Queen Mary
University of London

School of Electronic Engineering
and Computer Science

Mini-Project Coursework:
Improving Robot Employability in Restaurants

Module code: **EBU4201**
Module title: **Introductory Java Programming**
Hand-out date: **17th May 2016**
Hand-in date: **3rd June 2016**
Marks available: **40**
Feedback: **A marksheet will be provided with feedback comments and a mark out of 40.**

Mini-project Specification:

In April 2016, the Shanghaiist reported on the problems of employing robots to serve food in Chinese restaurants, and many in Guangzhou have been 'given the sack'!

Guangzhou restaurant fires its robot staff for their incompetence

Figure 1: A robot serving food in a Guangzhou restaurant [1]

The reported problems included: robots only being able to perform a small number of tasks and frequently breaking down. Additionally, the robots could not carry soup or pour hot water and, importantly, they could not take customer orders.

[1] http://shanghaiist.com/2016/04/06/restaurant_fires_incompetent_robot_staff.php

Your task is therefore to design a graphical user interface (GUI) for display on the robot's chest, with the aim of improving the robot's functionality, and their employability!

Required functional capability of the robot interface:

The GUI, in general, should present information to the customer (e.g., *a list of food dishes available to order*, *a total price for all food dishes ordered*). A stock list should be maintained which records the available quantities of each dish. This should be decremented each time a dish is ordered. The interface should allow the customer to communicate information to the robot (e.g., *a food order*, *a request to pay for all dishes ordered*):

(1) When the robot first approaches the customer, it should output a randomly-selected greeting on the GUI (e.g., "*Hello*", "*Hi*", "*How are you today*?", "*Isn't the weather great!*"). An example is given in Figure 2 below.

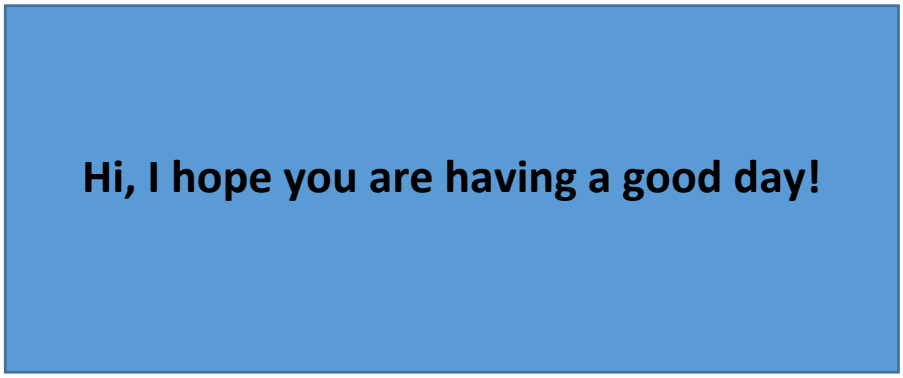

Figure 2

(2) The GUI should then ask the customer if they wish to order a:

1. Fish dish (e.g., *qīngzhēng lúyú*, *xiàmén shuǐ zhǔ huóyú*, *sìchuān huángmèn mán*),
2. Meat dish (e.g., *gōng bǎo jī ding*, *bō luó gū lǎo ròu*, *suàn tái ròu sī*),
3. Rice dish (e.g., *hóng cháng chǎofàn*, *huángjīn dà pái fàn*, *huǒtuǐ chǎofàn*),
4. Noodle dish (e.g., *miàntiáo*, *chǎomiàn*, *tāngmiàn*), or
5. Drink (e.g., *lǜchá*, *hóngchá*, *mòlìhuā chá*).

These can be selected using a number-based scheme i.e., option **1** refers to a *fish dish*, **2** to a *meat dish*, and so on. The interface should be similar to that shown in **Figure 3**.

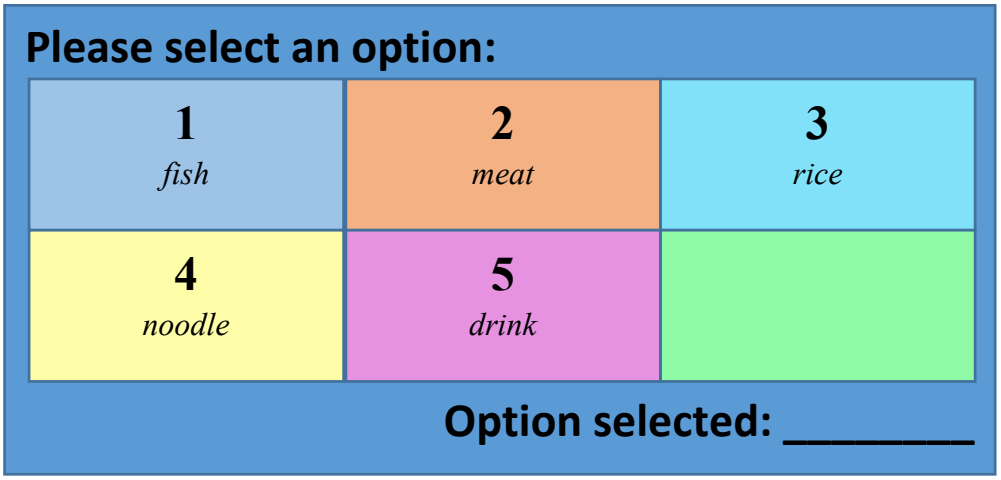

Figure 3

Based on the option selected, the robot should present a list of suitable dishes or drinks, with the prices (in RMB) for each. Each dish or drink can also be selected using a number-based scheme, e.g. within the rice category, option **1** can refer to *hóng cháng chăof*àn, **2** to *huángjīn dà pái fàn*, and so on (see **Figure 4** for an example).

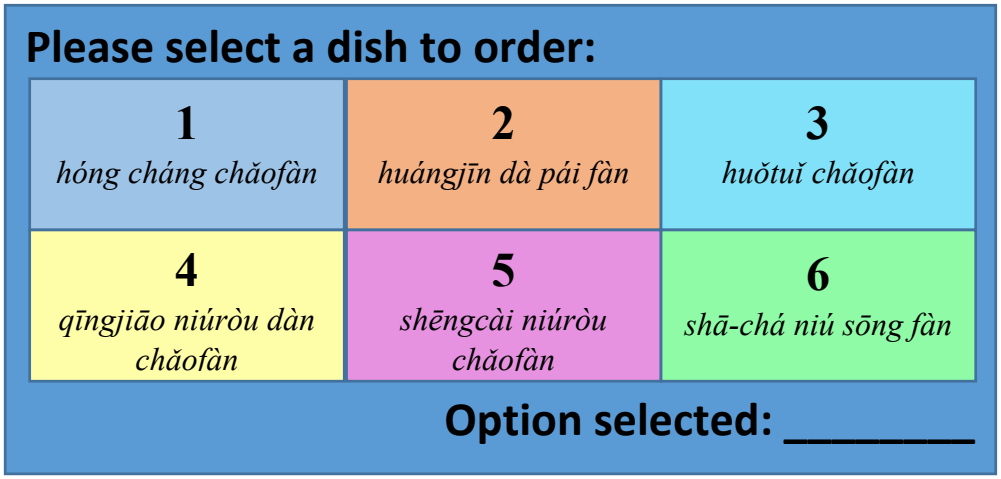

Figure 4

(3) The user interface should allow a customer to enter an order of one dish or drink.

The robot should inform the customer if the dish or drink is available using its stock list after a choice has been made, and either ask the user to confirm the order if it is available.

(4) If the dish or drink selected is not available to order, the robot should recommend alternative dishes (within the same category) or drinks to the customer, until they eventually select one which is available. See **Figure 5** for an example.

Figure 5

Each time a dish or drink is ordered, the stock list for the relevant dish or drink should be decremented by one.

(5) When the robot brings the dish or the drink to the customer, it should stay with the customer and tell jokes for entertainment; these jokes should be randomly-selected from a set of 4 defined by you.

(6) The customer may, at any point, order another dish or drink, or ask the robot to stop telling jokes.

(7) When the customer decides to leave the restaurant, the robot should display the total price for all dishes and drinks ordered during the visit.

(8) The robot should also display a farewell message to the customer, randomly-selected from a set defined by you.

(9) When the robot is inactive for 30 seconds, it should enter a sleeping mode, with reduced operation but potential to be awakened at any time to serve a customer.

Required technical capability of the robot interface:

(1) The interface should include fault checking so that, in the event that the customer enters an incorrect option, the robot returns an error message. For example, if there are 6 options on the menu, the robot will return an error message if the user enters 8, and ask the user to enter a valid option instead.

(2) All output presented to the customer by the robot should be read from pre-defined files. For example, a file named **`Jokes`** should retain the list of 4 jokes which the robot tells the customer. Separate files should similarly be created for the welcome messages (named **`WelcomeMessages`**) and dishes (named **`FishDishes`**, **`RiceDishes`**, and so on), and be placed within a directory named **`Files`**.

Code instructions:

The robot interface code must be named **`RobotInterface.java`**.
The main class of the GUI must be named **`RobotGUI`**.
All files generated must be placed in a directory called **`Robot`**.

Documentation to be submitted with your program code includes:

(1) Generated *Javadocs.*
(2) Internal comments in your code.
(3) The user manual, which must be no more than 2 A4 pages long, and must include instructions on how to run your Java application from the command line (both how to start and how to use it). It should be named **`UserManual`**, and its format must be .pdf.

All documentation must be placed in a directory called **`Documentation`**.

Submission Instructions:

You should zip all the following files:

(1) The **`Code`** directory, including all .java files produced.
(2) The **`Files`** directory, including all text files used to communicate with the customer.
(3) The **`Documentation`** directory, including all *Javadocs* and **`UserManual`**.

Name your .zip file **14xxxxxxx.zip**, where **14xxxxxxx** is your QMUL student number.