Distributed Systems -SMARTKITCHEN

Kevin Clarke (13101722), Eoin Sutton (13116053), Conor Prunty (13102311)

BSHCE4

Table of Contents

[Background 2](#_Toc479577554)

[Objectives 2](#_Toc479577555)

[Project Problem 2](#_Toc479577556)

[Project Solution 2](#_Toc479577557)

[Implementation 3](#_Toc479577558)

[JmDNS and JSON 4](#_Toc479577559)

[Error Handling 5](#_Toc479577560)

[Screenshots 6](#_Toc479577561)

[References 10](#_Toc479577562)

# Background

The SmartKitchen application has been created in order to allow users to interact with a collection of kitchen appliances within the home. A user can carry out various tasks such as powering devices on/off in addition to several other status options. It is estimated that more than half of all new business processes and systems will use some form of Internet of Things (IoT) by 2020 (BetaNews, 2017), so the features we implemented here will likely become extremely popular in the near future.

SmartKitchen allows our team to demonstrate our ability to apply the knowledge we have gained in relation to designing and building distributed systems.

# Objectives

### Project Problem

The task we set out to solve required our group to create a collection of protocols/messages that build a reference implementation of devices, in an environment such as a SmartKitchen/Home. We were required to create four or more services. The issue with this task is effectively determining what type of services to incorporate, and the various features each should have available to them. We needed to ensure that each feature made sense for that particular service to incorporate. For instance, it wouldn’t make sense for a coffee maker service to implement the same temperature setting functionality as our oven service.

In addition to the key features detailed above; we were also asked to create a Graphical User Interface (GUI) that would allow a non-technical user to interact with the SmartKitchen application and its services.

### Project Solution

To effectively solve the problem detailed above we created a SmartKitchen application with four services.

Our first is a Microwave, which accepts the actions Open/Close, On/Off, Change Heat and Set Timer. Our second device is a Fridge which accepts the actions Open/Close, Set Temperature and On/Off. The third device is an Oven which also accepts the actions Open/Close, Set Temperature and On/Off. Our fourth and final device is a Coffee Maker, which accepts the actions On/Off and Set Timer.

We ran into difficulty with the creation of a GUI using the NetBeans IDE, however after much discussion, we determined that the best way to proceed would be the implementation of the GUI through the use of jPanels executed through the command line.

# Implementation

The SmartKitchen application contains four key methods within the KitchenServant.java file.

The first Method is our method to turn on/off the four services controlled by the SmartKitchen application.

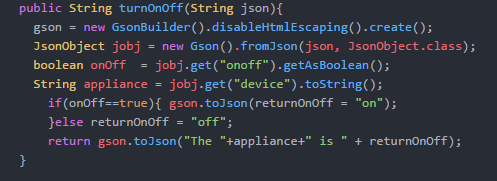


Figure – Turn On Off method

The method contained in Figure 1 takes the users chosen action (either on or off) for the chosen appliance and returns the value in JSON format where it is displayed by the applications interface. JSON is a way to store information in an organized, easy-to-access manner, and gives us a human-readable collection of data (Copter Labs, 2017).

The next method is a method that allows the user to alter the temperature of any appliance that this method would apply to (oven & microwave).

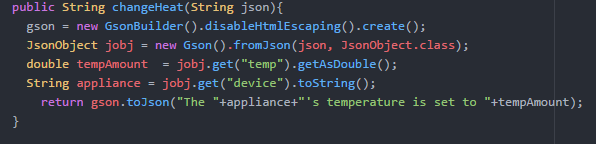


Figure – Change Heat method

Figure 2 depicts the changeheatMethod. This method allows the users to pass a chosen heat from the user interface which is then set as the new temperature and the temperature once set, is returned in JSON format to be displayed by the user interface.

The next key method that is implemented in the SmartKitchen application is the open/close door method. This method allows for the oven, refrigerator and microwave to be opened and closed through the use of the user interface.

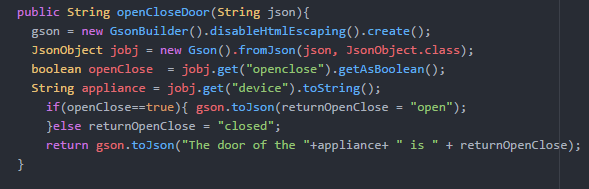


Figure – The Open Close Door method

The method shown in figure 3 allows for JSON input to be passed to the method, which then controls the status of the appliances and returns to the user whether they have opened or closed the door by returning a response in JSON format to the user interface.

The fourth key method that allows the SmartKitchen system to interact with the appliances is the setTimer method. This method allows interaction between the system and both the coffeemaker and the microwave.

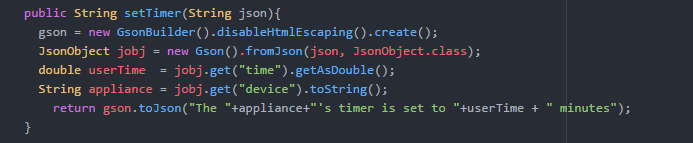


Figure – The Set Timer method

This method takes the time to be set from the user input and changes the status of the chosen appliance. The status is then returned in JSON format to the user interface.

### JmDNS and JSON

JmDNS is a Java implementation of multi-cast DNS and can be used for service registration and discovery in local area networks (Frisch, 2017). To implement JmDNS we decided to give users the option to turn on or off all devices at the same time. This is a multicast message, implemented in the first dialog in our jFrame. You can select All Devices here and you are given the option to turn on or off – this message is then sent via JSON to the turnOnOff method in the KitchenServant, which accepts device names and actions.

For JSON, at the client level we compose JSON strings that contain the desired action and the device name. In the servant, we use Google’s Gson to deconstruct the JSON string into the required Strings or Booleans which we use for the device logic.

### Error Handling

Sometimes things go wrong. If the user enters an invalid option on an application, or they click the wrong button. We have implemented a bit of error handling to try get around these errors.

JOptionPane in Java returns null by default if the user selects the ‘cancel’ option. To get around this, we have implemented a few lines of code which will ensure if that is what they choose, that it will display a message and give them the option to select a device again.

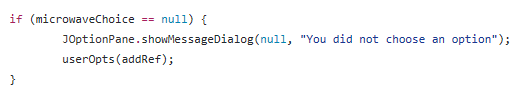


Figure - Prevent users selecting 'cancel'

Another implementation of error handling was required when user has the option to set the temperature. A free-text box is displayed, and we needed some code to ensure they entered a numeric value, and not a string of letters.

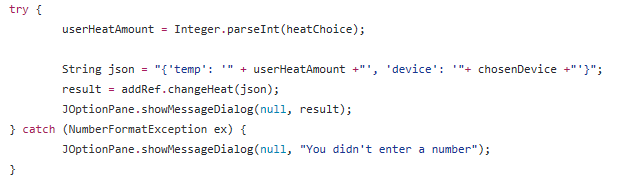


Figure - Try-Catch block to ensure numeric value entered

# Screenshots

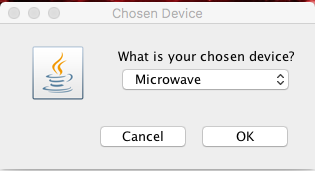


Figure - Selecting a device to interact with.

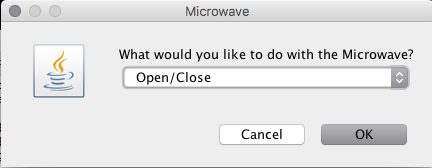


Figure – Users are presented with a choice of activities for the chosen device.

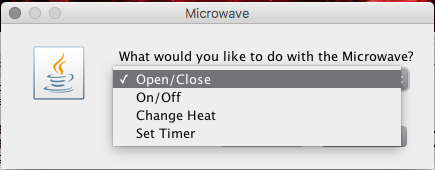


Figure -Selecting an interaction for the device

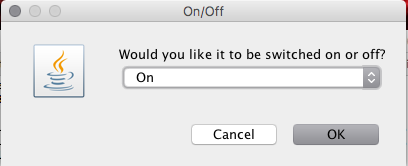


Figure - Finalising the chosen action for a device.

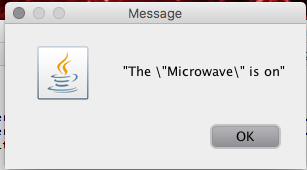


Figure - Message confirming the chosen activity has been carried out

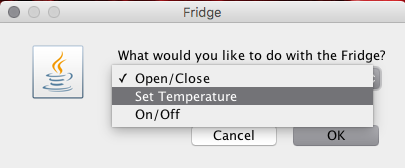


Figure - Activities for the Fridge

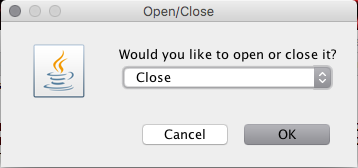


Figure - Choosing to close the fridge

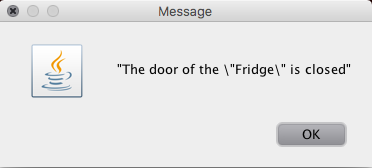


Figure - Fridge status confirmation

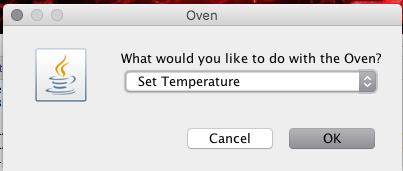


Figure - Selecting Set Temperature for the oven

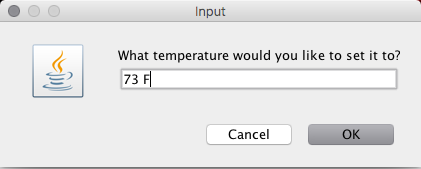


Figure -Setting the oven Temp to 73 degrees

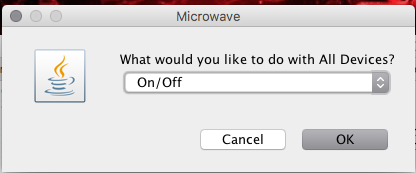


Figure - Choosing an activity to apply to all 4 devices

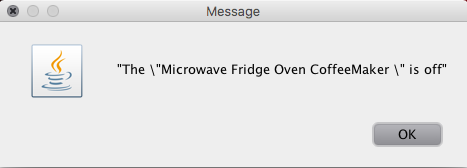


Figure - Confirmation that all appliances were turned off

# References

BetaNews. (2017). *Internet of Things' rising popularity will increase security risks, business costs*. [online] Available at: https://betanews.com/2016/01/19/internet-of-things-rising-popularity-will-increase-security-risks-business-costs/ [Accessed 08 Apr. 2017].

Copter Labs. (2017). *JSON: What It Is, How It Works, & How to Use It - Copter Labs*. [online] Available at: https://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/ [Accessed 10 Apr. 2017].

Frisch, P. (2017). *JmDNS*. [online] Jmdns.sourceforge.net. Available at: http://jmdns.sourceforge.net/ [Accessed 08 Apr. 2017].