```
// VendMach.java - VENDING MACHINE CONTROLLER SIMULATOR
//
// MODULE INDEX
// NAME                                 CONTENTS
// VendMach                             Vending machine controller class
// VendMach.VendMach                    Construct a vending machine controller
// VendMach.main                        Main line
//
// MAINTENANCE HISTORY
// DATE         PROGRAMMER AND DETAILS
// 03-09-05     JS      Original
// 06-09-05     LSH     Modified to resend response for repeated messages
//
//-----------------------------------------------------------------------------

// IMPORTATIONS

import java.io.*;
import java.util.*;
import java.net.*;

//-----------------------------------------------------------------------------

// VENDING MACHINE CONTROLLER CLASS

class VendMach
{
    //-------------------------------------------------------------------------

    // CONSTRUCT A VENDING MACHINE CONTROLLER

    VendMach ()
    {
        ServerSocket    acceptSocket;   // Socket to accept client connections
        Socket          comSocket;      // Socket to communicate w/ client
        InputStream     comInputStream; // Communications input stream
        OutputStream    comOutputStream; // Communications output stream
        int             ofs;            // Offset in receive buffer
        int             rcvdLen;        // Received length
        byte []         reqBuf;         // Request buffer
        VendMux.Unpack  unpack;         // Unpacker reference
        VendMux.Pack    pack;           // Packer reference
        long            serialNo;       // Request serial number
        long            prevSerialNo;   // Previous request serial number
        long            available;      // Available funds
        long            consumed;       // Consumed amount
        Random          rand;           // A random number generator

        // Instantiate a random number generator

        rand = new Random ();

        // Initialize variables (make lint happy)

        prevSerialNo = -1;
        consumed = 0;

        // Open a socket to accept connections from the clients

        acceptSocket = null;
        try {
            acceptSocket = new ServerSocket (VendMux.VEND_PORT);
        }
        catch (IOException e) {
            throw new RuntimeException ("open listen port: " + e.toString());
        }

        // Loop to accept connections

        for (;;) {
            // Accept a connection from the listening socket

            try {
                comSocket = acceptSocket.accept ();
                comInputStream = comSocket.getInputStream();
                comOutputStream = comSocket.getOutputStream();

                // Loop to process requests
```

```
                for (;;) {
                    // Receive a request from the client

                    reqBuf = new byte [ VendMux.REQUEST_LEN ];
                    ofs = 0;
                    while (ofs < VendMux.REQUEST_LEN) {
                        rcvdLen = comInputStream.read (
                            reqBuf, ofs, VendMux.REQUEST_LEN-ofs
                        );
                        if (rcvdLen == -1)
                            throw new IOException ("end-of-file");
                        ofs += rcvdLen;
                    }

                    // Decode the request

                    unpack = new VendMux.Unpack (reqBuf);
                    serialNo = unpack.unpackNumber (4);
                    available = unpack.unpackNumber (4);

                    // If it's not a repeated message

                    if (serialNo != prevSerialNo) {

                        // Select the consumed amount

                        consumed = rand.nextInt ((int)available + 1);

                        // Update the previous serial number

                        prevSerialNo = serialNo;
                    }

                    // Encode the response

                    pack = new VendMux.Pack ( VendMux.RESPONSE_LEN );
                    pack.packNumber (serialNo, 4);
                    pack.packNumber (consumed, 4);

                    // Send the response

                    comOutputStream.write (pack.packBuf, 0, pack.packLen);

                    // Log the transaction

                    System.out.println ("VendMach: available=" + available
                        + "  consumed=" + consumed);
                }
            }
            catch (IOException e) {
                System.err.println ("Warning: vending machine: "
                    + e.toString());
            }
        }
    }

    //--------------------------------------------------------------------------

    // MAIN LINE

    static public void
    main (
        String []       argv)           // Argument values
    {
        new VendMach ();
    }
}
```