Inf2C-SE 2014/2015, Tutorial 2 for week 4

Monday 6th October, 2014

Similar to the first tutorial, you can attempt these questions *before* your tutorial then your time at the tutorial may be more beneficial, spent discussing your solutions with your fellow students and tutor. Alternatively these exercises can be done within your tutorial. In both cases working in pairs or groups is encouraged.

1 Introduction

In this tutorial you will develop some of the example scenarios from the previous tutorial in to class diagrams. Recall the three systems presented in the first tutorial and reshown here below. For each

1.1 Thermostat - Basic

A thermostat is a simple system to control the temperature in a heated home. The user sets the desired temperature, if any (monitored) room in the house is below that temperature the heating is turned on. When all monitored rooms are above the desired temperature the heating is turned off.

Design some use-case and requirements specifications to handle this system. Take note of what happens when the user first starts the system, and also when the user changes the desired temperature.

1.2 Elevator - Basic

We have all used elevators before. Consider a single elevator system, and simply consider the scenario from the perspective of a user wishing to use the elevator. In other words, do not worry about the scenario of multiple simultaneous users. So we are simply looking at the functionalilty of a call button. The user presses a button indicating in which direction they intend to travel in the elevator. At some point the elevator should arrive at the floor to collect the waiting user.

A small part of this is that the call button should indicate to the user whether or not the elevator is coming. Generally this involves some kind of light embedded into the button. This is useful feedback to the user such that they know their request has been registered and they have not been dismissed/forgotten about. Hint: to have any effect, this light needs to be extinguished when the current request is met.

As an extension you could add a 'cancel' button. If a user has waited too long they may instead decide to use the stairs and may cancel their request.

1.3 Shopping List App

It's common for a person to quickly jot down a few items they require from the shops before going. In particular items required for a particular recipe. In the first tutorial you were asked to write some use-cases for a smartphone application to solve this problem. Here, develop these use cases into a class diagram.

The user should be able to store recipes such that *all* the required ingredients are added to the current shopping list. They should in addition be able to remove some of those ingredients if they already have them.

Another good feature would be substitutions, such as ingredients which could be used in place of the desired ingredient/item if none are available.

Of course the user should be able to tick of items as they are placed into their basket or bought at particular shops. An implementation could make this the same operation as for removing items from a recipe that the user already has.

2 Your Task

For this tutorial consider at least one of the above example systems. Preferably one that you tackled in the previous tutorial. Create a class diagram for each of the systems that you consider.

For each of the systems for which you have created a class diagram, answer the following questions:

- Say how your design is structured to the *problem* rather than the *solution*.
- Was your design completed top-down? Most of the time that is what designers are theoretically aiming for, practical matters intercede. What parts were not done top-down, and why?
- Is your design a functional design or an object-oriented design?

3 Sequence Diagram

The lecture slides contain a sequence diagram for the thermostat system. Provide the same for the elevator system. Choose which use-case the sequence diagram is for. Do not forget that the user has to select a destination floor once the elevator has arrived at their current floor.

4 Advanced - Negative Commenting

This section is mostly to promote discussion if there is time during the tutorial. One problem with class diagrams and modelling design in general is the possibility to be overly complex and hence too prescriptive. In the lecture you saw an example of trimming the possible objects before beginning the actual class diagram. Now that you are finished with your class diagram try to comment on each class describing what would happen if you were to remove this from the class diagram. This is known as *"Negative Commenting"*. Hopefully for most of your classes the effect of removing the class from the class diagram should be quite drastic.

Finally then, consider whether negative commenting is useful? Is it useful to the clients, the developers, both or neither?