Fontys Hogescholen

DATE: **3 April 2018**
TIME : **8:45 – 10:45**
DURATION: **120 minutes**

Module	CSA
Lecturer	GRTS / KPYS
Room	2.40 / 2.84

Distribution of points:

Assignment	1	2	3	4	5	Total
Points	20	20	20	20	20	100

Remarks:
For each assignment, exam points will be given if and only if the required functionality described in the assignment works (runs with no exceptions)!

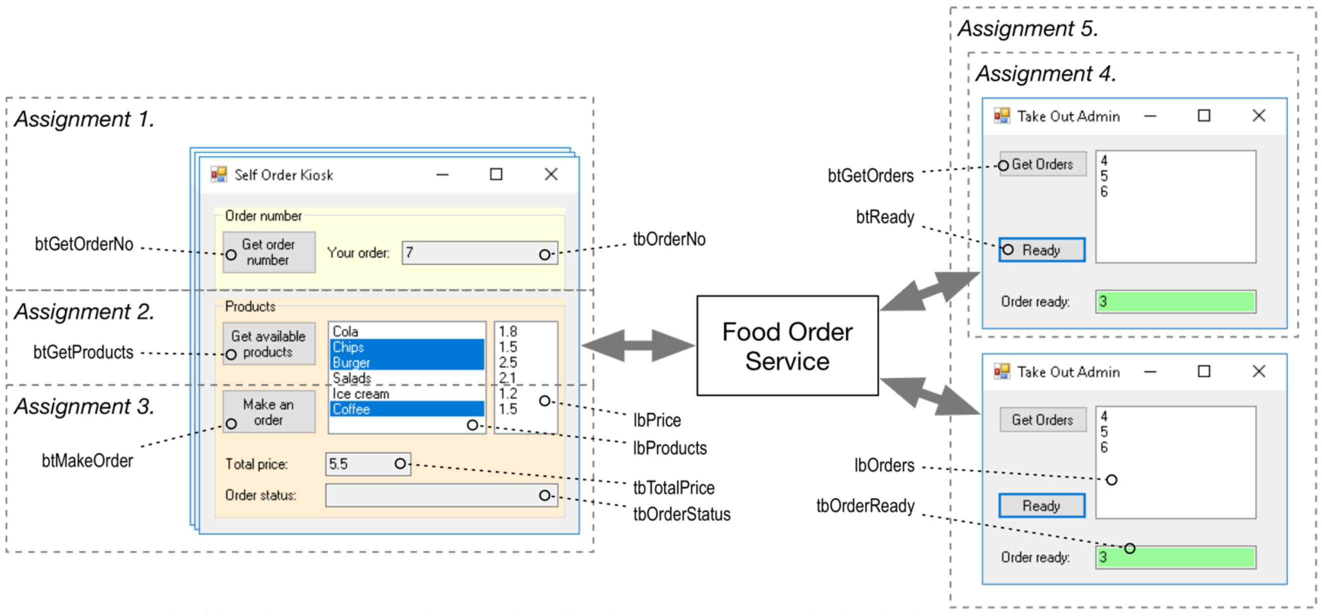

Figure 1. Example of the Client-Service application for self-order system in "McBob" fast food restaurant.

Introduction

In this exam, you are going to implement a client-service system for a fast food restaurant.

The core of the system is the food ordering service for a fast food restaurant "McBob" in Eindhoven. To optimize order preparation performance, the restaurant implemented self-order kiosks.

A client can make an order and see the price for it. Every new order receives a unique order number in the restaurant. When a new order is created, the order number appears in administrator screens, which will be delivered when ready. All administrators will see the number of the order that is ready to be delivered. Thus, the client receives his ordered food.

The service is responsible for storing information about available products and pending orders (order numbers). Each `Product` has:

- `Name` (string, e.g., "Cola", "Burger", "Chips", etc.)
- `Price` (double, indicating product price)
- `Stock` (integer, indicating available stock of a product)

Start the "FastFoodService" solution from the "Exam_Start" folder. This solution already contains two Client projects, each with one Form. The `SelfOrderClient` project is used by a client to make food order. The `TakeOutAdmin` is used by an employee of "McBob" to confirm an order. You should either add one new service project to the same solution, or you can make a new solution for the service project. In this exam you are free to decide which type of endpoint configuration you will be using (programmatic or administrative).

Assignment 1 [20 points]

Create several 'hard coded' products in the Service.

Create functionality which creates a unique order number, after button "Get order number" is clicked, and displays it in the `tbOrderNo` text box. An order number must always be unique between all running clients (see *Assignment 1* in Figure 1).

Assignment 2 [20 points]

By clicking on the "Get Available Products" button, the `SelfOrderClient` should retrieve a list of all 'available' products from the Service and show names of these products in `lbProducts` and prices in `lbPrice`. (see *Assignment 2* in Figure 1). A product is 'available' if its' stock is positive.

Note: for each product, <u>only</u> `name` and `price` should be accessible to the Clients. The `stock` must be hidden from the Clients.

Assignment 3 [20 points]

When a user selects one or more products in the `lbProducts` list, the total price must be calculated and shown in the `tbTotalPrice` text box.

By clicking the "Make an order" button (see *Assignment 3* in Figure 1), the `SelfOrderClient` sends selected items and order number to the service to create an order. The `SelfOrderClient` should:

- Show info message in the `tbOrderStatus` text box:
    - "The product *{product name}* is out of stock.", if one of the products in the order is not available any more (stock is less than 1).
    - "The order *{order number}* is being prepared", if an order was successfully added.

- If the order was successfully processed, clear `tbOrderNo, tbTotalPrice, lbProducts, lbPrice` fields.

When a `SelfOrderClient` places an order, the Service application should do the following:

- Accept an order if and only if all the products in the list are ‘available’ (stock is bigger than 0). In other words, if at least one product in the order is not ‘available’, then the whole reservation fails.
- If all products in the reservation are ‘available’, the order is accepted by modifying the stock of available products. Additionally, order number is stored in a list (later used by “Take Out Admin” client).

Note: In the service, we only need to store numbers of all orders. We do not track which items where added in which order.

Assignment 4 [20 points]

Add new service business endpoint to the service: the Admin endpoint will be used by an employee responsible for delivering prepared order.

By clicking “Get orders”, the `TakeOutAdmin` client must use newly created endpoint to retrieve from the service all order numbers added to the service.

When an order is ready (is prepared to be delivered to a client), an employee selects an order and clicks “Ready” button. The selected order number must be displayed in the text box `tbOrderReady` and it must be removed from both `lbOrders` list and from orders list in the service.

Assignment 5 [20 points]

The service should work with multiple `TakeOutAdmin` clients at the same time. Whenever a new order number is added to the service, it should automatically notify all other connected `TakeOutAdmin` clients about this event (you are free to implement this either with callbacks or events). Therefore, all `TakeOutAdmin` clients should display a new order number in their `lbOrders` list.

Additionally, if any of `TakeOutAdmin` clients deliver an order, its number should be removed from all active `TakeOutAdmin` clients.

At the startup, each `TakeOutAdmin` client application should automatically register for receiving this automatic notification. Similarly, each `TakeOutAdmin` client should automatically unregister from automatic notification at shutdown of the `TakeOutAdmin` client application.