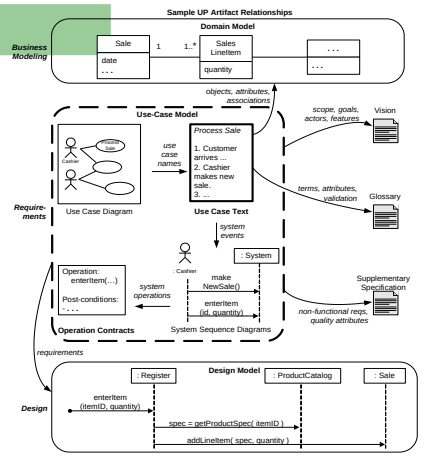
面向对象程序的分析与设计
Object-Oriented Analysis and Design

Lecture 5

Prof. S. Xu



2

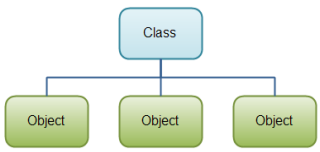
Contents

- Class and Class Diagram
- Domain Model

UML
Class and Class Diagram

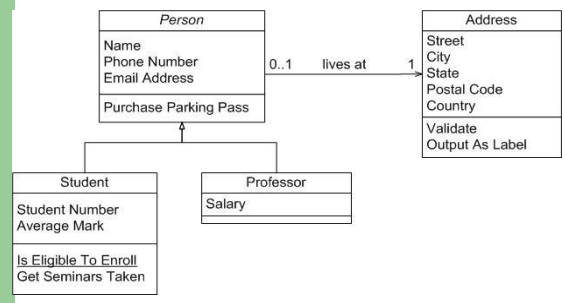
Class Diagram - Class

- A description of a set of objects that share the same attributes, operations, methods, and relationships.



5

Class Diagram – An Example



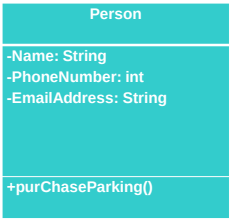
6

A Class in UML class diagram

Class name

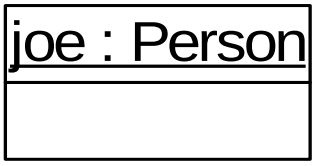
Attributes

Operators

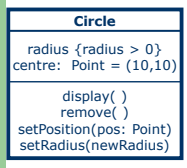


An Object in UML

object name and class



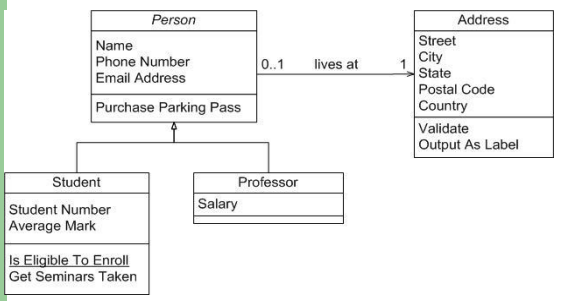
Code corresponding to a class




```
class Circle
{
   int radius;
   Point centre = Point(10,10);

   public void display( )
   { ...
   }
   public void remove( )
   { ...
   }
   public void setPosition(Point pos)
   { ...
   }
   public void setRadius(int newRadius)
   {
      if ( newRadius > 0 )
      { radius = newRadius; ...
      }
   }
}
```

Class Diagram – An Example



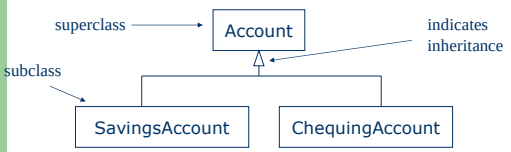
10

Class Relationships in UML

- Generalization
- Dependency
- Association

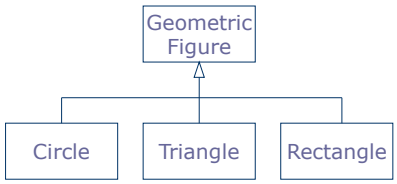
- These can represent inheritance, using, association/aggregation, etc.
- Association relationship is between instances

Inheritance Hierarchy (Generalization)



- An instance of SavingsAccount possesses ("inherits") the attributes, operations, and constraints of an Account.
  - Plus: any additional features.
- "Is-A" relationship

Class Hierarchy for Screen Objects (Generalization)



Exercises - Generalization

- Village vs Municipality
- Province vs Country
- Student vs TA
- President vs Employee

Association

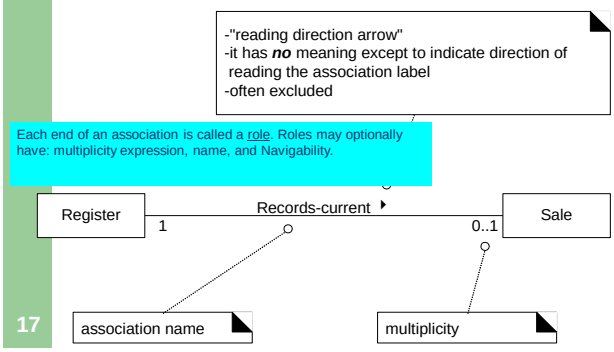
- Association:
  - A relationship between objects of different classes.
  - "has –a " relationship

Association

- Associations may have:
  - A name.
  - A direction (not shown here).
  - Role names.
  - Multiplicities.
    - ***** indicates an indeterminate number.
    - **..** indicates a range.

Association

- An association is represented as a line between classes with a capitalized association name .



17

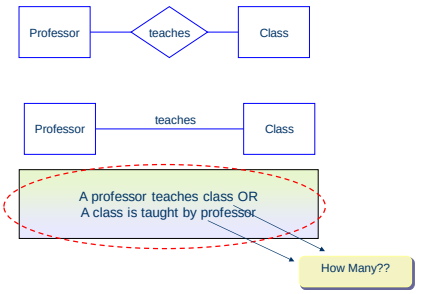
Cardinality and Connectivity

- Relationships can be classified as either
  - one – to – one
  - one – to – many
  - many – to –many

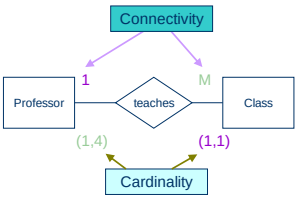
  } ***Connectivity***

- ***Cardinality*** : minimum and maximum number of instances of class B that can (or must be) associated with each instance of class/entity A.

Cardinality and Connectivity



Cardinality and Connectivity



Aggregation

- Aggregation is a "has a" relationship.
  - The diamond indicates which class is the aggregate (contains the parts).
    - ◇ Indicates that parts can exist independently of aggregate.

More examples?

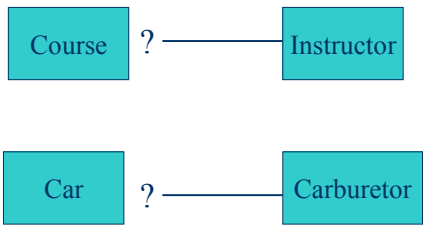
Composition

- Composition is a specific type of aggregation:
  - A "contains a" relationship.
    - ◆ Indicates that parts only exist when the aggregate exists (complete containment).

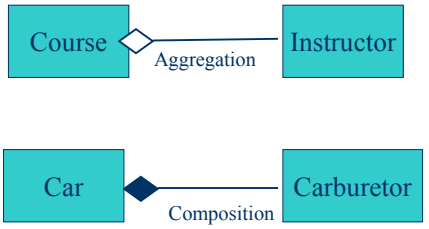
Aggregation/Composition

- Code example

Composition versus Aggregation



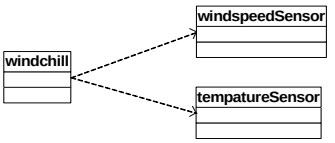
Composition versus Aggregation



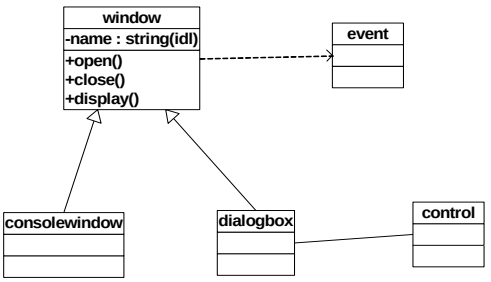
25

Dependency

- Represents a "using" relationship
- If a change in specification in one class effects another class (but not the other way around) there is a dependency



Class diagram – Another Example



Review

- What relationship is appropriate between the following classes: aggregation/composition, inheritance or neither?
  1. University – Student
  2. Student – TeachingAssistant
  3. Student – Freshman
  4. Student-Professor
  5. Car – Door
  6. Truck-Vehicle
  7. Traffic-TrafficSign
  8. TrafficSign-Color

28

Review

- What statement best describes a dependency relationship?

  A: "HAS A" relationship.
  B: "IS A" relationship
  C: "IMPLEMENTS" relationship
  D: "USES" relationship
  E: "IS PART OF" relationship

- What statement best describes an inheritance relationship?

  A: "HAS A" relationship.
  B: "IS A" relationship
  C: "IMPLEMENTS" relationship
  D: "USES" relationship
  E: "IS A MEMBER" relationship

29

Review

- A car has four wheels. The class Car and the class Wheel have what type of relationship?
- A: Dependency
  B Aggregation
  C: Inheritance
  D: Realization

- Inheritance relationships can best be modeled using which construct?

  A. association
  B. aggregation
  C. recursion
  D. Generalization

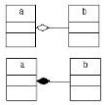
30

Review

Which pairs of symbols both mean class b is a part-of class a (has by value)?

A.

B.



31

2015/10/27

Review

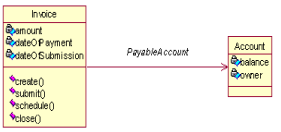
- Refer to the attached diagram, the arrow indicates: (Pick 1)

A. Refers to

B. Association

C. Dependency

D. Navigability



- Future Banking

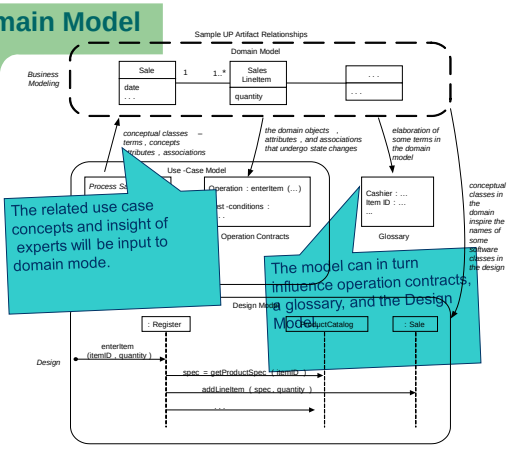
Domain Model

Customize UP, The UP Development Case

Table 2.1. Sample Development Case. s - start; r - refine

Discipline	Practice	Artifact	Incep.	Elab.	Const.	Trans.
		Iteration →	I1	E1..En	C1..Cn	T1..T2
Business Modeling	agile modeling req. workshop	Domain Model		s		
Requirements	req. workshop vision box exercise dot voting	Use-Case Model	s	r		
		Vision	s	r		
		Supplementary Specification	s	r		
		Glossary	s	r		
Design	agile modeling test-driven dev.	Design Model		s	r	
		SW Architecture Document		s		
		Data Model		s	r	
Implementation	test-driven dev. pair programming continuous integration coding standards	...				
Project Management	agile PM daily Scrum meeting	...				
...						

Domain Model



36

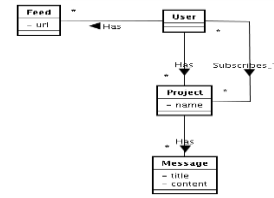
Domain Model (in UML notation)

a partial domain model drawn with UML class diagram notation. It illustrates that the *conceptual classes* of Payment and Sale are significant in this domain, that a Payment is related to a Sale in a way that is meaningful to note, and that a Sale has a date and time, information attributes we care about.

37

Domain Model (conceptual model)

- A domain model is a ***visual representation*** of conceptual classes or real-situation objects in a domain.
- Applying UML notation, a domain model is illustrated with a ***class diagram*** in which *no operations (method signatures) are defined*.



38

Domain Model

- Domain Model is a visualization of things in a real-situation domain of interest, ***not*** of software objects such as Java or C# classes, or software objects with responsibilities .
- Therefore, the following elements are not suitable in a domain model:
  - Software artifacts, such as a window or a database,
  - Responsibilities or methods.

39

Domain Model

Figure 9.3. A domain model shows real-situation conceptual classes, not software classes.

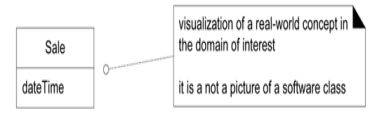
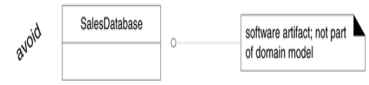


Figure 9.4. A domain model does not show software artifacts or classes.



40

Domain Model

- Ok, we know that domain model describes conceptual classes in a domain. So what is a conceptual class?
- Informally, a conceptual class is an idea, thing, or object.
- More formally, a conceptual class may be considered in terms of its symbol, intension, and extension

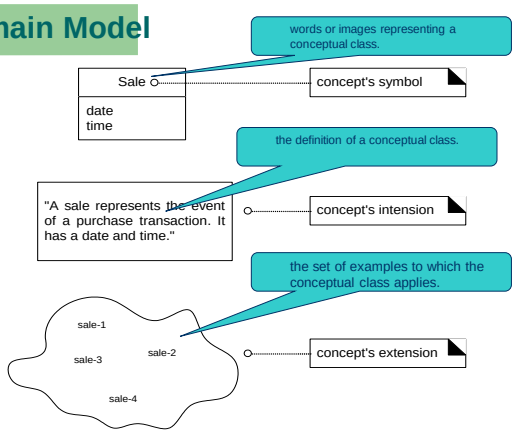
41

Domain Model

- Symbol: words or images representing a conceptual class.
- Intension: the definition of a conceptual class.
- Extension: the set of examples to which the conceptual class applies.

42

Domain Model



43

Domain Concept

- Examples of domain concept/entities:
  - Person: EMPLOYEE, STUDENT, PATIENT
  - Place: STORE, WAREHOUSE
  - Object: MACHINE, PRODUCT, CAR
  - Event: SALE,REGISTRATION, RENEWAL
  - Concept: ACCOUNT, COURSE




Domain Concept

- Guidelines for naming and defining class/entity types:
  - An concept class/entity type name is a singular noun
  - An entity type should be descriptive and specific
  - An entity name should be concise
  - Event entity types should be named for the result of the event, not the activity or process of the event.

Attributes

- Example of concept class/entity types and associated attributes:

  STUDENT: student_ID, student_Name, home_Address, phone_Number, major

- Guidelines for naming attributes:
  - An attribute name is a noun.
  - An attribute name should be unique
  - To make an attribute name unique and clear, each attribute name should follow a standard format
  - Similar attributes of different entity types should use similar but distinguishing names.

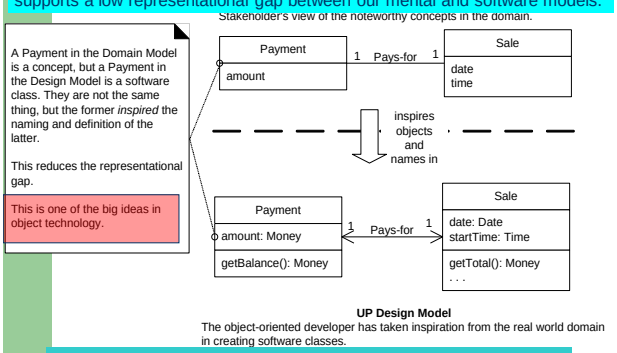
Domain Model

- Domain model To help understanding and more…
  - To understand the key concepts and vocabulary in a domain
  - To support a lower gap between the software representation and our mental model of the domain.
    - Lower representation Gap with OO modelling
    - A key idea in OO.

47

Domain Model

supports a low representational gap between our mental and software models.



UP Design Model
The object-oriented developer has taken inspiration from the real world domain in creating software classes.

48 here's a source-code payroll program written in 1953: 1000010101000111101010101010001010101010101111010101 … the

Domain Model

- How to create a domain model?
- ***Bounded*** by the current iteration requirements under design:
  - Find the conceptual classes
  - Draw them as classes in a UML class diagram.
  - Add associations and attributes.

49

Noun Phrase Identification

- Identify the nouns and noun phrases in textual descriptions of a domain, and consider them as ***candidate*** conceptual classes or attributes.
- The use cases are an excellent description to draw from for this analysis.

Main Success Scenario (or Basic Flow):

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale**.
3. **Cashier** enters **item identifier**.
4. System records **sale line item** and presents **item description, price**, and running **total**. Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment**.
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions**) and **Inventory** systems (to update inventory).
9. System presents **receipt**.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

. . .

50

Main Success Scenario (or Basic Flow):

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
2. **Cashier** starts a new **sale**.
3. **Cashier** enters **item identifier**.
4. System records **sale line item** and presents **item description, price**, and running **total**. Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5. System presents total with **taxes** calculated.
6. Cashier tells Customer the total, and asks for **payment**.
7. Customer pays and System handles payment.
8. System logs the completed **sale** and sends sale and payment information to the external **Accounting** (for accounting and **commissions**) and **Inventory** systems (to update inventory).
9. System presents **receipt**.
10. Customer leaves with receipt and goods (if any).

Extensions (or Alternative Flows):

. . .

5

Noun Phrase Identification

- Some of these noun phrases are candidate conceptual classes
- some may refer to conceptual classes that are ignored in this iteration (for example, "Accounting" and "commissions"), and

52

Noun Phrase Identification

- Different noun phrases may represent the same conceptual class or attribute, among other ambiguities.
- excluding abstract nouns (may become class attributes)
- excluding nouns outside problem boundary
- Some may be simply attributes of conceptual classes.
  - Such as price

53

Example: Case Study: POS Domain

Sale	**Cashier**
CashPayment	**Customer**
SalesLineItem	**Store**
Item	**ProductDescription**
Register	**ProductCatalog**
Ledger	

54

Example: Case Study: POS Domain

- In practice, one can immediately draw a UML class diagram of the conceptual classes as we uncover them.

Register	Item	Store	Sale
Sales LineItem	Cashier	Customer	Ledger
Cash Payment	Product Catalog	Product Description	

55

- Future Cars

Guideline: Report Objects - Include 'Receipt' in the Model?

- Should we include “Receipt” in domain model?

57

Guideline: Report Objects - Include 'Receipt' in the Model?

- Should we include “Receipt” in domain model?
  - No. In general, showing a report of other information in a domain model is not useful since all its information is derived or duplicated from other sources.
  - Yes. On the other hand, it has a special role in terms of the business rules: It usually confers the right to the bearer of the (paper) receipt to return bought items..

58

Guideline: Report Objects - Include 'Receipt' in the Model?

- The decision:
  - Since item returns are not being considered in this iteration, Receipt will be excluded.

59

Guideline: A Common Mistake with Attributes vs. Classes

- The common mistake -- to represent something as an attribute when it should have been a conceptual class.
- A rule of thumb is:
  - *If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute*

60

Guideline: A Common Mistake with Attributes vs. Classes

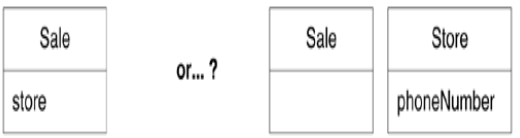
- As an example, Should "**destination**" be an attribute of "**Flight**", or a separate conceptual class "**Airport**"?

  - In the real world, a destination airport is ***not*** considered a number or text - it is a massive thing that occupies space. Therefore, Airport should be a concept.

61

Guideline: A Common Mistake with Attributes vs. Classes

- As an example, should "**store**" be an attribute of "**Sale**", or a separate conceptual class "Store"?



  - In the real world, a store is ***not*** considered a number or text - the term suggests a legal entity, an organization, and something that occupies space. Therefore, Store should be a conceptual class.

62

Guideline: When to Model with 'Description' Classes?

- A description class contains information that describes something else.
  - For example, a *ProductDescription* that records the price, picture, and text description of an Item.
- Why?
- The same situation that calls for a description class happens a lot….

63

Guideline: When to Model with 'Description' Classes?

- Assume the following:
  - An Item instance represents a physical item in a store; as such, it may even have a serial number.
  - An Item has a description, price, and itemID, which are not recorded anywhere else.
  - Every time a real physical item is sold, a corresponding software instance of Item is deleted from "software land."

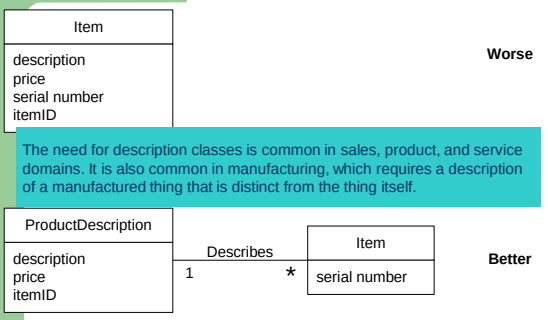
64

Guideline: When to Model with 'Description' Classes?

- To solve the Item problem, what is needed is a ***ProductDescription*** class that records information about items.
- A ProductDescription does not represent an Item, it represents a description of information about items.

65

Guideline: When to Model with 'Description' Classes?



The need for description classes is common in sales, product, and service domains. It is also common in manufacturing, which requires a description of a manufactured thing that is distinct from the thing itself.

Switching from a conceptual to a software perspective, note that even if all inventoried items are sold and their corresponding Item software instances are deleted, the ProductDescription still remains.

66

Guideline: When are Description Classes useful?

- **Add a description class (for example, ProductDescription) when:**
  - There needs to be a description about an item or service, independent of the current existence of any examples of those items or services.
  - Deleting instances of things they describe (for example, Item) results in a loss of information that needs to be maintained, but was incorrectly associated with the deleted thing.
  - It reduces redundant or duplicated information.

67

Example: Descriptions in the Airline Domain

- Consider an airline company that suffers a fatal crash of one of its planes.
- Assume that all the flights are cancelled for six months pending completion of an investigation.
- Also assume that when flights are cancelled, their corresponding Flight software objects are deleted from computer memory. Therefore, after the crash, all Flight software objects are deleted.

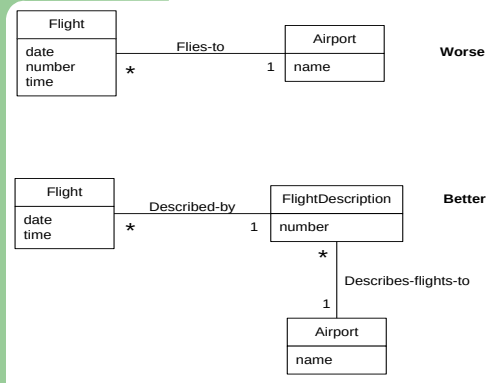
68

Example: Descriptions in the Airline Domain

- If the only record of what airport a flight goes to is in the Flight software instances, which represent specific flights for a particular date and time, then ***there is no longer a record of what flight routes the airline has.***
- The problem can be solved, both from a purely conceptual perspective in a domain model and from a software perspective in the software designs, with a ***"FlightDescription" class*** that describes a flight and its route, even when a particular flight is not scheduled

69

Example: Descriptions in the Airline Domain



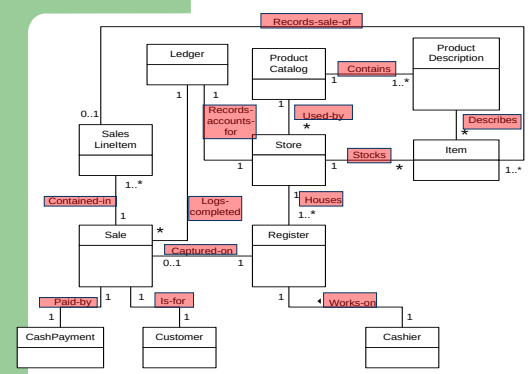
70

A Short Review Break

- Domain model To help understanding and more…
  - To understand the key concepts in a domain
  - To support a lower gap between the software representation and our mental model of the domain.
    - A key idea in OO.
- ***Bounded*** by the current iteration requirements under design:
  - Find the conceptual classes
  - Draw them as classes in a UML class diagram.
  - Add associations and attributes.
- Guideline: Report Objects - Include 'Receipt' in the Model?
- Guideline: Think Like a Mapmaker; Use Domain Terms
- Guideline: A Common Mistake with Attributes vs. Classes
- Guideline: When to Model with 'Description' Classes?

71

Example: NextGen POS



72

Exercise

Elevator control system which controls **n** elevators over **m** floors

Exercise

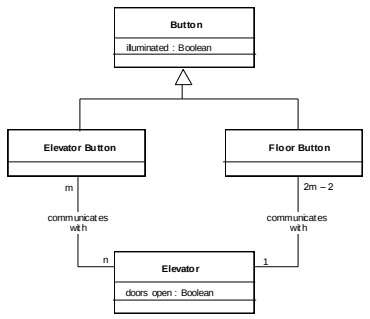
Push elevator bottom use case

- ***Buttons*** **in** ***elevators*** **and on** ***floors*** **control** ***movement*** **of n** ***elevators*** **in** ***building*** **with m** ***floors*** **.** ***Buttons*** **illuminate when pressed to request** ***elevator*** **to stop at specific** ***floor*** **;** ***illumination*** **is canceled when** ***request*** **has been satisfied. If** ***elevator*** **has no requests, it remains at its current** ***floor*** **with its** ***doors*** **closed.**

Exercise

- Noun extraction:
  - button, elevator, floor, movement, building, illumination, request, door
- exclude abstract nouns (may become class attributes)
  - movement, illustration, request
- exclude nouns outside problem boundary
  - floor, building, and door -> button (2 types), elevator

Exercise



Review

Refer to the diagram to answer the question.

From the specification perspective, this qualified association would imply an interface alone the lines of :

A. class Order {
public OrderLine getLineItems() ;
public void addLineItem(Number amount);

B. class Order {
public OrderLine getLineItems() ;
public void addLineItem(OrderLine lineItem);

C. class Order {
public OrderLine getLineItems(Product aProduct) ;
public void addLineItem(Number amount, Product aProduct);

D. class Order {
public Object getLineItems() ;
public void addLineItem(OrderLine lineItem);

Review

- Which model represents the following code generated by a Modeling tool ? (Single select)

class Order {
private Customer recipient;
// ...
}

A. Design A

B. Design B

C. Design C

Review

- Consider a system that manages the assignment of students and instructors to courses in a university. The CourseSchedule class has add(c:Course) and remove(c:Course) methods defined in it. What is the relationship between CourseSchedule and Course classes ?

A. Generalization relationship
B. Composition relationship
C. Inheritance relationship
D. Dependency relationship
E. Association relationship

Review

- Which of the following is true?

A: An object is an instance of a class
B: A class is an abstract definition for a set of objects
C: An object can be in more than one class
D: An object has a life span.
E: All of the above

-

Review

- If a "student signs up for a class," which type of relationship would you use to model the relationship between the two?

a. generalization
b. association
c. aggregation
d. subsetting

Review

- Which of the following could be a superclass of employee?

a. person
b. administrator
c. president
d. CEO

Review

- How does the design model (design class diagram) differ from the domain model?

- UML class diagrams may contain
  1. classes,
  2. associations between classes,
  3. attributes (a.k.a., fields), and
  4. operations (a.k.a., method signatures).

  Which of the items above is **not included** in a domain model illustration? (circle it)

83

Review

- Consider an on-line store that enables customers to order items from a catalog and pay for them with a credit card. Draw a UML diagram for the domain model that shows the relationship between these classes (with multiplicity if possible and the role as well):

Customer, Order, RushOrder, Product, Address, CreditCard

84

Review

- The following describes information used in a car loan system. Cars may be owned by persons, companies, or banks. A car loan from a bank may be involved in the purchase of a car. An owner can own several cars. A car can have several loans against it. Banks lend money to persons, companies, or other banks.

Draw the domain model using UML class diagram for the car loan system.

85

Review

- Consider a simple smart card door security system such as on the outside doors of the building. To gain access when the door is locked, the smart card is entered into the reader which beeps if the insertion is the wrong way around. The details of validly entered cards are sent to the control computer for checking. If access is permitted the indicator on the reader turns from red to green and door controller is unlocked for ten seconds allowing the person to enter. Otherwise the reader beeps, remains indicating red and the door stays locked.
- draw the use case diagram for the system
- Create a conceptual model (domain model) and illustrate it in UML notation. Show concepts, associations, and attributes

86

- What is IoT?