

Com S 362: Object-Oriented Analysis and Design
Homework 4

Due Dates: Deliverable have separate due dates, see below

Learning Objectives: The objective of this homework is to learn to design for change by explicitly and consciously anticipate changes in design parameters for the system under design. You are expected to carry out the following activities independently.

Fine prints: You MAY use the WWW and other books for this homework. You MAY obtain code snippets from the WWW and other books for this homework, provided the code snippet is available under a public license (GPL, MPL, etc) and that you have cited the source in the comments marking the beginning and the end of the code snippet. You may NOT consult with other students about this homework, post solution to this homework, or make the solution available to others in any format tangible or intangible. Failure to comply with these regulations will result in a grade of 'F' for this course without any question. The matter will also be pursued to the fullest extent possible under the university's regulation.

[Due Date: Fri, Nov 3: Deliverable 1, 2 & 3]

For the problem description on the following page construct a list of design parameters **[Deliverable #1: List of design parameters, please include a textual description of the design parameters.]**

Construct a design structure matrix that shows the interdependence of these design parameters. **[Deliverable #2: Initial design structure matrix, please include a description of each interdependence (X) that you have identified in this matrix. You may use the notation (1,3) to refer to the interdependence between the initiating parameter 1 and resulting parameter 3. In other words, the column 1 and row 3 of the DSM. The diagonal interdependence (.) need not be discussed.]**

In the context of the initial design structure matrix, think about each design parameter. Consider whether there is a likelihood of change in this parameter. If the design parameter changes what other design parameter will it influence. If there is a likelihood of change in that parameter, how can we modularize the system by introducing design rules in a way that minimizes the impact of the change? **[Deliverable #3: Improved design structure matrix. Please include a description and justification of newly introduced design rules. Please also re-organize your design structure matrix so that clearly shows the modules in your system. Refer to slides from the lectures on class diagrams.]**

[Due Date: Fri, Nov 10: Deliverable 4 & 5]

Represent your improved design structure as a class diagram in ArgoUML. **[Deliverable #4: class diagram of the application]**

Implement your improved design structure as a Java application taking advantage of interfaces and polymorphism to preserve the design structure in the source code. **[Deliverable #5: source code of the application as a zipped eclipse project]**

Problem Description: A simple cooking recipe query system. For simplicity at this point, persistent storage of cooking recipes is not considered. The domain description follows:

1. A recipe has three main components: ingredients, equipments, directions
2. Ingredients is a “set”, where each element of this set is an ingredient
3. An ingredient is a 3-tuple (amount, unit, name), where amount is a floating point number, unit is an element of the set {tsp, tbsp, cup, oz, gallon, lbs}, and name is a string. For example, 1 tsp salt.
4. Equipments is also a “set”, where each element of this set is an equipment
5. Directions is a “collection” of 1 or more step
6. A step is a 3-tuple (action, ingredient, equipment), where ingredient and equipment are defined above, and an action is an element of the set {measure, blend, cream, add, bake, fry, stir}
7. The dish described in the recipe can be part of breakfast, lunch, or dinner course

The query system initializes itself with a bunch of recipes in the form given above. It creates these recipes before prompting the user for query.

A query works as follows:

1. Display query options: By courses, by ingredient, by equipment, by action
2. If user picks by courses, display available set of courses to the user, in this case breakfast, lunch, dinner
3. Ask the user for course choice
4. Output all recipes for that course choice. Perform the query by action similarly.
5. If user picks query by ingredient, ask the user for ingredient, and output all recipes in which the ingredient is present. Perform the query by equipment similarly.