Software Design Patterns

PEC 2: Design and Responsibilities Assignment Patterns (evaluated out of 50 points)

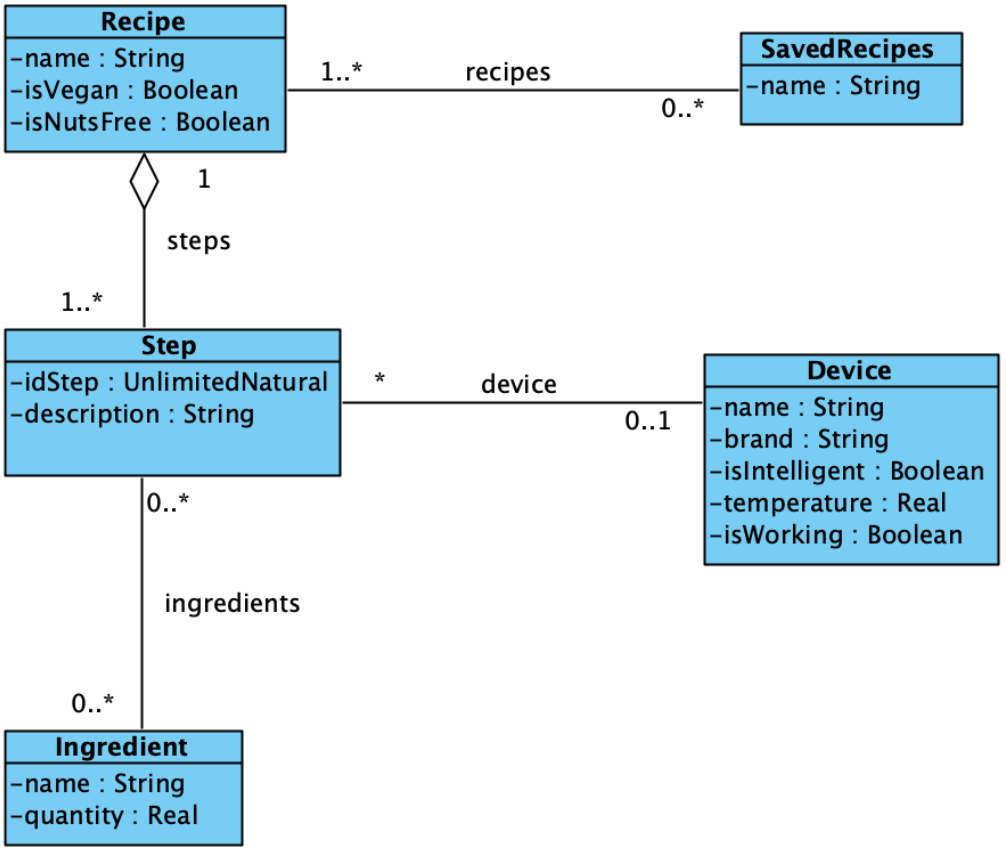
We are part of a company dedicated to the world of 'Guided Cooking'. This term is quite broad, but to simplify it, we can say that it is responsible for accompanying users in the cooking process. This can occur either in a traditional cooking instrument (such as a frying pan or a cast iron pot), or in a smart device (such as an oven or a smart pot).

The domain covered by this problem is quite broad, but for the moment we are only interested in recipes.

First we have the *Recipe* class, which represents a recipe. This is identified by *name*. In addition, it has a series of *Steps*, which are the different steps in a recipe. The recipe also has a couple of boolean attributes that are used to determine if it is vegan and if it is suitable for people with nut allergies, so the attributes are *isVegan* and *isNutsFree*. Each *Step* has an *Ingredient* and a *description* (a short description of the recipe). In addition, *Step* is identified by an *idStep*. It also has a *Device* in case a device is needed. In turn, *Ingredient* is composed of *name* and *quantity*. For simplicity we will omit the units of measurement.

Another important concept is *SavedRecipes*, which represents a collection of recipes that a user is able to create, thus saving the recipes that he/she has liked the most. For the time being, we will ignore the concept of user in the diagram and we will only consider the relationship between *SavedRecipes* and *Recipe*. Finally (although we have already mentioned it above) we have *Device*, which represents a device. This is identified by *name* and also has a pair of attributes *brand* and *isIntelligent*, which will allow us to distinguish, for example, a frying pan from an intelligent oven. It also has *temperature* (a real number that informs about the temperature. We will not take into account the unit) and *isWorking* (determines if it is on).

The static class diagram of the problem is as follows:



Question 1 (15 points)

Statement

In the Guided Cooking system, we want to coordinate the execution of all the devices that are integrated in our ecosystem. There is a critical configuration that must be managed in a unique way and accessible from anywhere in the system. The configuration includes parameters such as the server URL, the number of open recipes and the users that are cooking. A mechanism is therefore required to ensure that this configuration is consumed and modified in a secure way and does not lead to inconsistencies.

It is required:

a) (5 points) Which pattern would you recommend using in this case? Justify your choice, analysing the advantages and disadvantages compared to other alternatives.
b) (5 points) Show the class diagram illustrating your solution from the previous section.
c) (5 points) Write the pseudocode of the classes involved in the diagram you have provided.

Question 2 (12.5 points)

Statement

The concept and use of SavedRecipes is something fundamental for users. Recall that it is a feature whereby users can save their favourite recipes in collections. We are therefore asked to create a functionality that allows users to explore these recipe collections in a flexible way. This exploration must be flexible because the user must be able to choose different exploration strategies. Such approaches dictated how recipes are browsed in the collection. For example, one might require 'show only vegan recipes' and another might be 'show recipes sorted by cooking time'. The system should allow for the future addition of new ways of displaying the explored recipes, such as detailed visualisations, summaries, or even exporting to other formats.

It is required:

a) (5 points) Which pattern(s) would be the most appropriate to solve this problem?
b) (7.5 points) Implements the sequence diagram for the operation of getting all the recipes of a given collection by filtering out vegan recipes. That is, at the end our given collection will have no vegan recipes. In addition, when a recipe is vegan, it will have to be shown in a Log, so that we know for information purposes that it is being eliminated.

Note: A vegan recipe is one in which all ingredients are vegan. Furthermore, we will assume that there is a method in Recipe that already tells us whether a recipe is vegan or not without having to check all the ingredients.

Question 3 (15 points)

Statement

Going one step further, we need to create a remote control system for all smart devices. This way, users can send instructions to the system to control the devices, such as ‘start’, ‘pre-heat’, ‘pressure cook’, ‘stop’. With this we will for example control the temperature and if the device is working. The system we create must be flexible and must allow us to add new instructions in a simple way. In addition, we must be able to have a history of the instructions that we have executed so that we can undo or redo them as we see fit.

By way of clarification, the redo step makes sense when for example something is not done well enough. The undo step is not as common, but it can happen. For example, in a pressure cooker with two possible pressure releases, slow and fast. You may select first slow and then fast, but you realise that fast is too explosive and decide to undo and go back to slow. The important thing here is that our solution must allow both redo and undo.

It is required:

a) (5 points) What design pattern would be most appropriate to address this problem, and what would be the main reasons for this?
b) (5 points) Provide the class diagram.
c) (5 points) Show the pseudocode of the classes related to this problem.

Note: Make any modifications you see necessary to the classes we have provided in the activity statement. If you think that a method is missing from any class and that it is necessary to illustrate your solution, add it.

Question 4 (7.5 points)

Statement

We're working on a key part of our guided cooking app: a way for users to create their own Smart Cooking Plans. Basically, these plans are like super recipes that put together several normal recipes and tell you exactly how to use your ovens, smart pots, and so on, step by step. Each of these steps details which recipe(s) to make, which devices to use (and how to set them up with temperatures, times, etc.), and whether to wait for something to be ready before moving on to something else (like waiting for a cake to bake for half an hour).

The system has to be able to understand these Smart Cooking Plans and organise everything for you. It has to figure out the best way to use the available devices, sending orders to ovens and so on, and keep an eye on what's going on (like if an oven is heating up too slowly). And of course, you have to be shown all the progress so you know what's going on with everything. Our goal is to create a system that can handle how complicated it is to cook like this and that can be made bigger in the future. In particular, what we need to decide is how to allocate responsibilities for the different functions that are there.

It is required:

a) (4.5 points) How would you organise the classes in the system and how do they work together?Take into account both the design patterns you see necessary and important principles such as High Cohesion and Low Coupling.Use the classes proposed in the opening statement as a basis and add some if necessary.
b) (3 points) In addition, we want to integrate an external artificial intelligence service to adapt Smart Cooking Plan recipes to users' dietary preferences (vegan, gluten-free, etc.). This AI service implies a cost per call, so it is crucial to optimise its use in order to minimise expenses.Moreover, the recipe model in the AI service does not have to match ours. What pattern would be the most appropriate for this casuistry?