The University of Queensland
School of Information Technology and Electrical Engineering
Semester 1, 2012

CSSE4004/CSSE7014- Assignment 1

Issued: 14/03/2012

Due: 2pm on 20/04/2012

Weighting: 22%

1. Introduction

In this assignment students will be required to develop a simplified implementation of a distributed context-aware application. It is an application of an intelligent house ("smart home"). The intelligence embedded in the house makes this home adaptive to user needs. One of the goals in research on smart homes is that they should support disabled or elderly people living independently. For example, the home can monitor the health status of its inhabitants and can call for help if a medical emergency arises. This allows the inhabitants to be monitored continuously without needing a constant human presence. The information gathered by the home can also be accessible over the Internet to medical workers and family members of the inhabitants. This assignment is a very simplified version of context-aware applications providing an intelligent home functionality for two occupants.

The application will make use of the Elvin Notification/Subscribe architecture for communication between its different components.

2. Elvin and communication

Elvin is a notification/subscribe communication framework developed by DSTC/ Mantara Software Inc. In Elvin, Producer processes send messages (called notifications) to an Elvin Server. Consumer processes register subscriptions with the Elvin Server to express interest in the types of notifications they want to receive. Upon receiving a notification from a Producer, the Elvin Server evaluates all the Consumer subscriptions and forwards the notification to any interested Consumers.

More information on Elvin is available at www.elvin.org.

In this assignment two forms of communication will be used: (1) Elvin notifications and (2) Pseudo-RPC.

Elvin notifications are already provided by Elvin. Pseudo-RPC communication must be implemented by students to provide RPC-like functionality on top of Elvin notifications.

Pseudo-RPCs look like local method invocations to the client process. All Elvin communication should be encapsulated in the appropriate client stub.
For the assignment, Pseudo-RPC client stub must have the following characteristics:

- Accepts an RPC ***server name***, a ***method to call on that server*** and any ***parameters*** required for the method call
- Produces a Pseudo-RPC notification containing all of the information from mentioned above.
- Subscribes to a response notification from the Pseudo-RPC server
- waits for a response notification before it returns the response notification to the Pseudo-RPC client
- Uses appropriate Java threading primitives to emulate the blocking behaviour of a procedure call. The client stub must not "busy wait".

Similarly to the client, Pseudo-RPCs from a client process look like local method invocations to the server process. All Elvin communication should be encapsulated in the appropriate server stub.

The Pseudo-RPC server stub must have the following characteristics:

- Subscribes to Pseudo-RPC notifications for its server
- When it receives a Pseudo-RPC notification, it calls the ***method on the server*** (i.e., the method has to be located on the server, and not the stub itself) specified in the Pseudo-RPC notification using the parameters in the Pseudo-RPC notification
- Produces a response notification containing the result of the method call

The Pseudo-RPC client stubs will be implemented in
`<AssignmentComponenetName>PseudoRPCClientStub.java`

The Pseudo-RPC server stubs will be implemented in
`< AssignmentComponenetName>PseudoRPCServerStub.java`

Students are encouraged to develop ***generic RPC Client/Server stubs*** that are wrapped or extended for each specific RPC interaction.

3 Assignment components

The Smart Home application manages the current contexts of users (e.g., heart rates, blood pressure and home temperature, etc.) so as to issue health warnings at appropriate times and maintain indoor temperature at a suitable degree.

The Smart Home application that students should implement for this assignment has three components:

- Sensors/devices - a temperature sensor, a location tracker, a day clock, and a heart rate and blood pressure (BP) monitor. **Sensor.java**
- A home manager (HM) – a component that receives sensor readings, manages home temperature, issues health warnings and responds queries from the user interface. **HomeManager.java**
- A user interface (UI) for managing the Smart Home. **SmartHomeUI.java**

3.1 Sensors/devices

The context-awareness of the smart home is based on context information collected by four types of sensors. These sensors are:

- A temperature sensor that produces integer readings in the range 0 to 30.
- A location tracker that keeps track the location of the home occupants where a location is one of “away” or “home”.
- A heart rate and blood pressure monitor (BP) that monitors the user’s current heart rate and blood pressure.
- A day clock that tracks the current day of the week (i.e. Monday to Sunday).

Sensors are producers of information that will be consumed by the managing components. The temperature sensor should be able to:

- produce readings periodically, once a second **OR**

- produce readings only when the temperature goes outside a specified range(set by the Home Manager), and do not produce another readings until there is a temperature change and such a change is outside that specified range.

In other words, the HM should be able to subscribe to the temperature sensor for periodic updates every second on sensor readings or notifications about a particular reading range (more details in section 3.3.1, HM Adjusting Temperature).

All other sensors produce readings every second.

When sensors receive the shutdown request from the HM they deregister their subscriptions from Elvin and exit.

3.1.1 Starting the sensors

The temperature sensor, the location tracker, the heart rate and blood pressure monitor (BP) and the day clock can be started from the command line using the command:

```
java Sensor [type][predefined-data-file][ElvinURL]
```

where:

`[type]` is the type of the sensor: temperature, BP (heart rate and blood pressure), location or clock.

`[predefined-data-file]` is a file containing the readings that the sensor should produce. For location and BP sensors, the name of the data file may be named as `"<username1>Location.txt"`, and `"<username1>bp.txt"`, respectively. The `<username>` is to be ***sent to the Home Manager*** along with every reading the sensors produce. There are maximum of two occupants being monitored at one time (i.e., maximum of two separate predefined-data-file for each of the location and BP sensor). For other sensors, the name of the data files is simply `"Temperature.txt"` and `"Clock.txt”`.

`[ElvinURL]` is the ElvinURL for the Elvin Server the sensor should connect to.

The following is an alternative way that students can explore for starting the sensors. Firstly, the Temperature sensor and Clock can be started using the command line. Then having either one of them listening for subscriptions from the Home Manager, so that when a user logs in from the User Interface, the Home Manager can notify the sensor to start the remaining sensors (such as location and BP sensors) automatically using the predefined data file corresponding to that user.

3.1.2 Using the Predefined Data File

The sensors are required to produce readings according to those in the predefined data files for that sensor type.

The following are the required formats for the predefined data file for each sensor type. Note that students are not required to perform any file error handling or file format checking. However, it is necessary to produce files that strictly follow the formats specified for testing purposes.

Each line of the `predefined-data-file` is in the format:

```
value, number of seconds
```

`value` is the value that should be in the update notification.
`number of seconds` is the number of seconds that value should be used.

For example, a location tracker file might be:

```
home,20
away,10
```

Possible values for the clock are all days of the week in full, with the first letter capitalised (e.g. `Monday`).

Possible values for the heart rate and blood pressure monitor are:

```
60_120,20
75_130,10
```

where the 1st value before the underscore indicates the current heart rate of the user and the 2nd value following the underscore indicates the systolic blood pressure of the user. The number after the comma represents the number of seconds that values should hold for.

Once the end of the file is reached, the sensor will return to the beginning, repeating the first reading in the file, second and so on.

3.1.3 Exiting the Sensors

Upon receiving a "shutdown request" from the HM, each sensor deregisters its Elvin subscription and exits gracefully.

3.2 Home Manager (HM)

3.2.1 Starting the HM

The HM is started from the command line using the command:

```
java HomeManager [ElvinURL]
```

3.2.2 Temperature Adjusting

The Home Manager receives temperature sensor and location tracker readings. The behaviour of the HM is based on the user's location. HM can support up to ***two occupants***. When any of the users is at home, the HM keeps the temperature stable at 22 degrees Celsius by periodically evaluating the temperature sensor readings. If the temperature is not at 22 degrees, the HM adjusts the air-conditioning. During the ***adjustment*** of the air-conditioning, the HM does not evaluate any temperature or location reading for 5 seconds. After the adjustment the HM immediately returns to the periodic evaluation of temperature readings.

When all the users are away from home the temperature may be permitted to decrease to 15 degrees or increase up to 28 degrees inclusive. The HM should ***notify the temperature sensor*** to only produce readings if the temperature is outside that specified range so that the HM does not have to incur the high cost of receiving periodic updates. For example, if the current temperature is at 30 degrees, the HM should only receive a notification of 30 degrees once, and then it adjusts the air-conditioning for 5 seconds. The HM should ***only*** receive another notification when there is a change to the temperature (i.e., not 30 degrees), and such a temperature change is outside the specified range (i.e., temperature <15 or temperature >28).

When any one of the users return home, the HM should notify the temperature sensor to produce readings periodically and maintain the temperature stable at 22 degrees by adjusting it accordingly.

3.2.3 Logging – Temp. Adjustment

The Home Manager logs events that the user can get a dump of later on through the User Interface.

When an air-conditioning adjustment is made it is documented by logging information about the adjustment:

```
<Day>: Air-conditioning adjusted.
Temperature: at <X> degrees
At Home: <username1> and/or <username2>
```

Where Day is the current day of the week provided by the day clock, X is the temperature that triggered the HM to adjust the air-conditioning.

3.2.4 Health Warning - Heart rate and blood pressure monitoring

When the current heart rate is above 100 BPM or systolic blood pressure is above 140 mmHg, the HM sends a warning notification to the UI that includes the current ***heart rate, blood pressure, username and current day*****.** The UI generates a warning with this information and display it to the user (regarding the display format of the warning message on the UI, please see section 3.3.2).

When there is a change to the heart rate or blood pressure value, the HM re-evaluates the reading again and issues a new warning notification to the UIs if the threshold is exceeded.

3.2.5 Logging - Health Warning

The Home Manager also logs the warning information that it has sent to the UI. The user can query the warning history through the UI by specifying two arguments – a starting day and en ending day. The HM retrieves a list of warning information that was issued within this period (inclusive), and returns the information (e.g., heart rates and blood pressure, etc.) for those days to the UI. For example, if the starting day is Wednesday and the ending day is Tuesday, all the warning information recorded on Wed, Thurs, Fri, Sat, Sun, Mon and Tue are returned to the UI. Regarding the display format for viewing the warning log on the UI, please see section 3.3.4)

3.2.5 Queries from the User Interface

The HM must respond to two different queries from the User Interface ***via Pseudo-RPC:***

1. View log – temperature Adjustment
2. View log - Health Warnings for Day(s)

View log – temperature Adjustment
The HM returns all the log as previously specified.

View log - Health Warning for Day(s)
The HM takes two arguments, a starting day and en ending day. The HM returns a list of warnings that were issued within this period (inclusive) as previously specified.

3.2.6 Exiting the HM

When all UIs have exited (i.e. all UI programs are closed), the HM automatically issues a blanket shutdown request to the sensors, deregisters its Elvin subscriptions and exits gracefully.

3.3 Smart Home User Interface (UI)

The Smart Home User Interface provides an interface through which users can interrogate the Home Manager's log. User interacts with the UI using a text-based menu. User menu choices are read from standard input. All spec-related UI output is printed to standard output.

3.3.1 Starting the UI

The UI is started from the command line using the command:

```
java SmartHomeUI [ElvinURL]
```

The UI initially asks for a name of the user:

```
Welcome to the Smart Home Monitoring System
Please enter your user name:
```

Then a main menu of the UI appears as:

```
Welcome to the Smart Home Monitoring System
Please select an option:
1. View Log – Temperature adjustment
2. View Log – Health warnings for day(s)
E. Exit
```

The user then has the option of selecting 1, 2 or E. Any other input will have the following printed:

```
Invalid command
```

Pressing of the Enter Key returns the user back to the Main Menu.

3.3.2 Receiving health warning

When the heart rate or blood pressure thresholds (as specified in 3.2.4) are exceeded and a warning is received from the HM, the UI automatically displays the warning message of the following format to the user:

```
<Day>: Health Warning!
<Username>,   your   current   heart   rate   and   systolic   blood
pressure  are  <Current  heart  rate>  bpm  and  <Current  blood
pressure> mmHg, respectively.

Please relax and consider taking your medication.
```

It returns to the main menu following the display of the warning.

3.3.3 Option 1:

The UI issues a Pseudo-RPC to the Home Manager which responds with the log data as described in the Home Manager section. If no log has yet been recorded the following is printed:

```
Log of temperature adjustment is empty
```

Irrespective of whether the query succeeds or fails, pressing the Enter key returns the user to the Main Menu.

3.3.4 Option 2:

The user is presented with a prompt requesting the start day of the health warning of interest.

```
Please enter a start day:
```

The user enters the start day. Another prompt is displayed requesting the end day of the warning.

```
Please enter an end day:
```

The user enters the end day. Note if the start day or end day is not any day of the week in full with the first letter capitalised (e.g. `Monday`), the following is displayed immediately after the offending value:

```
Day must be a day of the week in full with the first letter
capitalised (e.g., Monday)
```

Then a Pseudo-RPC is made to the HM. The HM retrieves the necessary information (e.g., the heart rate and blood pressure, etc.) recorded between the specified days for a particular user from the log. The UI renders the information returned by the Home Manager and display them in the following format:

```
Health Information of Warnings issued between <start day> and
<end day> for <Current User>:
The recorded heart rate and systolic blood pressure on <start
day> were: <1st recorded Heart rate> <1st recorded blood
pressure>… and <Last recorded Heart rate> <Last recorded blood
pressure>.
```

Repeat this sentence for all days until the end day is reached.

For example, if a user searched for warnings from Saturday to Sunday, the following could be printed:

```
Health Information of Warnings issued between Saturday and
Sunday for Username1:

The recorded heart rate and systolic blood pressure on
Saturday were: 110 bpm 160 mmHg, 82 bpm 158 mmHg and 128 bpm
134mmHg.

The recorded heart rate and systolic blood pressure on Sunday
were: 90 bpm 154 mmHg and 104 bpm 120 mmHg.
```

If there are no log has yet been recorded the following message is printed to the screen:

```
Log of health warning information is empty
```

Irrespective of whether the query succeeds or fails, pressing the Enter key returns the user to the Main Menu.

3.3.5 Option E:

The UI first notifies the HM that it is exiting, then deregisters itself from Elvin subscriptions and exits gracefully. When all UIs have exited (i.e. all UI programs are closed), the HM issues a "shutdown request" to sensors and closes the whole application,

Additional Output

Students may wish to display debugging messages while implementing the assignment. Students are requested that only output described in the specification be written to standard output, all other output should go to standard error.

4. Assessment

Students will have to provide a demonstration individually of how their application works for assessment purposes. The application will be demonstrated against a set of test cases provide by the tutor. Students have the options of demonstrating their applications on their own laptops or the lab machines. Students are required to demonstrate their application within 2 week of its submission, without a demonstration no mark will be awarded.

To maintain academic integrity, the applications used for demonstration must be same as the one that students have submitted. That is, if the java files used for demonstration are different to the ones that the student has submitted, he/she will be subject to investigation.

Marks are allocated for the different components of the assignment as follows:

Assessment item	**Marks allocated**
Coding style (neat layout and code is commented)	2
Home Manager - correct log of temperature adjustments (i.e., supporting up to 2 users) - RPCs (to/from UI) are correctly implemented - issue and retrieval of health warning correctly - request to shutdown supported as per specification (i.e. shutdown when all UIs have exited)	7
Sensors - all types of sensors supported - sensors provide appropriate readings - temperature sensor can be instructed to operate in periodic and non periodic modes - deregistration of subscriptions on Elvin Server and exiting when receiving shutdown notification	5
User interface - user interface and its output as per specification - queries issued and returned correctly (i.e., supporting up to 2 users) - warning message displayed as per specification - issues shutdown request, deregisters subscription on Elvin Server and exits.	6
Pseudo-RPC - implementation of Pseudo-RPC meets specification	2
TOTAL	22

Partial marks will be awarded for assignments that are missing components, do not compile, or only partially fulfil the stated specifications. Assignments will be returned to students no later than 3 weeks after the assignment submission deadline. Assignment marks will be emailed to each student’s UQ email account and will also be accessible on the course website.

Submission details

The assignment must be submitted by the 2 pm deadline on 20/4/2012 through Blackboard. Information on how to submit will be provided closer to the submission date. Students that submit their assignments late will receive a penalty of 10% (2.2 marks) for each working day (or part there of) that the assignment is late. Each assignment submission must contain the following files:

```
HomeManager.java
Sensor.java
SmartHomeUI.java

HomeManagerPseudoRPCClientStub.java
HomeManagerPseudoRPCServerStub.java
```

Students should also submit their predefined data files for each of the sensors as well as any other "helper" classes required.

Updates to the specification

Clarifications and updates to this specification may be made up to one week prior to the submission deadline. If such clarifications or updates are made they will be communicated to students via the subject website, the subject newsgroup and also emailed directly to students' UQ email accounts. It is expected that students will check the subject newsgroup, the subject website and their email on a daily basis.

Plagiarism

The assignment should be an individual work. All students should read and understand the University's policy on plagiarism. Any cases of plagiarism detected will be dealt with according to the University's Plagiarism Policy.