

Today

- More on generalization/specialization hierarchies
 - (Finish chapter 26 on the domain model)
 - This is closely connected with inheritance
- More on refining the domain model (chap. 27)
- Update to HW statement: mainly, to email haoli@iastate.edu to make an appt. to show your system

Defining Subclasses

- These will, upon design/implementation, be
 - ...classes that inherit from their base class
- Recall an example: Figures 26.1 & 26.5
 - (Note boxed guidelines p. 399)
- Table 26.2 summarizes the big four criteria
 - (see table)

From Subclasses to Superclasses

- Let A be a subclass of B
 - T or F: a superclass relationship can be deduced

From Subclasses to Superclasses

- Let A be a subclass of B
- ___ is a superclass of ___

From Subclasses to Superclasses

- Let A be a subclass of B
- ___ is a superclass of ___
- We've looked at the situation where you
 - ...have a class C
 - ...want to express variations of that class
 - ...consider defining them as subclasses of C
- You can also have similar classes, and should
 - ...consider defining a superclass of them

Superclasses

- Let's suppose your domain model contains
 - A checkPayment class
 - A creditThePayment class
 - A payByCash class
- T or F: Should these be given similar names?
 - checkPayment, creditPayment, and cashPayment
- Think-pair-share: Why?

Superclasses

- Let's suppose your domain model contains
 - A checkPayment class
 - A creditPayment class
 - A cashPayment class
- T or F:
 - Should a payment superclass be defined?
- Let's see if the boxed points (p. 403) apply

Superclasses: Overdoing It

- If one diagram is
 - (a) more complex than another, and
 - (b) does not add value, then
 - it must be a worse diagram
 - ...because (a) makes it worse, and (b) does not compensate
- Example: Figures 26.9 vs. 26.10
- Note –
 - It is not clear which is better
 - ... but (b) tells us how to decide

Abstract Superclasses

- (Reference: section 26.7)
- Let class1 be an **Abstract class**:
 - A class with no members directly (no constructor)
 - but its subclasses have members (each IS-A class1)
 - This is true for conceptual classes
 - e.g. in a domain model
 - This is also true for Java software
 - you can define a class as abstract
- "If every member of a class C must also be a member of a subclass" of C, is C abstract?

Abstract Classes: the UML

- UML notation for abstract classes:
 - Italicize the class name
 - See Figure 26.12

Sub/Superclasses and State Changes

- See Figure 26.13
- If the candidate subclasses are
 - simply alternate states of the candidate superclass, then
 - avoid defining the subclasses
 - state change should be modeled in other ways
- Are check, credit, and cash payments different states of a payment concept?

Beyond Super/Subclasses

- Superclasses and subclasses are one way to refine a domain model
- There are other ways as well (see chap. 27)
 - Adding association classes
 - Adding aggregations
 - Modeling time
 - Modeling roles
 - Breaking a domain model into packages (subsystems)

Association Classes

- Consider this:
 - For a person class, would **phoneNumber** be a good attribute?
 - Think about it for a minute
 - (If the vote is non-unanimous I might ask you to jot something down and pass it up)

Association Classes

- A guideline:
 - If an object of class C can have
 - ...**multiple** values for a given attribute
 - ...put them in other objects
 - ...and hook them to C (i.e. associate them to C)
 - What data structure might be good for this?

Taking Association Classes Further

- Association classes are often associated
 - ...with two other classes at once
- A person is related to a place by some set of phone numbers
 - See Figure
 - The association itself, conceptually, has attributes
 - Implementationally,
 - objects of each non-association class...
 - ...hook to...
 - ...objects of the association class

Aggregation

(ref.: Section 27.2)

- We've seen IS-A relationships
- There are also part-of relationships - hence
 - Hand aggregates fingers
 - Sale aggregates SaleLineItems
- UML: use a filled diamond on the arc
 - See Figure 27.5
 - (If fingers can float around by themselves, use a hollow diamond)

Aggregation II

- Two POS examples of aggregations
 - Sale aggregates SaleLineItems
 - ProductCatalog aggregates productSpecifications
- Draw class diagrams for these situations...