

Today

Finish chapter 27 (associations, packages)
Mostly skip chapter 28 (re-applies things covered earlier to advance the POS system)

Chapter 29 (Statechart diagrams)

- Four more lectures
- 1.1 more assignments (110 pts.)
- Next Assignment is due in 1 week,
- ...will be posted by 5:00 p.m.

Qualified Associations

- See Figure 27.14
- Note the **qualifier**
- The qualifier says...
 - ...how to distinguish one object from another
- Does the qualifier apply to...
 - ...the adjoining class, or
 - ...the class at the far end of the association?
- Note the change in multiplicity in the association

Associations and Multiplicity

- Does (a) say that
 - each product catalog contains 1-n product specs, &
 - **1-n** product specs are contained in each catalogor
 - each product catalog contains 1-n product specs, &
 - **each** product spec is contained in 1 catalog?

Associations and Multiplicity II

- See Figure 27.15; does it say
 - Each child has 2 parents, and
 - Each parent has 0-infinite childrenor
 - 0-infinite children have two parents, and
 - Two parents have 0-infinite children?
- Note also the possibility of **reflexive associations**

Associations with Ordered Elements

- Sometimes in a 1-n relationship
 - The n items are unordered
- Examples:
 - Stars in the night sky
 - Vehicles crossing a particular intersection on a sample day
- Sometimes items are ordered
 - A Sale has SalesLineItem objects in order of scanning
- To specify ordering, see Figure 27.16
 - This is optional – do it if it enhances clarity

Associations and Packages

- Recall a package is a
 - Module
 - A large module
 - A grouping of classes together
 - A named grouping
- Which of the following classes are logical to group into the same package?
 - T/F: classes that are closely related by concept of purpose
 - T/F: classes that are in the same inheritance heirarchy
 - T/F: classes that participate in the same use cases
 - T/F: classes that are strongly associated (see p. 425)

Associations and Packages II

- Which of the following classes are logical to group into the same package?
 - T/F: design model classes that are also in the domain model
 - T/F: design model classes that are **not** in the domain model
 - (like what?)
 - T/F: classes that are used pervasively ("core" classes)
 - T/F: classes with no obvious other package ("miscellaneous" classes)

Associations and Packages III

- The text suggests a core/misc. package
- Packages can form a heirarchy (packages within packages – see Figure 27.19)

Statechart Diagrams

- Reference: chapter 29
- Statechart diagrams are analogous to finite state machines
 - In FSMs, each state is a node
 - Arcs express transitions between states
- Statechart diagrams involve the same idea of states and their transitions
- More specifically, statecharts involve events, states, and transitions
 - Event: something noteworthy happens
 - State: status or condition between events
 - There can be a state of change (water tank is filling)
 - This is a difference from the more low-level FSMs
 - Thus state is not a statement about physics, but about software structure
 - Transition: a link between 2 states labeled with an event
 - When the event occurs, the current active state changes per the link

Statechart Diagrams II

- The graphical notation for statechart diagrams has
 - Many details in the UML, of which some are usually or always used
 - See Figure 29.1
 - Note rounded rectangles (state, event, or transition?)
 - Note solid lines with arrowheads (state, event, or transition?)
 - Note labels of the arcs (state, event, or transition?)
 - Note solid dot (initial "state" pointing to initial state)
 - "Idle" is initial state of software once it is running
 - Dot indicates status of software prior to initialization

Statechart Diagrams III

- Statechart diagrams can describe
 - An entire system
 - A particular object
 - A use case
 - Anything that can have different states and transitions among them

Statechart Diagrams IV

- Statecharts can be used for Use Cases
- See Figure 29.2
- What objects should (not) have statechart diagrams?
 - If the object always reacts the same way to an event
 - It is state-independent (wrt that event)
 - Objects that are state-independent for all events of interest need not have statechart diagrams
 - (If it did have one, what would it look like?)

Statechart Diagrams V

- Statecharts, internal events, and external events
 - Internal events are messages
 - Do we have a way of depicting the effects of messages? If so, how?
 - External events (system events) are unpredictable from within the system boundaries
 - ...but the system must react
 - Statecharts are especially useful for these

Statechart Diagrams VI

- Statecharts, internal events, and external events
 - Internal events are messages
 - Do we have a way of depicting the effects of messages? If so, how?
 - External events (system events) are unpredictable from within the system boundaries
 - ...but the system must react
 - Statecharts are recommended for these
 - Temporal events occur as a result of time reaching a certain point
 - These happen independently of other things in the system
 - They are like system events (clock is outside the system?)
 - Statecharts are recommended for these

Statechart Diagrams VII

- Statecharts have some other notations available in UML
 - Figure 29.4 shows
 - transition actions
 - guards
 - Figure 29.5 shows
 - nested states
- A useful exercise to settle these new ideas in your mind:
 - Draw a statechart diagram related to the registration system or a component of it
 - You may work individually or in groups
 - Hand it in with your names (why? I'm haven't figured exactly yet)
- This concludes chapter 29!