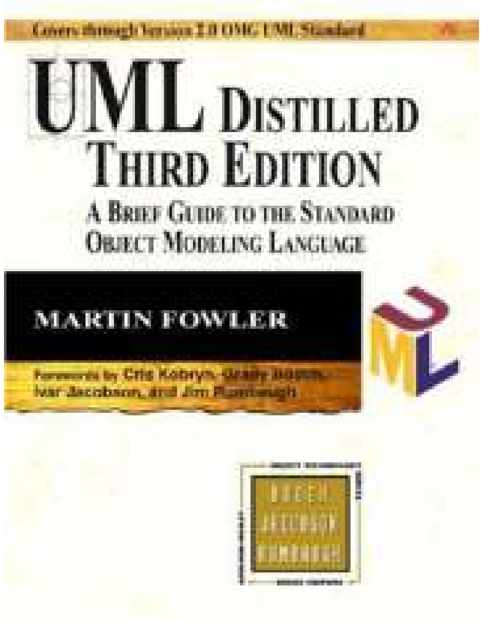
UML



Covers through Version 2.0 OMG UML Standard

UML DISTILLED
THIRD EDITION

A BRIEF GUIDE TO THE STANDARD OBJECT MODELING LANGUAGE

MARTIN FOWLER

Forewords by Cris Kobryn, Grady Booch, Ivar Jacobson, and Jim Rumbaugh

UML Sequence Diagrams

CSE 403

UML sequence diagrams

- **sequence diagram**: an “interaction diagram” that models a single scenario executing in the system
  - one of the most common UML diagrams

- relation of UML diagrams to other exercises:
  - CRC cards -> class diagram
  - use cases -> sequence diagrams

Key parts of a sequence diagram

- **participant**: an object or an entity;
  the sequence diagram actor
  - sequence diagram starts with an unattached "found message" arrow

- **message**: communication between objects

- the axes in a sequence diagram:
  - horizontal: which object/participant is acting
  - vertical: time (↓ forward in time)

Sequence diagram from use case

Basic Course

The Customer specifies an author on the Search Page and then presses the Search button.

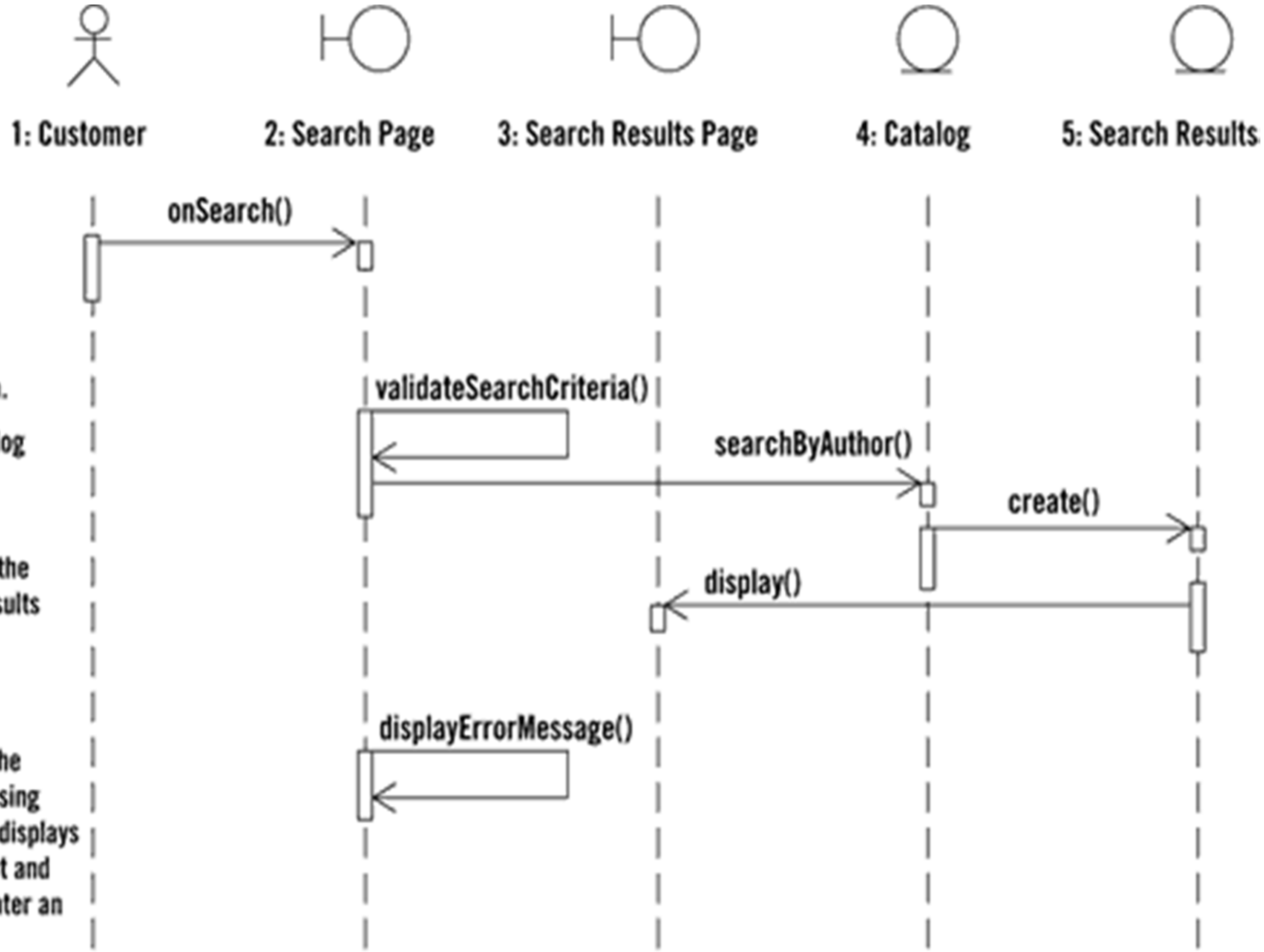
The system validates the Customer's search criteria.

The system searches the Catalog for books associated with the specified author.

When the search is complete, the system displays the search results on the Search Results Page.

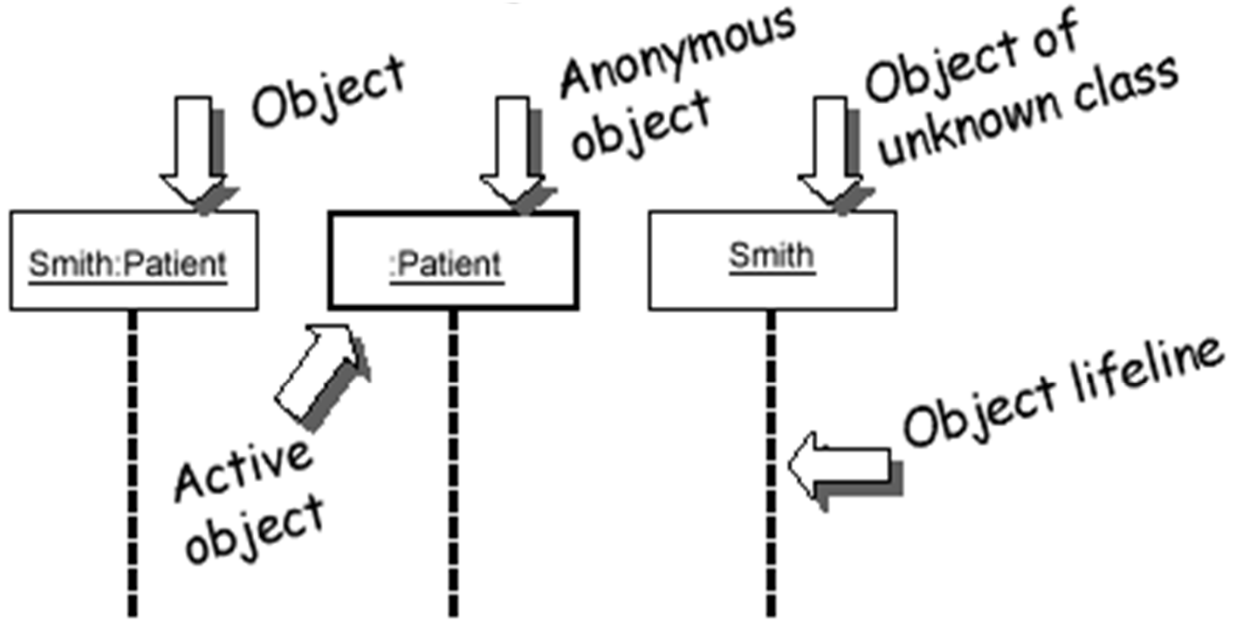
Alternate Course

If the Customer did not enter the name of an author before pressing the Search button, the system displays an error message to that effect and prompts the Customer to re-enter an author name.



Representing objects

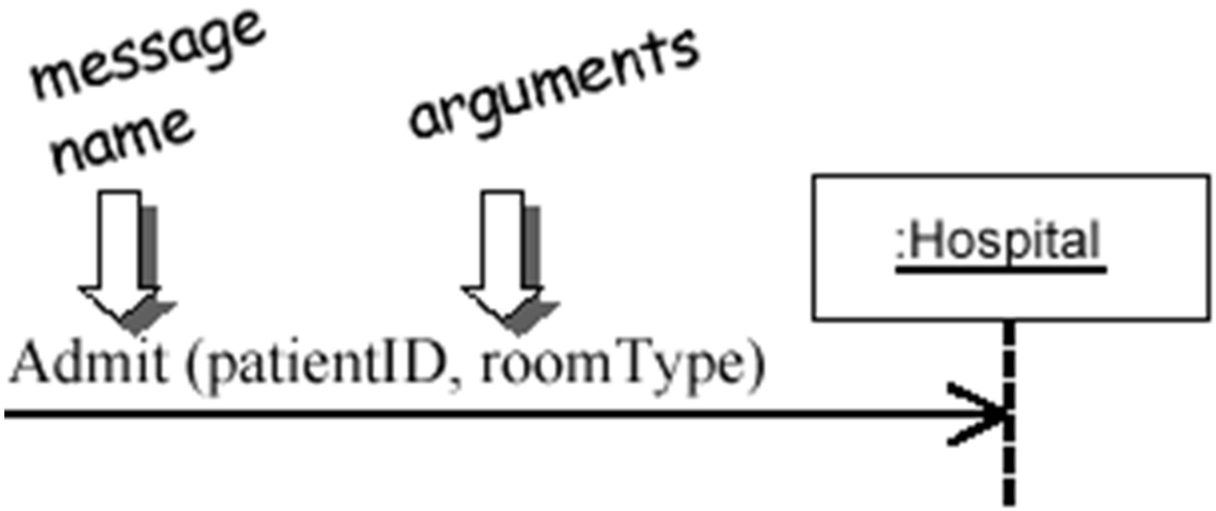
- An object: a square with object type, optionally preceded by object name and colon
  - write object's name if it clarifies the diagram
  - object's "life line" represented by dashed vert. line



Name syntax: <objectname>:<classname>

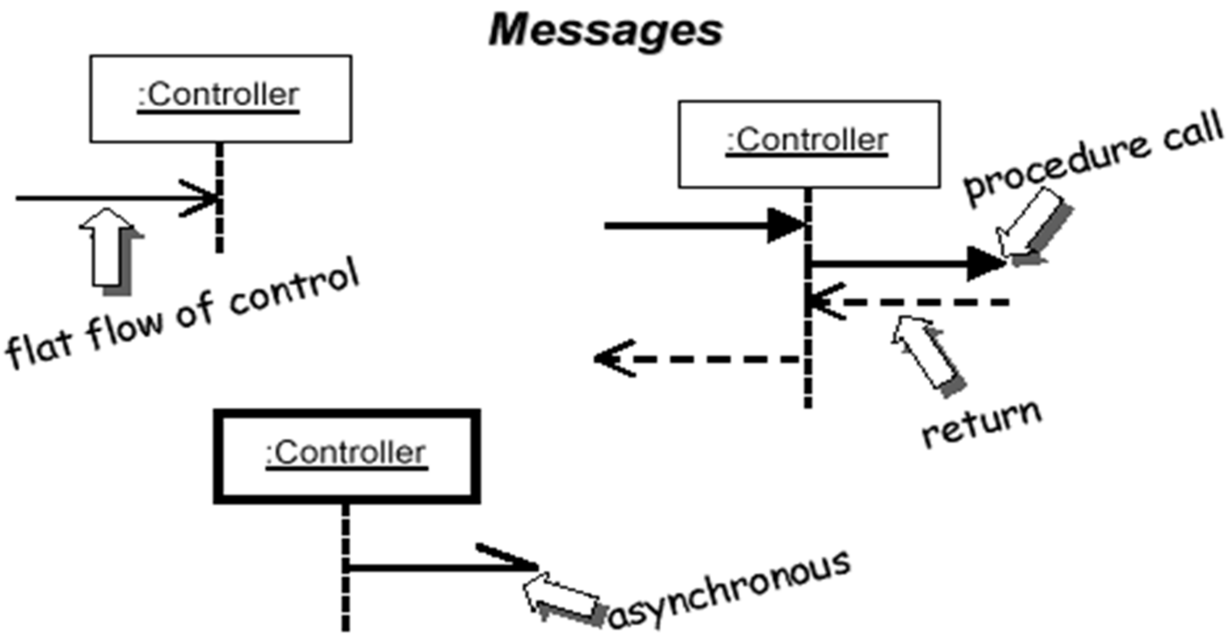
Messages between objects

- message (method call): horizontal arrow to other object
  - write message name and arguments above arrow



Different types of messages

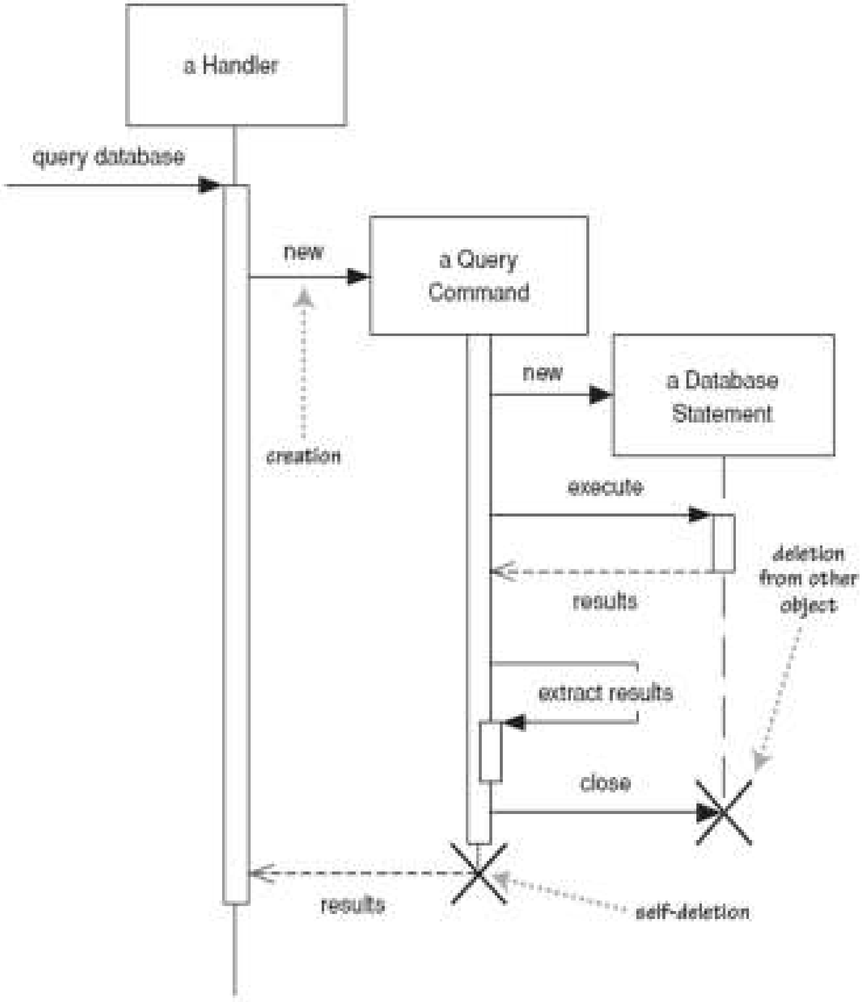
- Type of arrow indicates types of messages
  - dashed arrow back indicates return
  - different arrowheads for normal / concurrent (asynchronous) methods



Lifetime of objects

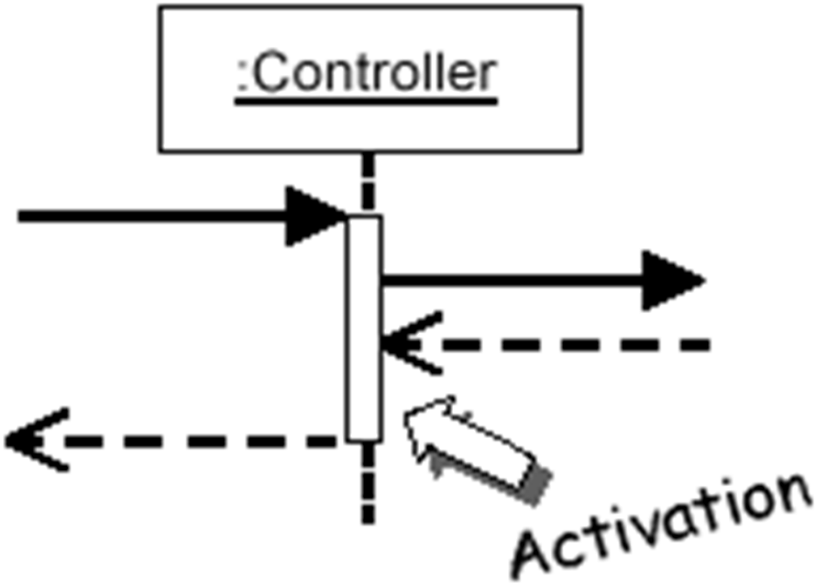
- *creation*: arrow with 'new' written above it
  - an object created after the start of the scenario appears lower than the others

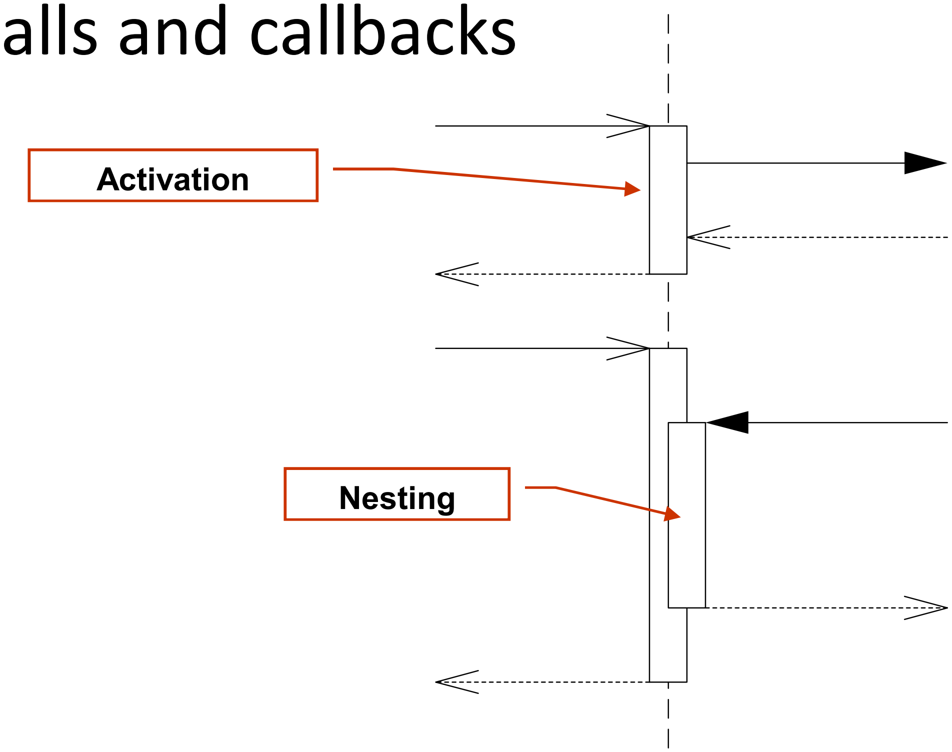
- *deletion*: an X at bottom of object's lifeline
  - Java doesn't explicitly delete objects; they fall out of scope and are garbage collected



Indicating method calls

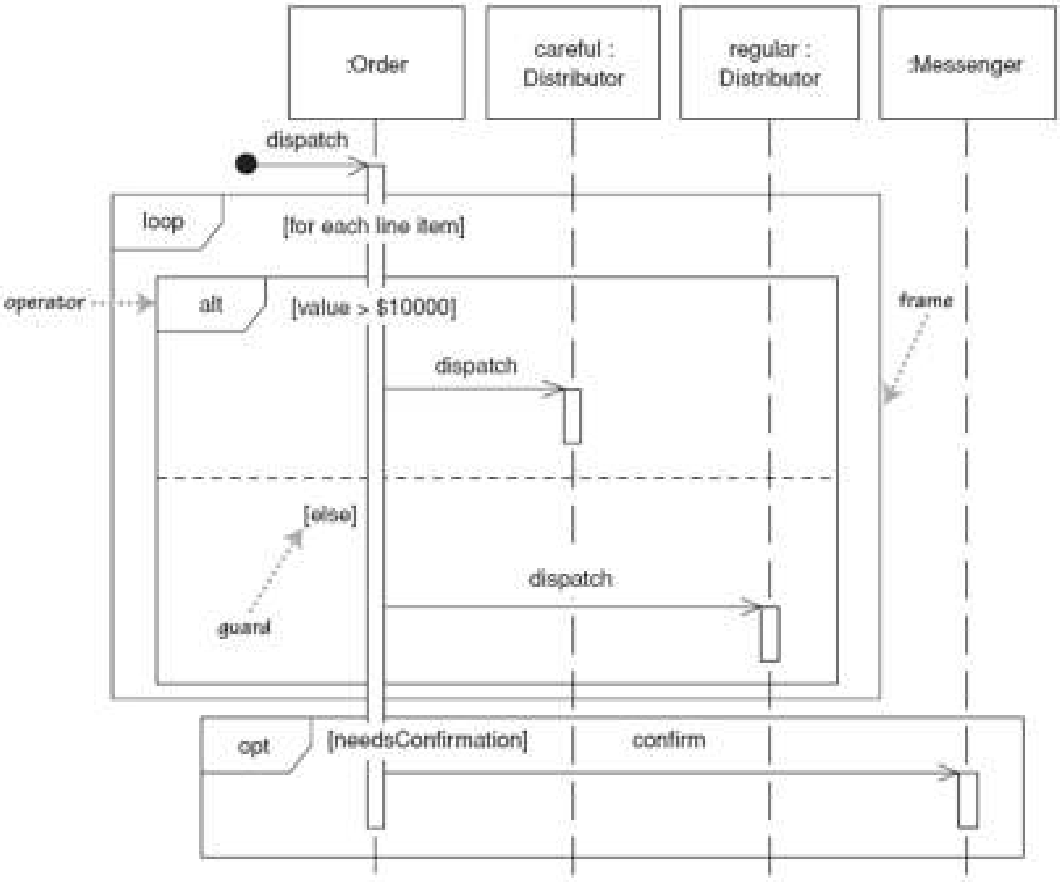
- **activation**: thick box over object's life line
  - Either: that object is running its code or it is on the stack waiting for another object's method
  - nest to indicate self-calls and callbacks





Selection and loops

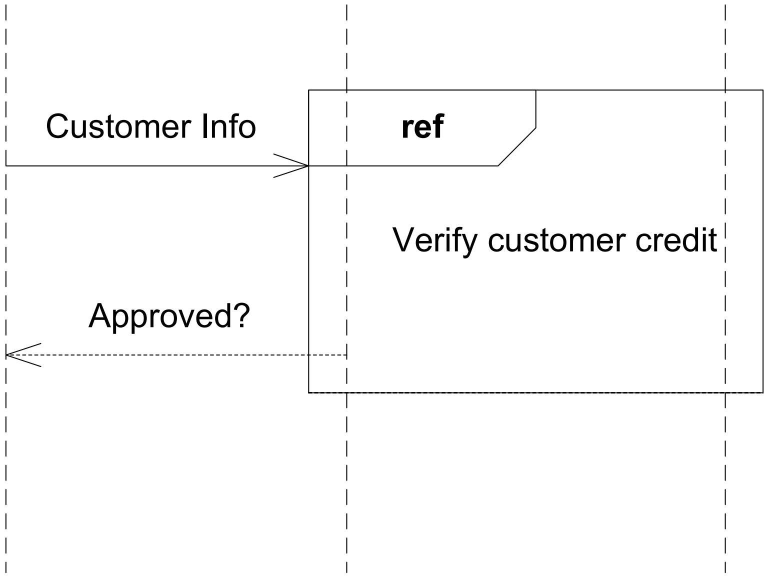
- frame: box around part of a sequence diagram to indicate selection or loop
  - `if` -> (opt) [condition]
  - `if/else` -> (alt) [condition], separated by horizontal dashed line
  - loop -> (loop) [condition or items to loop over]



Linking sequence diagrams

If one sequence diagram is too large or refers to another diagram:

- an unfinished arrow and comment
- a "ref" frame that names the other diagram



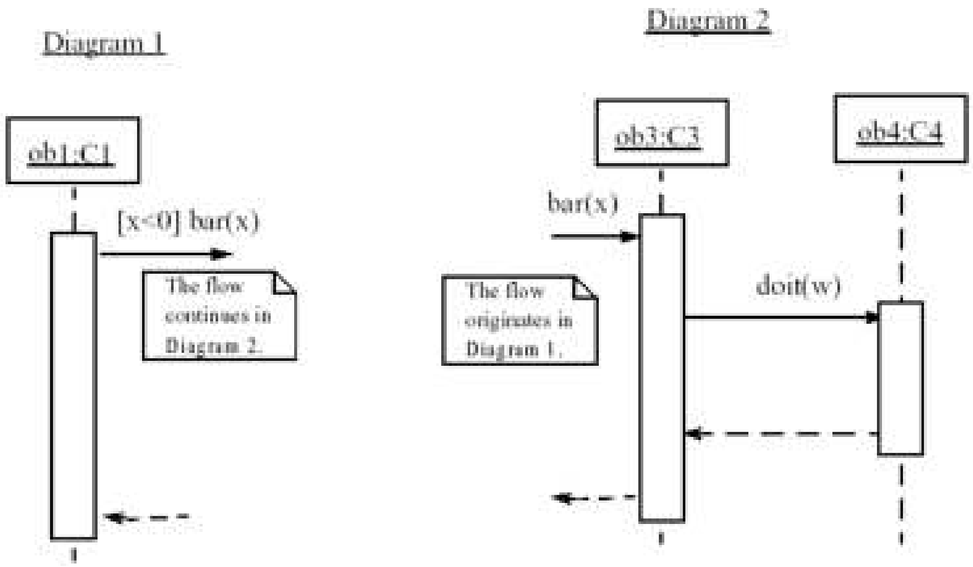
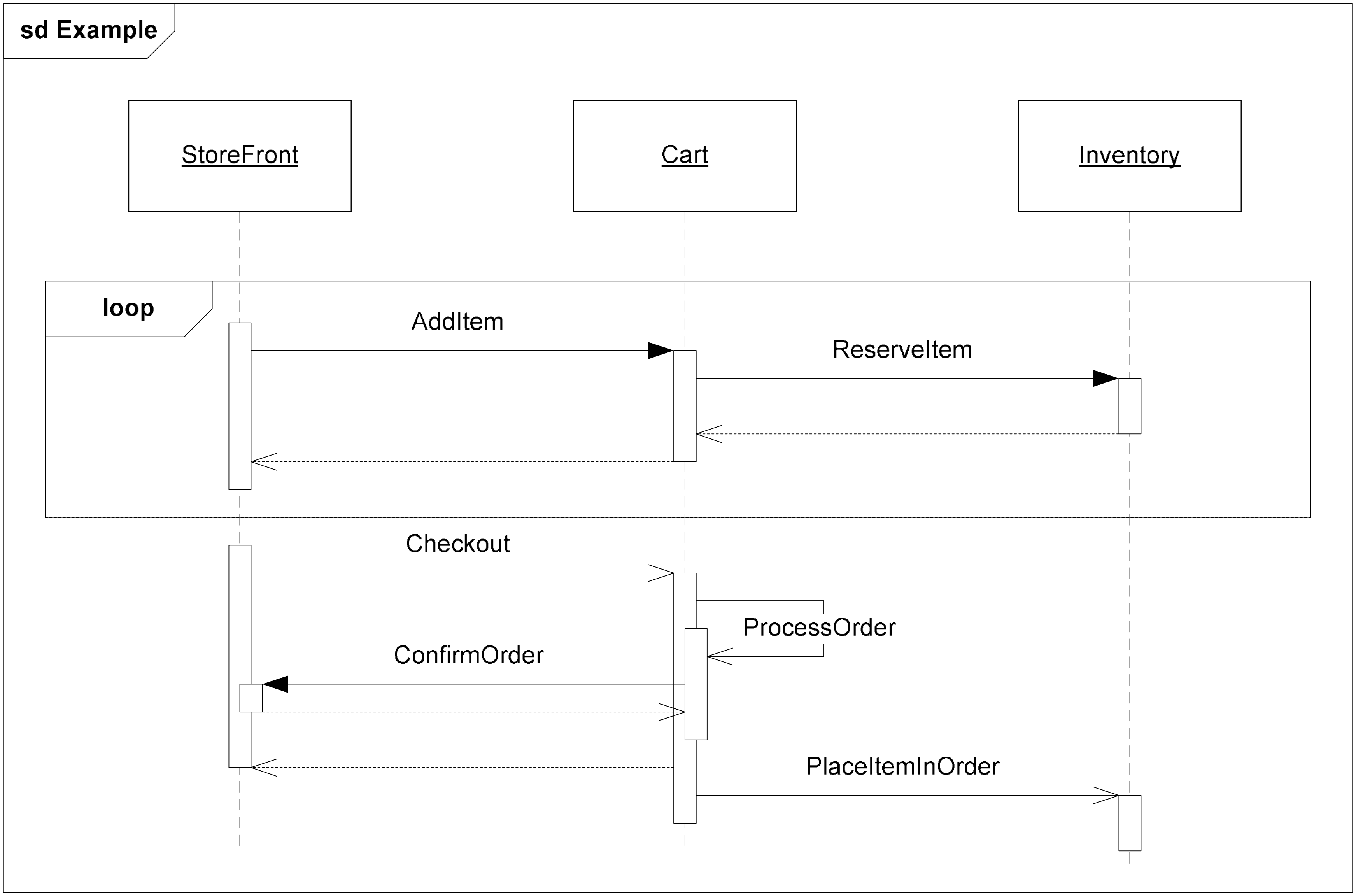


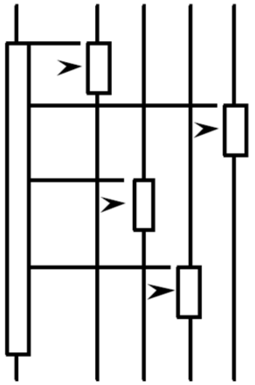
Diagram 1
ob1:C1
[x<0] bar(x)
The flow continues in Diagram 2.
Diagram 2
ob3:C3
ob4:C4
bar(x)
The flow originates in Diagram 1.
doit(w)

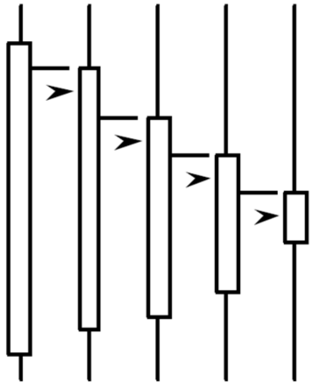
Example sequence diagram

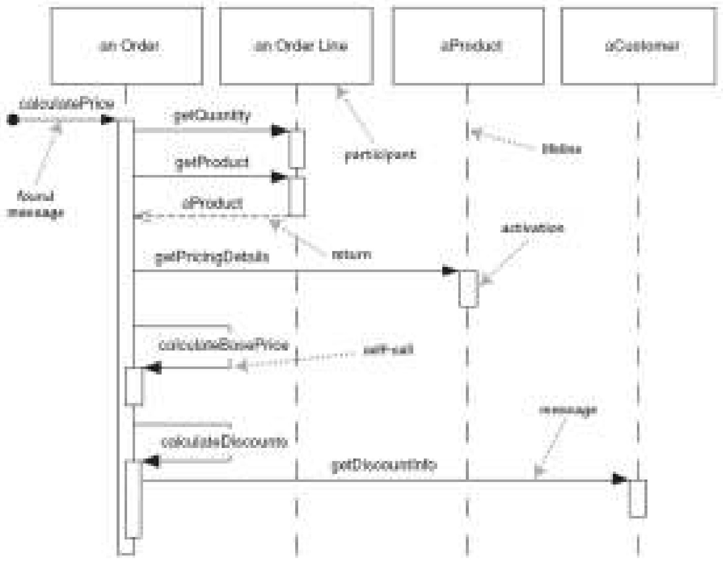


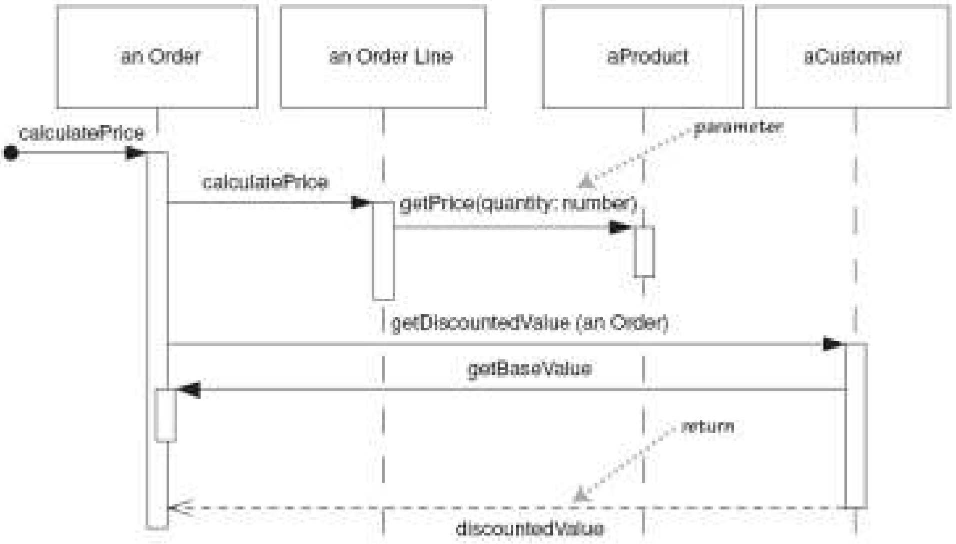
Forms of system control

- What can you say about the control flow of each of the following systems?
  - Is it centralized?
  - Is it distributed?









Why sequence diagrams?
Why not just code it?

- a good sequence diagram is still a bit above the level of the real code (not all code is drawn on diagram)
- sequence diagrams are language-agnostic (can be implemented in many different languages
- non-coders can do sequence diagrams
- easier to do sequence diagrams as a team
- can see many objects/classes at a time on same page (visual bandwidth)

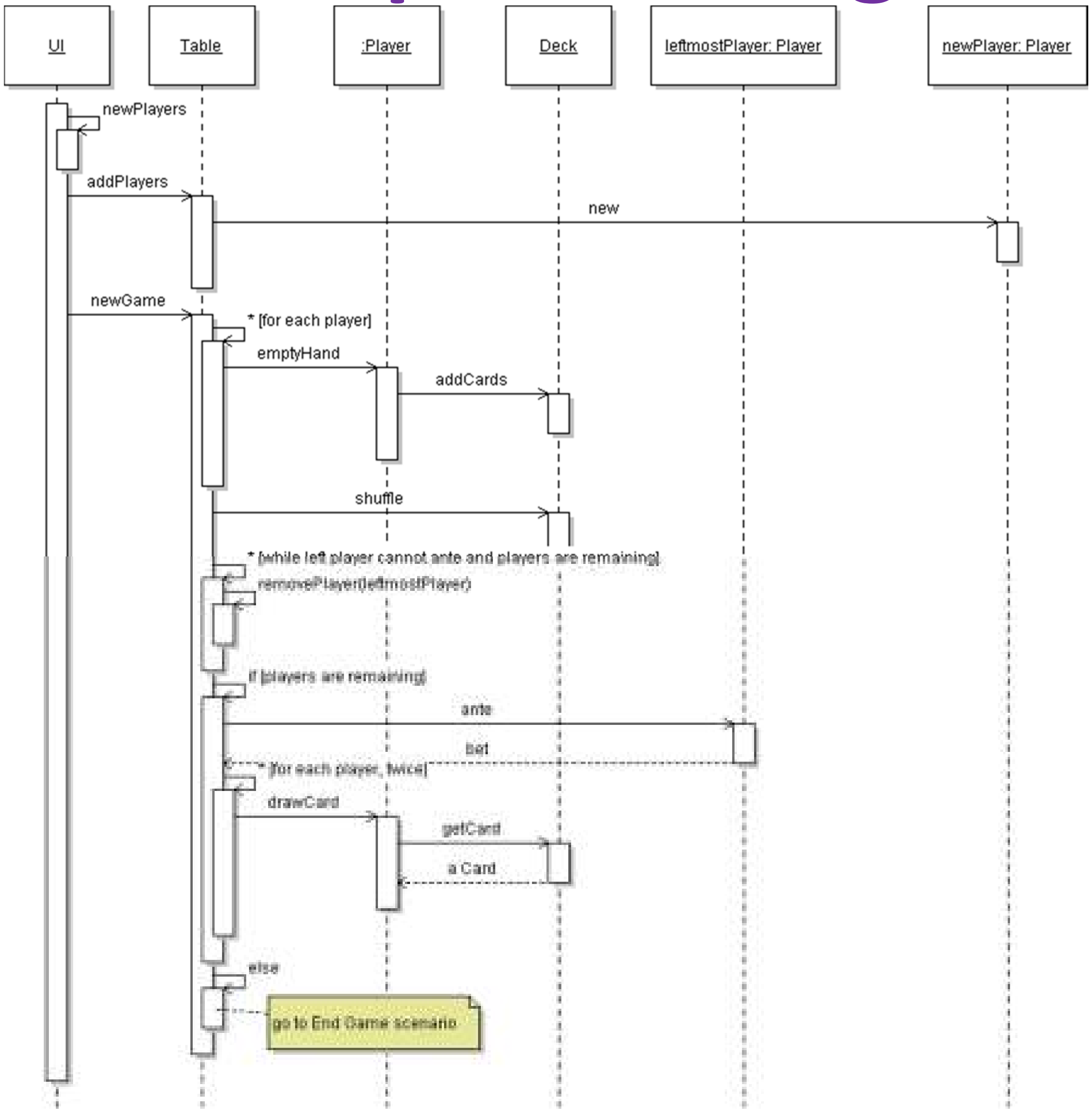
Sequence diagram exercise: Poker use case, *Start New Game Round*

The scenario begins when the player chooses to start a new round in the UI. The UI asks whether any new players want to join the round; if so, the new players are added using the UI.

All players' hands are emptied into the deck, which is then shuffled. The player left of the dealer supplies an ante bet of the proper amount. Next each player is dealt a hand of two cards from the deck in a round-robin fashion; one card to each player, then the second card.

If the player left of the dealer doesn't have enough money to ante, he/she is removed from the game, and the next player supplies the ante. If that player also cannot afford the ante, this cycle continues until such a player is found or all players are removed.

Poker sequence diagram



Sequence diagram question: poker use case, *Betting Round*

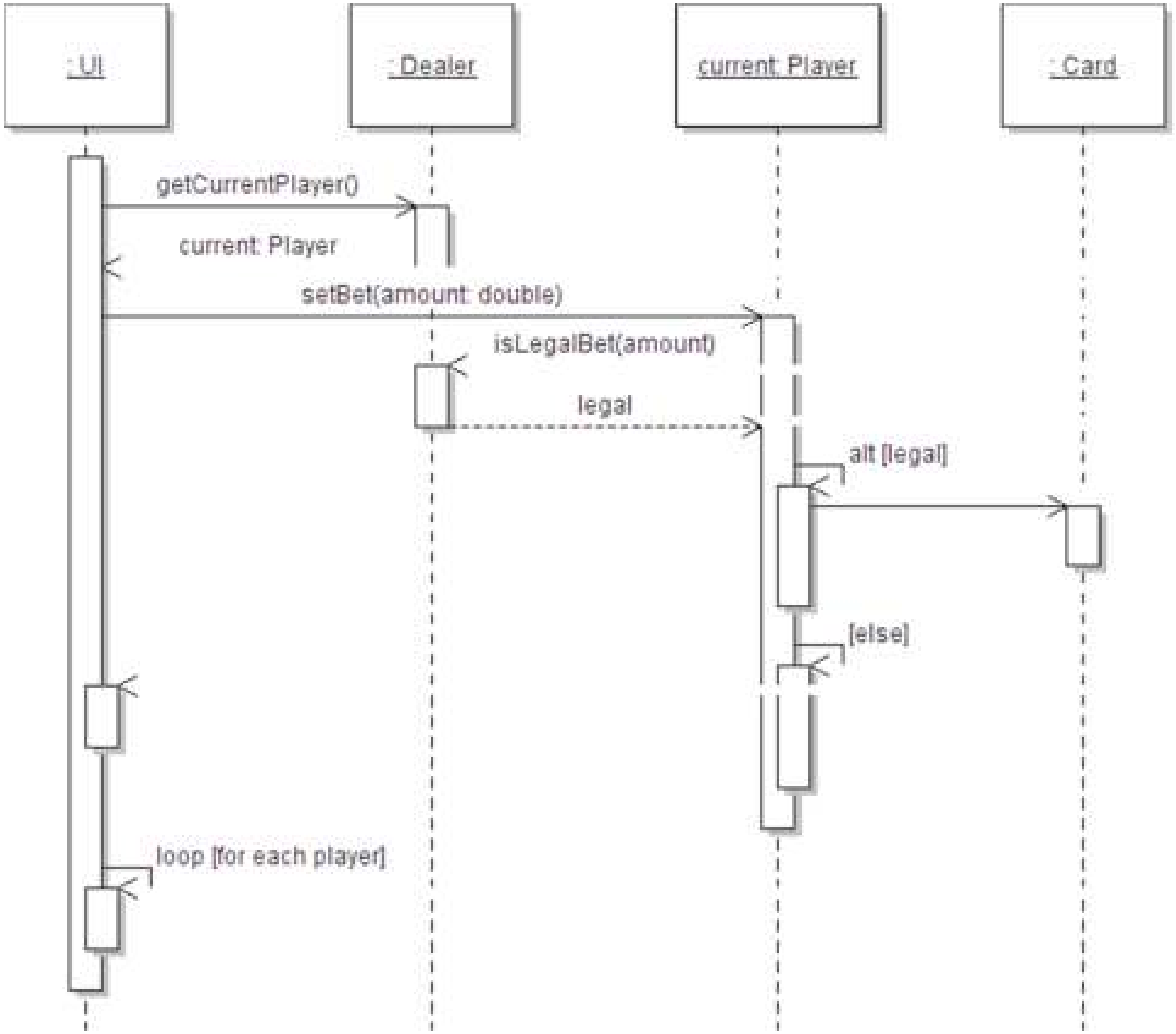
- Why is it hard to diagram this case as a sequence diagram?

The scenario begins after the Start New Round case has completed. The UI asks the first player for a bet. That player chooses to either bet a given amount, or check (no bet).

The next player is asked what to do. If the prior player placed a bet, the next player must either match ("see") it, or match it plus add an additional bet ("raise"), or choose not to match and exit the round ("fold"). This continues around the table until an entire pass is made in which all players have either matched all other players' bets or folded.

If the next player doesn't have enough money to match the current bet, the player is allowed to bet all of their money. But they can then win only up to the amount they bet; the rest is a "side pot" among the more wealthy players remaining in the round.

Poker sequence diagram 2



Exercise: Scheduler app use case, *Add Calendar Appointment*

The scenario begins when the user chooses to add a new appointment in the UI. The UI notices which part of the calendar is active and pops up an Add Appointment window for that date and time.

The user enters information about the appointment's name, location, start and end times. The UI will prevent the user from entering an appointment that has invalid information, such as an empty name or negative duration. The calendar records the new appointment in the user's list of appointments. Any reminder selected by the user is added to the list of reminders.

If the user already has an appointment at that time, the user is shown a message and asked to choose an available time or replace the appointment. If the user enters an appointment with the same name and duration as an existing meeting, the calendar asks the user whether he/she intended to join that meeting instead. If so, the user is added to that meeting's list of participants.