```java
package dk.atisa.hs07;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;

/**
 * Support for invoking a service with the HS07 protocol
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class Invoker {
    /**
     * Invoke a service using the HS07 protocol
     *
     * HTTP GET is used and the resource at
     *
     *     <location><method>[?key0=value0&...&keyN=valueN]
     *
     * is requested. The response to HTTP GET is assumed to be a single line
     * the response
     *
     * @param location of the service
     * @param method to be invoked
     * @param parameters of the service on the form of key0, value0, ..., keyN, valueN
     * @return the result of invoking the service (as a String)
     * @throws MalformedURLException
     * @throws IOException
     */
    public static String invoke(String location, String method, String ... parameters) throws
 MalformedURLException, IOException {
        String result = null;
        StringBuffer parameterString = new StringBuffer();
        if (parameters != null) {
            parameterString.append("?");
            for (int i = 0; i < parameters.length; i = i + 2) {
                parameterString.append(parameters[i] + "=" + parameters[i + 1]);
                if (i != parameters.length - 2) {
                    parameterString.append("&");
                }
            }
        }
        URLConnection connection = new URL(location + method + parameterString).
openConnection();
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.
getInputStream()));
        result = in.readLine();
        in.close();
        return result;
    }
}
```

```java
package dk.atisa.hs07;

import java.net.URL;

import javax.servlet.Servlet;

import org.mortbay.jetty.Server;
import org.mortbay.jetty.servlet.Context;
import org.mortbay.jetty.servlet.ServletHolder;

/**
 * An HS07 service
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public abstract class Service {
    /**
     * Creates and starts an HS07 service using the Jetty servlet container
     *
     * @param location of the service
     * @throws Exception
     */
    public Service(String location) throws Exception {
        Server server = new Server(new URL(location).getPort());
        Context root = new Context(server, "/", Context.SESSIONS);
        Servlet servlet = new ProtocolServlet(getController());

        root.addServlet(new ServletHolder(servlet), "/*");
        server.start();
    }

    /**
     * Override this to provide a POJO that implements this service
     *
     * @return the controller
     */
    public abstract Object getController();
}
```

```java
package dk.atisa.hs07;

import java.io.IOException;
import java.lang.reflect.Method;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * Support for implementing a service using the HS07 protocol
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class ProtocolServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private Object controller;

    public ProtocolServlet(Object controller) {
        this.controller = controller;
    }

    /**
     *
     * Lookup method on controller based on parameters of request URL
     *
     * The method is invoked using reflection and the invocation is neither type safe, nor
     * safe regarding number of arguments
     *
     */
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        boolean found = false;
        String methodName = req.getRequestURI().substring(1);
        for (Method method: controller.getClass().getDeclaredMethods()) {
            if (method.getName().equals(methodName)) {
                found = true;
                try {
                    Object[] parameters = new String[req.getParameterMap().keySet().size()];
                    int i = 0;
                    for (Object key: req.getParameterMap().keySet()) {
                        parameters[i++] = req.getParameterValues((String)key)[0];
                    }
                    Object result = method.invoke(controller, parameters);
                    if (result != null) {
                        resp.getWriter().append(result.toString());
                    }
                } catch (Exception e) {
                    resp.getWriter().append(e.toString());
                }
            }
        }
        if (!found) {
            resp.getWriter().append("java.lang.NoSuchMethodException: " + methodName);
        }
    }
}
```

```java
package dk.atisa.hs07.gateway;

import java.io.IOException;
import java.net.MalformedURLException;
import java.util.Collection;
import java.util.LinkedList;

import dk.atisa.hs07.Invoker;

/**
 * The controller for the gateway service. Implements the responsibilities
 * of the gateway service
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class Gateway implements Runnable {
    private static final long SLEEP_TIME = 1000;
    private Collection<String> thermometers = new LinkedList<String>();
    private Collection<String> observers = new LinkedList<String>();

    /**
     * Register a thermometer which is used for estimating temperature
     *
     * @param location of the thermometer. Must implement "getTemperature()"
     */
    public void registerThermometer(String location) {
        thermometers.add(location);
    }

    /**
     * Register an observer which is notified when a new temperature
     * estimate is available
     *
     * @param location of the observer. Must implement "notify(temperature)"
     */
    public void registerObserver(String location) {
        observers.add(location);
    }

    /**
     * Notify observers of temperature change
     *
     * @param temperature
     * @throws MalformedURLException
     * @throws IOException
     */
    public void notifyObservers(double temperature) throws MalformedURLException, IOException
 {
        for (String location : observers) {
            Invoker.invoke(location, "notify", "temperatur___e", "" + temperature);
        }
    }

    /**
     *
     * @return the average of the temperatures measured by thermometers in
     * the home
     * @throws MalformedURLException
     * @throws IOException
     */
    public double getTemperature() throws MalformedURLException, IOException {
        double sum = 0;
        for (String location : thermometers) {
            sum += Double.parseDouble(Invoker.invoke(location, "getTemperature"));
        }
        return sum/thermometers.size();
    }

    /**
     * Run the control algorithm which is decentralized in this case, i.e.,
     * observers are responsible for concluding based on temperature data
     */
    public void run() {
        while (true) {
            try {
                Thread.sleep(SLEEP_TIME);
                if (thermometers.size() > 0) {
                    double temperature = getTemperature();
                    System.out.println("Average temperature: " + temperature + " for " +
thermometers.size() + " thermometer(s)");
                    notifyObservers(temperature);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

```java
package dk.atisa.hs07.gateway;

import dk.atisa.hs07.Service;

/**
 *
 * Gateway service. A home has one gateway service. It has three responsibilities:
 *
 * - providing an Internet access point for the home from the outside
 * - running a control algorithm for the heating system
 * - providing a known address to other services in the home
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class GatewayService extends Service {
    public GatewayService(String gatewayLocation) throws Exception {
        super(gatewayLocation);
        System.out.println("Started gateway service at " + gatewayLocation);
    }

    /**
     * Start the controller as a Thread since we run the control
     * algorithm continuously
     */
    public Object getController() {
        Runnable controller = new Gateway();
        new Thread(controller).start();
        return controller;
    }

    public static void main(String[] args) throws Exception {
        new GatewayService(args[0]);
    }
}
```

```java
package dk.atisa.hs07.actuator;

/**
 * An HS07 radiator that may be turned on and off
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class Radiator {
    /**
     * Maximum temperature for control algorithm
     */
    public static final double MAX_TEMPERATURE = 20.5;
    /**
     * Minimum temperature for control algorithm
     */
    public static final double MIN_TEMPERATURE = 19.5;

    private boolean state = false;

    /**
     * Run the control algorithm upon notification of temperature change
     *
     * @param _temperature
     */
    public void notify(String _temperature) {
        double temperature = Double.parseDouble(_temperature);
        if (temperature < MIN_TEMPERATURE) {
            System.out.println("Turn on radiator");
            setState(true);
        } else if (temperature > MAX_TEMPERATURE) {
            System.out.println("Turn off radiator");
            setState(false);
        }
    }

    public void setState(boolean state) {
        this.state = state;
    }

    public boolean getState() {
        return state;
    }
}
```

```java
package dk.atisa.hs07.actuator;

import java.net.URL;

import dk.atisa.hs07.Invoker;
import dk.atisa.hs07.Service;

/**
 *
 * Radiator service. A home may have a number of radiator services
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class RadiatorService extends Service {
    public Object getController() {
        return new Radiator();
    }

    /**
     * Construct a radiator service and register the service with the gateway
     * as an observer
     *
     * @param gatewayLocation
     * @param thisLocation
     * @throws Exception
     */
    public RadiatorService(String gatewayLocation, String thisLocation) throws Exception {
        super(thisLocation);
        URL url = new URL(thisLocation);
        Invoker.invoke(gatewayLocation, "registerObserver", "location", url.toString());
        System.out.println("Started radiator service at " + url);
    }

    public static void main(String[] args) throws Exception {
        URL baseUrl = new URL(args[1]);
        for (int i = 0; i < Integer.parseInt(args[2]); i++) {
            String location = "http://" + baseUrl.getHost() + ":" + (baseUrl.getPort() + i) +
 "/";
            new RadiatorService(args[0], location);
        }
    }
}
```

```java
package dk.atisa.hs07.sensor;

/**
 * An HS07 thermometer that may be queried for the current temperature
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class Thermometer {
    private double  temperature = 20;

    /**
     * Simulate taking a temperature measurement
     *
     * @return the current temperature
     */
    public double getTemperature() {
        temperature += Math.random() - 0.5;
        return ((int)(temperature*10))/10.0;
    }
}
```

```
package dk.atisa.hs07.sensor;

import java.net.URL;

import dk.atisa.hs07.Invoker;
import dk.atisa.hs07.Service;

/**
 * Thermometer service. A home may have a number of thermometer services
 *
 * @author Klaus Marius Hansen, klaus.m.hansen@daimi.au.dk
 *
 */
public class ThermometerService extends Service {

    public Object getController() {
        return new Thermometer();
    }

    /**
     * Create a thermometer service and register with the gateway
     * @param gatewayLocation
     * @param thisLocation
     * @throws Exception
     */
    public ThermometerService(String gatewayLocation, String thisLocation) throws Exception {
        super(thisLocation);
        URL url = new URL(thisLocation);
        Invoker.invoke(gatewayLocation, "registerThermometer", "location", url.toString());
        System.out.println("Started thermometer service at " + url);
    }

    public static void main(String[] args) throws Exception {
        URL baseUrl = new URL(args[1]);
        for (int i = 0; i < Integer.parseInt(args[2]); i++) {
            String location = "http://" + baseUrl.getHost() + ":" + (baseUrl.getPort() + i) +
 "/";
            new ThermometerService(args[0], location);
        }
    }
}
```