**4010-362 Engineering of Software Subsystems**

Mid-Term Exam

Instructor: Lutz, Reichlmayr, Hawker

**INSTRUCTIONS:** This exam is a take-home exam. It is open book and notes. You may take as much time as necessary to design the system described here**.** Rename this file to aaaNNNN‑midterm.doc where aaaNNNN is you SE Department login id. Create a design for the system described on the next page, and describe it by answering the questions on the following pages. Deposit this file into the Mid-Term Design Exercise dropbox in the myCourses site for your section by **11:59pm on either Wednesday, 10 April 2013, or Thursday, 11 April 2013, depending on which is a class day for you.**

**Work on this design problem is to be an individual effort. From this point forward, you are not permitted to ask for help from anyone or provide help to anyone on any matter related to this exam. You may refer to your textbook, notes and any work you submitted for this course. If you need clarification of any of the requirements, place a question in the Mid-Term Design Exercise Clarifications discussion forum. Your instructor will provide the clarification in that discussion forum so that everyone in the class can see the answer.**

**Your personal integrity will ensure that you maintain academic honesty.**

### Wegmans Inventory Tracking System



This system tracks the inventory in individual Wegmans supermarkets and determines the product that will be shipped to each store. The system has the following requirements:

1. Information is received in real-time from each store to indicate what is being sold.
2. The communication between the distribution center and each individual store is handled by a communications front-end executes an inventory exchange protocol. This protocol is best described by an event-driven state machine. The system handles several types of protocols because they unfortunately have not yet standardized on a single uniform protocol.
3. As output, all the protocols generate what are known as product inventory events. There are several different types of product inventory events.
4. Multiple clients want to be alerted when these events occur. For this design you will need to alert the store tracking client which keeps track of the inventory on the shelves in the individual stores. Be sure that the design can easily handle a variable number of other clients who want to receive alerts also.
5. For some reason the inventory system does not use a database but rather stores product information internal to the program. In the inventory structure, products are grouped into product families. Families can be nested to an arbitrary depth. The product categories are known to stay relatively stable.
6. There are several operations that the managers like to perform on the store inventory, such as, find lowest in stock, find overstocked items to put on sale, create truck loading inventory.
7. The managers will also sometimes perform “what-if” operations on the inventory. Starting with the current inventory the distribution manager will create a promotional sale and through simulation “run it over” the inventory to see the effect on the inventory.
8. For each promotion the distribution manager will select one of several standard sales prediction algorithms.
9. Managers will test many different promotions and test one promotion with different sales predictions. After a promotion is tested on the inventory, the inventory structure must be returned to the state it was in prior to the test.

**Note:**  You are only responsible for design elements described by these requirements. System elements that are not described in these requirements can be treated as classes, interfaces, or subsystems that your design uses.

1. **(5 points)**

List all of the nouns and verbs in the system description. Take as much space as you need.

|  |  |
| --- | --- |
| **Nouns** | **Verbs** |
|  |  |

1. **(35 points)**

Create a class-responsibility-collaborator description of each class in your system. Copy the template table as many times as you need. You do not need to provide a CRC table for each subclass in an inheritance hierarchy. You can use a single table to describe all of the subclasses, if you feel that sufficiently describes the design. List **specific** responsibilities for each class. Verbs such as handles, holds, tracks, manages, controls followed by one or more nouns are **not** specific responsibilities for the class and should not appear. Itemize the attributes held by a class with a single responsibility statement beginning with “This class has the following attributes: *list-of-attributes*.”

|  |  |
| --- | --- |
| **Class:** |  |
| **Responsibilities:** | |
| **Collaborators** | |
| **Uses:** | **Used by:** |

|  |  |
| --- | --- |
| **Class:** |  |
| **Responsibilities:** | |
| **Collaborators** | |
| **Uses:** | **Used by:** |

1. **(35 points)**

Describe each usage of a design pattern in the design you created. Be sure to name each pattern within the context of this application. Your description of the role for each participant should not be the generic role specified in the GoF text for a pattern participant. Describe the role in the context of this application. It should be clear to the reader that 1) the class is indeed playing the specified role in the pattern, and 2) the role addresses a feature, requirement, or need stated in the system description. Identify anyplace where you needed to stretch the standard pattern to fit this application. Copy the template table as many times as you need.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name:** | | | **GoF pattern:** |
| **Participants** | | | |
| **Class** | **Participant** | **Role of participant** | |
|  |  |  | |
|  |  |  | |
|  |  |  | |
| **Deviations from standard pattern:** | | | |

1. **(25 points)**

Provide UML class structure diagrams that visualize your design along with a short description that highlights the important aspects of the structure. Note that this requests diagram**s**. You will probably want to present your design in a few diagrams to make it easier to read and explain. Specify the relationships among the classes (inheritance, association, aggregation, composition, cardinality, navigability, and stereotype labels). You do not need to specify instance data or methods in the class diagrams. That information should be evident in the CRC descriptions provided in question #2.