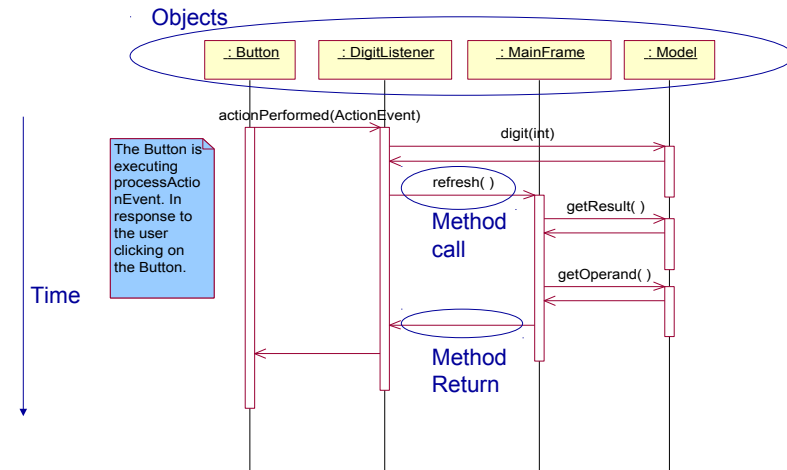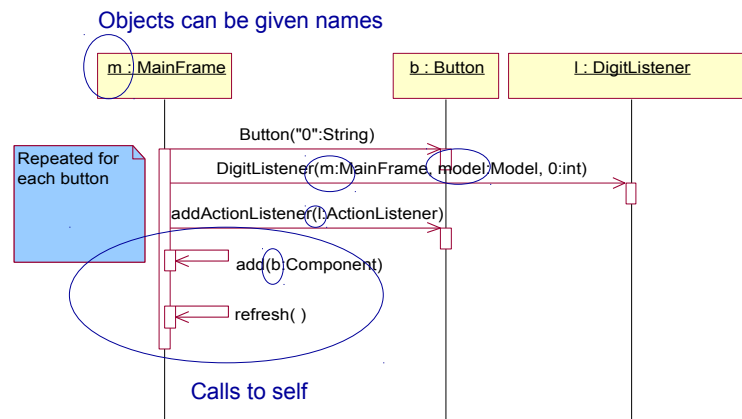# Sequence and Communication Diagrams

- Class Diagrams give a static view of the system. Time is not involved.
- We need also a way to describe the dynamics of the system.
- Sequence diagrams describe typical sequences of method calls.
- Example...

A sequence diagram

Objects

: Button　: DigitListener　: MainFrame　: Model

actionPerformed(ActionEvent)

digit(int)

The Button is executing processActionEvent. In response to the user clicking on the Button.

refresh( )

Method call

getResult( )

getOperand( )

Time

Method Return

Named object and Calls to self

Objects can be given names

m : MainFrame　　b : Button　　l : DigitListener

Repeated for each button

Button("0":String)

DigitListener(m:MainFrame, model:Model, 0:int)

addActionListener(l:ActionListener)

add(b:Component)

refresh( )

Calls to self

Show creation and deletion

a Handler

query database

new

a Query Command

creation

new

a Database Statement

execute

deletion from other object

results

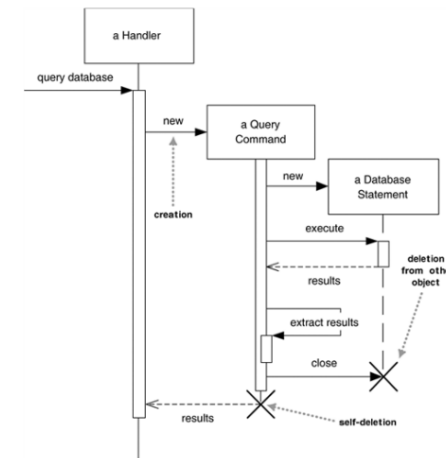extract results

close

self-deletion

results

Figure 4.3. Creation and deletion of participants

- For a language with garbage collection, the big X just means the object is available for removal.

Loops and conditionals

- Sequence diagrams are not ideal for showing loops and conditionals
  - It is better to use pseudocode or other diagram (activity or state diagram)
  - Or simply describe your control flow in additional comments or documentations
- If you really want to, the following slide (from "UML Distilled") gives an example of the notation...

- The following slide gives a sequence diagram for this procedure:

```
procedure dispatch
  foreach (lineitem)
    if (product.value > $10K)
      careful.dispatch
    else
      regular.dispatch
    end if
  end for
  if (needsConfirmation) messenger.confirm
end procedure
```

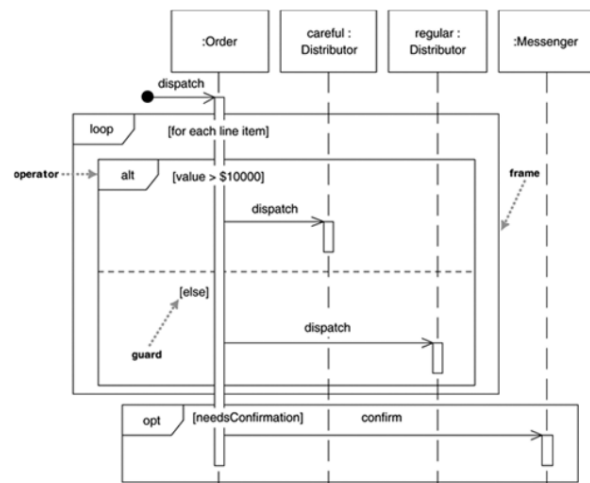

Figure 4.4. Interaction frames

```
procedure dispatch
  foreach (lineitem)
    if (product.value > $10K)
      careful.dispatch
    else
      regular.dispatch
    end if
  end for
  if (needsConfirmation) messenger.confirm
end procedure
```

Sequence Diagrams …

- Hint at but do not describe algorithms
  - For algorithms use pseudo code and/or activity charts and/or state diagrams
- Are good at showing typical or important interactions between a few objects
- Cut across levels of abstraction

Communication Diagrams

- Show the same information as sequence diagrams and also links between the objects
- Show the ordered tree structure of calls by hierarchical numbering

: Button

1.3. | 1. actionPerformed(ActionEvent)

: DigitListener

1.2. refresh( )

1.1. digit(int)   1.2.3.

1.1.1.   1.2.1.1.
1.2.2.1.

: MainFrame

1.2.1. getOperand( )

1.2.2. getResult( )

: Model