UML diagrams

These slides are based on the materials on:
UML distilled: a brief guide to the standard object modelling language, 3rd Ed. by Martin Fowler

1

What is UML

- When building complex systems, it might be worthwhile to plan things out before you start coding
  - This would allows us (and others) to visualise the system.
- UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting software systems.

2

UML Class Diagram

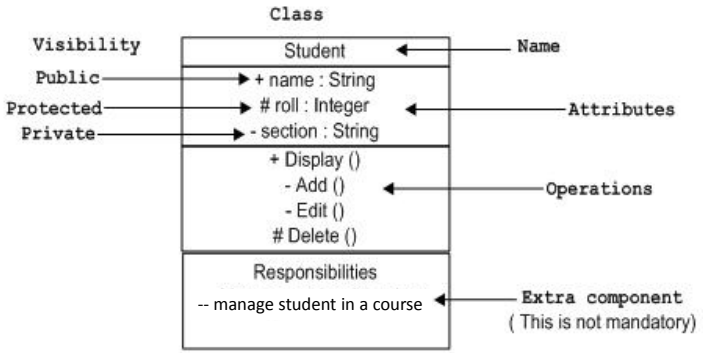
- The purpose of the class diagram is to model the static view of an application.
- A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them.
- Class diagrams also show the properties and operations of a class.
- Properties represent structural features of a class.
  - You can think of properties as corresponding to fields in a class.
- Properties are a single concept, but they appear in two quite distinct notations
  - attributes and associations.

3

Drawing UML class diagram

- The class diagram is divided into four parts.
  - The top section is used to name the class.
  - The second one is used to show the attributes of the class.
  - The third section is used to describe the operations performed by the class.
  - The fourth section is optional to show any additional components.

4



5

attribute

- The attribute notation describes a property as a line of text within the class box itself.
- The full form of an attribute is:

  visibility name: type multiplicity = default {property-string}
- An example of this is:

  - name: String [1] = "Untitled" {readOnly}

6

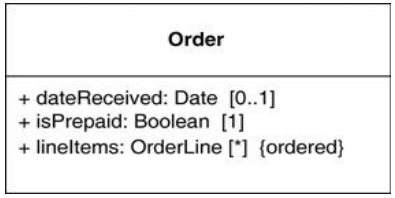
- visibility indicates whether the attribute is public (+), private (-), protected (#)
- name: The name of the attribute—how the class refers to the attribute
  - roughly corresponds to the name of a field in a programming language.
- type: The type of the attribute indicates a restriction on what kind of object may be placed in the attribute.
  - You can think of this as the type of a field in a programming language.

7

- default is the value for a newly created object if the attribute isn't specified during creation.
- The {property-string} allows you to indicate additional properties for the attribute.
  - In the example, {readOnly} indicates that clients may not modify the property.
  - By default, the attribute is modifiable.

8

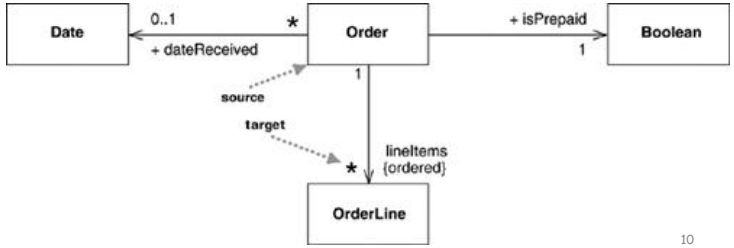
- The multiplicity of a property is an indication of how many objects may fill the property. The most common multiplicities are
  - **1** (An order is either paid or unpaid.)
  - **0..1** (An order can have at most one received date recorded)
  - A corporate customer may or may not have a single sales rep.)
  - ***** (An order might contain zero or several items and these items should appear in order)



9

Associations

- The other way to notate a property is as an association.
- An **association** is a solid line between two classes, directed from the source class to the target class.
- The name of the property goes at the target end of the association, together with its multiplicity.
- The target end of the association links to the class that is the type of the property.



10

Programming Interpretation of Properties

```
public class Order {
    public Date dateReceived;
    public Boolean isPaid;
    public OrderLine[] lineItems;
    …
}
```

11

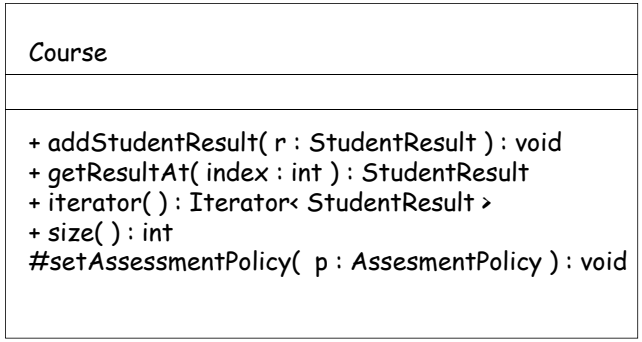
Operations

- Operations are the actions that a class knows to carry out.
- Operations correspond to the methods on a class.
- Normally, the operations that simply manipulate properties (e.g. get and set) are not shown
- The UML syntax for operations is:
  visibility name (parameter-list) : return-type {property-string}
- A parameter is
  direction name: type = default value
  - Parameters are separated by commas
- Example
  + balanceOn (date: Date, accountNum: int) : Money {query}

12

- The visibility marker: public (+), private (-), etc
- The name is a string.
- The parameter-list is the list of parameters for the operation. Each parameter is separated by a ","
- The return-type is the type of the returned value
  - If no value is returned, the type is void
- The property-string indicates property values that apply to the given operation.
- The name, type, and default value are the same as for attributes.
- The direction indicates whether the parameter is input (in), output (out) or both (inout). If no direction is shown, it's assumed to be in.

13

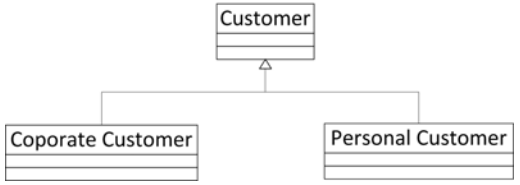


14

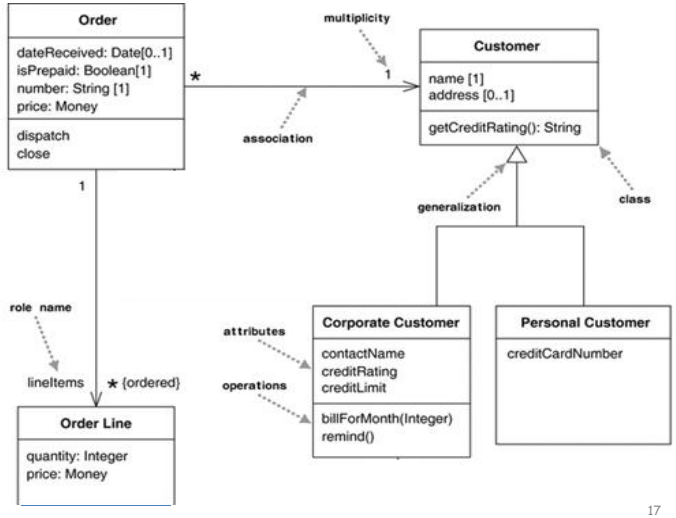
Generalization

- Generalization corresponds to inheritance in OO.
- For example, a business has personal and corporate customers.
  - They have differences but also many similarities.
  - The similarities can be placed in a general Customer class, with Personal Customer and Corporate Customer as subclass.
- Generalization is a solid line and fat triangular arrow from the subclass to superclass

15

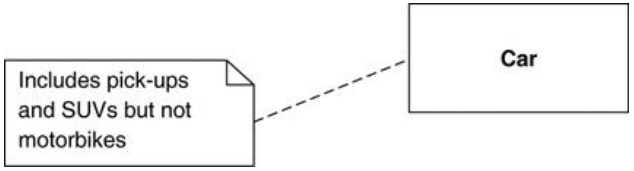


16



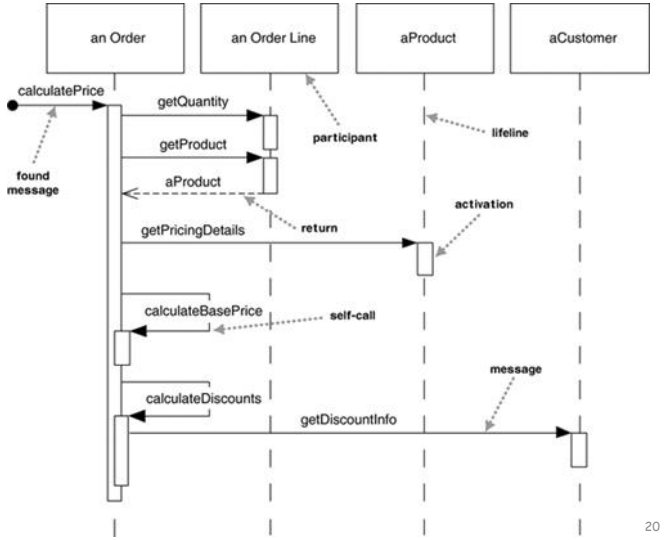
Notes and Comments

- Notes are comments in the diagrams.
- Notes can stand on their own, or they can be linked with a dashed line to the elements they are commenting



sequence diagram

- A sequence diagram captures the behavior of a single scenario.
- The diagram shows a number of objects and the messages that are passed between these objects within the scenario.
- Sequence diagrams show the interaction of participants by showing each participant with a lifeline that runs vertically down the page and the ordering of messages by reading down the page.



- We have an order and are going to invoke a command on it to calculate its price.
- To do that, the order needs to look at all the line items on the order and determine their prices, which are based on the pricing rules of the order line's products.
- Having done that for all the line items, the order then needs to compute an overall discount, which is based on rules tied to the customer.

21

- Named the participants using the style anOrder (i.e. anClass)
  - A fuller syntax is name : Class, where both the name and the class are optional
    - order:Order or :Order
- Each lifeline has an activation bar that shows when the participant is active in the interaction.
  - This corresponds to one of the participant's methods being executed.
- You don't need to show all the return values as they might clutter things
- The first message is called a found message as it doesn't have a participant that sent it.

22

review

- Understand the meaning of class diagram and sequence diagram.
- Able to relate a class diagram to Java code definition.
- Able to tell how the objects interact with each other from a sequence diagram.

23