The University of Queensland
School of Information Technology and Electrical Engineering
Semester 1, 2015

CSSE4004/CSSE7014 - Assignment 1

Issued: 20/03/2015

Due: 4pm on 23/04/2015

Weighting: 24%

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.

This assignment is a very simplified version of a context-aware application providing intelligent home functionality for two occupants. The home supports automated temperature control, monitoring of energy consumption and management of media (music) files.

The application will make use of the IceStorm Publication/Subscription architecture for communication between its different components.

2. Ice distributed computing platform

The Ice distributed computing platform (http://www.zeroc.com) is a reasonably new computing platform but already used by many companies for software development in which speed (performance) and reliability are key factors. You will find a listing of such companies on the Ice website. **Ice provides two communication paradigms**: RMI and notifications (publish/subscribe). You should use both in the assignment and for this you need to evaluate which communication should be of the type RMI and which one is more suitable for publish/subscribe.

A short screencast that shows how to create an Ice client and server with Java and Eclipse taken from http://www.zeroc.com/download/screencasts/IceIn20Minutes.mov has been put on Blackboard for downloading.

Publish/subscribe is provided in Ice by IceStorm (www.zeroc.com/icestorm/index.html). IceStorm acts as a mediator between message publishers and message subscribers. IceStorm messages are unidirectional notifications. A component/process indicates its interest in

receiving messages by subscribing to a topic. A topic is essentially equivalent to an application-defined Slice interface (Slice is the specification language (IDL) for Ice): the operations of the interface define the types of messages supported by the topic. A publisher uses a proxy for the topic interface to send its messages, and a subscriber implements the topic interface (or an interface derived from the topic interface) in order to receive the messages. For more information, please refer to the introduction of IceStorm: http://www.zeroc.com/doc/Ice-3.4.1/manual/IceStorm.45.3.html.

The demos can be downloaded from here: http://www.zeroc.com/download.html#demo

3 Assignment components

The Smart Home application has two major subsystems - one that manages the house environment (temperature and energy usage) and one that performs media centre tasks (in this case, managing a collection of media files such as music tracks).

The Smart Home application that students must implement for this assignment has four components:

- Sensors/devices (a temperature sensor, energy monitor and a location tracker). **Sensor.java**
- A home manager (HM) – a component that receives sensor readings, manages home temperature, displays energy usage warnings and fields queries from the user interface. **HomeManager.java**
- An electronic media manager (EMM). **EMM.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 three 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 of the location of the home occupants where a location is one of “away” or “home”.
- An energy monitor that gauges the household electricity usage.

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 particular range, and not produce another reading until there is a temperature change and such a change is outside the 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 3.3.2, *Temperature Adjustment*).

All other sensors produce readings every second.

When sensors receive the shutdown request from the HM they deregister from IceStorm and exit.

3.1.1 Starting the sensors

The temperature sensor, the location tracker and the energy monitor are started in Eclipse, taking the following arguments:

`[type][predefined-data-file]`

where:

`[type]` is the type of the sensor (**temperature, energy or location**).

`[predefined-data-file]` is a file containing the readings that the sensor should produce. For location sensors, the data file may be named as "`<username1> Location.txt`". There are at most two occupants being monitored. For other sensors, the names of the data files are simply "`Temperature.txt`" and "`Energy.txt`".

The arguments can be set in Eclipse "run->Run configurations...->Arguments"

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
```

In the case of the energy data file (Energy.txt), the values are integers representing energy readings in watts. For example, the values

3200,15
2000,20
4300,10
represent 3.2 kW for 15 seconds, then 2 kW for 20 seconds, then 4.3 kW for 10 seconds.

Temperature readings are integers, e.g.

28,3
21,5
meaning 28 degrees (Celsius) for 3 seconds, then 21 degrees for 5 seconds.

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 subscription and exits gracefully.

3.2 Electronic Media Manager (EMM)

3.2.1 Starting the EMM

The Electronic Media Manager is s tarted from Eclipse, taking the following argument:

```
[predefined-data-file]
```

where:
`[predefined-data-file]` is a file containing media file listings.
The Electronic Media Manager maintains a list of media files which it populates using a EMM file read at start-up. This file contains a listing for each media file. Each listing has:

- a file name
- a title
- a disc title
- a track number

An example of a listing format:

filename: track1.mp3
title: Two of us
disc: Let It Be
track: 1

filename: track2.mp3
title: Dig a pony
disc: Let It Be
track: 2

You can assume each field in the listing occurs on a single line.

Each file may contain an arbitrary number of listings, and each listing is separated by a newline character.

3.2.2 EMM Queries

The EMM responds to queries that are initiated from HM via ***RPC/RMI***.

There are four different RMIs:

- **Get Title**

  Get Title takes a single argument - the file name - and returns the title (e.g., `Dig a pony` for the file `track2.mp3`).

- **Get Disc**

  Get Disc takes a single argument – the file name – and returns the corresponding disc title (e.g., `Let It Be` for the file `track2.mp3`).

- **Get Tracks**

Get Tracks takes a disc title and returns a list of track names for that disc. For example, for the disc `Let It Be`, the query returns a list containing the track titles `Two of us` and `Dig a pony`. The query should return the track titles ordered by ascending track number, even if the tracks are out of order in the data file.

- **Get Files**

  The Get Files query returns a complete listing of all file names ordered alphabetically.

3.2.3 Exiting the EMM

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

3.3 Home Manager (HM)

3.3.1 Starting the HM

The HM is started in Eclipse, taking no arguments.

3.3.2 Temperature Adjustment

The Home Manager receives temperature sensor and location tracker readings. The behaviour of the HM is based on the user's location. The 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 subscribe for temperature** readings of the temperature being outside that specified range when the user is away 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 the user returns home, the HM should subscribe for periodic readings from the temperature sensor and maintain the temperature stable at 22 degrees by adjusting it

accordingly. (so the temperature sensor should produce two kinds of sensor readings, one is periodic and another one is out-of-range, HM should choose which one to subscribe based on the users' location).

3.3.3 Logging – Temperature 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:

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

where `X` is the temperature that triggered the HM to adjust the air-conditioning.

3.3.4 Energy Usage Monitoring

When the current electricity consumption is above 4 KW, the HM invokes a warning method on the UI interface which passes the current electricity usage (regarding the display format of the warning message on the UI, please see section 3.4.4).

When there is a change to the electricity reading, the HM re-evaluates the reading again and issues a new warning to the UIs if the threshold is exceeded.

3.3.5 Queries from the User Interface

The HM must respond to three different queries from the User Interface ***via RPC/RMI*** (described in Section 2)***:***

1. View log – Temperature Adjustment
2. View media files
3. List tracks

View log – Temperature Adjustment
The HM returns the entire log as previously specified.

View media files
The HM returns a complete listing of all media files (one per line), including the file names, titles and disc titles separated by commas, e.g.:

```
track1.mp3, Two of us, Let It Be
track2.mp3, Dig a pony, Let It Be
```

List tracks
Like the Get Tracks query on the EMM, this query takes the name of an album and returns the corresponding track titles ordered by ascending track number. This query should simply call the corresponding EMM query.

3.3.6 Exiting the HM

When UI has exited, the HM automatically issues a blanket shutdown request to the sensors, deregisters its notification subscriptions and exits gracefully.

3.4 Smart Home User Interface (UI)

The Smart Home User Interface provides an interface through which users can interrogate the Home Manager's log. A 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.

3.4.1 Starting the UI

The UI is started in Eclipse, taking no arguments.

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 media files
3. View disc tracks
E. Exit
```

The user then has the option of selecting 1, 2, 3 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.

3.4.2 Option 1:

The UI issues a request 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.

3.4.3 Option 2:

The UI issues a request to the Home Manager which responds with the listing of media files, including the file names, titles and disc titles, as described previously. If the listing of media files is empty (i.e., there are no files), the following is printed:

```
No media files were found
```

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

3.4.4 Option 3:

The UI prompts the user for a disc title as follows:

```
Please enter the disc title:
```

The UI issues a request to the Home Manager, supplying the entered disc title, and then displays the list of returned track titles. If there are no tracks for this disc, the UI displays the following message:

```
The disc '<title>' was not found in the media collection
```

Where `<title>` is the disc title entered by the user.

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

3.4.4 Receiving warning of high energy consumption

When the electricity usage threshold is exceeded and a warning is received from the HM, the UI displays a warning message of the following format to the user:

```
Energy Usage Warning: Current electricity consumption is
<Current Electricity Usage>.

Please consider the environment before switching on any
```

```
electrical appliance.
```

After displaying the message, the UI continues with its previous task. For example, if the UI was awaiting a menu option from the user, it remains in this state after printing the message.

The communication mode between the UI and HM follows the style of client/server (with reversible roles). The client (UI) issues a request and the server (HM) responds with corresponding reply. However, the HM should also play the role of client when sending warnings to the UI. When the HM forwards the warning message to the UI, you should check if the UI has started or not (socket function), otherwise, there will be a socket exception.

3.4.5 Option E:

The UI first notifies the HM that it is exiting (when the user selects the option "E. Exit"), then shutdowns the communicator and exits gracefully. Then the HM publishes a "shutdown request" to the IceStorm, shutdowns the communicator and unsubscribes itself from the IceStorm. When sensors receive the "shutdown" notification from the IceStorm the sensors destroy the topic and exit.

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 provided 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 submitting the assignment. 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, supporting up to 2 users - RPC/RMIs (to/from UI) are correctly implemented - issue and retrieval of media manager data correctly - request to shutdown supported as per specification (i.e., shutdown when all UIs have exited)	5
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 IceStorm and exiting when receiving shutdown notification	4
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 IceStorm and exits.	5
Electronic Media Manager - responds to queries as per specification - deregistration of subscriptions on IceStorm and exiting when receiving shutdown notification	4
RPC/RMIs - implementation of RPC/RMIs and Publish/Subscribe meet specification	4
TOTAL	24

Partial marks will be awarded for assignments that are missing components, do not compile, or only partially fulfil the stated specifications.

Assignment marks will be emailed to each student's UQ email account no later than 3 weeks after the assignment submission deadline.

Submission details

The assignment must be submitted as a zipped file through Blackboard by the 4 pm deadline on 23/4/2015. Students who submit their assignments late will receive a penalty of 10% (2.4 marks) for each working day (or part thereof) that the assignment is late. In the event of exceptional personal or medical circumstances that prevent on-time hand-in, you should contact the lecturer and be prepared to supply appropriate documentary evidence (e.g. medical certificate).

Late submissions must be made directly to the lecturer or tutor (late submissions through Blackboard will be unsuccessful).

Each assignment submission must contain the following files:

```
HomeManager.java
Sensor.java
SmartHomeUI.java
EMM.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 Blackboard, course newsgroup (uq.itee.csse4004) and also emailed directly to students' UQ email accounts. It is expected that students will check the course newsgroup 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.

Appendix

A simple way to run the demo of IceStorm.

1. Install Ice-3.5.1.msi

2. Install Eclipse plug-in

3. Download Ice-3.5.1-demos.zip

4. Create a java project, in which create new files subscriber.java, publisher.java and clock.ice the same as the files in the directory of \Ice-3.5.1-demos\demoj\IceStorm\clock. (clock.ice should be created as slice file)

5. Copy the file configuration files from \Ice-3.5.1-demos\demoj\IceStorm\clock to the bin directory of the java project. (config.icebox config.pub config.sub config.service)

6. Import the IceStorm.jar into the java project

7. Use the command line and enter the bin directory of java project. Input “icebox --Ice.Config=config.icebox” to start the server (make sure that the bin directory of Ice-3.5.1 is included in the Path, an environment variable)

8. Start the publisher from the Eclipse, and then start the Subscriber from the Eclipse.