Donald's restaurantplan.

Pieter van den Hombergh
Ferd van Odenhoven

March 12, 2018

The plan

Donald has been in the restaurant business for years and is making up a plan to provide for a proper pension plan, now he has come into the dusk of his famous years and notices he has put up little to provide for his elderly years. With the Olympics ahead he comes up with a plan to increase his income substantially. To do this he adapted his menu to the chinese kitchen (for what he thinks Chinese would eat or rather what fellow country ducks would enjoy as Chinese food). Completely in style with Chinese restaurant usage in the Netherlands, you can order your meal 'by the numbers'.

Figure 1: Duck, Donald;
class: **Aves**;
order: *Anseriformes*;
family: `Anatidae`;
this instance: rather stupid

Donald is not known as the smartest of ducks and his current restaurant implementation in Java already has some flaws. As an example: the kitchen is very fussy and throws exceptions big time. Compare this with the way Gordon Ramsay works in his TV shows.

The new employees

Because of the expected growth, Donald called for the help of his nephews Huey, Dewey and Louie. For them he has three jobs in mind: a waiter to take orders, a serving waiter and a cook to prepare the meals.

The simulation

To be prepared, Donald wants to simulate the restaurants behaviour on his computer. His target is of course to make his nephews work as efficient as possible, so that can take care of the till without further worries.

Your task

It is up to you, student of Fontys Venlo, to write this simulation program. Happily there already is a working program, which faithfully simulates the work like Donald does it on his own. In programming terms: Donald is modelled as one Thread only[1]. It is your job to add more threads, as a simulation for the jobs of the nephews. In the cooking part of the program `sleep()` calls are used to simulate preparation time; You are not allowed to add more sleeps. Of course we expect an optimal program, meaning that you should use the resources (CPU, Memory) sparingly.

A customer is shop, help.

Of course we have a customer too. In comparison to earlier versions, complaining (exception throwing) has been turned into serving a **404 meal** immediately. The customer will check and tell what has been ordered and served and will say if this differs. It *will* differ when the customer orders a non existing meal.

You should not (as in forbidden) move any of the methods for cooking or serving from the restaurant class into antother class. The work definition is part of the application (the restaurant) so leave it there. You ARE allowed to break the methods or loops apart, (e.g. move to new method) so you may solve the concurrent access and synchronization puzzle. Note that the queues are not the only things that are not thread safe. In the first version, leave the queue alone. Synchronize in the restaurant on any object that will do the trick.

- Document your work: explain how the code works that you have added.
- Comment on the results in the output.
- Using svn on fontysvenlo.org is mandatory.
- Hand in your labreport in printed form: for due date see website.
- Questions about this exercise: during labhours!

[1] Notice that ducks have tiny heads. They do not support very much of multithreading. Donald even less so.