INF 111 / CSE 121: Software Tools and Methods

Lecture Notes for Fall Quarter, 2007
Michele Rousseau
Set 18

(Some notes adapted from Susan E. Sim & UML Distilled)

Announcements

- **Homework Due 11/21 @ 3p**
- **TA will be available for questions in class Today & in discussion on Monday**

- **UML Links:**
- http://dn.codegear.com/article/31863#use-case-diagram

Previously in INF 111…

- **UML**
  - Class Diagrams
  - Use Case Diagrams
  - Sequence Diagrams

Review

Review (continued)

Today's Lecture

- **UML**
  - Sequence Diagrams
  - Package Diagrams

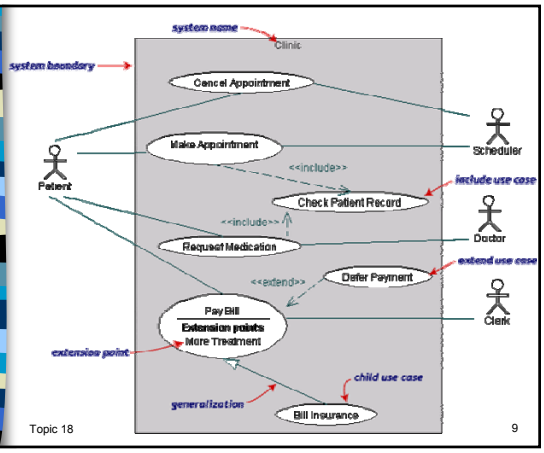
Use Cases:Include & Extendsrevisited

- **Includes**
  - A relationship in which one use case (the base use case) includes the functionality of another use case
  - Promotes reuse
  - Should be used when the inclusion case is common in two or more use cases

Both use similar notation, but are very different.
Represented with a dashed line and <<includes>> or <<extends>>

Include & Extend revisited

- **Extend:**
  - specifies that one use case (extension) extends the behavior of another use case (base).
  - reveals details about a system or application that are typically hidden in a use case
  - the extension use case is not meaningful on its own
  - Describes behavior sequences that can change the base case
  - Each behavior sequence can be inserted into the base use case at a different point, called an extension point
  - When do you use it
    - A part of a use case that is optional system behavior
    - A subflow is executed only under certain conditions
    - A set of behavior segments that may be inserted in a base use case



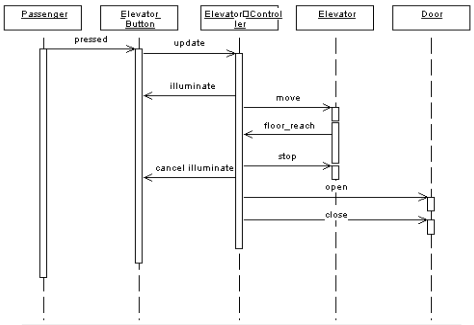
Sequence Diagrams (revisited)

- **Represent one scenario**
- **Describe the dynamic behavior of a system**
- **Good at**
  - describing the behavior of several objects within a single use case
  - Showing collaborations between objects
- **Not good at precise definition of the behavior**

Basic Elements of a Sequence Diagram

- Objects
  - **Lifelines: time goes from top to bottom**
  - **May be several instances of one class**
  - **Boxes on lifelines show if object is active**
- Messages
  - **Analogous to method calls in a program**
  - **Can have parameters**
- Special messages
  - **New — shown by position of object**
  - **Delete — shown with a big X**
  - **Return messages**
  - **Self-calls**

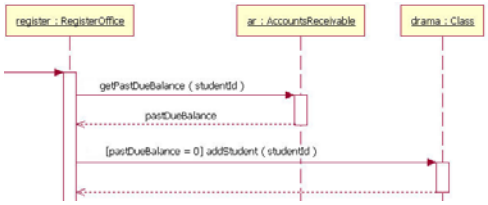
Elevator Example: Sequence Diagram



Sequence Diagram for Serving Elevator Button

Guards

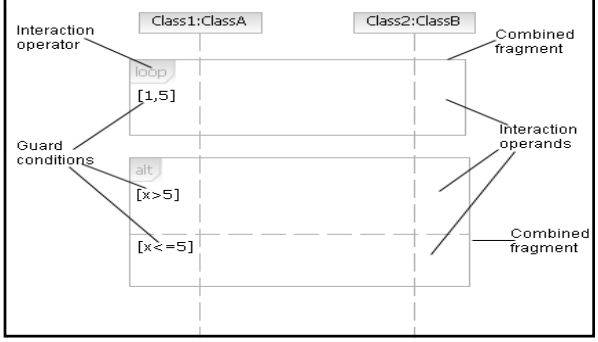
- **When a condition must be met before a message is sent**
- **Represented by brackets on the message line [*guard*]**



Frames

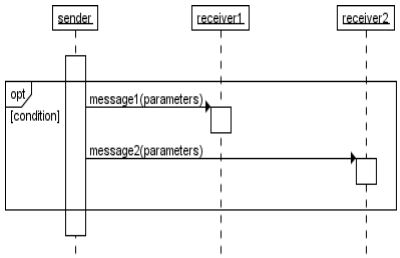
- Encloses a region of a sequence diagram
- Guard specifies condition
  - **Allows you to specify several interactions within a guard**
- Can be divided into one or more fragments
- Keyword specifies the type of frame
- Keywords:
  - **opt -Optional fragment that executes if guard is true**
  - **alt -Alternative fragment for mutual exclusive choice between two or more message sequences**
    - **Eg. If → then → Else**
  - **loop -Loop fragment while guard is true**
  - **par -Fragments that execute in parallel**
  - **region -Critical region within which only one thread can run**

Frames/Combined Fragments

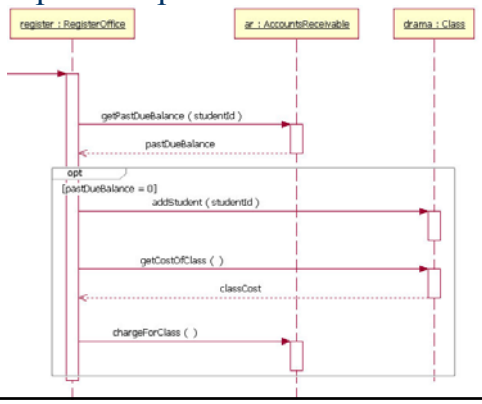


Frame: Option

- **Like a typical guard expanded**

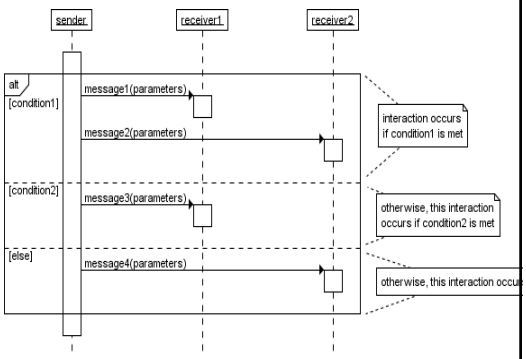


Opt Example



16

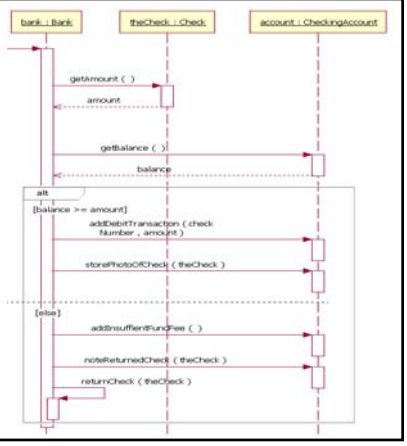
Frame: Alt



Topic 18

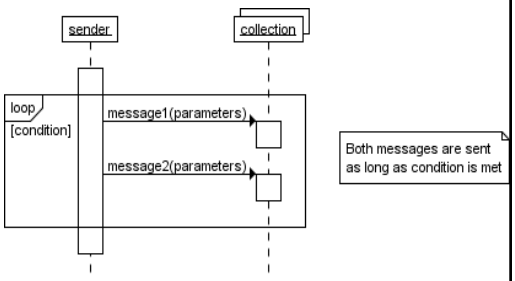
17

Alt Example

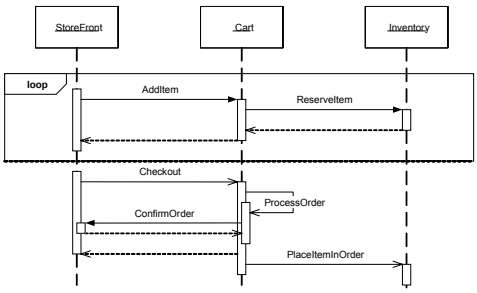


Topic 18

Frame: Loop

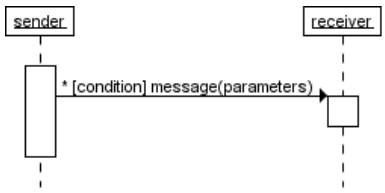


Loop Example



What if you only have 1 msg to loop?

- **Use the "*" symbol**
- **As long as the condition holds the message is sent**



Synchronous & Asynchronous Calls

- Synchronous →
  - **Some methods must finish before another can start**
- Asynchronous →
  - **Some methods can continue executing while others run**