

# Object Oriented Analysis

September 30, 2003

Tutorial 3

CSC407

Available at <http://www.cs.utoronto.ca/~wl/teach/407/>

## General Comments

- Don't create a big complex class diagram
- Break down into small diagrams each has some central theme
- No need to repeat associations and attributes in multiple diagrams unless you have a good reason for it (or the tool does it automatically which is the case for the examples used here ☺)
- Capture the important information embedded in the requirement statements
- Look for the best representation of the relationships possible
- Always keep it as **SIMPLE** as possible

## Example: Online Grocery Store

- We want a program that can compute the schedules of grocery deliveries (This is taken from A1 of csc407 spring term in 2003)
- Develop Class, Object, and Use Case Diagrams
- **Our goal:** to understand how to develop the model step-by-step and present it
- You are expected to present your A2(a) in a similar fashion or better

09/30/2003

W.Liu

3

## Class Diagrams

- Identify important subjects
- Use them to break down the class diagram
- Make them the themes of these smaller class diagrams
- We have the following class diagrams:
  - Order
  - Bin
  - Schedule
  - Van

09/30/2003

W.Liu

4

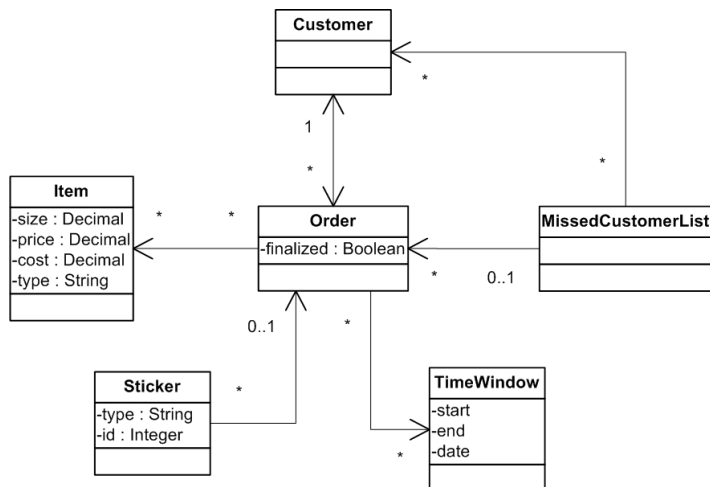
# Order

- Order is an important concept here
- It has many relationships with other concepts
- Note that for simplicity reasons, the relationship between Sticker and Item is not shown here, but shown in the next diagram: Bin. You will see more of such examples throughout

09/30/2003

W.Liu

5



09/30/2003

W.Liu

6

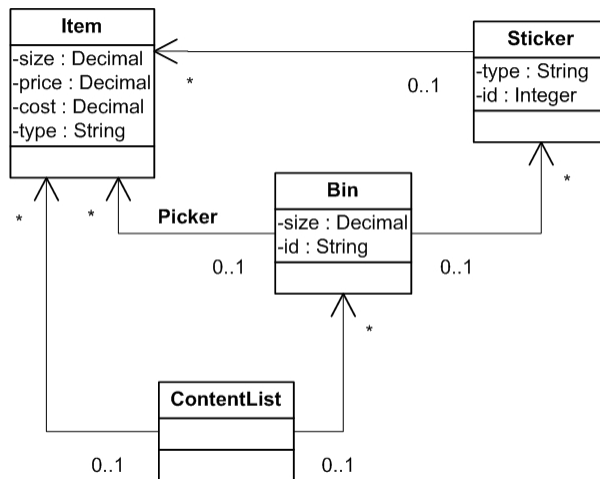
# Bin

- The Sticker class is introduced in order to resolve the seemingly conflicting requirements:
  - Bins change sticker when reused, and
  - Must keep track of Bin's content for all day
- We choose not to model Sticker as an association class because the navigation direction

09/30/2003

W.Liu

7



09/30/2003

W.Liu

8

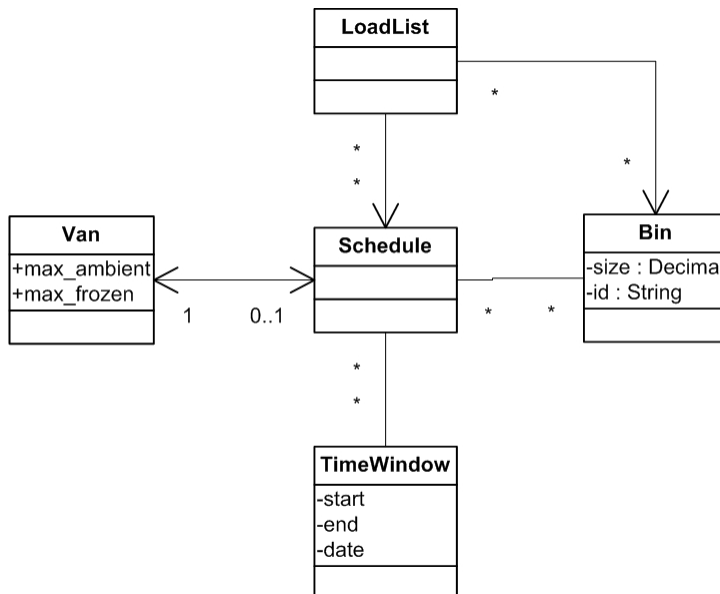
# Schedule

- Modeling the Schedule class separately helps us to understand how it might be used
- It also allows us to choose a simple representation for the complex relationships thus achieve simplicity

09/30/2003

W.Liu

9



09/30/2003

W.Liu

10

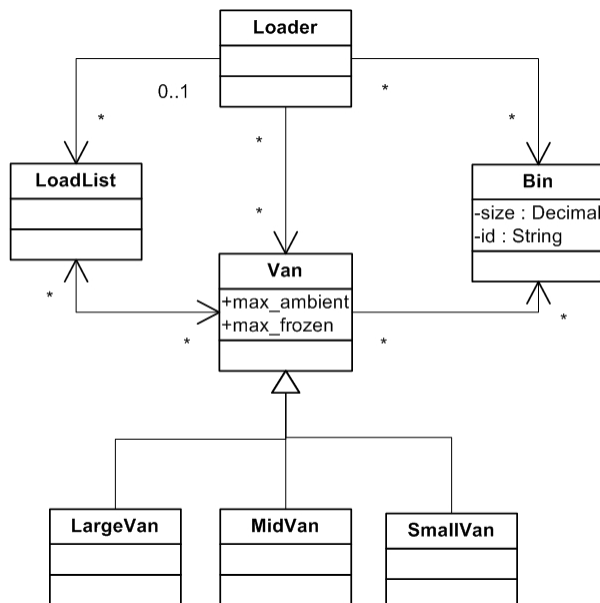
# Van

- Note that the association between Van and Bin is only shown in this class diagram because of the navigation direction
- The subclasses of Van can also be eliminated using attribute *size*
- Note that Loader doesn't have to be a class (like Picker), but it is easy to show the 3 way relation here

09/30/2003

W.Liu

11



09/30/2003

W.Liu

12

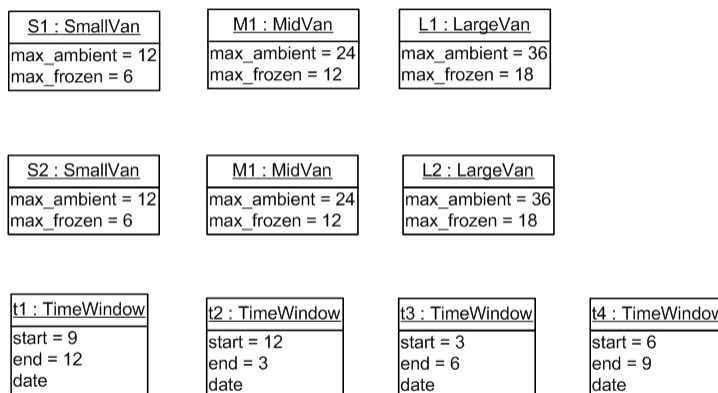
# Object Diagram

- You must be able to model the instance information that is given
- The following diagram does not have interesting relationships, but has real information
- You may also model a slice of the running system to illustrate some tricky points using OD

09/30/2003

W.Liu

13



09/30/2003

W.Liu

14

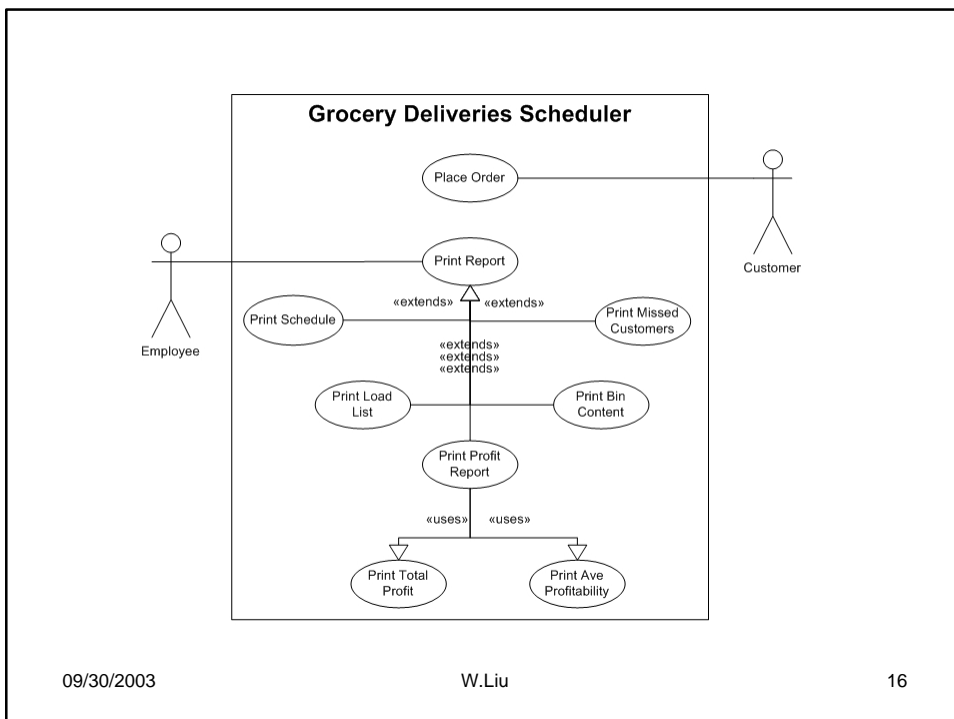
# Use Case Diagram

- Use cases are about the To-Do's
- Use cases should encode what your software should achieve
- Identify relevant, main use cases and actors
- Actors are external agents, they don't have to be humans, they can be machines as well

09/30/2003

W.Liu

15



09/30/2003

W.Liu

16

# Qualities Your TA Expects

- Overall simplicity
- Presentation
  - UML
  - written text
  - subdivision
- Analysis (both text and UML model)
  - completeness
  - correctness
  - simplicity
  - appropriateness
- Include all of Class Diagrams, Object Diagrams and Use Case Diagrams

09/30/2003

W.Liu

17