```java
import java.io.*;
import java.net.*;

public class EchoClient {
    public static void main(String[] args) throws IOException {

                Socket echoSocket = null;
                PrintWriter out = null;
                BufferedReader in = null;

                try {
                        String hostName = args[0];
                        echoSocket = new Socket(hostName, 7);
                        out = new PrintWriter(echoSocket.getOutputStream(), true);
                        in = new BufferedReader(new InputStreamReader(
                                echoSocket.getInputStream()));
                } catch (UnknownHostException e) {
                        System.err.println("unkown host");
                        System.exit(1);
                } catch (IOException e) {
                        System.err.println("Couldn't get I/O for "
                                + "the connection to host");
                        System.exit(1);
                } catch (ArrayIndexOutOfBoundsException aiobe) {
                        System.err.println("wrong usage: enter hostname");
                        System.exit(1);
                }

                // establish an input stream to read from the standard input
                BufferedReader input = new BufferedReader(
                        new InputStreamReader(System.in));
                String userInput;
                while ((userInput = input.readLine()) != null) {

                        // writer line to output stream
                        out.println(userInput);
                        out.flush();

                        // print received echo result
                        System.out.println("echo: " + in.readLine());
                }

                // close streams and socket
                out.close();
                in.close();
                input.close();
                echoSocket.close();
    }
}
```

```java
import java.net.*;
import java.io.*;

class EchoServer {

    public static void main(String[] args) {
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(7);
        } catch (IOException ioe) {
            System.err.println("Couldn’t listen on port 7");
            System.exit(1);
        }

        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException ioe) {
            System.out.println("Accept failed: 7");
            System.exit(-1);
        }

        try {
            PrintWriter out =
                    new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));

            String input = in.readLine();
            out.println(input);

            out.close();
            in.close();
            clientSocket.close();
            serverSocket.close();
        } catch (IOException ioe) {
            System.err.println("Couldn t communicate with client");
        }
    }
}
```

```java
import java.net.*;
import java.io.*;

class MultiClientEchoServer {

    public static void main(String[] args) {
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(7);
        } catch (IOException ioe) {
            System.err.println("Couldn’t listen on port 7");
            System.exit(-1);
        }

        while(true) {
            try {
                Socket clientSocket = serverSocket.accept();
                EchoClientHandler handler =
                        new EchoClientHandler(clientSocket);
                (new Thread(handler)).start();
            } catch (IOException ioe) {}
        }
    }
}

class EchoClientHandler implements Runnable {
    private Socket clientSocket;
    public EchoClientHandler(Socket clientSocket) {
        this.clientSocket = clientSocket;
    }
    public void run() {
        try {
            PrintWriter out =
                    new PrintWriter(clientSocket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));

            String input = null;
            while ((input = in.readLine()) != null) { // read from the client
                out.println(input);                   // write to client
            }

            out.close();
            in.close();
            clientSocket.close();
        } catch (IOException ioe) {
            System.err.println("couldn t communicate with client");
            System.exit(-1);
        }
    }
}
```

```
import java.net.*;
import java.io.*;

class SimpleClient {
        public static void main (String[] args)         {
                try {
                        Socket connection = new Socket (args[0], Integer.parseInt(args[1]));
                        System.out.println("connection establish");
                        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

                        PrintWriter writer =
                                new PrintWriter(new OutputStreamWriter(connection.getOutputStream()), true);

                        BufferedReader input = new BufferedReader(new InputStreamReader(connection.getInputStream()));
                        String line;
                        while(!(line = reader.readLine()).equalsIgnoreCase("exit")) {

                                writer.println(line);
                                String fromServer = input.readLine();
                                System.out.println("<from server> " + fromServer);
                        }
                        connection.close();
                }catch(IOException ioe) {
                        ioe.printStackTrace();
                }
        }
}

```

```java
import java.io.*;
import java.net.*;

public class TimeClient {

        // The Time protocol defined in RFC 867 returns the server's time

        private static final int TIME_PORT = 13;

    public static void main(String[] args)  throws Exception{
                String hostName = null;
                try {
                        hostName = args[0];
                        System.out.println(getServerTime(hostName));
                } catch (UnknownHostException e) {
                        System.err.println("Don't know about host" + hostName);
                        System.exit(1);
                } catch (IOException e) {
                        System.err.println("Couldn't get I/O for "
                                                                + "the connection to:" + hostName);
                        System.exit(1);
                } catch (ArrayIndexOutOfBoundsException aiobe) {
                        System.err.println("wrong usage: enter hostname");
                        System.exit(1);
                }
        }

        // returns the date from the specified host name using the time protocol
        private static String getServerTime(String hostName)
                throws IOException, UnknownHostException {
                        Socket timeSocket = new Socket(hostName, TIME_PORT);
                        BufferedReader reader = new BufferedReader(
                    new InputStreamReader(timeSocket.getInputStream()));
                        String serverTime = reader.readLine();
                        timeSocket.close();
                        return serverTime;
        }
}
```

```java
import java.io.*;
import java.net.*;

public class TimeClientUDP {

                // The Time protocol defined in RFC 867 returns the server's time

        // The well known time port
                private static final int TIME_PORT = 13;

        public static void main(String[] args)  throws Exception{
                        String hostName = null;
                        try {
                                hostName = args[0];
                                System.out.println(getServerTime(hostName));
                        } catch (UnknownHostException e) {
                                System.err.println("Don't know about host" + hostName);
                                System.exit(1);
                        } catch (IOException e) {
                                System.err.println("Couldn't get I/O for "
                                                                                + "the connection to:" + hostName);
                                System.exit(1);
                        } catch (ArrayIndexOutOfBoundsException aiobe) {
                                System.err.println("wrong usage: enter hostname");
                                System.exit(1);
                        }
                }

                private static String getServerTime(String hostName)
                        throws IOException, UnknownHostException {

                    DatagramSocket socket = new DatagramSocket();
                // send request
                byte[] buf = new byte[256];
                InetAddress address = InetAddress.getByName(hostName);
                DatagramPacket packet = new DatagramPacket(buf, buf.length, address, TIME_PORT);
                socket.send(packet);

                        // get response
                packet = new DatagramPacket(buf, buf.length);
                        socket.receive(packet);
                        BufferedReader input = new BufferedReader(new InputStreamReader(
                                        new ByteArrayInputStream(packet.getData())));

                         // display response
                String serverTime = input.readLine();
                socket.close();
                        return serverTime;
                }
}
```

```java
import java.net.*;
import java.io.*;

public class MonitorServer {

	public static void main (String[] args) throws IOException {

		ServerSocket serverSocket = new ServerSocket (Integer.parseInt(args[0]));
		while (true) {
			Socket socket = serverSocket.accept();
			String hostAddress = socket.getInetAddress().getHostAddress();
			String hostName = socket.getInetAddress().getHostName();
			int port = socket.getPort();
			System.out.println("==========================");
			System.out.println("Connection Established");
			System.out.println("client address: " + hostAddress);
			System.out.println("client name: " + hostName);
			System.out.println("client port: " + port);
			System.out.println("==========================\n\n");
		}
	}
}
```

```java
import java.net.*;
import java.io.*;

class SimpleClient {
	public static void main (String[] args)		{
		try {
			Socket connection = new Socket (args[0], Integer.parseInt(args[1]));
			System.out.println("connection establish");
			BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

			PrintWriter writer =
				new PrintWriter(new OutputStreamWriter(connection.getOutputStream()), true);

			BufferedReader input = new BufferedReader(new InputStreamReader(connection.getInputStream()));
			String line;
			while(!(line = reader.readLine()).equalsIgnoreCase("exit")) {

				writer.println(line);
				String fromServer = input.readLine();
				System.out.println("<from server> " + fromServer);
			}
			connection.close();
		}catch(IOException ioe) {
			ioe.printStackTrace();
		}
	}
}
```

```java
import java.net.*;
import java.io.*;

public class SimpleServer {

        private static final int PORT = 8888;
        private static final String GREETING = "hello !";

        public static void main(String[] arg) throws IOException {
                ServerSocket server = null;
                server = new ServerSocket(PORT);
                System.out.println("server is waiting for connection");
                Socket connection = server.accept();
                System.out.println("connection astablished");

                //talking
                BufferedReader reader = new BufferedReader(
                        new InputStreamReader (connection.getInputStream()));

                PrintWriter writer = new PrintWriter(
                        new OutputStreamWriter(connection.getOutputStream()), true);

                String line;
                while ((line = reader.readLine()) != null) {
                        System.out.println("<recived from client> " + line);
                        writer.println(line);
                }
                connection.close();
                server.close();

        }
}
```

```java
import java.io.*;
import java.net.*;

public class KnockKnockClient {

    public static void main(String[] args) throws IOException {

                Socket kkSocket = null;
                PrintWriter out = null;
                BufferedReader in = null;

                try {
                        kkSocket = new Socket(args[0], Integer.parseInt(args[1]));
                        out = new PrintWriter(kkSocket.getOutputStream(), true);
                        in = new BufferedReader(new InputStreamReader(kkSocket.getInputStream()));
                } catch (UnknownHostException e) {
                        System.err.println("Don't know about host: " + args[0]);
                        System.exit(1);
                } catch (IOException e) {
                        System.err.println("Couldn't get I/O for the connection to: " + args[0]);
                        System.exit(1);
                }

                BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
                String fromServer;
                String fromUser;

                while ((fromServer = in.readLine()) != null) {
                        System.out.println("Server: " + fromServer);
                        if (fromServer.equals("Bye."))
                                break;

                        fromUser = stdIn.readLine();
                        if (fromUser != null) {
                                System.out.println("Client: " + fromUser);
                                out.println(fromUser);
                        }
                }

                out.close();
                in.close();
                stdIn.close();
                kkSocket.close();
    }
}
```

```java
import java.net.*;
import java.io.*;

public class KnockKnockProtocol {
    private static final int WAITING = 0;
    private static final int SENTKNOCKKNOCK = 1;
    private static final int SENTCLUE = 2;
    private static final int ANOTHER = 3;

    private static final int NUMJOKES = 5;

    private int state = WAITING;
    private int currentJoke = 0;

    private String[] clues = { "Turnip", "Little Old Lady", "Atch", "Who", "Who" };
    private String[] answers = { "Turnip the heat, it’s cold in here!",
                                 "I didn’t know you could yodel!",
                                 "Bless you!",
                                 "Is there an owl in here?",
                                 "Is there an echo in here?" };

    public String processInput(String theInput) {
        String theOutput = null;

        if (state == WAITING) {
            theOutput = "Knock! Knock!";
            state = SENTKNOCKKNOCK;
        } else if (state == SENTKNOCKKNOCK) {
            if (theInput.equalsIgnoreCase("Who’s there?")) {
                theOutput = clues[currentJoke];
                state = SENTCLUE;
            } else {
                theOutput = "You’re supposed to say \"Who’s there?\"! " +
                            "Try again. Knock! Knock!";
            }
        } else if (state == SENTCLUE) {
            if (theInput.equalsIgnoreCase(clues[currentJoke] + " who?")) {
                theOutput = answers[currentJoke] + " Want another? (y/n)";
                state = ANOTHER;
            } else {
                theOutput = "You’re supposed to say \"" +
                            clues[currentJoke] +
                            " who?\"" +
                            "! Try again. Knock! Knock!";
                state = SENTKNOCKKNOCK;
            }
        } else if (state == ANOTHER) {
            if (theInput.equalsIgnoreCase("y")) {
                theOutput = "Knock! Knock!";
                if (currentJoke == (NUMJOKES - 1))
                    currentJoke = 0;
                else
                    currentJoke++;
                state = SENTKNOCKKNOCK;
            } else {
                theOutput = "Bye.";
                state = WAITING;
            }
        }
        return theOutput;
    }
}
```

```java
import java.net.*;
import java.io.*;

public class KnockKnockServer {
    public static void main(String[] args) throws IOException {

                ServerSocket serverSocket = null;
                try {
                        serverSocket = new ServerSocket(4444);
                } catch (IOException e) {
                        System.err.println("Could not listen on port: 4444.");
                        System.exit(1);
                }

                Socket clientSocket = null;
                while (true) {
                        try {
                                clientSocket = serverSocket.accept();
                        } catch (IOException e) {
                                System.err.println("Accept failed.");
                                System.exit(1);
                        }

                        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                        BufferedReader in = new BufferedReader(
                                new InputStreamReader(
                                clientSocket.getInputStream()));
                        String inputLine, outputLine;
                        KnockKnockProtocol kkp = new KnockKnockProtocol();

                        outputLine = kkp.processInput(null);
                        out.println(outputLine);

                        while ((inputLine = in.readLine()) != null) {
                                outputLine = kkp.processInput(inputLine);
                                out.println(outputLine);
                                if (outputLine.equals("Bye."))
                                        break;
                        }
                        out.close();
                        in.close();
                        clientSocket.close();
                }
        }
}
```

```java
import java.io.*;
import java.net.*;

public class KnockKnockClient {

    public static void main(String[] args) throws IOException {

                Socket kkSocket = null;
                PrintWriter out = null;
                BufferedReader in = null;

                try {
                        kkSocket = new Socket(args[0], Integer.parseInt(args[1]));
                        out = new PrintWriter(kkSocket.getOutputStream(), true);
                        in = new BufferedReader(new InputStreamReader(kkSocket.getInputStream()));
                } catch (UnknownHostException e) {
                        System.err.println("Don't know about host: " + args[0]);
                        System.exit(1);
                } catch (IOException e) {
                        System.err.println("Couldn't get I/O for the connection to: " + args[0]);
                        System.exit(1);
                }

                BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
                String fromServer;
                String fromUser;

                while ((fromServer = in.readLine()) != null) {
                        System.out.println("Server: " + fromServer);
                        if (fromServer.equals("Bye."))
                                break;

                        fromUser = stdIn.readLine();
                        if (fromUser != null) {
                                System.out.println("Client: " + fromUser);
                                out.println(fromUser);
                        }
                }

                out.close();
                in.close();
                stdIn.close();
                kkSocket.close();
    }
}
```

```
import java.net.*;
import java.io.*;

public class KnockKnockMultiServer {

	private static final int PORT = 4444;
	public static void main(String[] arg) {

		ServerSocket serverSocket = null;
		try {
			serverSocket = new ServerSocket(PORT);
		} catch (IOException e) {
			System.err.println("could not connect server");
			e.printStackTrace();
		}
		Socket connection = null;
		while(true) {
			try {
				connection = serverSocket.accept();
				new Thread(new KnockKnockTask(connection)).start();
			}catch (IOException ioe) {
				System.err.println(ioe);
				System.exit(1);
			}
		}
	}
}
```

```java
import java.net.*;
import java.io.*;

public class KnockKnockProtocol {
    private static final int WAITING = 0;
    private static final int SENTKNOCKKNOCK = 1;
    private static final int SENTCLUE = 2;
    private static final int ANOTHER = 3;

    private static final int NUMJOKES = 5;

    private int state = WAITING;
    private int currentJoke = 0;

    private String[] clues = { "Turnip", "Little Old Lady", "Atch", "Who", "Who" };
    private String[] answers = { "Turnip the heat, it’s cold in here!",
                                 "I didn’t know you could yodel!",
                                 "Bless you!",
                                 "Is there an owl in here?",
                                 "Is there an echo in here?" };

    public String processInput(String theInput) {
        String theOutput = null;

        if (state == WAITING) {
            theOutput = "Knock! Knock!";
            state = SENTKNOCKKNOCK;
        } else if (state == SENTKNOCKKNOCK) {
            if (theInput.equalsIgnoreCase("Who’s there?")) {
                theOutput = clues[currentJoke];
                state = SENTCLUE;
            } else {
                theOutput = "You’re supposed to say \"Who’s there?\"! " +
                            "Try again. Knock! Knock!";
            }
        } else if (state == SENTCLUE) {
            if (theInput.equalsIgnoreCase(clues[currentJoke] + " who?")) {
                theOutput = answers[currentJoke] + " Want another? (y/n)";
                state = ANOTHER;
            } else {
                theOutput = "You’re supposed to say \"" +
                            clues[currentJoke] +
                            " who?\"" +
                            "! Try again. Knock! Knock!";
                state = SENTKNOCKKNOCK;
            }
        } else if (state == ANOTHER) {
            if (theInput.equalsIgnoreCase("y")) {
                theOutput = "Knock! Knock!";
                if (currentJoke == (NUMJOKES - 1))
                    currentJoke = 0;
                else
                    currentJoke++;
                state = SENTKNOCKKNOCK;
            } else {
                theOutput = "Bye.";
                state = WAITING;
            }
        }
        return theOutput;
    }
}
```

```
import java.net.*;
import java.io.*;

public class KnockKnockTask implements Runnable {

        private Socket connection;

        public KnockKnockTask (Socket connection) {
                this.connection = connection;
        }

        public void run(){
                try {
                        BufferedReader in = new BufferedReader(
                                new InputStreamReader(connection.getInputStream()));
                        PrintWriter out = new PrintWriter(new OutputStreamWriter(
                                connection.getOutputStream()),true);
                        String inputLine , outputLine;
                        KnockKnockProtocol protocol = new KnockKnockProtocol();


                        outputLine = protocol.processInput(null);
                        out.println(outputLine);

                        while ((inputLine = in.readLine()) !=null) {
                                outputLine = protocol.processInput(inputLine);
                                out.println(outputLine);
                                if(outputLine.equals("Bye."))
                                        break;
                        }
                        out.close();
                        in.close();
                        connection.close();
                }catch(IOException e) {
                        System.out.println(e);
                        System.exit(1);
                }
        }
}
```

```java
import java.util.*;

/**
 * A stateless protocol for system properties
 * @pattern Singleton
 */
public class PropertiesProtocol {

	private static PropertiesProtocol instance;
	public static String FINAL_MESSAGE = "BYE";
	private Properties properties;


	// constructor private - singleton
	private PropertiesProtocol () {
		properties = System.getProperties();
	}

	/**
	 * returns the single instance of this protocol
	 */
	public static PropertiesProtocol getInstance () {
		if (instance == null) {
			instance = new PropertiesProtocol();
		}
		return instance;
	}

	/**
	 * process the client input; returns a string result for the input
	 */
	public String processInput (String input) {
		if (input.equalsIgnoreCase("list")) {
			return listProperties();
		}
		if (input.equalsIgnoreCase(FINAL_MESSAGE)) {
			return FINAL_MESSAGE;
		}
		String value = properties.getProperty(input);
		if (value == null) {
			return "no such property " + input;
		}
		return value;
	}

	// returns a String of all properties seperated by a semicolon
	private String listProperties () {
		StringBuffer buffer = new StringBuffer();
		Enumeration enum = properties.keys();
		while (enum.hasMoreElements()) {
			String next = (String)enum.nextElement();
			buffer.append(next);
			buffer.append(";");
		}
		return buffer.toString();
	}
}
```

```java
import java.net.*;
import java.io.*;

/**
 * A Server that responds to client queries about the server' system properties
 */
public class PropertiesServer {

        private static final int PORT = 4444;

        public static void main(String[] arg) {

                ServerSocket serverSocket = null;
                try {
                        serverSocket = new ServerSocket(PORT);
                } catch (IOException e) {
                        System.err.println("could not connect server");
                        e.printStackTrace();
                }
                Socket connection = null;
                while(true) {
                        try {
                                // for every client accepted handle conversation in a new thread
                                connection = serverSocket.accept();
                                new Thread(new PropertiesTask(connection)).start();
                        }catch (IOException ioe) {
                                System.err.println(ioe);
                                System.exit(1);
                        }
                }
        }
}

// this class handles the task of a conversation with a single client
class PropertiesTask implements Runnable {

        private Socket connection;

        /**
         * Constructs a PropertiesTask for the given Socket connection
         */
        public PropertiesTask (Socket connection) {
                this.connection = connection;
        }

        /**
         * performs the conversation with the client untill the server sends a BYE. message
         */

        public void run(){
                try {
                        BufferedReader in = new BufferedReader(
                                new InputStreamReader(connection.getInputStream()));
                        PrintWriter out = new PrintWriter(new OutputStreamWriter(
                                connection.getOutputStream()),true);
                        String inputLine, outputLine;

                        PropertiesProtocol protocol = PropertiesProtocol.getInstance();
                        while ((inputLine = in.readLine()) !=null) {
                                outputLine = protocol.processInput(inputLine);
                                out.println(outputLine);
                                if(outputLine.equals(PropertiesProtocol.FINAL_MESSAGE))
                                        break;
                        }
                        // close streams and socket
                        out.close();
                        in.close();
                        connection.close();
                }catch(IOException e) {
                        System.out.println(e);
                }
```

```
                }
        }
```

```java
import java.net.*;
import java.io.*;

class SimpleClient {
        public static void main (String[] args)         {
                try {
                        Socket connection = new Socket (args[0], Integer.parseInt(args[1]));
                        System.out.println("connection establish");
                        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

                        PrintWriter writer =
                                new PrintWriter(new OutputStreamWriter(connection.getOutputStream()), true);

                        BufferedReader input = new BufferedReader(new InputStreamReader(connection.getInputStream()));
                        String line;
                        while(!(line = reader.readLine()).equalsIgnoreCase("exit")) {

                                writer.println(line);
                                String fromServer = input.readLine();
                                System.out.println("<from server> " + fromServer);
                        }
                        connection.close();
                }catch(IOException ioe) {
                        ioe.printStackTrace();
                }
        }
}
```

```java
import java.net.*;
import java.io.*;

public class DateClient {

    private static final String HOST_NAME  = "localhost";
    private static final int PORT_NUMBER = 8899;

    public static void main(String[] args) throws IOException {
        DatagramSocket socket = new DatagramSocket();
        byte[] buf = new byte[256]; // prepare empty packet
        InetAddress address = InetAddress.getByName(HOST_NAME);
        DatagramPacket packet =
                new DatagramPacket(buf, buf.length, address, PORT_NUMBER);
        socket.send(packet);        // send packet to server
        packet = new DatagramPacket(buf, buf.length);
        socket.receive(packet);     // wait for packet from server
        int len = packet.getLength();
                String received = new String(packet.getData()).substring(0,len); // extract date
        System.out.println("date received: " + received);
        socket.close();
    }
}
```

```java
import java.net.*;
import java.io.*;
import java.util.Random;
import java.util.Date;

public class DateServer {
    public static void main(String[] args) throws IOException {
        new Thread(new SendDateTask()).start();
    }
}

class SendDateTask implements Runnable{

    private static final int PORT_NUMBER = 8899;
    protected DatagramSocket socket = null;
    protected BufferedReader in = null;
    private Random random = new Random();

    public SendDateTask() throws IOException {
        socket = new DatagramSocket(PORT_NUMBER);
    }

    public void run() {
        try {
            while (true) {
                byte[] buf = new byte[256];    // create buffer for packet
                DatagramPacket packet = new DatagramPacket(buf, 256);
                socket.receive(packet);        // wait for client s packet
                String date = new Date().toString();
                buf = date.getBytes();        // copy number to packet

                // create a new datagram with the client s IP address and port
                InetAddress address = packet.getAddress();
                int port = packet.getPort();
                packet = new DatagramPacket(buf, buf.length, address, port);
                socket.send(packet);          // send packet to client
            }
        } catch (IOException e) {}
    }
}
```

```
import java.io.*;
import java.net.*;

public class MultiDateClient {
    public static void main(String[] args) throws IOException {
        String groupIP="230.0.0.1";
        int port = 4446;
        MulticastSocket socket = new MulticastSocket(port);
        InetAddress group = InetAddress.getByName(groupIP);
        socket.joinGroup(group);
        DatagramPacket packet;
        for (int i = 0; i < 5; i++){
            byte[] buf = new byte[256];
            packet = new DatagramPacket(buf, buf.length);
            socket.receive(packet);
            buf = packet.getData();
            int len = packet.getLength();
            String received = new String(buf).substring(0,len);
            System.out.println("Todays date: " + received);
        }
    }
}
```

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

public class MultiDateServer {
    private DatagramSocket socket;
    private boolean broadcast = true;
    private String group = "230.0.0.1";
    private int port = 4446;
    private int delay = 2000;

    public void start() throws Exception{
        DatagramPacket packet;
        InetAddress address = InetAddress.getByName(group);
        socket = new DatagramSocket(port);
        while (broadcast) {
            try {
                byte[] buf = new byte[256];
                String dString = new Date().toString();
                buf = dString.getBytes();
                packet = new DatagramPacket(buf, buf.length, address, port);
                socket.send(packet);
            } catch (IOException e) {
                broadcast = false;
            }
            try{
                Thread.sleep(delay);
            } catch (InterruptedException e){
                System.exit(0);
            }
        }
        socket.close();
    }
    public static void main(String[] args) throws Exception{
        MultiDateServer server = new MultiDateServer();
        server.start();
    }
}
```